MINISTRY OF EDUCATION, CULTURE AND RESEARCH OF REPUBLIC OF MOLDOVA
TECHNICAL UNIVERSITY OF MOLDOVA
FACULTY OF COMPUTERS, INFORMATICS AND MICROELECTRONICS
DEPARTMENT OF SOFTWARE ENGINEERING AND AUTOMATICS

DSL for Fitness and Nutrition Management
Project report

Mentor: asis., Cretu Dumitru
Students: Belih Dmitrii, FAF-232
Rudenco Ivan, FAF-232
Mihalevschi Alexandra, FAF-232
Bujor-Cobili Alexandra, FAF-232
Tatarințev Denis, FAF-232

Chișinău, 2025

Abstract

The project entitled was developed by the students Belih Dmitrii, Rudenco Ivan, Mihalevschi Alexandra, Bujor-Cobili Alexandra, Tatarintev Denis, from Technical University of Moldova. Our project designed to help individuals seamlessly plan their workouts and post-exercise meals using an intuitive Domain-Specific Language (DSL). Many people struggle to balance fitness training with proper nutrition, often relying on separate tools that do not communicate effectively. Our solution integrates both aspects into a single, user-friendly system that generates personalized recommendations based on fitness goals, dietary preferences, and nutritional needs. By simplifying the planning process and ensuring consistency, our project empowers users to make informed decisions that enhance their overall health and well-being.

Content

1 Introduction . . . **4**
1.1 Problem Statement . . . 4
1.2 Who is Affected? . . . 5
1.3 Where and When the Problem Arises . . . 6
2 Literature Review . . . **8**
2.1 Nutrition Management . . . 10
2.2 Macronutrient Breakdown . . . 11
2.3 Nutrient Timing and Hydration . . . 11
2.4 Fitness Management . . . 12
2.5 Review of the Existing Systems/Applications . . . 14
3 Domain Analysis . . . **21**
3.1 Domain Scope and Context . . . 21
3.2 System Analysis . . . 22
3.3 Stakeholder Analysis . . . 24
3.4 Technical Challenges . . . 25
4 Comparative analysis with other projects . . . **26**
4.1 Analysis of Cooking-Recipe-DSL . . . 26
4.2 Analysis of wlang2 . . . 26
4.3 Comparative Analysis of DSL's . . . 27
4.4 Summary . . . 29
5 Implementation . . . **30**
5.1 Grammar Definition and Lexical Analysis . . . 30
5.2 Implementation of Lexer . . . 38
5.3 Implementation of Parser . . . 40
5.4 Visualization . . . 43
5.5 Interpreter and Core Functionality . . . 45
Conclusions . . . **57**
6 Appendix . . . **61**

1 Introduction

1.1 Problem Statement

Despite the growing reliance on digital tools for health management, individuals trying to optimize their fitness and nutrition face a critical problem: **no integrated solution exists to seamlessly connect workout planning with dietary requirements**. *Current apps* specialize in either fitness tracking or meal planning, forcing users to manually coordinate between these systems. This fragmentation leads to inefficiency, inconsistent progress, and potential health risks from poor aligned exercise and nutrition plans. Busy adults, particularly those aged 25–44, struggle to balance time-consuming meal prep and fitness planning with work and family commitments. The lack of customization in existing tools further worsens the problem, as users with unique goals or dietary restrictions cannot easily tailor plans to their needs [1].

The problem lies in the lack of an intuitive and flexible tool that allows users to plan and optimize both their workouts and post-workout meals in a smooth manner. While various fitness apps and meal planners exist, they often function independently rather than in an integrated manner. Users must manually piece together fitness routines and meal plans, leading to inefficiencies, inconsistencies, and suboptimal results.

Causes and Contributing Factors

- The complexity of fitness and nutrition science makes it difficult for individuals to create well-balanced plans without expert knowledge.
- Existing fitness and nutrition platforms often supply to general audiences rather than offering personalized solutions.
- Users lack a simple and effective way to structure their training programs along with the corresponding nutritional needs.
- Time constraints and limited resources discourage individuals from conducting extensive research on fitness and nutrition.

Lacking the Function of Body Mass Index Calculator

The lack of a Body Mass Index, further referenced as BMI, calculator is the main problem with most fitness self-management systems nowadays. The most popular and basic approach for measuring someone's health is BMI. It provides an estimate of body fat and a reliable indicator of one's vulnerability to diseases that may develop as body fat increases. Diseases such as high blood pressure, diabetes, heart disease as well as other symptoms will be caused by a high BMI. For instance, there are a few fitness self-management applications in the market such as Google Fit: Health and Activity Tracking, Calories Counter – MyFitnessPal, Lifesum – Diet Plan, Macro Calculator and Food Diary, Da – fit , FitBit, Noom Weight Loss Coach, and Pam App, all of them are lacking the feature of calculating the BMI. Lacking this capability makes it hard to beginners in their fitness journey. The system would be encouraged to include a BMI [2] calculator function therefore the novices might refer to the BMI rate as their health goal.

Less of the Daily Activity Report Generator

The fitness-app creates a daily activity report that provides a summary for the user to then review. This report includes daily details, such as: weight, BMI, water intake, exercise, calories burned, and so forth. After being successfully generated, users can view their daily activity report to later then use as base for improving their future training. Furthermore, users can easily refer to their previous progress as a reference for upcoming fitness plans.

High Costs of Fitness Plans and Meal Programs

Many individuals face the challenge of some costs associated with fitness plans and meal programs. Although there are many fitness apps and meal planning services available, most of them come with subscription fees. These costs can be a barrier for those who want to maintain a consistent fitness routine or healthy eating habits, especially on a low budget.

A Domain-Specific Language (DSL) that unifies fitness and nutrition planning is essential to automate personalized workout routines and meal plans, reduce manual effort, and improve health outcomes.

1.2 Who is Affected?

Everyday people who like to stay active, sports players, fitness enthusiasts, athletes and those try be healthier are affected by this problem. Personal trainers, nutritionists, and fitness coaches also experience sometimes difficulties in providing customized plans efficiently. Lets not forget, people with special food needs, health problems, or or unique fitness goals can't easily find the right advice that will work for them

Needs of the Target Audience

To address this problem effectively, the target audience requires:

- An easy tool to create structured and personalized workout programs.
- Automated recommendations for post-workout meals based on training intensity, goals, and dietary preferences.

- A way to modify and adjust plans based on progress and changing needs.
- A DSL (Domain-Specific Language) that simplifies the planning process while allowing customization.

1.3 Where and When the Problem Arises

People face this problem when exercising at home, at the gym, or anywhere else, we do not judge. It gets even more depressing when with inconsistent progress comes the lack of motivation, which then leads to inadequate planning, unachievable goals and eventually burnout.

Why Addressing This Problem is Important

An effective solution will help individuals see that there is hope for achieving a fitness goal, thus, maintaining correct nutrition will get easier, and that will gradually enhance their overall well-being. By integrating fitness and nutrition planning into a single place, there is a lower probability for users to get tired, hence, it will save time, reduce confusion, and make informed decisions about their health.

How Can We Fix This?

The development of a DSL (Domain-Specific Language) specifically designed for fitness and nutrition planning can address these issues. The DSL will enable users to:

- Define workout routines with precise parameters such as intensity, duration, and type of exercises.
- Generate meal plans based on workout data, dietary preferences, and nutritional requirements.
- Adjust plans dynamically based on progress tracking and user feedback.
- Automate recommendations while allowing manual customization for advanced users.

Measuring the Impact

The effectiveness of the solution can be measured by:

- User approval and interest levels.
- Improvements in fitness and nutrition among users.
- Reduction in time spent manually planning workouts and meals.
- Feedback from fitness professionals and nutritionists on the tool's ease of use and effectiveness.

Scope and Timeline

This problem impacts many people wanting better health and fitness. The development and implementation of the DSL will require collaboration between fitness experts, nutritionists, and software developers. The initial phase will be based on research, prototyping, and testing, followed by making improvements, adjustments based on user feedback.

By addressing this problem through our DSL, people will have access to a simple, integrated tool that improves their fitness journey while ensuring proper nutrition, eventually leading to better health outcomes.

2 Literature Review

The next whole chapter will be about how we discover from different articles how many people are affected, how they are affected, what is the real situation, what are the trends, how to measure calories and other important meal metrics, and of course, how to plan fitness and create movement plans overall. That will help us to analyze the domain more effectively and perform the domain analysis from the next chapter.

How big is the problem we are analysing from the point of view of statistics?

The issue is fairly notable, because we observe that a lot of adult people, engaged in studies or work, have possibility to try the most basic fitness recommendation. For example, in the USA, almost 46% of adults aged 18 and over do not meet the guidelines for any type of physical activity [3]. The cause for such a high percent of poorly engaged into fitness people might be due to lack of motivation (41%), work schedule (22%), lack of energy (45%) and other difficulties [4]. The people who try using fitness apps usually look for diet nutrition planning (50%), training courses and plans (42.17%), or physical assessment (26.96%) as stated in the table below (Table 3.1) [5]. They might be discouraged by the result, since not all apps include meal prepping functions or any meal related functions.

Table 2.0.1 - Interest of users in the meal prep/fitness apps, based on info from source [5]

Options	**Proportion of agreed people**
Physical assessment and physical fitness monitoring	26.96%
Health and exercise data recording	42.17%
Exercise guidance and help	43.48%
Training courses and plans	42.17%
Selling sports equipment and other peripherals	27.83%
Diet nutrition planning	50%

Moreover, people tend to spend a lot of time by choosing the meal for the day/part of the day, exactly 48 minutes per day. This leads people to make impulsive food choices, not cooking/eating at all, taking a delivery food or other choices that might harm their health or fitness goal [6]. People who plan their meals can provide themselves a more healthy and balanced nutrition, and worry less about what to eat at the beginning of the meal itself [7].

How many people are influenced in words of statistics?

A significant number of people rely on digital solutions for fitness and meal planning. In the United States, 24% of adults aged 18–64 regularly use their smartphones as personal trainers. This suggests a growing demand for automated fitness guidance and meal planning tools [8]. Furthermore, using the tools Google Trends provides, searching such words as meal prep or meal plan has been increasing significantly in last years (Figure 3.1).

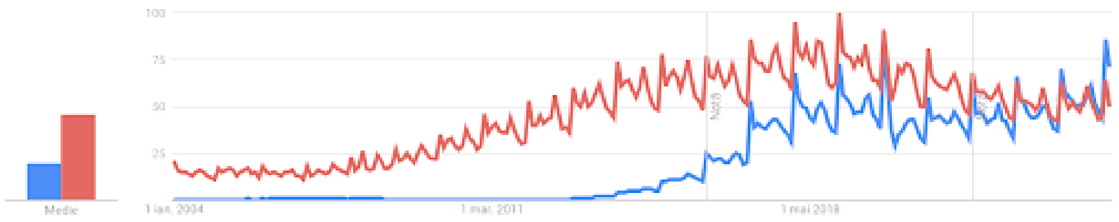


Figure 2.0.1 - Search trends on words meal prep and meal plan, based on the data offered by Google Trends on 19 of February, 2025 [9]

In the United States, 9 in 10 Americans have planned their meal at least once in their life, but less than 44% say that they meal prep regularly [10]. Millions of people around the world already use or plan to use digital tools for fitness and meal planning, so a well-designed DSL could make meal prep and workouts even easier and more accessible for them.

What are the main issues fitness and nutrition enthusiasts/pro's face?

Considering the statistics from before, it's important now to state what are the main problems present in the domain and why our solution can try to solve some explained problems. Firstly, let's take a closer look at the various barriers that fitness enthusiasts often face when it comes to maintaining a consistent and effective workout routine:

- Routine: People get bored with their daily exercise routines, which can make them give up or not be consistent.
- Reaching dead point: Also people may hit a dead point where they no longer notice progress, which can be disappointing.
- Lack of joy: Some people follow workout routines they don't enjoy because they think it's the ”right” thing to do.
- Personalized approach: Every client is different, and a one-size-fits-all system doesn't work [11].

Secondly, it is important to take in account the struggles of professional and amateur nutritionists when it comes to basic meal planning and motivation:

- Develop long-term healthy habits: It's challenging for people to maintain healthy eating habits over time. Motivation tends to fade, leading them to turn back to their old unhealthy habits.
- Food preparation: Many people have busy schedules and struggle to find time for meal preparation.
- Understanding nutrition in correlation to chronic diseases: While nutritionists can't provide medical nutrition therapy, they need to understand how certain foods affect chronic conditions [12].

In conclusion, both fitness enthusiasts and nutritionists face big challenges that can block progress. Our solution proposes to handle these issues by offering a more personalized approach to fitness and meal planning.

2.1 Nutrition Management

Among such aspects as timing and training the most important fact is also what and how we eat according to our activities and needs. And here the most essential part is what kind of activities people have. Since the majority of activities are about different type of races (running, marathons) and gym exercises (including gear and bodyweight) we'll consider this one and get more about the amount of time and training users get[13]. For now we'll analyze some important moments to consider for our DSL in the meal management part.

Basal Metabolic Rate (BMR)

There are different formulas that calculate exact amount of calories for every person, and calories must be the start point in anyone's journey through their nutrition and sport life[14].

- **For men:**

$$\text{BMR} = 13.397W + 4.799H - 5.677A + 88.362$$

- **For women:**

$$\text{BMR} = 9.247W + 3.098H - 4.330A + 447.593$$

where W is weight (kg), H is height (cm), and A is age (years).

