Pondera: A Personalized AI-Driven Weight Loss Mobile Companion with Multidimensional Goal Fulfillment Analytics

George Pashev
University of Plovdiv "Paisii Hilendarski"
georgepashev@uni-plovdiv.bg

Silvia Gaftandzhieva
University of Plovdiv "Paisii Hilendarski"
sissiy88@uni-plovdiv.bg

Abstract

The global obesity epidemic is a significant challenge to public health, necessitating innovative and personalized solutions. This paper presents Pondera, an innovative mobile app revolutionizing weight management by integrating Artificial Intelligence (AI) and multidimensional goal fulfilment analytics. Pondera distinguishes itself by supplying a tailored approach to weight loss, combining individual user data, including dietary preferences, fitness levels, and specific weight loss objectives, with advanced AI algorithms to generate personalized weight loss plans. Future development directions include refining AI algorithms, enhancing user experience, and validating effectiveness through comprehensive studies, ensuring Pondera becomes a pivotal tool in achieving sustainable weight loss and health improvement.

Keywords: AI; Weigh-Loss; Mobile application.

1 Introduction

Healthcare chatbots have significantly advanced medical technology by providing personalized, accessible, and engaging solutions in various domains such as mental health, chronic disease management, and weight loss. These chatbots deliver tailored dietary and exercise recommendations, essential for effective weight management. According to systematic reviews and meta-analyses by Franz et al. (2007) and Young et al. (2012, 2014), tailored interventions are crucial for sustained weight loss, underscoring the potential of chatbots in this area. By utilizing user data, chatbots enhance engagement and motivation through adaptive interactions and constant availability, which is crucial for users seeking weight loss support. The integration with wearable technology further personalizes feedback and recommendations, enhancing intervention effectiveness.

Despite their potential, deploying healthcare chatbots involves overcoming challenges related to information accuracy, user trust, and behavioral change. Ensuring the reliability of chatbot-provided information is critical, given the potential for negative health outcomes from inaccuracies. This necessitates rigorous sourcing and verification processes, ensuring information is derived from reputable, evidence-based medical sources (Franz et al., 2007). Furthermore, maintaining algorithmic transparency and mitigating bias are essential to ensure that chatbots provide unbiased, medically sound advice (Young et al., 2014).

Compliance with regulatory and ethical guidelines is vital for user trust and data security. In the U.S., the FDA regulates healthcare chatbots that offer diagnostic or therapeutic advice, detailing criteria for software oversight based on intended use and potential patient risks. The GDPR in the EU imposes strict data handling requirements, impacting chatbots that process personal health information. Similarly, the U.S.'s HIPAA mandates the protection of sensitive patient data, with additional international standards from ISO ensuring the reliability and safety of healthcare chatbots globally.

Healthcare chatbots are poised to revolutionize weight management and broader health interventions through their ability to provide personalized, dynamic support. However, realizing this potential requires continuous improvement, adherence to regulatory standards, and integration into comprehensive digital health ecosystems. Future advancements in AI will further enhance the personalization capabilities of

healthcare chatbots, making them indispensable tools in promoting healthier lifestyles and managing weight effectively.

Pondera's multidimensional analytics engine surpasses WeightMentor's (Holmes et al, 2019) basic goal-setting by simultaneously analyzing weight, diet, exercise, sleep, and stress. This holistic approach enables nuanced insights and personalized interventions. Our machine learning algorithms identify correlations between lifestyle factors, allowing for targeted goal adjustments. As an adaptive system, Pondera incorporates feedback loops and dynamic goal adjustment mechanisms. It continuously refines user goals and interventions based on real-time data, dynamically adjusting recommendations and support strategies. This adaptive architecture ensures personalized, effective support throughout the user's weight loss journey, optimizing outcomes and engagement.

2. Pondera: design and development

1.1 Pondera functionalities and components

Overview of Pondera Development Goals

Pondera aims to lead in weight management by effectively using AI and personalized analytics, as outlined in these specific goals:

G1 - Comprehensive Personalization: Utilize AI to analyze user data points to craft customized weight loss plans that evolve based on feedback.

