INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Enhancing Health and Fitness: A Hybrid Recommender System

IVO LOUREIRO CASTRO
Setembro de 2024

POLITÉCNICO
DO PORTO

ISEP INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

Enhancing Health and Fitness: A Hybrid Recommender System

Ivo Loureiro Castro

Dissertação para obtenção do Grau de Mestre em Engenharia Informática, Área de Especialização em Engenharia de Software

Orientador: Constantino Martins

Porto, Setembro 2024

Orientador: Constantino Martins

Porto, Setembro 2024

Declaração de Integridade

Declaro ter conduzido este trabalho académico com integridade.

Não plagiei ou apliquei qualquer forma de uso indevido de informações ou falsificação de resultados ao longo do processo que levou à sua elaboração.

Portanto, o trabalho apresentado neste documento é original e de minha autoria, não tendo sido utilizado anteriormente para nenhum outro fim.

Declaro ainda que tenho pleno conhecimento do Código de Conduta Ética do P.PORTO.

ISEP, Porto, 15 de September de 2024

Dedicatória

À minha Mãe

Resumo

A digitalização no setor da saúde tem vindo a aumentar significativamente com os avanços tecnológicos nos anos recentes, por esta razão, é possível observar uma transformação significativa na forma como se previne e trata doenças e mal-estar, recorrendo a soluções digitais para melhorar este processo.

A prática de exercício físico é uma forma eficaz de prevenção ao reduzir a probabilidade de ter doenças físicas ou mentais, mas, atualmente, a quantidade de informação disponível sobre este tópico consegue ser avassaladora e cresce a um passo acelerado. Por esta razão, sistemas de recomendação são uma opção viável para filtrar informação e ajudar no processo de tomada de decisão.

Nesse sentido, este documento detalha o desenvolvimento de uma aplicação que utiliza *Python* e *a framework Django* para incorporar um sistema de recomendação híbrido que alterna entre a abordagem de filtragem por conteúdo e a filtragem colaborativa. O objetivo desta aplicação é auxiliar o utilizador a planear uma rotina de exercícios físicos.

Para esse efeito, o primeiro passo foi realizar a revisão da literatura para perceber o estado da arte de sistemas de recomendação aplicados à saúde. Este estudo incide sobre uma variedade de artigos científicos selecionados utilizando a metodologia PRISMA para garantir a relevância da informação. Esta análise permitiu identificar não só falhas, mas também diretrizes na literatura, orientando assim o desenvolvimento do projeto. As lacunas identificadas estão relacionadas com o estudo da eficácia percetível dos sistemas de recomendação, com a pequena amostra de exemplos concretos de soluções que asseguram a privacidade e proteção de dados do utilizador e com a fraca presença da inclusão do processo de esquematização da experiência do utilizador.

É esperado que este documento represente um contributo significativo para a compreensão das potencialidades e desafios dos sistemas de recomendação aplicados na saúde.

Abstract

This master's thesis details the development of a recommender system integrated into a health application, aimed at enhancing of enhancing physical activity.

This study's main objective is to analyse, design and implement a framework for a hybrid recommender system framework that provides personalized health recommendations to individual users. Following an ethical data management and prioritizing user privacy, this study aims to bring clarity to the complexities of personal health data usage, exemplifying a model for secure and user-centric application development.

This research aims to offer insights into key features of an effective health recommender system, by evaluating the system's impact on user behaviour and health outcomes and demonstrating how it can influence user engagement. Contributing to the broader understanding of digital health innovations and the potential of personalized health recommendations.

Keywords: Recommender System, Health Application, physical activity, exercise, ethics, data privacy.

Agradecimentos

Gostaria de agradecer à minha Mãe e a toda a minha Família por incutirem em mim os valores necessários para enfrentar qualquer desafio.

Ao meu Orientador, Professor Constantino Martins, por toda a atenção, tempo, mentoria e empenho para que este documento fosse o melhor possível.

Aos meus amigos, neste contexto, ao Ricardo e ao Pedro, os profissionais da área da saúde que me ajudaram a desenvolver este sistema de recomendação.