BMR represents the minimum number of calories required to sustain life without any physical activity. This number is then multiplied by an activity factor (ranging from 1.1 to 1.9) based on the individual's exercise level.[15]

2.2 Macronutrient Breakdown

Now we will break down what are the macronuntrients vital for living and muscular development (and not only):

Table 2.2.1 - Nutrient recommended intake and sources

Nutrient	**Recommended Intake**	**Sources**
Carbohydrates	45-65% of daily intake, 3-8 g/kg/day	Complex: Whole grains, potatoes, rice, oatmeal Simple: Fruits, vegetables
Fats	20-35% of daily intake, minimum 1 g/kg/day	Unsaturated: Olive oil, nuts Limit saturated and trans fats
Proteins	10-25% of daily intake, 1-2 g/kg/day	Meat (red and white), fish, dairy, eggs

A balanced diet should also consider micronutrients such as calcium, iron, and the whole range of vitamins, especially A and C. Junk food should be minimized due to its low micronutrient value.

2.3 Nutrient Timing and Hydration

Not only food composition is essencial for the living, but also the timing when we eat and hydration during workout.

Pre-Workout Nutrition: A pre-workout meal should be high in complex carbohydrates and low in proteins and sugars to sustain energy levels. Recommended timings:

- Whole meal: 3-4 hours before exercise.
- Light snack: 1-2 hours before exercise.
- Hydration: 500 mL of water 2-4 hours before exercise.

During Workout Nutrition: Hydration is essential as the body loses fluids through sweat. Suitable options for hidratation include:

- Water (best option).
- Sports drinks (without caffeine) and juices for additional carbohydrate intake.
- For endurance events (4+ hours): 30-90 g of carbohydrates via sports gels or high-carb drinks.

Post-Workout Nutrition: A post-workout meal should replenish energy (carbohydrates) and support muscle recovery (protein). The optimal intake window is 1-2 hours after training.

Caloric intake, macronutrient distribution, and meal timing should be adjusted based on individual goals, preferences, and training regimens. Experimentation with drastic dietary changes is discouraged if current habits are already yielding positive results.

2.4 Fitness Management

Along with nutrition, fitness is also an important aspect that we want to consider in our DSL. So, we analysed the following information about what are one of the best approaches to plan the fitness.

Fitness goals and their impact

Setting the goal is a big and important element in fitness and any kind of physical activity, as well as in meal planning. It is a teachnique that helps in mantaining health and creating some robust behaviours. Researchers state that goals helps to rise the fiber consumption, as well as better exercise dedication, reduced sodium uptake, and of course fewer dropouts in physical activity. However, in order to have higher effect, the goals setted must be specific and challenging [17]. Personal trainers recommend setting goals that are created using the SMART method:

- S - Specific goal
- M - Measurable way to track progress
- A - Attainable and realistic time set
- R - Relevant motivation to reach the goal
- T - Timely and real deadline for the goal

Examples of good goals might be achieving a strict number of steps a month, work out a specific number of days in a month (taking into considerations that the body and muscles must also relax at least 48 hours), strech for 15 minutes after each workout and so on [17]. Some realistic goals and rates for weight loss or muscle gain are the following:

- Weight loss: 0.5-1% of body weight per week. This rate must represent the calorie-per-day deficit. This is good for a general goal like loosing weight.
- Muscle gain: 0.25-0.5% of body weight per weight. Paired with correct fitness plan and nutritious meal, this number will represent the calorie-per-day surplus, which will lead to muscle growth [18].

Fitness planning

Many individuals worry that they don't know how correctly to choose their fitness plan, since there are a lot of exercises,and each of them are targeted to specific muscules. Basic rules state that for a healthy life, an adult should get at least 150 minutes of moderate aerobic activity per week or 75 minutes of more energetic aerobic activity per week. Aerobic activity are exercises made specifically to increase and improve heart rate, mantain a healthy weight and sustain musculature. If there is a specific need for an increase, the aim is in general 10% for each week for a safe progress. Strength training, and more specifically, major muscle groups, require 2 times a week of trainings. As a good median is considered 12 to 15 repetitions [19]. Usually, personal trainers recomend to do strength workout based on following 4 types of compound exercises (that involve more types and groups of muscles in the same time) [nerdfitness site]:

- Quads (front of the legs): squats, lunges, one-legged squats, box jumps
- Butt and hamstrings: deadlifts, hip raises, straight leg deadlifts, good mornings, step-ups
- Push (chest, shoulders and even triceps) - overhead press, push-ups, dips, bench press, dumbbell press
- Pull (back, biceps, forearms) - chin-ups, pull-ups, bent-over rows, bodyweight rows

If mobility is limited or is not developed fully, there are easier modifications of the exercises that involve chairs, doors, mats, and other small improvements, that moght help such people like seniors or people recovering from injuiries [20].

Intensity metrics and how to choose fitness exercises based on them

Another important moment that we should consider before choosing the exercices for physical activity - their intensity and the metrics we should consider, as well as their complexity.

In order to understand what is the recommended intensity of the exercises, a person can determine by using their age and heart rate (%HRR - Percentage of Heart Rate Reserve): Firslty, to multiply the current age by 0.7 and substract from the total 208, this will be the xmax heart rate. After that, the person can find the recommended heart rate for their exercise:

- Moderate intensity: 50% to 70% of the maximum heart rate.
- High intensity: 70% to 85% of the maximum heart rate.

If the person is not fit or is a beginner - the intensity should be oriented on the lower end of the spectrum. If the person is in good shape, they can focus on a higher end of the spectrum [19].

Another way of measuring is using AIM's, Acceleration-based Intensity Metrics. Common AIMs include:

- Player Load per minute (PL/min) – external training load using triaxial accelerometers.
- Mean Amplitude Deviation (MAD) – dynamic acceleration by analyzing variations in

movement over short time intervals (e.g., 5 seconds).

These are much more complex measurements and might not be the best choice for enthusiasts and are more used by advanced personal trainers or professional sportsmen [21].

2.5 Review of the Existing Systems/Applications

Nowadays, fitness applications are extremely popular. Recent studies show that 42% of smartphone or tablet owners use one or more fitness apps. As accessible technology spreads and becomes more widely available, more customers are linking their smart wearables to applications in order to track their fitness and health. A recent burst of new goods promising to track steps or count calories in order to help customers get fitter, stronger, and healthier is seen. Several of these apps, meanwhile, struggle to hold users' interest over time. Thus, a literature review will be conducted by reviewing some similar systems.

Google Fit: Health and Activity Tracking

Brief overview on app functionality

Google Fit: Health and Activity Tracking [22] is a physical activity tracking platform developed by Google. A single set of APIs is used to combine data from various apps and devices. In order to give users an all-encompassing view of their fitness, Google Fit utilises sensors in their activity tracker/mobile device to record physical fitness activities (walking, cycling, etc.)

Strengths of the app

Google Fit: Health and Activity Tracking supports the capability of personalization workout plans. Instead of adopting the fixed training plan that the system has produced, the user can arrange their workout activities according to their ability. Additionally, Google Fit: Health and Activity Tracking offers activity reports including data on daily steps, time spent walking, a comparison chart for the days, and more. In furthermore, it offers the user the ability to create goals and keep track of their progress over time. This system places a strong emphasis on intercommunal communication since it allows users to share their daily progress with others. Last but not least, creating an account is essential in order to guarantee the user's privileges and data security

Limitations and problems

According to Google Fit: Health and Activity Tracking's app functionality, tracking BMI is not a feature that is available in body measurements. Besides that, the menstruation calendar and notification generating functions are excluded too. It is also confusing to a new user of the system because it lacks a search window.

Recommendations

It is recommended to include the BMI result in the activity report, develop a female-friendly space that has a menstrual calendar function and sends notifications prior to the estimated menstruation day, add a search panel inside the page in order to deal with users who are unfamiliar with how the system works.

Figure 2.5.1 - GoogleFit App design

Calorie Counter – MyFitnessPal

Brief overview on app functionality

Calorie Counter - MyFitnessPal is a self-management tool for fitness with a focus on weight loss [23]. The software uses gamification parts to motivate users to keep up with their dietary and fitness aims. To keep track of their intake, users may manually look for nutrients in the app's vast pre-existing database or scan the barcodes of various food products. MyFitnessPal's calorie counter gives access to 14 million different foods and has metrics for tracking calories and exercise. Users can combine their fitness data onto a single platform by linking their Calorie Counter - MyFitnessPal account with others like FitBit, Samsung Health, and Apple Watch.

Strengths of the app

The functionality of goal setting is offered by the calorie counter in MyFitnessPal. This application's goal setting is divided into a few categories, including objectives for weight loss goal, dietary improvement's goal, physical fitness goal, and more. Moreover, it allows users to share their training progress with other people. In order to protect the user's rights and the security of the data, it is also required to register for an account prior to logging into the system.

Limitation and problems

Calorie Counter - MyFitnessPal does not offer customized features, resulting in the in-ability of the user to create their fitness plan. Likewise, it has the drawback of not supporting the generation of activity reports. The user was unable to access their daily workout summary. The lack of a function that displays BMI precludes Calorie Counter – MyFitnessPal users from tracking their basic body fat measurements. Additionally, the system does not provide menstrual cycle notifications, which are unfamiliar to female users. A new user of the system finds it unfamiliar because a search window is missing

Recommendations

It is recommended that the Calorie Counter - MyFitnessPal implement the missing functionality to address the before mentioned constraint and satisfy user desire, add a portal that allows users to customize their fitness schedule, a session of activity reporting, BMI calculator via the report of activity summary, a menstrual reminder, a search panel at the top of the navigation bar to assist the inexperienced user in using the application.

Figure 2.5.2 - MyFitnessPal App Design

Lifesum - Diet Plan, Macro Calculator & Food Diary

Brief overview on app functionality

An app for digital self-care called Lifesum [24] enables users to eat healthier in order to achieve their weight and health objectives. Find a healthy diet that fits the user's lifestyle and eating habits. Use one of the many active diets, such as Clean Eating and High Protein, to take control of the user's everyday routines. According to Lifesum, users should receive four pre-planned recipes each day and stick to a 7–21-day meal plan. There are a variety of meal plans available, including Keto Burn and Vegan for a Week, depending on the user's health objectives

Strengths of the app

In order to create a customised plan depending on the user's needs, the Lifesum app asks users for details like their height, weight, age, and specific goals when they first join up. Additionally, it contains a weekly health test that provides information on the user's habits as well as possible areas for improvement. Besides that, it has a goal-setting feature with reminders. It indicates other users how the individual is progressing with their training.

Limitations and problems

The lack of customised options in Lifesum unable to let users developing their own exercise plans. The production of activity reports is not supported, which is another downside. The user is unable to see their summary of their daily workouts. Lifesum cannot track their fundamental body fat measurement due to the lack of function that displays BMI.

Recommendations

It is recommended to incorporate the lacking feature, such as a portal where users can choose their own workout routine, review on daily progress in an activity reporting session that is designed for them. Besides, adding the BMI calculator in the activity summary data, a menstrual reminder, a search panel at the top of the navigation bar to help a novice user navigate the application would be great additions to the app.

Figure 2.5.3 - Lifesum App Design

Da-fit

Brief overview on app functionality

A health management tool called Da-fit [25] was introduced by Mo Young Limited firm and is used to track physical activity. It features precise motion recording, sleeping information, and exercise analysis. It has the capacity to store all the information obtained by a smartwatch or activity tracker. The user of this application can quickly browse through all their activity status thanks to the straightforward, well-organized layout.

Strengths of the app

Da-fit allows users to customise their fitness schedule to suit their preferences. Secondly, it has the benefit of creating activity reports every day so that users can assess their progress. Furthermore, the user has to enter their goal in order for the system to suggest a training regimen for them. Additionally, the system places a strong emphasis on community communication as it gives users the ability to share progress.

Limitations and problems

Due to the absence of a function that displays BMI, Da-fit is unable to track their basic body fat measurement. Notifications for menstrual cycles are also lacking from the system. The system seems strange to a new user because there isn't a search box. Additionally, the system doesn't include the feature that requires new users to create an account.

Recommendations

Lack of a BMI calculator can be remedied by incorporating this feature through Da-fit's activity report. Furthermore, a menstrual reminder is developed to advise female users to scale back their exercise. Establishing a search panel at the top of the menu to guide a new user through the programme.

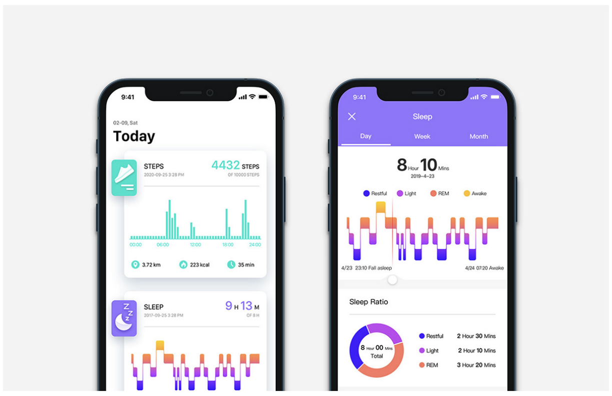


Figure 2.5.4 - Da-fit App Design

Fitbit

Brief overview on app functionality

The fitness app Fitbit [26] focuses on the areas of diet and weight control. It has technologies for daily readiness scores, health indicators including SpO2 and AFib assessments, stress management, continuous heart rate monitoring, sleep tracking, voice assistance, and more. It can connect to the smartwatch and retrieve information, which it can then record. It has a high rating in the Play Store, receiving 4.2 stars out of 5. Additionally, it is the first top-grossing health and fitness app available on Google Play

Strengths of the app

Fitbit allows users to customise their fitness schedules to suit their preferences. Additionally, it has the benefit of creating activity reports every day so that users can assess their progress. Beyond that, the user must specify their goal in order for the system to suggest a training regimen for them. Additionally, it encourages individuals to update the community on their exercise progress [27]. Apart from that, it has a unique feature that female users are familiar with menstruation notification. Last but not least, creating an account is necessary for new users before they can log in to the system to ensure the security of the users' data.

Limitations and problems

Fitbit is unable to monitor their basic body fat measurement because it lacks a feature that displays BMI. Since there isn't a search box, the application feels lost when using the application to a new user.