G2 - Interactive User Assessment: Improve quizzes to understand users' weight loss goals and challenges, including psychological factors.

G3 - Nutrition and Fitness Integration: Offer diverse dietary and fitness options tailored to individual preferences and needs.

G4 - Behavioral Change Support: Implement habit formation, motivation, and progress tracking tools to encourage lasting changes.

G5 - User Engagement and Community Building: Develop features allowing users to share experiences and motivate each other.

G6 - Data Privacy and Security: Ensure robust data protection measures adhering to GDPR and HIPAA regulations.

G7 - Adaptive Learning and Feedback Loops: Continuously refine plans based on user feedback and changing circumstances.

G8 - Comprehensive Health Integration: Track and improve overall health metrics, promoting holistic well-being.

G9 - Partnerships with Health Professionals: Collaborate with experts to enhance credibility and effectiveness.

G10 - Continuous Research and Innovation: Stay at the forefront of AI, machine learning, and nutrition/fitness developments.

These goals guide Pondera's development to not only assist users in weight management but also support broader health and well-being objectives. By focusing on these goals, Pondera can truly revolutionize weight management, offering users a unique and effective tool to achieve their weight loss and health objectives.

For the ***development of Pondera***, a mobile application designed for personalized weight management and training plans, a comprehensive software architecture involving multiple technologies is required.

It contains 4 software components:

- User Interface (UI): This layer includes the presentation and interaction layer of the application, built with HTML, JavaScript, and Bootstrap. It allows users to input their goals, preferences, and other required details (see Fig. 1).
- Front-End: The front-end is responsible for sending requests to the back-end via AJAX calls and updating the UI based on the data received. It is built using JavaScript and interacts with the Flask CORS back-end for data processing.
- Back-End (API): The Flask application serves as the back-end, handling API requests from the front-end, processing data, interacting with the SQLite3 database, and communicating with external APIs like GPT-3.5. It employs Cross-Origin Resource Sharing (CORS) to enable secure cross-origin requests and responses.
- Database (SQLite3): This database stores all the static data required by Pondera, including user information, goals, user groups, available resources (meals, training sets), and their associated parameters (Zone diet blocks, calories, vegetarian index, ketogenic index, HIIT index, etc.).

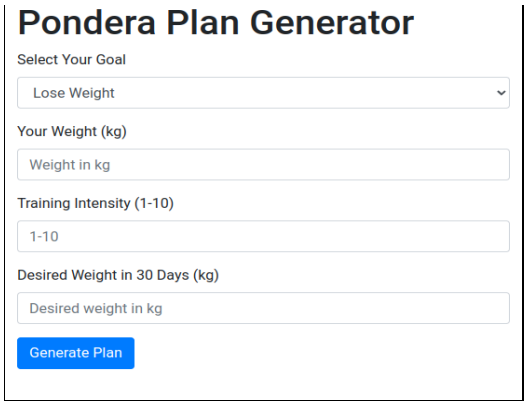

Figure 1. The data input form for Plan Generation

1.2 Software prototype

This system provides personalized training and eating plans tailored to user goals, dietary preferences, and exercise intensity, developed using Python Flask CORS, JavaScript, HTML, Bootstrap, and SQLite3.

The Pondera database includes eight key entities: Users, Goals, UserGroups, UserGoals, Resources, Meals, Trainings, and UserPlans. Users have attributes like UserID, Username, and Weight; Goals include GoalID and GoalDescription; Resources and Meals detail items such as ResourceID and Calories; Trainings and UserPlans track elements like VideoURL and daily assignments.

Relationships within the database include One-to-Many between Users and UserGoals, Many-to-Many between Users and UserGroups, and One-to-One between Resources and either Meals or Training. UserPlans detail the many-to-many relationships between Users, Meals, and Training, organizing daily meal and training assignments.

The app integrates with the GPT-3.5 API to update meal and training data dynamically, ensuring complete information for generating personalized plans.