Table of Contents

List of Figures

Orientador: Constantino Martins

Porto, Setembro 2024

List of Tables

List of Code Snippets

List of Equations

Acronyms and Symbols

List of acronyms

API – Application Programming Interface

BFP – Body Fat Percentage

BMI – Body Mass Index

CSV – Comma-Separated Values

DRF – Django Rest Framework

HTTP – Hypertext Transfer Protocol

JWT – JSON Web Tokens

MAE – Mean Absolute Error

MSE – Mean Squared Error

ORM – Object-Relational Mapping

REST – Representational State Transfer

RMSE – Root Mean Square Error

SQL – Structured Query Language

SVD – Singular Value Decomposition

URL – Uniform Resource Locator

1 Introduction

Digital health applications have a growing impact on the healthcare field. The advancements in computing power, data storage systems and artificial intelligence allow the creation of software-based solutions that not only help in treating illness but also in preventing them. Nowadays, by utilizing data from devices like smartphones and wearables, digital health solutions can help users monitor and prevent health risks. The collaboration between technology and healthcare is paving the way for more personalized, efficient and accessible patient care (Park et al,2019).

Technology has a crucial role when dealing with the large volume of data that exists in the healthcare field of research. Data filtering and mining are proving to be more important in aiding end-users with health-related decisions. Recommender systems are a type of information filtering that can be used as a personal health advice tool, enhancing users' well-being (Jangde and Ramaiya,2022).

The objective of this development is to explore and assess the creation of a health-focused application featuring a recommender system tailored for physical activity. It is expected that this document contributes to field of digital health solutions, while serving as a model for ethical development pratices.

The developed recommender system is a hybrid model that combines content-based and collaborative filtering techniques to provide exercise recommendations. Its performance and effectiveness were evaluated using a range of statistical and predictive metrics, demonstrating the system's capability to adapt to various user profiles and interactions.

The chapter begins with the problem statement that identifies challenges and gaps in physical activity and general well-being. Then a subsection that outlines the main and specific objective of this thesis, followed by the expected outcomes and contributions to the field of digital health. Ethical considerations are also discussed to provide a clear view of what ethical challenges arise in the development of a recommender system. Lastly, a detailed schedule of activities that divides the development and documentation process into phases, followed by an explanation of the thesis' organization and its chapters.

1.1 Problem statement

The regular practice of physical activity brings numerous benefits to health. Despite that, many people ignore this reality, either due to lack of time or simply out of laziness (Sequeira, 2024). Increasing physical fitness can prevent heart problems, obesity, diabetes, among other benefits, in addition to improving mood and self-esteem and reducing the risk of depression and anxiety. This practice can be done by anyone of any age (Wellhub Editorial Team, 2024).

According to the 2017 Eurobarometer of the European Union, 68% of Portuguese people do not engage in any physical activity, making this a significant area for growth for the sector, which has set the goal of reaching one million participants in physical activities in gyms by 2025 (Principal, 2018).

In Europe, in 2017, it was estimated that 59.98 million people were attending gyms, and this number is increasing (Gough, 2021). Due to the high number of gym-goers and the unavailability of personal trainers, it becomes more challenging to provide continuous guidance to these individuals, users may perform exercises that do not match their needs or abilities, making it necessary to use applications that can replace a personal trainer.

The goal of the developed recommender system is to assist users in their daily exercise routines, whether at the gym or at home, by providing tailored exercise recommendations that suit their individual fitness levels, goals, and needs.

1.2 Objectives

The main objective of this development is to answer the following question “How can hybrid techniques in a recommender system be utilized to improve health and fitness outcomes?”.

For that reason, the primary goal is the development of a hybrid recommendation system that is able to assign an exercise routine according to the customer's needs and profile. The use of this system should be intuitive and easy so that anyone can benefit from it.

To tackle this challenge, specific steps must be achieved. Firstly, we must perform a literature review to study existing applications in the market and the usefulness of this type of application in the field of health. This step will allow us to understand what good practices exist and the gaps that should be addressed.

Following, with the objective of providing personalized recommendations, it’s important to identify different user profiles to understand different needs and preferences for a greater level of tailoring. More so, to perfect the system it’s also necessary to track and register user data and their goals, in other to fine-tune the recommendations.

Critical to the development of this recommender system is the study of physical exercises that can be done at home, supporting the need for alternatives to the gym. The system should consider the specific requirements of each exercise to ensure that users receive training plans tailored to their profile and goals, promoting accessibility regardless of their available equipment.

Additionally, since the main objective encompasses general well-being, studying and defining a correct and healthy diet is a critical, as is studying and incorporating therapeutic exercises to tackle muscle discomfort, offering a complete solution.