Recommendations

The activity data on the Fitbit can be used to add a BMI calculator, which will solve the problem that aforementioned. Adding a search panel to the menu at the top will help a new user navigate the application.

Figure 2.5.5 - Fitbit How it looks

Table 2.5.1 - Comparison of features, strengths, limitations and recommendations of apps

App Name	**Strengths**	**Limitations**	**Recommendations**
Google Fit	• Personalized workout plans • Activity reports (steps, walking time, etc.) • Goal tracking & progress sharing	• No BMI tracking • No menstruation calendar/notifications • No search window	• Add BMI results in reports • Include menstrual calendar & notifications • Add search panel
MyFitnessPal	• Goal setting (weight loss, diet, fitness) • Progress sharing • Account security	• No custom fitness plans • No activity reports • No BMI tracking • No menstrual reminders • No search window	• Allow custom fitness plans • Add BMI calculator • Include menstrual reminders
Lifesum	• Personalized diet plans based on user data • Weekly health test • Goal setting & progress sharing • Account security	• No custom fitness plans • No activity reports • No BMI tracking • No search window	• Enable custom fitness plans • Add activity reports & BMI calculator • Add search panel
Da-fit	• Customizable fitness plans • Daily activity reports • Goal-based training suggestions • Progress sharing	• No BMI tracking • No menstrual reminders • No account registration	• Add BMI calculator in reports • Add search panel • Implement account registration
Fitbit	• Flexible fitness plans • Daily activity reports • Menstrual notifications	• No BMI tracking • No search window	• Add BMI calculator in reports • Include search panel

3 Domain Analysis

Analyzing the literature that we found about fitness and meal prepping, we are ready to make analysis on the domain, in which we will discuss the domain scope and context, existing systems, stakeholders, and of course, possible technical challenges.

3.1 Domain Scope and Context

This project focuses on the intersection of fitness technology, nutrition, and healthy habits. The main goal is to create tools that help automate and personalize health management. A big challenge is that workout plans and diet plans are often treated as separate things, even though they are closely connected.

This DSL aims to offer a more easy way to manage both fitness and nutrition together, because they are usually treated separately, even though these themes are closely linked. [28].

Here are the key features that we plan to focus on:

- **Fitness planning**
  - Exercise routines design

    Designing workouts tailored to goals (e.g., strength, endurance, weight loss).
  - Intensity tracking (heart rate, reps, sets)

    Monitoring metrics like heart rate, reps, sets, and recovery time.
  - Progress analysis and dynamic adjustments

    Measuring improvements (e.g., muscle gain, stamina) and adjusting plans dynamically.
- **Nutritional management**
  - Caloric and macronutrient balancing

    Aligning intake with energy expenditure (e.g., protein for muscle repair).
  - Dietary restrictions handling

    Accommodating allergies, ethical choices (veganism), or medical conditions (diabetes).
  - Meal timing optimization
- **Behavioral Integration**
  - Habit formation mechanisms
  - User motivation strategies

3.2 System Analysis

From the analysis of existing literature, it is evident that while fitness apps have advanced in tracking physical activity and caloric intake, a significant gap remains in integrating these with personalized and culturally relevant diet plans. Many apps either focus on Western diets or generic recommendations, which limits their effectiveness for users from different cultural backgrounds, such as those who prefer Indian, Mediterranean or other diets. Additionally, the lack of AI-driven features like dynamic adaptation of workout plans and real-time health monitoring creates a gap for users who need continuous updates to their routines and nutrition plans based on progress. Integration of a cultural diet component, real-time AI feedback, and its universal approach to fitness and nutrition offers a more robust solution compared to existing apps. This allows for a unique experience, encouraging better adherence to fitness and health regimes. This system analysis illustrates how it can fill existing gaps in the fitness and nutrition market by offering a more comprehensive and adaptive approach to health and wellness [29].

Existing Systems

Thousands of applications are everywhere, as technologies show — they help users in every possible way to take the whole of their good health/lifestyle/fitting. Some of the most popular ones include MyFitnessPal, Lose It, and Fitbit. They keep track of the calories you expend, your workouts, and give health tips to your users. Most are focused either on diet or exercising and fail to integrate their features well. While they provide meal and workout ideas, you risk using numerous applications to cover all aspects of your fitness and diet. Most of the contemporary systems possess a certain dietary regime and rigid fitness schedule, which often times clashes with personal tastes or cultural food habits. They lack a supportive environment or persona coaching to keep their users interested and engaged [30]. AI support functions have been formerly viewed as instant, which now represent a very important supplement to the various health journeys [31].

Proposed System

Our proposed Domain-Specific Language (DSL) is designed to allow users in managing their well-being, nutrition, and fitness routines through **flexibility, personalization, and user-centric design.** By prioritizing individual preferences, lifestyle compatibility, and practical constraints, the system avoids rigid health mandates and instead focuses on adaptable planning. Below are the core components of the DSL:

- *Health-Centric Flexibility*

The DSL highlights user autonomy by merging personal goals, preferences, and real-world constraints into the plans. Health guidelines are treated as optional recommendations rather than strict rules, reducing friction and fostering long-term adherence.

Key Features:

- Plans adapt to user-provided data (e.g., dietary restrictions, ingredient availability, energy levels).
- Generation of recipes using ingredients already available in the user's pantry.
- Avoidance of rigid calorie counting or nutrient limits unless explicitly requested.
- Supports mental and emotional health by aligning plans with user comfort (e.g., rest days, favorite foods).

• *Adaptive Scheduling*

Users can design **personalized schedules** for meals, workouts, and recovery that align with their daily routines. The system serves to both structured and flexible lifestyles.

Key Features:

1. **Predefined templates:** Common time blocks (e.g., morning workouts, evening meals) based on research into popular routines.
2. **Dynamic adjustments:** Shift activities seamlessly if conflicts arise (e.g., reschedule a workout due to a busy workday).
3. Integration with calendars and real-time notifications to maintain consistency [32].

• *Intelligent Food Planning*

The DSL simplifies meal planning by balancing dietary tastes, nutritional goals, and ingredient accessibility.

Key Features:

1. Generates recipes tailored to user preferences (e.g., vegan, gluten-free) and ingredient availability.
2. Suggests grocery lists and meal-prep strategies to minimize waste.
3. Optional nutritional insights (e.g., protein intake, balanced macros) for users seeking guided support.
4. Supports meal variety to prevent monotony (e.g., rotating cuisines or seasonal ingredients).

• *Goal-Oriented Sport Planning*

Workout plans are dynamically created based on users' fitness goals, available time, equipment access, and physical capabilities.

– Customizable workouts (e.g., strength training, cardio, mobility) with adjustable intensity and duration.
– Adapts to equipment constraints (e.g., home workouts without gym machines).
– Tracks progress and iteratively updates plans to align with evolving goals (e.g., weight loss, muscle gain).
– Includes recovery recommendations (e.g., stretching, hydration) to prevent burnout.

System Philosophy

The main idea behind this DSL is to give users control over their own meal and training plans, making them easy to follow, enjoyable and fun, and suited to their personal lifestyle. Instead of forcing strict health rules, it focuses on adaptability and personalization, so more people can use it without feeling overwhelmed. Whether someone wants to try new recipes, keep up with a healthy routine, or create some healthy habits in a busy schedule, this DSL acts as a helpful and supportive tool for long-term well-being [33].

3.3 Stakeholder Analysis

Another foundation for our DSL is understanding who and what are our stakeholders. Different people have different expectations, as well as needs. Here we analysed 4 different stakeholders and target groups that we think might benefit from our DSL solution.

Table 3.3.1 - Stakeholders, their roles, and key needs in the DSL

Stakeholder	**Role**	**Needs/Pain Points**
End users	Fitness enthusiasts, athletes	• Time-efficient planning • Personalized routines • Unified tracking interface
Fitness trainers	Personal trainers	• Client management tools • Evidence-based recommendations
Nutrition adepts	Nutritionist, nutrition amateurs, meal prepping enthusiasts	• Meal planning tools • Based recommendations
Researchers	Sport science researchers, data analysts	• Tracking system • Based recommendations

As stated by this table, there are in our opinion 4 different shareholders or interested groups, that might benefit. End users could use the DSL for their own purposes, while fitness trainers, nutrition adepts and researchers could use the more advanced tolls that could give them more information or recommendations.

3.4 Technical Challenges

Creating a DSL for meal planning and training management comes not only with benefits, but also with some technical challenges, which include:

- **Data standardization**
  - Inconsistent calorie measurement
  - Adding various ingredients and possible meals
- **Algorithm accuracy**
  - Limitations of generic formulas
  - Need for constant oversight
  - Creation of reliable algorithm
- **Learning curve for DSL and adaptability for user**
  - Balancing simplicity and flexibility of the DSL
  - Ensuring accessibility for non-technical users
  - Providing some clear documentation for enthusiasts
- **Integration with external systems**
  - Compatibility with different fitness apps and/or devices
  - Possible synchronisation with existent nutrition databases
- **Security concerns, data privacy**
  - Protecting sensitive data (like health/diet information, age, habits and other information)
  - Authentication mechanisms for safe authorization
  - Compliance with data protection regulation in our country

In conclusion, while building a DSL for meal planning and training management has many advantages for the users, there are also some technical challenges to take into the account. All these issues might slow down the development of the solution or create some barriers for the users that are interested in using our app.

4 Comparative analysis with other projects

Besides the concurrent apps that we discused in second chapter, we found some related DSL's and want to analyse them compaed to our solution.

4.1 Analysis of Cooking-Recipe-DSL

The Cooking-Recipe-DSL, accessible on GitHub beneath the username "sharknoon" 1, is portrayed as a basic domain-specific language for cooking recipies, planned to show the functions of a DSL in Scala 1. It parses a cooking recipe composed in a certain graphical plot and change over it into verbal yield 1.The language is written in German and includes such tokens as ("Lasagne"), length ("120 min"), number of servings ("2 Personen"), creator ("Simon Mennig"), categories ("Spanish, Pasta dish, Cooking"), trouble ("progressed"), fixings ("250 g meat pate", by the way), and cooking steps ("Make bolognese. Preheat stove."). After parsing, the data is changed into a more presentable setting, in usser browser.

We found other projects related to such DSLs. For example, the kolasu-kuki-languageserver project by Strumenta is a DSL for cooking recipes and shows also such features like syntax highlighting and error reporting. Similarly, the some project created by opscode includes modules like Chef::DSL::Recipe and Chef::DSL::Universal that are related to recipe and meal planning in the context of system computing.

These projects, along with the main Cooking-Recipe-DSL, focus on the culinary part, specifically on managing, presenting, and potentially automating recipe components. However, these projects do not include some sports planning related themes.

4.2 Analysis of wlang2

The "wlang2" project, found on GitHub and created by "enspirit", is a general-purpose DSL programmed on Ruby. It spins around code updating and paragraph control. The "wlang2" template engine is very flexible, allowing users to create their own plans and stamps and define how they behave. It also supports logic-free HTML generation, inspired by the Mustache template engine, which can create rich web pages. To improve performance, "wlang2" compiles templates to generate output quickly. It can also integrate with other web development tools and frameworks, like Tilt and Sinatra.

The GitHub shows that "wlang2" is useful for various content generation tasks far beyond simple HTML, such things as code generation or other types of content output where other engines might be more suitable. "wlang2" is mentioned in different contexts on GitHub, including its use in an ad tech project (”scala-openrtb” by ”Powerspace”) and as a topic in a computer science course on language analysis and laziness. These references highlight ”wlang2” as a general tool for developers to manage and enhance content.

Unlike the ”DSL for Wellness and Supply Management,” ”wlang2” does not have built-in features for health, nutrition, or broader health management. Using ”wlang2” in these areas would require significant custom development to add the specific logic and data structures needed for personalized health and wellness predictions.

4.3 Comparative Analysis of DSL’s

DSL for Fitness and Nutrition vs. Cooking-Recipe-DSL

The main difference between the ”DSL for Fitness and Nutrition Management” and the ”Cooking-Recipe-DSL” is their focus. The former is designed to manage both wellness and nutrition with a focus on personalized health and fitness, while the latter is mainly about managing and presenting cooking recipes. The ”DSL for Fitness and Nutrition Management” recognizes the important link between physical activity and dietary intake, offering a universal approach to health that the ”Cooking-Recipe-DSL” lacks.

The ”DSL for Fitness and Nutrition Management” includes many advanced health features that go beyond what the ”Cooking-Recipe-DSL” offers. Users can create personalized workout plans with precise parameters and benefit from dynamic adjustments based on their progress. It balances caloric and macronutrient intake according to activity levels and accommodates various dietary restrictions and medical conditions. Meal timing is optimized for better performance and recovery. Additionally, it includes features to encourage long-term habit formation and user motivation.

Notably, the ”DSL for Fitness and Nutrition Management” includes features like a BMI calculator, daily activity reports covering various health metrics, and a menstrual cycle calendar for female users. Flexible scheduling functions allow for simple integration of health plans into daily routines, and intelligent meal planning simplifies meal preparation with tailored recipes and grocery lists. In contrast, the ”Cooking-Recipe-DSL” is limited to parsing and displaying recipe information, lacking any features related to fitness planning, dietary analysis, listing ingredients, or integration with broader health management concepts.

DSL for Fitness and Nutrition vs. wlang2

The main difference between our DSL, ”DSL for Fitness and Food Management” and ”wlang2” is their purpose. ”DSL for Fitness and Food Management” is specifically designed

Table 4.3.1 - Feature comparison between fitness/nutrition and recipe DSLs

Feature	**Fitness/Nutrition DSL**	**Cooking-Recipe-DSL**
Core Purpose	Integrated Fitness & Nutrition Management	Recipe Management
Fitness Planning	Yes (Detailed)	None
Nutritional Management	Yes (Comprehensive)	Yes (Basic Recipe)
Personalization	High (Goals, Preferences, Restrictions)	Ingredient listing only
Integration	Fitness & Nutrition	Standalone Recipes
Advanced Health Features	Yes (BMI, Activity)	None
Behavioral Integration	Yes	None
Adaptive Scheduling	Yes	None
Intelligent Food Planning	Yes	None

for health and nutrition, while ”wlang2” is a general template engine for generating code and text.