Upon receiving user inputs (weight, desired training intensity, dietary preferences, and goal), the system utilizes a multidimensional vector space model to match and recommend a diverse yet ***personalized set of meals and training plans*** that align with the user's inputs and the Zone diet principles. Fig. 3 presents the process flow diagram of Pondera.

This architecture supports the dynamic generation of personalised weight management plans, leveraging the power of AI for data completion and offering users a tailored approach to achieving their weight loss goals:

- **User Interaction:** Users interact with the UI to enter their personal information, goals, and preferences.
- **Data Processing:** The front-end sends this data to the back-end via AJAX.
- **API Logic:** The Flask back-end processes the request, queries the SQLite3 database for matching resources, and communicates with the GPT-3.5 API as needed to complete missing data (see Fig. 4).

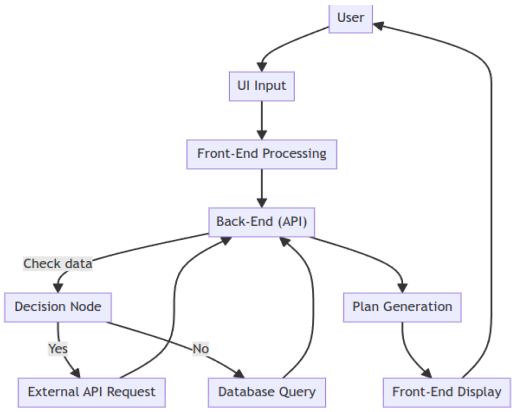

Figure 3. Process flow diagram of Pondera

```
from flask import Flask, request, jsonify
from your_plan_generator import generate_personalized_plan  # Placeholder for your actual function
app = Flask(__name__)
@app.route('/api/generate_plan', methods=['POST'])
def generate_plan():
    try:
        # Extracting user input from the request
        user_data = request.json
        weight = user_data.get('weight')
        training_intensity = user_data.get('training_intensity')
        daily_blocks = user_data.get('daily_blocks')
        desired_weight = user_data.get('desired_weight')
        # generate_personalized_plan returns a dict
        plan = generate_personalized_plan(weight,training_intensity, daily_blocks,desired_weight)
        return jsonify(plan), 200
    except Exception as e:
        return jsonify({'error': str(e)}), 400
if __name__ == '__main__':
    app.run(debug=True)
```

Figure 4. Python Flask API method endpoint for plan generation

Using algorithms, the back-end calculates the best match of meals and training sets based on the user's inputs and the multidimensional vector space model.

The vectorization process for Pondera's data involves converting the structured data from the

database into numerical vectors, which can be processed by machine learning models for generating personalized plans. This process includes:

- **Encoding Categorical Data**: Attributes like DietaryRestrictions, FitnessLevel, and GoalType are categorical and can be converted into numerical vectors using techniques like one-hot encoding or label encoding.
- **Normalizing Numerical Data**: Attributes such as Age, Height, Weight, Calories, Proteins, Carbs, and Fats should be normalized to ensure they're on a similar scale, typically between 0 and 1, to prevent any one feature from dominating the model's behaviour.
- **Text Vectorization**: For textual data, such as Ingredients in meals, techniques like TF-IDF (Term Frequency-Inverse Document Frequency) can be used to convert text into a meaningful vector of numbers.
- **Aggregating Data**: User profiles might need to aggregate data from Goals, Training Sets, and Meals based on user activity. This aggregated data can then be vectorized as a part of the user's profile vector.
- **Dimensionality Reduction**: After vectorization, dimensionality reduction techniques such as PCA (Principal Component Analysis) can be applied to reduce the number of features, if necessary, to simplify the model without losing significant information.

```
import pandas as pd
from sklearn.preprocessing import OneHotEncoder, MinMaxScaler
def preprocess_data(user_profiles, goals, training_sets, meals):
    # Fill missing values
    user_profiles.fillna(user_profiles.mean(), inplace=True)  # Numerical columns
    user_profiles.fillna('unknown', inplace=True)  # Categorical columns
    # Convert categorical data to numerical format
    encoder = OneHotEncoder(sparse=False)
    categorical_columns = cols  # Example columns
    encoded_features = encoder.fit_transform(user_profiles[categorical_columns])
    encoded_df = pd.DataFrame(encoded_features, columns=encoder.get_feature_names(categorical_columns))
    user_profiles = pd.concat([user_profiles.drop(categorical_columns, axis=1), encoded_df], axis=1)
    # Normalize numerical values
    scaler = MinMaxScaler()
    numerical_columns = ['Age', 'Weight']  # Example columns
    user_profiles[numerical_columns] = scaler.fit_transform(user_profiles[numerical_columns])
    # Similar preprocessing would be done for goals, training_sets, and meal data frames
    return user_profiles  # This function would return all preprocessed data frames in a real scenario
```