Finally, a key objective of this development is to provide a system accessible on various platforms, such as desktop and mobile, ensuring a user-friendly and interactive experience, that leads users to engage with the application and achieve their goals.

1.3 Contributions

This thesis aims to contribute to the field of digital health, specifically health recommender systems applied to physical activity. The expected outcomes are anticipated to address both technical advancements and practical applications in well-being.

An expected contribution is an enhanced understanding of user profiles, engagement, and satisfaction. This project will provide insights into user preferences, behaviors, and needs, as well as assess the impact of the recommender system on user engagement, satisfaction, and adherence to health and physical activity recommendations.

The implementation of the recommender system is anticipated to be an example of the capability that these types of systems have in integrating user data and provide tailored recommendations to users. Furthermore, given the sensitive nature of this information, the project will also provide insights on how it can be done while preserving user privacy and an ethical use of user data.

The integration of the recommender system into an intuitive multiplatform solution is expected to contribute to the ways in which health and fitness applications can provide different types of users with home-based exercise and diet management, amplifying health technology offerings and providing alternatives to non-technological approaches.

In conclusion, this thesis is expected to significantly advance how health applications with recommender systems can promote healthier lifestyles, offering insights not only to the development community but also for public health and individual well-being.

1.4 Ethical Considerations

The development of a recommender system raises ethical challenges, in particular, a recommender system for a health application. This section brings light to some ethical challenges and principles that were taken in consideration for the development of this system.

User data is a central asset of recommender systems, handling such information requires a thoughtful approach. For this reason, we align our practices with the General Data Protection Regulation (GDPR) to ensure user privacy and protection (European Parliament, 2016)

Clear communication with users about how their information is used to provide a personalized experience is crucial to ensure the system's transparency and explainability. Only with this approach is it possible to build a trustworthy system that allows the user to feel comfortable with the information provided by the recommender system (Albahri *et al.*, 2023).

A crucial step in the development of a recommender system is extensive testing, to validate accuracy and reliability of the information provided and mitigate the risk of misinformation, which represents another ethical concern (Fernandez and Bellogín, 2020).

Another priority is the user's well-being and optimal health outcomes, for this purpose a user-centric approach is adopted. This involves designing the system to ensure that recommendations are not just technically correct but also beneficial and safe, something that is essential in health technology (Petersen *et al.*, 2022).

In conclusion, these ethical considerations ensure the development of a valuable recommender system that upholds ethical integrity in health applications and confirms compliance with the polytechnic code of ethics as well (Diário da República, 2021).

1.5 Schedule of activities

To develop a recommender system and its documentation, an organized approach is required. For that purpose, **Table 1** was elaborated, the project will be divided into six distinct phases, each with activities and goals to achieve. The objective of this schedule of activities is to layout the necessary steps to ensure a methodical and well-planned approach to build a high-quality solution.

Table 1: Schedule of Activities

Phase	Dates	Activities
Project initiation and technology selection	February 15 – March 15	Initial drafting of the conceptual framework for the recommender system
Requirements Analysis and System Design	March 15 – April 15	Functional and non-functional requirements analysis
		Recommender system specification
		High-level design documentation
Prototype Development	April 15 – June 15	Design data models and algorithms
		Development of recommender system prototype
		Test basic recommendation functionalities
		Refine recommender system
System Integration	June 15 – July 15	Application back-end development
		Recommender system integration
System Evaluation	July 15 – August 5	System performance testing
Documentation	Continuous	Documentation of the system

The first phase, the project's conceptual framework will be drafted to understand what techniques to apply to build the recommender system and what technologies are must suited to implement this system.

The second phase combines requirements analysis and system design. The objective is to outline functional and non-functional requirements to determine what the system needs to accomplish and to understand what type of architecture better accommodates the system's needs.

The Prototype Development phase involves designing the data models and determining what algorithms to implement in an earlier stage. Following this, the recommender system is developed and tested for its capabilities, with refinements made to enhance its accuracy and reliability.

The fourth phase involves developing the server-side application and integrating the recommender system. This phase also involves making the system capable of receiving and storing user information.

The System Evaluation phase assesses the performance of the recommender system and evaluates how well the application meets the requirements defined during the second phase.

Finally, the Documentation phase is a continuous phase throughout the project's development. It involves recording all aspects of the system, including design decisions, implementation details and testing results.