”DSL for Fitness and Food Management” has built-in features for sport and nutrition, like planning workouts, meal plans based on dietary restrictions, and BMI calculations. This makes it very effective for managing health and food. On the other hand, ”wlang2” doesn’t have any built-in health or wellness features. It’s good for creating content based on templates, but it would need a lot of customization to handle wellness and food management. Adding features like personalized workouts or nutrition analysis in ”wlang2” would require building them from scratch, which is a lot of work.

In summary, ”DSL for Fitness and Food Management” is a better choice for managing health and nutrition because it’s specifically designed for that purpose, while ”wlang2” is more general and would need significant modifications to be used in this way.

Table 4.3.2 - Comparison between fitness/nutrition DSL and wlang2

Aspect	**Fitness/Nutrition DSL**	**wlang2**
Type of Language/Tool	Domain-Specific Language	General-purpose
Primary Domain	Fitness and Nutrition Management	Code/Text Generation
Built-in Health Concepts	Yes (Workouts, Meals, BMI, etc.)	No
Personalization for Health	Yes	No (Requires Customization)
Focus	User Health and Wellness	Text Output Generation
Ease of Use (for end-users)	Designed for end-users	Primarily for Developers

4.4 Summary

In conclusion, the ”DSL for Fitness and Nutrition Management” project is a big step forward in health tools. Its main strength is its integrated and personalized approach to managing wellness and nutrition, addressing the gaps in many existing solutions. Unlike the ”Cooking-Recipe-DSL,” which only focuses on recipes without considering wellness and individual health needs, the ”DSL for Wellness and Nutrition Management” offers a more complete solution for overall health management.

Similarly, while ”wlang2” is a powerful general-purpose template engine, it lacks specific features for wellness and nutrition management. It would need a lot of development to handle these areas effectively.

The ”DSL for Fitness and Nutrition Management” can integrate wellness and nutrition planning, provide personalized recommendations based on goals and restrictions, and include advanced health-tracking features. This makes it a better tool for people looking to take control of their health and fitness journeys.

5 Implementation

5.1 Grammar Definition and Lexical Analysis

The following section is dedicated to the analysis of the grammar, Lexer, Parser, as well as tokens and other important elements of the implemented DSL.

Lexical Elements (Tokens)

The lexical components shape the elemental building pieces of our domain-specific dialect. These tokens are recognized by the lexer amid the primary stage of compilation, changing the crude source code into a organized grouping of tokens that can be handled by the parser.

Our language contains a full set of tokens defined particularly for fitness and nutrition:

- **Keywords:** `using`, `create`, `Person`, `select`, `where`, `generate`, `output`, `await`, `for`, `while`, `include`, `list`, `add`, `foreach`, `time`, `amount`, `type`, `goal`, `ingredients`, `rep_time`, `rest_time`, `meal`, `added_meals`, `ID`, `Diet`, `Allergic`, `Exercises_weights`, `Weight`, `Height`, `Busy_Time`, `Training`, `recipe_list`, `recipe_link`, `parameters`, `schedule`, `update`, `Breakfast`, `Lunch`, `Snack`, `Dinner`, `Calories`, `Proteins`, `Carbs`, `Fats`, `total`, `result`.
- **Identifiers:** Variable names (e.g., `person`, `training`, `ingredient`, `meal`).
- **Literals:**
  - Numbers
  - Strings (e.g., `"nuts"`, `"Shakshuka with Toast"`).
  - Time values (`11:30`, `13:00`).
- **Operators:** `=`, `<`, `>`, `*`, `+`, `-`, `.`, `[]`, `{}`.
- **Punctuation:** `:`, `;`, `,`, `(`, `)`, `//`.

Syntactic Structures (BNF)

The Backus-Naur Form (BNF) defines the syntax of our language, showing how tokens can be combined to create valid statements and expressions. This formal grammar is essential for the parser to validate code structure and build an abstract syntax tree.

Our grammar defines a hierarchical structure starting from the Program level, which consists of multiple Statements. Each Statement can be one of several types (CreateStmt, GenerateStmt, etc.), each with its own specific grammar rules. The grammar supports nested structures through Block definitions and allows for complex expressions and conditions.

```
Program          = { Statement } ;
Statement        = CreateStmt | GenerateStmt | SelectStmt |
UsingStmt | OutputStmt | LoopStmt ;
CreateStmt       = "create" Identifier "{" { Attribute } "}" ;
Attribute        = Identifier ":" ( Literal | Array | Object ) ;
GenerateStmt     = "generate" Identifier "{" { Param } "}" ;
Param            = Identifier ":" ( Literal | Reference ) ;
SelectStmt       = "select" Identifier "where" Condition ;
Condition        = Identifier Operator Literal ;
Operator         = "=" | ">" | "<" ;
LoopStmt         = WhileStmt | ForStmt ;
WhileStmt        = "while" "(" Condition ")" Block ;
ForStmt          = "for" Identifier "in" Iterable Block ;
Iterable         = Identifier | Array ;
OutputStmt       = "output" Expression [ "in" "table(" Identifier ")" ]
   ;
UsingStmt        = "using" Identifier ;
Block            = "{" { Statement } "}" ;
Expression       = MathExpr | Reference | Literal ;
MathExpr         = Number Operator Number ;
Reference        = Identifier { "." Identifier } ;
```

Basic Semantics

The semantics of our language define the meaning and behavior of the programs. Understanding these semantics is crucial for both language implementation and effective use.

Key Constructs

Our language provides specialized constructs for fitness and nutrition planning:

- `create Person`: Initializes user profiles with various attributes like weight, height, dietary preferences, and allergies. The system validates numerical attributes to ensure they fall within reasonable ranges.
- `generate training`: Creates personalized workout plans by analyzing user data against the exercise database. It considers factors like available equipment, fitness goals, and physical limitations.
- `select...where`: Filters database records based on specific conditions, enabling targeted data retrieval for meal and exercise planning.

- Loops: `while` checks training time limits, `for` iterates ingredients.
- `output`: Supports various presentation formats, including tabular displays for workout schedules and meal plans.

Execution Rules

Our language follows a logical execution flow designed for fitness and nutrition planning programs:

1. First, relevant data sources are loaded using the `using` statement, establishing connections to food and exercise databases.
2. Next, user profiles are created with the `create` statement, capturing essential information like physical attributes, dietary restrictions, and fitness goals.
3. The system then filters relevant records using `select` statements, narrowing down potential meals and exercises based on user requirements.
4. Plan generation occurs through `generate` statements, which create comprehensive meal and workout schedules.
5. Finally, loop-based improvement processes optimize the generated plans, ensuring they meet all constraints and preferences.

Error Conditions

The language implementation must handle various error conditions to ensure robust operation:

- **Lexical errors** occur when invalid tokens are encountered, such as abnormal time values or unrecognized symbols.
- **Semantic errors** arise when code references missing features or entities, like trying to access allergies for a user profile that doesn't define them.
- **Type mismatches** happen when operations are performed on incompatible data types, such as trying to calculate with string values.
- **Constraint violations** represent domain-specific errors, such as including allergens in a meal plan or scheduling workouts during busy periods.

Grammar Complexity Evaluation

The complexity of our grammar directly impacts the implementation difficulty and the language's expressive power. This evaluation helps understand the computational resources required for parsing and the potential for language extensions.

Table 5.1.1 - Grammar Complexity Metrics

Metric	**Classification**
Number of Productions	25 rules
Recursion Depth	Medium (up to 3 levels)
Lexical Complexity	High (44 token types)
Syntactic Complexity	High (nested blocks, optional formats)
Ambiguity	Unambiguous

This Domain-Specific Language (DSL) is of a quite high level of complexity, as shown by its 25 production rules and medium recursion depth of up to three levels. It has high lexical complexity with 44 token types and syntactic complexity due to nested blocks and optional formats. Despite these complexities, our DSL is unambiguous, ensuring clear and precise interpretation of its constructs.

Use cases

Based on the information we already provided about the grammar and tokens, we can already analyse different base use cases and offer a breakdown of all the important data that can be deduced or generated using the code provided.

Meal Plan

```
using dbo.food
using person.ID = 1
foreach person.food.parameter {
  include parameter into generation
}

while (parameters != done)
{
  generate food.recipe schedule update
}

for (meals)
{
  output meal.recipe.link
}
```

In this example, we can see that there are used loops, like while and for, as well as constructs like foreach and using for database calls. Using this code snippet, it is expected to obtain the following results:

- **Breakfast:**
  - **Recipe:** Shakshuka with Toast
  - **Calories:** 600
  - **Time:** 8AM (30 min prep)
- **Lunch:**
  - **Recipe:** Sweet Potato with Beef Steak
  - **Calories:** 750
  - **Time:** 12PM (45 min prep + recommendation before 10AM)
- **Snack:**
  - **Recipe:** Cookies with Milk
  - **Calories:** 400
  - **Time:** 3PM (0 min prep)
- **Dinner:**
  - **Recipe:** Chicken Heavy Cream Pasta
  - **Calories:** 850
  - **Time:** 6:30PM (45 min prep)

Total Result

- **Meals:** 4
- **Calories:** 2600
- **Proteins:** 100
- **Carbs:** 3025
- **Fats:** 100

Creating Users Data

```
using dbo.food
create Person person{
  ID (1)
  BMI (BMI parameters)
  Calories (Calories parameters)
  Goal (Maintainance)
  Time { Busy (10AM-5PM, 11PM-6AM) }
  Weights {
    // here will be chosen weights for sport
    }
  Diet (Standard)
  Alergic (nuts.all)
}
```

This example shows us how to create users and append to them the relevant data. There are used such keywords as BMI, ID, Calories, Goal, Time, etc.

Training Plan

```
using dbo.sport
using dbo.food
using person.ID = 1
generate training (time avg, amount avg_not_specified, type
    w_gear_not_specified, goal fit, Person person)
output training as description table

while (time.max)
{
  output rep time
  output rest time
}
```

The above code snippet is responsible for creating a training plan. We can observe the appropriate training-themed keywords and loops that might help to generate the result. Here is the expected result:

- **Result:** 1 Training
- **Time:** 55-65 min (3 min rest)
- **Amount:** 4 exercises $\times$ 4 sets $\times$ 8-10 reps
- **Type:** Gym Geared Full-Body
- **Goal:** Keep Fit

The training plan is generated based on information from the table (Table 5.1.2):

Table 5.1.2 - Training Plan Exercises

Exercise	**Chest Press**	**Biceps Curls**	**Leg Press**	**Leg Raises**
Set 1	0.9 × person.w × 8 reps	0.9 × person.w × 8 reps	0.9 × person.w × 8 reps	0.9 × person.w × 8 reps
Set 2	1 × person.w × 8 reps	1 × person.w × 8 reps	1 × person.w × 8 reps	1 × person.w × 8 reps
Set 3	1.1 × person.w × 10 reps	1.1 × person.w × 10 reps	1.1 × person.w × 10 reps	1.1 × person.w × 10 reps
Set 4	1.1 × person.w × 8 reps	1.1 × person.w × 8 reps	1.1 × person.w × 8 reps	1.1 × person.w × 8 reps

Here are also another examples of use cases. The first code snippet shows how to trigger the function for finding the optimal time for exercises. This function takes into account various factors such as user preferences, available time slots, and exercise intensity to determine the best time for a workout. The second code example shows how to generate meals from an existent ingredient, thus the function provides a list of possible meals.

Code Examples for find optimal times

```
using person.ID = 123

generate optimalTimes for user {
    BusyTime({"08:00-12:00", "13:30-15:00"})
    Exercises = {
        Running { Sets = 3, Reps = 10 }
    }
}

output optimalTimes as table
```

Code example for find meals by ingredient

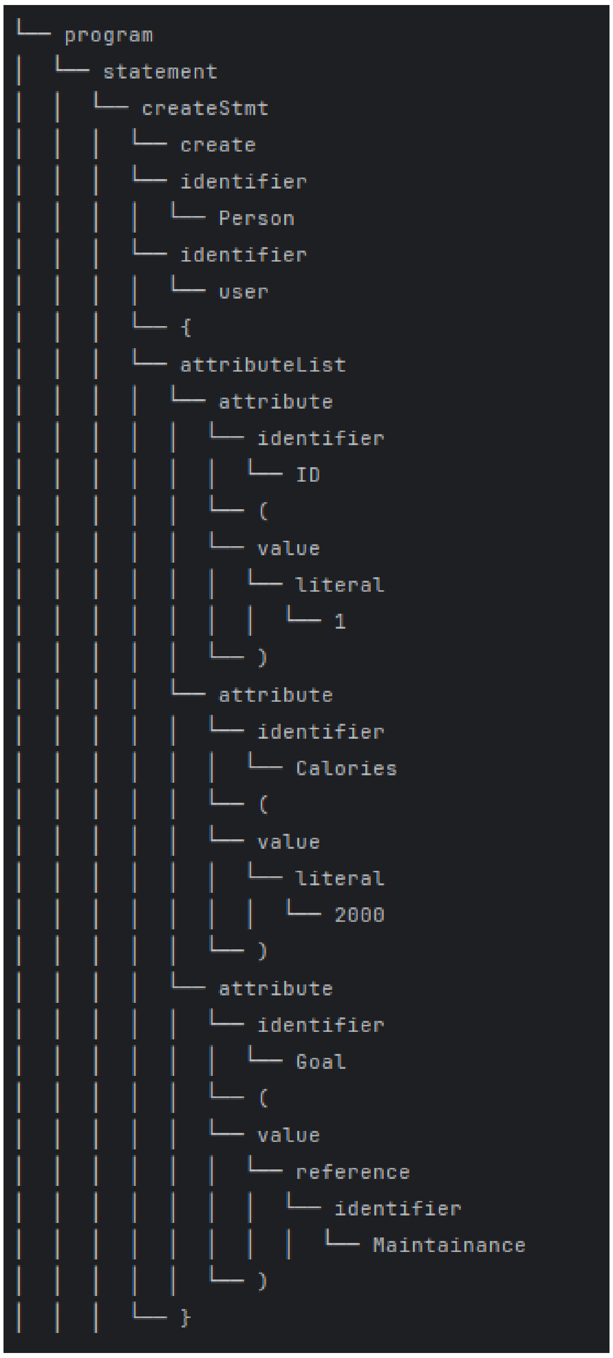
```
generate meals where ingredient = "chicken"

output meals as table
```