Figure 5. Data Preprocessing Process

This structured approach enables the creation of a comprehensive vector space that represents the multifaceted data involved in personalizing weight loss plans. With the vectors ready, machine learning algorithms can then be applied.

Moving forward to the ***Feature Extraction step***, we'll build upon the preprocessed data. The goal of feature extraction is to convert the raw data into a set of features that can be used for creating machine learning models. This involves identifying which attributes of the data are most relevant to the problem you're trying to solve and possibly creating new features from the existing ones to better capture the underlying patterns in the data. Feature extraction in Pondera involves:

- Selecting relevant nutritional information from meals (e.g., calories, proteins, carbs, fats) that aligns with dietary goals.
- Extracting key attributes from training sets (e.g., difficulty level, duration, calories burned).
- Incorporating user-specific goals and progress metrics into the features.

Fig. 6 shows a part of the code for implementing feature extraction in Python.

```
def extract_features(user_profiles, goals, training_sets, meals):
    # Example of extracting nutritional features from meals
    nutritional_features = meals[['Calories', 'Proteins', 'Carbs', 'Fats']]
    # Example of extracting training set features
    training_features = training_sets[['DifficultyLevel', 'Duration', 'CaloriesBurned']]
    user_goals_features = pd.merge(user_profiles, goals, on='UserID', how='left')
    return nutritional_features, training_features, user_goals_features
```

Figure 6. Feature extraction

In the Feature Extraction step following data preprocessing, we select key attributes from the data for vectorization to help craft personalized weight loss plans:

- User Profiles: Important features include age, current weight, dietary restrictions, and fitness level, which influence meal and workout recommendations.

- Goals: Factors like goal type (weight loss, muscle gain), target weight, and target date are crucial for plan personalization and need precise quantification.

- Training Sets: Attributes such as difficulty level, duration, and calories burned are vital for aligning workout plans with user goals and fitness levels.

- Meals: Essential nutritional details include calories, proteins, carbs, fats, and compatibility with dietary restrictions, critical for meal plan formulation.

The Vectorization Process requires three steps:

- **Numerical Features**: Numerical features like age, weight, calories, proteins, carbs, and fats are already in a suitable format for most machine learning algorithms. However, they might require normalization to ensure all features are on the same scale.
- **Categorical Features**: Categorical features, especially those that have been one-hot encoded, are already in a vectorized form. However, it's essential to ensure that the vectorization is consistent across the dataset to match the one-hot encoding schema used during training.
- **Combining Features**: Once all features are in a numerical format, we combine them into a single vector for each user profile and meal. This vector represents the input to our machine-learning models.

Fig. 7 presents a simple vectorization process for features implemented in Pondera where user**_features** and **meal_features** are pandas DataFrames containing our preprocessed and extracted features.

```
import numpy as np
def vectorize_features(user_features, meal_features):
    user_vectors = user_features.to_numpy()
    meal_vectors = meal_features.to_numpy()
    return user_vectors, meal_vectors
```

Figure 7. Vectorization process

Normalization and dimensionality reduction are essential for optimizing machine learning algorithms. Normalization adjusts each feature to scale uniformly, typically with a mean of 0 and standard deviation of 1, or within a range like 0 to 1. This uniform scaling reduces bias from features with larger scales, enhancing algorithm performance and speeding up algorithms like gradient descent.