1.6 Document Structure

This thesis focusses on the several steps required to develop a health application that encompasses a recommender system. This section outlines the structure of the thesis and of each sections purpose and contents.

The initial section serves as an introduction, starting with background information and context. This is followed by the problem statement that outlines the specific issues the system intends to address. The following chapter focusses on the objectives of the thesis and system to resolve the identified challenges. A separate chapter highlights the benefits of this thesis to the field of research and expected outcome, followed by chapter on ethical considerations that explains the challenges in the development of a health recommender system from an ethical point of view. The last chapter is the schedule of activities that illustrates the organization of the necessary phases to complete this thesis.

The second section encompasses a literature review process, providing insight into the state-of-the-art in health recommender systems, highlighting the most recent developments and challenges that relate to this project.

The third section of this document is about the system development and the phases necessary for a correct development process. This includes, requirement analysis, system design, prototype development and system integration.

The fourth section describes the implementation, detailing the development process and the algorithms used to build the recommender system and the server-side application

The fifth section focuses on the evaluation, detailing the testing metrics applied and providing an analytical review of the results produced by the system.

The sixth section summarizes the work, the findings and contributions to the field of health recommender systems, along with considerations for future development.

2 Literature Review

In this chapter, we explore the implementation of recommender systems in health applications, with the intent of highlighting advantages and disadvantages.

Nowadays, an extensive amount of health-related data exists, scattered across different platforms, this can be overwhelming and unproductive to improve the well-being of the general audience (Ngoc *et al.*, 2020). Consequently, the use of artificial intelligence systems in the medical process is growing. These systems aid in several critical areas, such as early detection, diagnosis, and, as this study focus on, personalized health solutions (Ali *et al.*, 2023).

In this review, we will examine critical concepts relevant to a health recommender system, with a focus on their effectiveness and ethical dimensions. This involves, exploring the outcomes of using health recommender systems, assessing how these systems ensure user privacy and protection, if the user well-being is the priority of the development, understanding what ethical challenges arise and how they are handled, examining the adherence the general audience has to digital health solutions and what algorithms can optimize well-being.

To this end, four research questions have been formulated with the intent of examining these critical concepts. The first question focusses on the effectiveness of health recommender systems, with the second question we explore data privacy and ethical considerations, the third question allows us to investigate adherence and reception of health digital solutions by users and the fourth question brings clarity about what types of recommender system exist and which are more suitable for health applications.

This literature review is divided in seven subsections: the research questions that conduct the investigation, the data sources that were selected to collect relevant articles, the search query string was used in the data sources, the eligibility criteria, the information extraction process following the PRISMA method, the contribution of retrieved articles to answer the research questions and, in the final section, the conclusions drawn from this study and the identification of gaps to be addressed by this project.

2.1 Research Questions

As mentioned previously, four research question were formulated to conduct this literature review and the PRISMA 2020 (Page *et al.*, 2021) method has applied to identify and evaluate research articles relevant for those questions.

RQ1. How effective are health applications that incorporate a recommendation system to enhance well-being through physical activity in comparison with non-technological solutions?

RQ2. How is user data protected and what efforts are made to ensure an ethical use of information in health applications that incorporate a recommendation system?

RQ3. What is the motivation for users to adopt health applications to improve well-being and how can these applications provide guidance?

RQ4. What type of recommendation system is more suitable for a health application that focuses on physical activity and dietary recommendations?

2.2 Data Sources

The data sources utilized in this literature review are presented in **Table 2**.

Table 2: Data Sources

Data Source	URL
ACM Digital Library	https://dl.acm.org/
ScienceDirect	https://www.sciencedirect.com/
B-on	https://www.b-on.pt/
Google Scholar	https://scholar.google.com/

To choose data sources it is necessary to consider their credibility, scope and relevance to the field of research. The data sources present in Table 2 were selected because they are widely recognized and peer-reviewed repositories by the scientific community.

2.3 Search Terms

The keywords for the search query are health applications, e-health, recommendation systems, physical activity, nutrition, exercise, user adoption, privacy and data protection.

Combining these keywords with the OR operator and with the AND operator the following search query was created ("health applications" OR "e-health") AND ("recommendation systems") AND ("physical activity" OR "nutrition" OR "exercise") AND ("user adoption" OR "privacy" OR "data protection").

The objective of this search query is to refine the search results to include research articles that contain information related to healthcare supported by information technology. Specifically, articles that include the use of recommender systems, with connections with physical activity and nutrition and discuss aspects such as user adoption and privacy.