Example analysis

Let's see now how the Lexer analyses a concrete example from the FitnessNutrition grammar. Here is a simple code for creating a user:

```
create Person user {
  ID (1)
  Calories (2000)
  Goal (Maintainance)
}
```

The Lexer parses the introduced code and creates the following parse tree:



```
└── program
│   └── statement
│   │   └── createStmt
│   │   │   └── create
│   │   │   └── identifier
│   │   │   │   └── Person
│   │   │   └── identifier
│   │   │   │   └── user
│   │   │   └── {
│   │   │   └── attributeList
│   │   │   │   └── attribute
│   │   │   │   │   └── identifier
│   │   │   │   │   │   └── ID
│   │   │   │   │   └── (
│   │   │   │   │   └── value
│   │   │   │   │   │   └── literal
│   │   │   │   │   │   │   └── 1
│   │   │   │   │   └── )
│   │   │   │   └── attribute
│   │   │   │   │   └── identifier
│   │   │   │   │   │   └── Calories
│   │   │   │   │   └── (
│   │   │   │   │   └── value
│   │   │   │   │   │   └── literal
│   │   │   │   │   │   │   └── 2000
│   │   │   │   │   └── )
│   │   │   │   └── attribute
│   │   │   │   │   └── identifier
│   │   │   │   │   │   └── Goal
│   │   │   │   │   └── (
│   │   │   │   │   └── value
│   │   │   │   │   │   └── reference
│   │   │   │   │   │   │   └── identifier
│   │   │   │   │   │   │   │   └── Maintainance
│   │   │   │   │   └── )
│   │   │   └── }
```

Figure 5.1.1 - Terminal Parse Tree

In our specific example, the terminal tree shows the complete decomposition of the create Person statement, displaying each grammatical component as a nested hierarchy. While technically accurate, this representation can become challenging to read with more complex statements, especially as the nesting grows deeper.

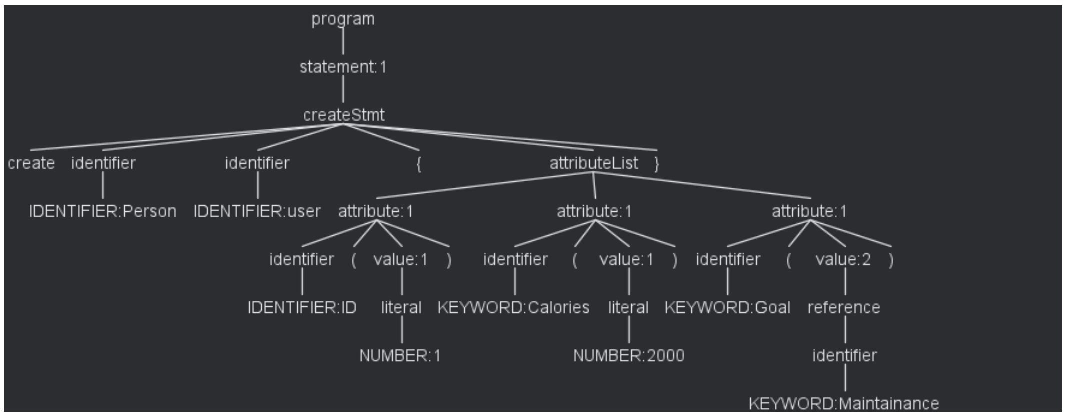


Figure 5.1.2 - Graphical Parse Tree

The graphical parse tree offers a more intuitive and visually comprehensible alternative. Generated through specialized ANTLR visualization tools or integrated development environment (IDE) plugins, this representation transforms the same parsing information into a more digestible format.

For our Fitness Nutrition DSL, these parse trees demonstrate how a simple create statement gets broken down into its constituent parts - showing how the language parser understands and validates each element of the code.

5.2 Implementation of Lexer

The FitnessNutritionLexer is an automatically generated lexical analyzer (tokenizer) created using ANTLR (ANother Tool for Language Recognition).

The implementation of the lexer for our Fitness and Nutrition DSL transforms the raw source code into a structured sequence of tokens.

```
public static final int
    T__0=1, T__1=2, T__2=3, T__3=4, T__4=5, T__5=6, T__6=7, T__7=8,
    T__8=9, T__9=10, T__10=11, T__11=12, T__12=13, T__13=14, T__14=15,
    T__15=16, T__16=17, T__17=18, T__18=19, T__19=20,
    T__20=21, T__21=22, T__22=23, T__23=24, T__24=25, T__25=26,
    T__26=27, KEYWORD=28, IDENTIFIER=29, STRING=30, NUMBER=31,
    TIME=32, BOOLEAN=33, COMMENT=34, WS=35;
```

This section defines the token types that our lexer recognizes. Each token is assigned a unique integer value that the parser uses to identify token types. The specialized tokens like TIME (32) are particularly important for our fitness and nutrition domain, allowing the representation of workout times and meal schedules.

```java
private static String[] makeLiteralNames() {
    return new String[] {
      null, "'create'", "'{'", "'}'", "'('", "')'", "'generate'",
            "','", "'select'", "'where'", "'!='", "'done'", "'='",
            "'>'", "'<'", "'while'", "'for'", "'in'", "'foreach'",
            "'include'", "'into'", "'output'", "'as'", "'table'",
            "'using'", "'.'", "'*'", "'-'"
    };
  }
```

The makeLiteralNames method maps literal strings to their corresponding token types. When the lexer encounters ”create” in the input, it recognizes it as token type 1 (T–0). This mapping enables the lexer to identify domain-specific keywords like ”generate” for workout plans and ”select” for filtering nutrition data. During lexical analysis, when the lexer processes a statement like:

```
create Person person {
   ID (1)
   BMI ( BMI parameters )
   Calories ( Calories parameters )
}
```

It identifies ”create” as T–0, ”Person” and ”person” as IDENTIFIERs, ”” as T–1, and so on. This tokenization is important for the parser to understand the structure of user profile creation.

For time values like ”11:30” or ”13:00” in busy time specifications, the lexer uses pattern matching to recognize them as TIME tokens. This allows the system to properly schedule workouts and meals around the user’s busy periods.

When processing nutrition-related code like:

```
Alergic ( nuts . all )
```

The lexer identifies ”Alergic” as an IDENTIFIER, ”(” as T–3, ”nuts” as an IDENTIFIER, ”.” as T–24, ”all” as an IDENTIFIER, and ”)” as T–4. This tokenization enables the system to understand dietary restrictions and allergies.

For loop constructs like:

```
while ( time . max ) {
   output rep time
   output rest time
}
```

The lexer identifies the output directive and its formatting options, enabling the system to present workout and meal plans in user-friendly formats.

In the case of errors, such as an invalid time format or unrecognized symbol, the lexer reports the error with context information, helping users correct their fitness and nutrition plans.

5.3 Implementation of Parser

This chapter presents an in depth look at the parsing logic defined in FitnessNutrition-Parser.java. The parsing logic is distributed across a collection of methods, each associated with a specific non-terminal in the grammar. These methods work together to analyze key-words, identifiers, values, blocks, expressions, and control flow statements. Before diving into the parser methods themselves, it is important to understand the underlying grammar from which the parser is generated.

ANTLR uses a grammar specification file—in this case, FitnessNutrition.g4—to define the language rules. This grammar file combines both lexical and syntactic rules and determines how valid DSL scripts are structured and recognized.

```
while ((((_la) & ~0x3f) == 0 && ((1L << _la) & ...) != 0)) {
    statement();
}
```

This loop defines the core structure of the program. The parser continuously reads and processes top-level statements by checking whether the next token belongs to a valid set. If so, it hands off processing to the statement() handler; otherwise, it exits the parsing flow.

```
switch (_input.LA(1)) {
    case T__0: createStmt(); break;
    case T__5: generateStmt(); break;
}
```

This switch identifies the starting point of each statement. By using LA(1) lookahead, the parser inspects the upcoming token and invokes the appropriate method, such as createStmt or generateStmt. This design allows clean branching based on language constructs.

Parsing Object Creation Blocks

The parser recognizes a creation block by detecting the keyword create, followed by two identifiers, and finally a set of attributes enclosed in curly braces. Each of these components is parsed in a specific order to ensure the structure matches the grammar.

The attributeList() function is responsible for parsing the contents inside these braces, the parser reads each attribute by checking if the token is a keyword or identifier. This is done using a simple loop that continually invokes the attribute rule:

```
while (_la==KEYWORD || _la==IDENTIFIER) {
    attribute();
}
```

This logic ensures that every attribute declaration is accounted for before moving forward. The attribute method itself supports multiple patterns depending on the nature of the attribute. These include simple key-value pairs, nested blocks for grouping attributes, and dynamic attributes with expressions.

```
identifier(); match(T__3); value(); match(T__4);
identifier(); match(T__1); attributeList(); match(T__2);
identifier(); match(T__3); expression(); match(T__4);
```

The first variant parses attributes like calories(200) — a standard key-value pair. The second supports nested attributes such as nutrition fat(10) protein(20) , which are useful for organizing related parameters. The third allows computed values like goal(weight - 5), giving the DSL a more dynamic and expressive capability.

Each version ensures that the syntax is strict: values must be enclosed in parentheses, and nested blocks in curly braces. The ability to mix static, structured, and computed data in a single block is a key strength of the DSL.

Handling Generate Statements

During the development, we needed to create the method: generateStmt. This method supports two syntax variations: block-style and inline parameters.

```
match(T__5);
identifier();
match(T__1);
paramList();
match(T__2);
```

This form is used when parameters are enclosed in curly brackets. It is structurally similar to the create block and useful for verbose configurations. The second form supports parentheses (round brackets), offering a concise syntax suitable for shorter configurations or quick calls. The paramList method parses comma-separated parameters:

```
param();
while (_la==T__6) {
    match(T__6);
    param();
}
```

Each parameter can consist of a single identifier or an identifier followed by multiple values:

```
identifier();
if (_la == T__3) {
    match(T__3);
    value();
    while (_la == T__6) {
        match(T__6);
        value();
    }
    match(T__4);
}
```

This logic allows parameters such as days(1, 2, 3) to be declared naturally. The parser checks for an opening parenthesis and parses one or more values before closing the group.

Value Parsing and Expression Handling

Values in the DSL can be direct literals, references, or identifiers.Direct literals are straightforward values like numbers or strings that are directly used in the code. References are pointers to other values or objects that have been defined elsewhere in the code. Identifiers are names given to variables or functions that help in identifying them within the code. The parser handles this by attempting each possibility in order:

```
literal();
reference();
identifier();
```

This polymorphic approach means that the same rule can match 200, user.age, or planType depending on the input. It gives the user flexibility in value definition. When expressions are used, particularly in dynamic attributes or conditions, the parser uses reference(); operator(); value();

This matches binary expressions such as calories ¿ 300. Each part is parsed using subrules to ensure syntactic and semantic validity.

Looping and Control Structures

The DSL supports multiple control structures: while, for, and foreach loops. These allow iterative operations within blocks.

A classic while loop is parsed as:

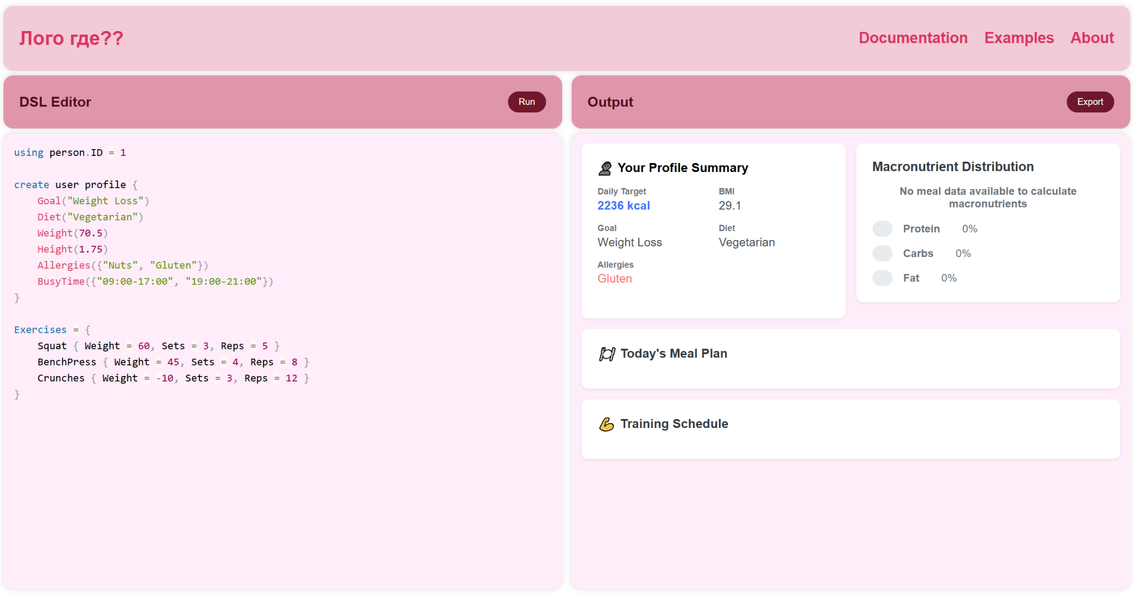
```
match(T__14);
match(T__3);
condition();
match(T__4);
block();
```

5.4 Visualization

When we started designing the frontend part of our DSL, we wanted to create something that didn't feel like just another boring nutrition app. Most health tracking tools look sterile and with white backgrounds and cold color schemes. We went in the opposite direction, and chose a warm pink-ish palette that reminds of strawberry milk.