Dimensionality reduction, often through methods like Principal Component Analysis (PCA), reduces the number of variables, retaining the most critical information with minimal data loss. This process simplifies models, decreases overfitting, and reduces computational demands, ultimately transforming the data into a lower-dimensional space that captures significant variance.

The use of tools like scikit-learn's StandardScaler and PCA in Python exemplifies these processes. StandardScaler normalizes features, while PCA reduces dimensions, preserving 95% of the data's variance to maintain essential information for model effectiveness.

To optimize daily meal and exercise plans, we employ a greedy algorithm that iteratively selects the optimal combination to balance calorie intake and expenditure while meeting nutritional goals. This method focuses on minimizing the difference between daily calorie consumption and burn, ensuring all nutritional needs (proteins, carbs, fats) are met within specified limits.

```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
def normalize_and_reduce_dimensions(data):
    scaler = StandardScaler()
    normalized_data = scaler.fit_transform(data)
    pca = PCA(n_components=0.95)  # retain 95% of the variance
    reduced_data = pca.fit_transform(normalized_data)
    return reduced_data
```

Figure 8. Normalization and dimensionality reduction

The implementation of this algorithm involves:

- Sorting meals and exercises by their caloric and nutritional values.
- Selecting meals that meet daily nutritional requirements without exceeding caloric limits.
- Choosing exercises that address any caloric surplus or help achieve calorie deficit goals.

This approach allows for practical daily planning, making decisions that are sufficient for day-to-day progress without needing to be globally optimal. The greedy algorithm simplifies decision-making by focusing on immediate objectives, efficiently balancing the diet and exercise routine to meet the user's health and weight loss targets.

Optimal meals are chosen from a sorted list based on their ability to meet daily nutritional needs without surpassing caloric limits. This selection is iterative, adding meals that enhance the nutritional profile, and updating remaining nutritional needs after each selection.

Exercises are selected to either match or exceed remaining caloric needs after meal choices. They aim to address any caloric surplus from meals or

create a deficit, selected for their high caloric burn to efficiently meet targets.

Placeholder functions help sort and select meals and exercises by calculating nutritional scores and adjusting needs based on meals' nutritional content. These functions, crucial for the algorithm, allow for informed decisions in the daily plan.

This algorithm iteratively selects meals and exercises that balance caloric intake and expenditure, aligning with nutritional requirements, to effectively meet fitness goals and cater to user preferences.

```
def optimize_daily_plan(selected_meals, selected_exercises,
daily_calorie_needs, daily_nutritional_needs):
    # Sort and select meals and exercises based on nutritional and
calorie needs
    sorted_meals = sorted(selected_meals, key=lambda x:
calculate_nutritional_score(x, daily_nutritional_needs))
    optimal_meals, remaining_needs = select_optimal_meals
(sorted_meals, daily_calorie_needs, daily_nutritional_needs)
    optimal_exercises = select_optimal_exercises(sorted(selected_exercises,
key=lambda x: x['calories_burned'], reverse=True),
remaining_needs['calories'])
    return optimal_meals, optimal_exercises
def select_optimal_meals(meals, calorie_needs, nutritional_needs):
    # Select meals that match nutritional and calorie requirements
    optimal_meals, remaining_needs = [], nutritional_needs.copy()
    for meal in meals:
        if meets_needs(meal, remaining_needs):
            optimal_meals.append(meal)
            update_remaining_needs(meal, remaining_needs)
    return optimal_meals, remaining_needs
def select_optimal_exercises(exercises, calorie_target):
    # Select exercises to fulfill or exceed remaining calorie needs
    optimal_exercises, calories_burned = [], 0
    for exercise in exercises:
        if calories_burned < calorie_target:
            optimal_exercises.append(exercise)
            calories_burned += exercise['calories_burned']
    return optimal_exercises
# Assume implementation details for placeholder functions
def calculate_nutritional_score(meal, needs): pass  # Calculate closeness
to nutritional needs
def meets_needs(meal, needs): pass  # Check meal meets remaining
nutritional needs
def update_remaining_needs(meal, needs): pass  # Update needs based
on meal
```

Figure 9. Part of the code for optimizing the daily plan

As a response, the personalized plan is sent back to the front-end, where it is displayed to the user.