This query has applied to all data sources.

2.4 Quality Assessment

To ensure a relevant review, eligibility criteria were implemented.

Inclusion criteria:

IC1. The source must have references to the field of Health recommender systems
IC2. Source is peer-reviewed
IC3. The source provides valuable insights to the key concepts of health recommender systems
IC4. The source examines recommender systems in effectiveness, user experience or other impact metrics relevant to health
IC5. The source includes data privacy or ethical considerations in the scope of recommender systems

Exclusion criteria:

EC1. The source is older than 3 years
EC2. The source is a book, conference paper or a survey
EC3. The source is not written in English
EC4. The source focus on recommender systems applied to another field of studies
EC5. The source studies the application of artificial intelligence solely in wearable devices and their performance

With these criteria it's possible to select which articles are relevant and robust to answer the research questions.

2.5 Data Extraction

The data extraction process was accomplished following the PRISMA method (Page *et al.*, 2021). **Figure 1** represents the flow diagram of this method.

This method consists of three steps:

Identification step consists in extracting only the articles that aren't older than 3 years and removing non-academic material and duplicates. This step resulted in 234 records to be screened.

Screening is composed by two phases, abstract screening and relevance re-assessment. In the first phase it's possible to eliminate irrelevant articles by reading the abstract and in the second phase possibly relevant papers are screened for more information. After this process, 22 reports were assessed for eligibility.

Eligibility in this phase the articles were read to search for information related to the Research Questions. After this process, 20 articles were included in this review.

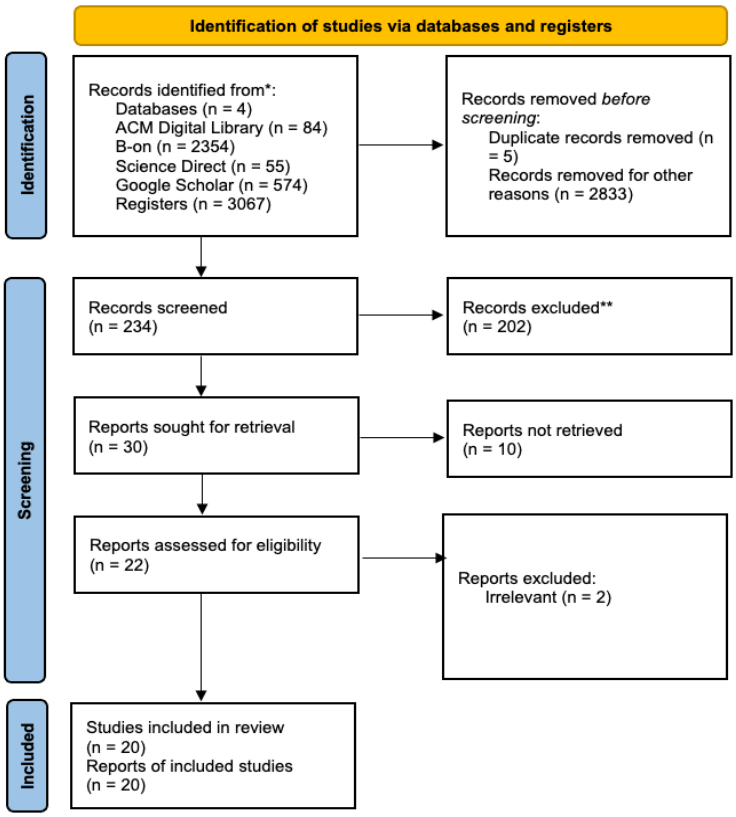

Figure 1: PRISMA Flow Diagram

2.6 Results

2.6.1 RQ1: How effective are health applications that incorporate a recommendation system to enhance well-being through physical activity in comparison with non-technological solutions?

From the studied articles, ten showcase how a recommender system can be implemented to provide tailored recommendations to users resulting in behaviour change, increased well-being and offering an alternative path to better health outcomes.

(Vairavasundaram *et al.*, 2022) and (Sengan *et al.*, 2021) describe recommender systems that support a physical activity planner to help users achieve a step count goal for the day.

Both systems aim to monitor and classify user behaviour with the help of tracking devices, providing accurate recommendations based on the applications' user classification methodology and predictive models.