The website is basically split into two parts: a code editor area that looks like something a developer would use, and an output area that breaks down your nutrition in a way that's actually interesting to look at and easy to understand. The chocolate and cream color scheme wasn't just a random choice - it's meant to make nutrition feel approachable, almost like you're reading a recipe instead of writing code and looking at data.

The left side lets you write in a our custom DSL, almost like you're coding your diet, and the right side translates that into something visual and understandable. It's technical enough for professionals and trainers but also soft enough for regular people who are just trying to improve their diet. For now, the most important parts of the webpage such as code editor and output area are already implemented, but additional features such as simplified input mode and output export arr still work in progress, and will be added soon.



```
using person.ID = 1

create user profile {
    Goal("Weight Loss")
    Diet("Vegetarian")
    Weight(70.5)
    Height(1.75)
    Allergies({"Nuts", "Gluten"})
    BusyTime({"09:00-17:00", "19:00-21:00"})
}

Exercises = {
    Squat { Weight = 60, Sets = 3, Reps = 5 }
    BenchPress { Weight = 45, Sets = 4, Reps = 8 }
    Crunches { Weight = -10, Sets = 3, Reps = 12 }
}
```

Your Profile Summary

Daily Target 2236 kcal
BMI 29.1
Goal Weight Loss
Diet Vegetarian
Allergies Gluten

Macronutrient Distribution

No meal data available to calculate macronutrients

Protein 0%
Carbs 0%
Fat 0%

Today's Meal Plan

Training Schedule

Figure 5.4.1 - Frontend part

The frontend part of our project is a single-page website designed to serve as an interactive code editor for our DSL. The left part of the page is a textarea that allows you to write DSL code. The right part is the output area, that shows the results of running the code. Since our website is structured in a modular way, with each feature having separate scope, we decided to use React.js during development, to split the webpage into components. We made a separate component for the core app, page header, code editor, and output section. JSX components are transformed into regular JavaScript using Babel. We used Vite as our build tool, since it is very easy to use and allows for Hot Module Replacement. The styling on our webpage is done with Sass, since it allows to write scalable style sheets and more easily manage a large amount of CSS. Using Sass allowed us to make it so that our webpage has a set color theme, which can be easily reconfigured to any other color palette on the fly. The DSL code editor on the right side of our webpage allows to write and run our DSL code. It supports syntax highlighting, and is configured to intelligently recognize DSL structure using Regular Expressions. The editor itself is implemented using the react-simple-code-editor library, which allowed us to configure and modify the way it displays code. Syntax highlighting is done with the Prism.js library, with a color theme that can be configured to better match our DSL's overall theme. The app is lightweight, has fast loading times, runs in any modern browser, and supports many different screen resolutions. The output shows the nutritional summary, charts and progress indicators, also giving the option to export the result.

5.5 Interpreter and Core Functionality

An interpreter is a program that directly executes instructions written in a programming or scripting language without requiring them to be compiled into machine code. Implementing an interpreter involves designing a system that reads and processes source code, translating it into actions that the computer can perform. Considering that we already analysed the breaking the source code into tokens and parsing, the last thing that remains is to see how the DSL executes the instructions from the source code. The next subsections will be dedicated to a throughout analysis of the interpreter core.

Core Components

- **Json_Handler** - includes a method SaveUser that will be included in the User Profile creation feature and in general it works with input Json file.
- **Txt_Handler** - gets the input from the user on frontend and saves it into a .txt file, which later is parsed by the DSL.
- **Program** - gets what exactly feature we want to use. Among all features there are Creating a User Profile, Creating a One Day Meals Plan or Creating a Weekly Training Schedule. For the first time, we are working with only one example of a user that we create in the User Profile, but now we don't even include any interactions with that part, so all functions work separately.
- **Process** - this part of the code defines the processing of features for the user such as the calculation of BMI, calories for the person and adding the meal plan and training schedule for the user
- **User_Profile** - gets such variables for the user as: ID, name, weight, height, goal (get fit, maintain, bulk, etc), busy time per day (i.e. from 2 PM to 6 PM), specific diet (vegetarian, high protein, etc), allergy ingredients (nuts, dairy, etc), exercises (i.e. Flat Bench Press / 50kg / 5 sets / 3 reps), BMI, calories per day, meal plan and training schedule. The feature of creating a User Profile is working the next way: the interpreter reads the input JSON, then calculates the BMI, calories per day and includes meal and training plans. We also developed a Modify User feature, that actually works pretty similar to user creation, but

instead of creating anew user it modifies the user.

Now, we managed connect all three parts and display them dynamically on the screen for the convenience of the user. It is really simple and usable for the user to work when there is a simple interface and a pleasant, fast and most importantly, understandable output format.

Meal Plan Structure

For the meal generation, we developed the following components:

- **Meal** - includes name, description, calories, carbs, proteins and fats
- **Meal_Plan** - includes lists of meals for breakfast, lunch, dinner and snack (in future will be changed on the list of meals like meal 1, meal 2, meal 3, etc)
- **Meal_Plan_Generator** - will include a lot of aspects that must be included in creating a meal plan. At the moment it represents a standard choice of meal planning for vegetarians or others. The data now is presented just as strings we wrote by hand.

In general Meal Planning at the moment includes creating a meal plan for vegetarians and for other people with predefined inputs and includes into the user's profile.

Instead of giving one "best" variant of meal plan, the output consists offers a list of several good meals for each meal type: breakfast, lunch, dinner, and snack. As an input this feature gets the same data as user creation. Some aspects that we have overviewed are considered with allergies and diet. In the future, we plan to add a warning note to the recipies that contain allergies, because in such way we let users have more variety of foods to get. Output here contain several meals found in database with their: name, list of ingredients, calories, carbs/proteins/fats and meal type (in future this may outline is food better to eat after trainings or right before going to sleep for example)

Training Schedule Structure

For the training generation, the components look in the following way:

- **Workout** -includes name, type of train (Wheighted, Cardio, etc)
- **Workout_Training_Schedule** - will be used to set a weekly plan by days. Includes 7 days of the week as a workout (monday workout, tuesday workout, etc)
- **Training_Schedule_Generator** - works in the same way as Meal Plan Generator, but with workouts instead. Depending on the goal of the user (now it is gaining muscles or losing weight) it generates a schedule for the user

Input here is the same as in user's creation and meal planning. In addition to basic user's data as id, name and goal with proper schedule. it gets the number of days with workouts, estimated burned calories (how much user will burn calories during the week doing those exercises), recommendations for any type of users and finally the schedule where each workout day has: name, type, duration of workout, intensity, difficulty, calories burned and scheduled time of workout. The last feature was created as a base for future connection of scheduling based on time.

Other Important Features

Find Optimal Time

The first new feature added to provide a possibility for a user to schedule his day automatically. As a result we got a feature that works technically but it doesn't consider human factor, so it just finds the best time it could get. Input here consists of user's busytime and exercises for the day. Right now time is not considered to work as real like (23:00 - 06:00) but it works with 24:00 and 00:00 as the end and begin of the day. Output here is pretty simple: time of each meal + time for training with a small note about.

Find Meals By Ingredients

Here is represented a new branch of our features that is a first step into "recipies" aspect of our DSL. Using a list of ingredients it find meals with those ingredients from the meals database Input here is a list of ingredients and a match. There are 2 types of matches: "any" that will output meals with any of those ingredients in recipies and "all" which will output meals where are included all ingredients from list the function outputs in addition to the input data the list of meals and count the number of meals. Each meal has: name, diet, meal type, list of all ingredients and nutrition list (calories/carbs/proteins/fats)

Database

We've developed a something that can be named as "noSQL Database" since our database inside interpretor consists of 4 json files:

- **Users.json** - contains all users that had been created, added or modified. Data is simialr to create user style
- **Meals.json** - contains all meals we've added already, is used for working with meal searching and meal plan generation. Each meal has it's meal type and category (diet type)

- **Ingredients.json** - contains all ingredients that can be used in Meals.json, which are sepeareated by type of ingredient
- **Workouts.json** - contains all workouts which are used to create a training schedule. Workouts, like meals have their own categories based on goals and type of training

JSON Transformation

The parse tree generated by ANTLR provides a comprehensive representation of the program's structure but is tightly coupled with ANTLR's internal representation. This makes it challenging to work with in application code. By transforming the parse tree into JSON, we create an intermediate format that preserves the structure while making it accessible through standard JSON processing libraries. The ParseTreeToJsonUtil.java class handles this initial conversion, producing a raw JSON representation that maintains the hierarchical nature of the parse tree.

Once the raw JSON version of the parse tree is available, a second transformation is required to extract the semantic meaning and organize it according to our domain model. This is where JsonTreeTransformer.java is needed, analyzing the raw tree JSON and converting it into a structured representation.

This section examines the implementation of a two-stage JSON transformation that processes the parse tree and converts it into a semantically rich structure that can be utilized by other components of the system.

Parse Tree to Raw JSON Transformation

The raw JSON representation serves as an intermediate format that bridges the gap between the syntactic structure captured by the parse tree and the semantic understanding required by the application. While this representation contains all the information from the parse tree, it is still highly coupled to the grammar structure and not directly usable by application logic. This requires a second transformation stage to extract and organize the semantic content.

The ParseTreeToJsonUtil class forms the foundation of this conversion process. This utility class encapsulates the logic required to transform an ANTLR parse tree into a structured JSON representation. The implementation preserves the hierarchical nature of the parse tree while making it more accessible for subsequent processing stages.

The primary method toJson initiates the conversion process by creating a root JSON object and invoking the recursive traversal method. The resulting JSON string is formatted with an indentation of 4 spaces for improved readability. This formatting is particularly beneficial during development and debugging phases when manual inspection of the JSON structure is necessary.

The core of the conversion process:

```java
private static void traverse(ParseTree tree, JSONObject json) {
    if (tree == null) return;

    if (tree instanceof TerminalNode) {
        Token token = ((TerminalNode) tree).getSymbol();
        json.put("text", token.getText());
    } else {
        String className = tree.getClass().getSimpleName();
        String ruleName = className.endsWith("Context") ?
                className.substring(0, className.length() - 7) :
                className;

        ruleName = Character.toLowerCase(ruleName.charAt(0)) + ruleName
            .substring(1);

        JSONArray children = new JSONArray();
        json.put(ruleName, children);

        for (int i = 0; i < tree.getChildCount(); i++) {
            JSONObject childJson = new JSONObject();
            children.put(childJson);
            traverse(tree.getChild(i), childJson);
        }
    }
}
```

Terminal nodes correspond to individual tokens in the input stream, representing concrete syntactic elements such as identifiers, literals, and operators. For these nodes, the utility extracts the token text and stores it directly in the JSON object under the key ”text”.

Non-terminal nodes, conversely, represent grammar rules that structure the relationships between tokens. For these nodes, the utility derives the rule name from the class name of the node. ANTLR automatically generates classes for each grammar rule with the suffix ”Context” appended to the rule name. This implementation removes that suffix to obtain the original rule name from the grammar. Additionally, the first character of the rule name is converted to lowercase to adhere to conventional JSON naming practices.

Semantic JSON Transformation

The second stage of the transformation involves converting the raw JSON representation into a semantically meaningful structure that aligns with the application's domain model. This step is crucial for abstracting away the syntactic details and focusing on the meaning of the language constructs.

The JsonTreeTransformer class implements this transformation by traversing the raw JSON structure and extracting relevant information based on the known grammar patterns.

```java
public static void main(String[] args) throws IOException {
    String inputPath = "parseTree.json";
    String outputPath = "output.json";

    String rawJson = new String(Files.readAllBytes(Paths.get(inputPath)
       ));
    JSONObject rawTree = new JSONObject(rawJson);

    JSONObject result = transform(rawTree);

    try (FileWriter writer = new FileWriter(outputPath)) {
        writer.write(result.toString(4));
        System.out.println("output.json saved");
    }
}
```

The main method demonstrates the practical application of the transformer. It begins by reading the raw JSON file, parsing it into a JSONObject, and invoking the transform method to create the semantic representation. The resulting semantic JSON is then written to an output file with proper formatting.

Core Transformation Logic

The transformation process gathers all the necessary information from the parse tree. It is necessary because information related to a single semantic entity might be scattered across different parts of the syntax tree.

The first pass focuses on identifier detection and preliminary entity type recognition. It specifically looks for 'using' statements that establish bindings between identifiers and values.

The transformed JSON structure focuses on entities like users, persons, and their attributes, rather than on syntactic constructs like statements and expressions. This semantic organization makes the data immediately usable by application logic without requiring additional parsing or interpretation.

```java
public static JSONObject transform(JSONObject rawTree) {
    JSONObject result = new JSONObject();
    JSONArray program = rawTree.optJSONArray("program");
    if (program == null) return result;

    JSONObject createObject = new JSONObject();
    String functionName = "create_user";

    // First pass to detect using statement and ID
    for (int i = 0; i < program.length(); i++) {
        JSONObject stmtWrapper = program.getJSONObject(i);

        // Check if this is a direct usingStmt at the top level
        if (stmtWrapper.has("usingStmt")) {
            JSONObject using = handleUsing(stmtWrapper.getJSONArray("
               usingStmt"));
            String reference = using.optString("reference");
            Object value = using.opt("value");
            ...
            }
        }
    }
    return result;
}
```

Handling Entity Creation

When our language creates a new thing or object (like a user or person), the next handler processes it, transforming it in an JSONObject.