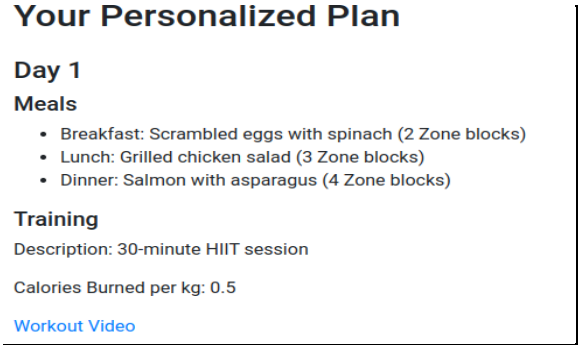

Figure 10. A view of an Item of a Generated Plan

The User Interface (UI) for displaying a generated plan in Pondera is designed to provide a clear and concise overview of a user's personalized training and eating plan. Utilizing Bootstrap's grid system, the UI is responsive, ensuring it looks good on various devices. Each day of the plan is clearly labeled (e.g., "Day 1"), with meals and workouts detailed under specific subheadings like "Meals" and "Training." These sections list meal names, their nutritional content like Zone diet block count, and a brief description of the day's exercise routine, including estimated calories burned and links to workout videos.

This structured presentation aids users in navigating their plan, enhancing readability and interaction. The UI also includes visual cues like icons or progress bars to provide feedback on the user's progress. Overall, the UI is not just informative but engaging, encouraging users to actively participate in their health regimen by following through with meals and workouts and interacting dynamically with the plan through features like marking activities as completed or navigating between plan days. This interface acts as a bridge between the complex backend algorithms and the user, making the personalized plans accessible, understandable, and actionable.

2 Experiment: Sample Request for Menu Generation

Objective: To evaluate the effectiveness of Pondera's AI-driven personalization algorithm in generating a week-long tailored meal plan for a user with specific dietary preferences and nutritional goals.

Methodology: A request was submitted to the Pondera system with the following user profile inputs:

Dietary Preferences: Vegetarian, Allergies: None, Nutritional Goals: Weight loss, with a focus on high protein and low carbs, Physical Activity Level: Moderate

Fig. 11 presents a generated menu.

```
   Day 1
   Breakfast: Scrambled tofu (150g) with spinach (50g) and avocado (100g) on whole-grain toast (2 slices).
   Lunch: Quinoa salad with chickpeas (100g), cucumbers (50g), tomatoes (50g), and feta cheese (30g).
   Dinner: Lentil soup (250g) with a side of roasted broccoli (75g) and carrots (75g).
   Snacks: Almonds (30g) and Greek yogurt (150g).

   Day 2
   Breakfast: Greek yoghurt (150g) with mixed berries (100g) and a sprinkle of chia seeds (15g).
   Lunch: Whole grain pasta (100g cooked) with pesto sauce (30g) and roasted vegetables (100g).
   Dinner: Grilled portobello mushroom (100g) with quinoa (100g) and steamed green beans (75g).
   Snacks: Sliced apple (150g) with peanut butter (15g).
```

Figure 11. Generated Menu Sample

Comments on the Generated Menu

Pros: **Personalization**: The menu adheres to the user's dietary preferences and nutritional goals, showcasing Pondera's ability to tailor recommendations. Nutritional Balance: Meals are well-balanced, providing a good mix of protein, healthy fats, and complex carbohydrates, aligning with the weight loss goal.

Cons: **Repetition**: The generated menu may lack variety over a week, potentially leading to diet fatigue. Including more diverse ingredients and cuisines could improve user satisfaction. **Practicality**: Some meals might require significant preparation time, which could be a barrier for users with busy schedules. Suggesting quicker options or meal prep tips could enhance usability.

The experiment indicates that while Pondera's menu generation feature is effective in creating personalized and nutritionally balanced meal plans, further refinement is needed in diversifying meal options and considering practicality for users with varying lifestyles.