Then, we start by setting up variables to track what we are doing, we look for attribute lists:

```java
// Go through all parts of the create statement
    for (int i = 0; i < createStmt.length(); i++) {
        JSONObject node = createStmt.getJSONObject(i);

        // Process attribute lists (things in curly braces)
        if (node.has("attributeList")) {
            JSONArray attributes = node.getJSONArray("attributeList");
            for (int j = 0; j < attributes.length(); j++) {
                JSONObject attrWrapper = attributes.getJSONObject(j);
                if (!attrWrapper.has("attribute")) continue;

                JSONArray attr = attrWrapper.getJSONArray("attribute");
                if (attr.length() == 0) continue;
```

This part looks for attributes in the create statement. Attributes are things like name, age, or weight. Next, we get the attribute name:

```java
// Get the attribute name
                String key = "";
                if (attr.getJSONObject(0).has("identifier")) {
                    key = formatKey(attr.getJSONObject(0).getJSONArray(
                        "identifier").getJSONObject(0).getString("text")
                        );
                } else if (attr.getJSONObject(0).has("text")) {
                    key = formatKey(attr.getJSONObject(0).getString("
                        text"));
                }

                if (key.isEmpty()) continue;
```

Once we have the attribute name, we handle it based on what kind of attribute it is. Some attributes need special handling, like exercises (which have names, weights, reps), allergies (which are lists), and busyTime (which shows when someone is not free). This looks for attributes written differently, like ”Weight(180)” instead of ”weight: 180”, and for weight and height, we extract simple values.

Hence, this code checks if a node has a ”text” attribute and processes it if it matches certain values like ”Weight” or ”Height”. It then tries to extract and store the value, handling it as a number if possible.

```
        if (node.has("text")) {
            String text = node.getString("text");

            if (text.equals("Weight") || text.equals("Height") ||
                    text.equals("Allergies") || text.equals("BusyTime")
                       ) {
                inDirectAttribute = true;
                currentAttribute = formatKey(text);

                if (i + 2 < createStmt.length() && createStmt.
                   getJSONObject(i+1).has("text") &&
                        createStmt.getJSONObject(i+1).getString("text")
                           .equals("(")) {
                    if (text.equals("Weight") || text.equals("Height"))
                        {
                        if (i + 3 < createStmt.length() && createStmt.
                           getJSONObject(i+2).has("text")) {
                            String valueStr = createStmt.getJSONObject(
                               i+2).getString("text");
                            try {
                                result.put(currentAttribute, Double.
                                   parseDouble(valueStr));
                            } catch (NumberFormatException e) {
                                result.put(currentAttribute, valueStr);
                            }
                        }
                    }
```

Handling Exercise Blocks

Exercises require special processing as they contain structured workout data. The handleExercisesStmt method processes exercise statements to extract key workout information:

```
private static JSONArray handleExercisesStmt(JSONArray exercisesStmt) {
    JSONArray exercises = new JSONArray();
    JSONObject currentExercise = null;
    for (int i = 0; i < exercisesStmt.length(); i++) {
        JSONObject stmt = exercisesStmt.getJSONObject(i);
        if (stmt.has("exerciseEntry")) {
            JSONArray entries = stmt.getJSONArray("exerciseEntry");
```

The method begins by initializing storage for exercises and a variable to track the current exercise being processed, then iterates through the exercise statement components, specifically looking for exercise entry nodes which contain the actual workout data.

```
            for (int j = 0; j < entries.length(); j++) {
                JSONObject entry = entries.getJSONObject(j);
                if (entry.has("text") && !entry.getString("text").
                   equals("{")) &&
                        !entry.getString("text").equals("}")) {
                    currentExercise = new JSONObject();
                    currentExercise.put("name", entry.getString("text")
                       );
                    exercises.put(currentExercise);
                }
```

When an exercise name is found (excluding brace characters), a new exercise object is created and added to the exercises array. The name serves as the primary identifier for each exercise.

```
                if (entry.has("exerciseParams") && currentExercise !=
                   null) {
                    JSONArray params = entry.getJSONArray("
                       exerciseParams");

                    for (int k = 0; k < params.length(); k++) {
                        JSONObject param = params.getJSONObject(k);
                        if (param.has("text")) {
                            String text = param.getString("text");
```

This critical section processes the exercise parameters, the numerical values that quantify the workout. The code examines each parameter node to extract weight, sets, and repetitions data.

Weight processing demonstrates the method's robust handling of both numerical values and text representations, ensuring no data is lost even if formatting isn't perfect. The method concludes by returning the fully populated exercises array, now containing all parsed workout data in a structured JSON format.

Getting Simple Values

The extractValue method serves as a versatile parser for basic attribute values, it handles both literal values and references, automatically converting between data types when possible.

```java
private static Object extractValue(JSONArray attr) {
    for (int i = 0; i < attr.length(); i++) {
        JSONObject node = attr.getJSONObject(i);

        if (node.has("value")) {
            JSONObject valNode = node.getJSONArray("value").
               getJSONObject(0);
```

Getting Array Values

For more complex data structures, extractArrayValue processes both string-formatted arrays and proper JSON arrays:

```java
public static String toJson(ParseTree tree) {
    JSONObject jsonObject = new JSONObject();
    traverse(tree, jsonObject);
    return jsonObject.toString(4);
}
```

Simple Helper Methods

The stripQuotes method removes both double and single quotes from string literals, ensuring consistent representation regardless of the quotation style used in the source code.

The formatKey method applies camelCase formatting to attribute keys by converting the first character to lowercase. This ensures consistency with common JSON naming conventions and makes the transformed data more natural to use in JavaScript and Java contexts.

The joinIdentifiers method composes dot-notation paths from chains of identifiers, which are commonly used for references and nested access patterns in the DSL. The method inserts dots between identifier segments while avoiding dots at the beginning or end of the path.

```java
private static String stripQuotes(String s) {
    return s.replaceAll("^\"|\"$", "").replaceAll("^’|’$", "");
}

private static String formatKey(String s) {
    return Character.toLowerCase(s.charAt(0)) + s.substring(1);
}
```

```java

private static String joinIdentifiers(JSONArray ref) {
    StringBuilder sb = new StringBuilder();
    for (int i = 0; i < ref.length(); i++) {
        JSONObject part = ref.getJSONObject(i);
        if (part.has("identifier")) {
            if (sb.length() > 0) sb.append(".");
            sb.append(part.getJSONArray("identifier").getJSONObject(0).
                getString("text"));
        }
    }
    return sb.toString();
```

By transforming the raw syntax JSON into a semantic JSON that represents real things, we make it much easier for the rest of our application to work with the data. Each part of our system can focus on what it does best, without needing to understand all the details of our language syntax.

Conclusions

In conclusion, our project has successfully developed a Domain-Specific Language (DSL) that helps individuals manage their fitness and nutrition more effectively. By combining workout and meal planning into one easy-to-use system, we aim to address the common problem many people face when trying to balance exercise and healthy eating. Our solution not only simplifies the planning process but also provides personalized recommendations based on users' fitness goals and dietary preferences.

Throughout this report, we have detailed the development of our DSL, emphasizing its role in integrating fitness and nutrition management. We explored the challenges individuals face in balancing these two aspects and how our solution addresses these issues by providing personalized recommendations. Furthermore, we highlighted the comparative analysis with other existing solutions, showcasing the unique advantages of our approach. By synthesizing these elements, we have demonstrated the potential of our DSL by creating a solution that is working well and has potential for future growth.

Overall, we recognize that there are still areas for improvement. For instance, incorporating a feature that allows users to select their own workout routines and track their daily progress would greatly enhance the user experience. Additionally, we are working on refining the website's functionality, such as adding a simplified input mode and output export options, to make it even more user-friendly. Overall, we believe that our project has a strong foundation for future developments in fitness and nutrition management. Hence we are excited about the potential impact it can have on users' lives.

Moreover, we plan on expanding the functionality and usability of our DSL to include features that might include social interaction among users and trainers. Additionally, we plan to explore partnerships with nutritionists and fitness experts to offer users access to professional advice and maybe more tailored programs.

Bibliography

[1] Cleveland Clinic. ”Body Mass Index (BMI).” 2018. https://my.clevelandclinic.org/health/articles/9464-body-mass-index-bmi. Accessed: 2025-02-20.

[2] Dr. Beth Oller. ”Nutrition for Athletes.” https://familydoctor.org/nutrition-for-athletes/.

[3] National Center for Health Statistics. ”Products - Data Briefs - Number 443 - August 2022.” https://www.cdc.gov/nchs/products/databriefs/db443.htm. August 2022.

[4] Statista. ”Most common barriers to fitness 2023.” https://www.statista.com/statistics/1445847/common-fitness-barriers/. 2023.

[5] ResearchGate. ”(PDF) Research and practice of personalized fitness plans based on Hmove platform.” https://www.researchgate.net/publication/380283902_Research_and_practice_of_personalized_fitness_plans_based_on_Hmove_platform. 2025.

[6] Michael J. Widener, Linda Ren, Chloe C. Astbury, Lindsey G. Smith, and Tarra L. Penney. ”An exploration of how meal preparation activities relate to self-rated time pressure, stress, and health in Canada: A time use approach.” *SSM - Population Health* 15 (2021). https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8170144/.

[7] ResearchGate. ”(PDF) Meal planning is associated with food variety, diet quality and body weight status in a large sample of French adults.” https://www.researchgate.net/publication/313265513_Meal_planning_is_associated_with_food_variety_diet_quality_and_body_weight_status_in_a_large_sample_of_French_adults. February 2025.

[8] Statista Daily Data. ”Infographic: Smartphone as Personal Trainer.” https://www.statista.com/chart/24702/gcs-share-smartphone-users-regularly-use-fitness-apps. April 2021.

[9] Google Trends, *Google Trends*. Available at: https://trends.google.com/trends/explore?date=now%201-d&gprop=youtube&q=meal%20plan,meal%20prep&hl=ro (Accessed: Feb. 19, 2025).

[10] MYPROTEIN™. ”How Do People Meal Prep Around the U.S.?” https://us.myprotein.com/thezone/motivation/how-do-people-meal-prep-around-the-u-s/. 2025.

[11] The Penguin India Blog. ”Conquering the challenges faced by fitness enthusiasts.” https://penguinindiablog.wordpress.com/2013/09/16/conquering-the-challenges-faced-by-fitness-enthusiasts/. September 2013.

[12] ISSA. ”What Are the Challenges of Being a Nutritionist?” https://www.issaonline.com/blog/post/what-are-the-challenges-of-being-a-nutritionist. 2025.

[13] Olivia Morgan. ”Sports and Nutrition: What Athletes Need to Know.” https://www.massgeneralbrigham.org/en/about/newsroom/articles/sports-and-nutrition.

[14] ALBERT STUMM. ”No Pain, No Gain? Hardly. This Year’s Fitness Buzzword is ’Recovery’.” https://apnews.com/article/2ffce1799725037b0142657db62d9e8d.

[15] Mia Erickson. ”Healthy Hacks: The Busy Person’s Guide to Staying Active.” https://www.dailytelegraph.com.au/lifestyle/the-busy-persons-guide-to-staying-active/news-story/18c6f15af0983505ebd164228ffb7dd7.

[16] Charles W. Cox. ”Family Food and Nutrition.” 2020. https://gucchd.georgetown.edu/products/FamilyFoodandNutrition.pdf. Accessed: 2025-02-20.

[17] Health, *10 Realistic Fitness Goals, Recommended by Personal Trainers*. Available at: https://www.health.com/fitness/fitness-goals (Accessed: 18 March 2025).

[18] Kamb, S., *How To Build Your Own Workout Routine (Plans & Exercises)*. Available at: https://www.nerdfitness.com/blog/how-to-build-your-own-workout-routine/ (Accessed: 17 March 2025).

[19] Mayo Clinic, *Can you sing while you work out?*. Available at: https://www.mayoclinic.org/healthy-lifestyle/fitness/in-depth/exercise-intensity/art-20046887 (Accessed: 13 March 2025).

[20] Harvard Health, *Importance of Exercise: Benefits & Recommended Types*. Available at: https://www.health.harvard.edu/exercise-and-fitness/exercise-fitness (Accessed: 3 March 2025).

[21] Chen, W.-H., Chiang, C.-W., Fiolo, N. J., Fuchs, P. X., & Shiang, T.-Y., *Ideal Combinations of Acceleration-Based Intensity Metrics and Sensor Positions to Monitor Exercise Intensity under Different Types of Sports*. Sensors, 22(7), 2583, 2022. DOI: https://doi.org/10.3390/s22072583. Available at: https://www.mdpi.com/1424-8220/22/7/2583 (Accessed: 20 March 2025).

[22] Google. ”Google Fit: Activity Tracking - Apps on Google Play.” 2022. https://play.google.com/store/apps/details?id=com.google.android.apps.fitness&hl=en&gl=US. Accessed: 25-Nov-2022.

[23] Google. ”MyFitnessPal: Calorie Counter - Apps on Google Play.” 2022. https://play.google.com/store/apps/details?id=com.myfitnesspal.android&hl=en&gl=US. Accessed: 25-Nov-2022.

[24] Google. ”Lifesum: Healthy Eating & Diet - Apps on Google Play.” 2022. https://play.google.com/store/apps/details?id=com.sillens.shapeupclub&hl=en&gl=US. Accessed: 25-Nov-2022.

[25] Google. ”Da Fit - Apps on Google Play.” 2022. https://play.google.com/store/apps/details?id=com.crrepa.band.dafit&hl=en&gl=US. Accessed: 25-Nov-2022.

[26] Google. ”Fitbit - Apps on Google Play.” 2022. https://play.google.com/store/apps/details?id=com.fitbit.FitbitMobile&hl=en&gl=US. Accessed: 25-Nov-2022.

[27] Mrs. G. Sailaja. ”Study on Exercise and Nutrition.” *International Journal of Progressive Research in Engineering, Management and Science*, 2024. https://www.ijprems.com/uploadedfiles/paper//issue_11_november_2024/37264/final/fin_ijprems1732863734.pdf. Accessed: 2025-02-20.

[28] JournalNX. ”Nutrition for Exercise and Health: A Brief Review.” *IT Medical Team*, 2024. https://www.itmedicalteam.pl/articles/

nutrition-for-exercise-and-health-a-brief-review.pdf. Accessed: 2025-02-20.

[29] Unknown. ”Sporting Performance and Food.” https://www.betterhealth.vic.gov.au/health/healthyliving/sporting-performance-and-food.

[30] Corina Tifrea and Valentin Cristian. ”Scientific Perspective on Athletic Studies.” *SEA Open Research*, 2024. https://seaopenresearch.eu/Journals/articles/SPAS_18_10.pdf. Accessed: 2025-02-20.

[31] CHUA CHU YAN. ”Final Year Project Report - IA 2024.” Universiti Tunku Abdul Rahman (UTAR), 2024. http://eprints.utar.edu.my/6606/1/fyp_IA_2024_CCY.pdf. Accessed: 2025-02-20.

[32] Muskan Sharma. ”Study on Fitness and Nutrition.” *Journal of Emerging Technologies and Innovative Research*, 2022. https://www.jetir.org/papers/JETIR2204456.pdf. Accessed: 2025-02-20.

[33] Anita Been. *Food for Fitness: How to Eat for Maximum Performance (4th Edition)*. 2020. https://students.aiu.edu/submissions/profiles/resources/onlineBook/v5z6e5_Food_for_Fitness_How_to_Eat_for_Maximum_Performance-_4th_Edition.pdf. Accessed: 2025-02-20.

6 Appendix

Team Contract

Team Contract Project Title

Semester: 4

Project theme: DSL for Fitness and Nutrition Management

Project period: February 2025 - June 2025

ECTS: 8

Supervisory team: Mentors: Cretu Dumitru

Project group number: Team 8

Team Members

Belih Dmitrii

Bujor-Cobili Alexandra

Tatarentev Denis

Rudenco Ivan

Mihalevschi Alexandra

Team Rules and Responsibilities

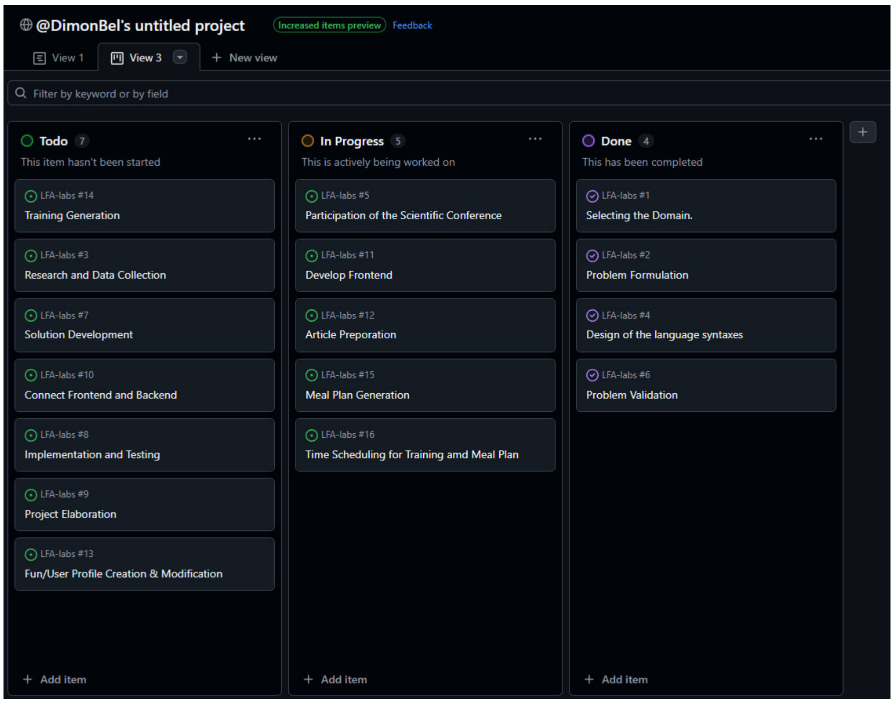
1. If the team members are not doing their work on time and with quality, they will be out of the team.
2. All of the team members are obliged to attend all the meetings, answer their messages, provide feedback, and be as involved as possible in this project.
3. All of the team members must listen to the team leader and, at their request, change and modify their tasks.
4. Each team member must document their work and progress.

By signing this document, each member of the group confirms participation on equal terms in the process of writing the project. Thus, each member of the group is responsible for all contents in the project.

Chisinau, 2025

Team Roles and Responsibilities

Member	**Responsibilities**
Belîh Dmitrii	Team leader. His job is to monitor the entire application development process and how the process is going.
Tatarințev Denis	Responsible for functionality and food. Everything related to food and articles is answered by Denis. The main logic that will work under DSL
Bujor-Cobili Alexandra	Develops the Lexer and collaborates on developing the parser.
Rudenco Ivan	Engaged in visualization of our part. How our DSL will look is his task. Then he will connect to the server.
Mihalevschi Alexandra	Include assistance in developing the DSL and also in developing the server part of the application.



Todo 7
This item hasn't been started

- LFA-labs #14 Training Generation
- LFA-labs #3 Research and Data Collection
- LFA-labs #7 Solution Development
- LFA-labs #10 Connect Frontend and Backend
- LFA-labs #8 Implementation and Testing
- LFA-labs #9 Project Elaboration
- LFA-labs #13 Fun/User Profile Creation & Modification

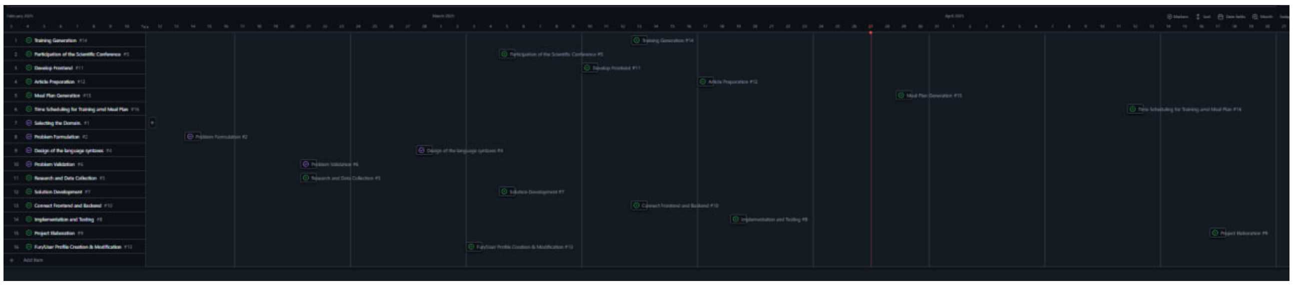
In Progress 5
This is actively being worked on

- LFA-labs #5 Participation of the Scientific Conference
- LFA-labs #11 Develop Frontend
- LFA-labs #12 Article Preporation
- LFA-labs #15 Meal Plan Generation
- LFA-labs #16 Time Scheduling for Training amd Meal Plan

Done 4
This has been completed

- LFA-labs #1 Selecting the Domain.
- LFA-labs #2 Problem Formulation
- LFA-labs #4 Design of the language syntaxes
- LFA-labs #6 Problem Validation

Figure 6.0.1 - Project Plan



1	Training Generation #14
2	Participation of the Scientific Conference #5
3	Develop Frontend #11
4	Article Preparation #12
5	Meal Plan Generation #15
6	Time Scheduling for Training and Meal Plan #16
7	Selecting the Domain. #1
8	Problem Formulation #2
9	Design of the language syntaxes #4
10	Problem Validation #6
11	Research and Data Collection #3
12	Solution Development #7
13	Connect Frontend and Backend #10
14	Implementation and Testing #8
15	Project Elaboration #9
16	Fun/User Profile Creation & Modification #13
+	Add item

Figure 6.0.2 - Project Deadline

Questions:

1) Who has a problem?

Individuals who want to optimize their fitness training and nutritional intake but struggle with planning effective workouts and well-rounded meals.

2) Who is involved in the problem situation?

Fitness enthusiasts, athletes, beginners, personal trainers, and nutritionists are involved. Additionally, software developers and health professionals play a role in providing solutions.

3) Who suffers from the problem?

People aiming to improve their physical health but lack knowledge or resources to create an optimal training and meal plan. This can lead to ineffective workouts, improper nutrition, or even health risks.

4) Who should take part in solving the problem?

Fitness experts, dietitians, software developers, and data analysts should collaborate to create a DSL that simplifies fitness and nutrition planning.

5) What happened?

Many people either fail to follow a structured fitness regimen or struggle to balance their post-workout nutrition. They often rely on scattered resources, making it difficult to create a cohesive plan that aligns with their goals.

6) What is the problem?

There is no streamlined, user-friendly tool that integrates workout planning with proper nutrition in an accessible way. Many existing fitness apps lack customization or require extensive prior knowledge. There are also applications that are specialized only for food or separate applications that are for fitness. There is no application that analyzes your nutrition and fitness at the same time.

7) What led you to identify the problem?

Studies indicate and gym experience that many individuals lack proper fitness guidance, leading to inconsistent results or injury. Additionally, research highlights the importance of nutrition in workout recovery, yet many struggle to plan appropriate meals. Existing tools either lack flexibility or require expertise, making it difficult for the average user to benefit fully.

8)What are the causes that affect those involved?

Current fitness and nutrition identification systems lack nutrition and settings between them, forcing users to manually coordinate their training and nutrition planning.

9)What are the needs of the target audience?

Users want a unified way to track both training and nutrition data, with automated links between exercise intensity, nutritional requirements, and analysis of their progress data.

10)What resources are needed?

The first step is to analyze the data in the food and fitness areas. You will also need a nutrition database, a team of developers, and 3-5 months of hard work.

11)Where did the problem arise?

As fitness apps became popular, they split into separate tools for workouts and nutrition, creating a need to bring these features back together. There is no app that will combine both nutrition and fitness into one common program.

12)When did the problem arise?

The issue became prominent as fitness apps grew in popularity but remained specialized in either workout or nutrition tracking, rarely combining both effectively.

13)When was the problem identified?

The problem emerged through analyzing conducting research on existing fitness applications.

14)When is it planned to be resolved?

The solution will be delivered by the end of the semester through our completed DSL implementation (approximately 3-5 months).

15)Why is there a need to solve the problem?

To improve fitness outcomes by automatically connecting workout data with nutrition requirements, eliminating manual coordination between different tracking systems and algorithm .

16)How can the problem be solved?

By developing our DSL with the main features Wellness, Scheduling, Food Planning and Sport Planning

17)How will the elimination of the problem be determined?

The issue is resolved when users can efficiently create personalized meal plans and workout routines using the DSL with minimal effort. Success is marked by reduced planning time, improved performance, user satisfaction, flexibility for diverse needs, and increased motivation as users feel more in control of their health.

18)How big is the problem?

The problem is significant for busy adults aged 25–44, who juggle work, family, and limited time for meal prep and fitness planning. With many spending nearly an hour on meal prep and seeking diet and exercise guidance, balancing these tasks is challenging. Automation offers a valuable solution to simplify their health management.

19)How much is the quantity/quality/number of people etc. affected?

A significant number of people rely on digital solutions for fitness and meal planning. In the adults aged 18–64 regularly use their smartphones as personal trainers. This suggests a growing demand for automated fitness guidance and meal planning tools. That millions of people worldwide want or plan to use already digital fitness and food related tools, a well-designed DSL could help make meal prep and training even more efficient and accessible.

20) How long will it take to resolve the problem?

Creating the DSL is just the first step. Once it's built, it will take some time for people to discover and start using it (like beta version and after that full app in production). It could take several months (like 3-5 months) before a significant number of users integrate it into their daily routines.

21) How many parties need to be involved in resolving it?

Developers – to design and refine the DSL. Fitness and nutrition experts – to provide logic for meal and workout plans. End users (individuals, trainers, meal prep businesses) – to test and provide feedback. Potential API providers – to integrate nutritional databases and fitness tracking tools.

22) Why our DSL would be better than any other app?

Our DSL offers several advantages over traditional apps:

- **Customization & Flexibility** – Users can create highly personalized meal and workout routines based on their specific preferences, dietary needs, and fitness goals

- **Automation & Efficiency** – Enables automatic generation and adjustment of plans, reducing time spent on repetitive tasks
- **Integration with Other Tools** – Can connect with fitness trackers, calorie calculators, and grocery lists
- **Scalability** – Useful for both individual users and professionals managing multiple clients

Here we will present the best features of our DSL that will include that.

Key Features:

1. **Wellness**: The DSL prioritizes user preferences and personal goals rather than enforcing strict health guidelines, ensuring flexibility in meal and workout planning. The main fact is that we have to include not the health features, because they are really strict and can just lead to more questions - instead we keep user's data as the start point of creating the plans, so it will depend only on personal feelings and preferences. So we won't pass through the health boundaries so it won't generate more problems. Example: the DSL will allow to include for user more recipes with the igredients he has
2. **Scheduling**: Users can personalize meal, workout, and recovery timing to fit their lifestyle, with options for predefined or flexible schedules. The DSL will include some features related to Time scheduling. We will research some most common options for time to train, getting meals and other parts of time management
3. **Food Planning**: The DSL provides structured meal planning based on dietary preferences, ingredient availability, and nutritional goals. Everyone needs to get recipes, best choices of foods and other similar stuff. So our DSL will give the possibility to get most usable features that will help to get the right recipes and maintain the usage of this DSL.
4. **Sport Planning**: The additional feature list will be based on planning the workouts that will help users to get to their goals. Based on their time, requirements and possibilities. So this will also be useful for anyone who can think about getting into this domain easily with help of our DSL