More experiments indicated that sometimes the menu may contain incompatible food.

3 Conclusion

This paper highlights the development of Pondera, a mobile app designed for personalized weight management using AI and goal fulfillment analytics. It outlines how AI algorithms, user assessment, and the integration of nutrition and fitness methodologies are utilized to create tailored weight loss plans. The paper emphasizes the importance of extensive AI testing, user experience design, and validation studies to ensure the app's effectiveness in real-world scenarios. It also points to the need for scaling the app to serve a diverse user base and integrating continuous feedback mechanisms. Looking ahead, further advancements in AI and digital health ecosystems could significantly boost the effectiveness of healthcare chatbots in managing weight and promoting healthier lifestyles, making them vital tools in combating obesity.

Acknowledgments

The paper is financed by the Scientific Research Fund at the University of Plovdiv "Paisii Hilendarski", project № MUPD23-FTF-019.

References

M. Franz, J. VanWormer, A. Crain, J. Boucher, T. Histon, W. Caplan, ... & N. Pronk. 2007. Weight-loss outcomes: A systematic review and meta-analysis of weight-loss clinical trials with a minimum 1-year follow-up. Journal of the American Dietetic Association, 107(10): 1755-1767.

M. Young, D. Lubans, C. Collins, R. Callister, R. Plotnikoff, P. Morgan, ... & T. Burrows. 2014. Behavioral interventions to reduce sedentary behavior in children and adolescents: Systematic review and meta-analyses. British Journal of Sports Medicine, 48(3): 147-155.

M. Young, R. Plotnikoff, C. Collins, R. Callister, P. Morgan. 2012. Social cognitive theory and physical activity: A systematic review and meta-analysis. Obesity Reviews, 13(12):, 1100-1111.

FDA. Digital Health Guidance Documents, https://www.fda.gov/medical-devices/digital-health-center-excellence/guidances-digital-health-content.

European Commission. 2016. General Data Protection Regulation, https://eur-lex.europa.eu/eli/reg/2016/679/oj.

U.S. Department of Health & Human Services. 2021. HIPAA for Professionals, https://www.hhs.gov/hipaa/for-professionals/index.html.

ISO. 2021. ISO/TS 22272:2021 Health Informatics - Methodology for analysis of business and information needs of health enterprises to support standards based architectures, https://www.iso.org/standard/78905.html.

Y. Oh, J. Zhang, M. Fang, Y. Fokushida. 2021. A systematic review of artificial intelligence chatbots for promoting physical activity, healthy diet, and weight loss. Int J Behav Nutr Phys Act, 18(160).

B. Sears. 2024. https://drsears.com/the-zone-diet/.

A. Paoli. 2014. Ketogenic diet for obesity: friend or foe?, Int. Journal of Environmental Research and Public Health, 11(2): 2092-2107.

K. Varady. 2011. Intermittent versus daily calorie restriction: Which diet regimen is more effective for weight loss?, Obesity Reviews, 12(7): e593-e601.

R. Estruch, E. Ros, J. Salas-Salvadó, M. Covas, D. Corella, F. Arós, ..., M. Martínez-González. 2018. Primary Prevention of Cardiovascular Disease with a Mediterranean Diet Supplemented with Extra-Virgin Olive Oil or Nuts. New England Journal of Medicine, 378(25):e34.

M. McMacken, S. Shah. 2017. A plant-based diet for the prevention and treatment of type 2 diabetes, Journal of Geriatric Cardiology, 14(5):342.

A. Rahmanti, H. Yang, B. Bintoro, A. Nursetyo, M. Muhtar, S. Syed-Abdul, Y. Li. 2022. SlimMe a Chatbot With Artificial Empathy for Personal Weight Management: System Design and Finding." Frontiers in Nutrition, 9:870775

S. Holmes, A. Moorhead, R. Bond, H. Zheng, V. Coates, M. McTear. 2019. WeightMentor, bespoke chatbot for weight loss maintenance: Needs assessment & Development, 2019 IEEE Int. Conference on BIBM, 2845-2851.