(Sartori, Savi and Talpini, 2022) describes a system to help people aged 35-45 years with a sedentary lifestyle to promote behaviour change. This application tailors recommendations based on users psychological and physical characteristics and integrates Behaviour Change Theories to enhance goal setting and self-monitoring of user behaviour.

(Liu *et al.*, 2023) is another system that provides personalized fitness recommendations, but this system focus on speed, heart rate and workout distance recommendations for a user's input target calories and specific type of sport like running or biking. The systems' accuracy was evaluated with a real-world data set and produced accurate predictions.

(Orue-saiz, Kazarez and Mendez-zorrilla, 2021) and (Samad *et al.*, 2022) are systematic reviews that highlight the inefficiencies and areas of improvement of nutritional recommender systems and food consumption tracking apps. In (Samad *et al.*, 2022) review it attributes the unsatisfactory results to the lack of food recognition features, the scarce use of food databases and the lack of evidence-based strategies.

(Hinojosa-Nogueira *et al.*, 2023) describes the development of an effective personalized nutrition recommender system. This application uses several data sources, expert knowledge, user data and image processing to provide the user with healthy menus. After testing with subjects, the app achieved an average quality score of 3.97/5, leading this study to consider the app a reliable tool for nutrition recommendation and a leap in this field.

(Caballero *et al.*, 2021) is an article about a recommender system that tracks and notifies users in real time about pollen counts to which the user is allergic, resulting in 90.5% of the people who tested the application to agree that it was useful to anticipate their symptoms.

(Coppens, De Pessemier and Martens, 2024)is an article that demonstrated the effectiveness of incorporating recommendation techniques, such as mood micro-profiling, content-based recommendations, and contextual post-filtering, by using two groups. The experimental group received personalized recommendations using these techniques, while the control group received only partially personalized suggestions through random selection. The study found that the experimental group experienced higher satisfaction, increased physical activity, and lower dropout rates, highlighting the effectiveness of recommendation systems in enhancing well-being.

(Coppens, De Pessemier and Martens, 2023) is an article that studied the impact of adding contextual and motivational variables to health recommender systems. The study's findings suggest that incorporating these features in health recommender systems leads to more relevant physical activity suggestions, improving user adherence and contributing to positive health outcomes, indicating their effectiveness in promoting behavior change.

Although the articles demonstrate that recommender system techniques in health applications can enhance physical activity and well-being, this literature review reveals a lack of evidence on their long-term effectiveness and a direct comparison with non-technological approaches.

2.6.2 RQ2: How is user data protected and what efforts are made to ensure an ethical use of information in health applications that incorporate a recommendation system?

All the articles studied in this literature review mention data privacy concerns, expressing at some point that due to the high volume of sensitive data being utilized by recommender systems, factors like transparency and data protection are crucial for the implementation of such a system. In six articles, an in-depth analysis is made on how this sensitive information can be protected and ethically used.

(Sengan *et al.*, 2021) and (Liu *et al.*, 2023), describe recommender systems that were developed having user privacy as a top priority. (Sengan *et al.*, 2021) proposes a model that employs Homomorphic Encryption to protect the user's information, with this method the system can give an accurate recommendation based on user context without compromising user privacy. (Liu *et al.*, 2023) is a system that seeks to minimize identity information from users, focusing on the use of sensory data from wearable Internet of Things devices. It also uses a multi-level deep learning framework for generating fitness recommendations that structures data analysis across multiple levels ensuring a greater degree of privacy.

(Wieczorek *et al.*, 2023), (Sun *et al.*, 2023) and (Valentine, D'Alfonso and Lederman, 2022) are articles that highlight the need for transparency and user consent for the handling of user's sensitive data and discuss the hard balance between personalization and privacy.

(Wieczorek *et al.*, 2023) is a systematic review that gives a broader view of the ethical considerations that should be considered in the field of retrieving information from self-tracking

devices. It discusses concerns on the ownership of the data collected, the misuse of data, the challenges of interpreting data and advocates for the need of further rules and regulations to protect users.

(Sun *et al.*, 2023) this study focuses on the use of social media data in health recommender systems, highlighting that this practice raises privacy concerns and diminish user trust. It concludes that users have higher trust in recommender systems that are based on their own data rather than influenced by social media data.

(Valentine, D'Alfonso and Lederman, 2022) this paper describes how digital mental health app can be enhance by incorporating recommender systems, but raises concerns of the ethical use of information, advocating that explainability and transparency play a crucial role in balancing personalization with privacy concerns.

(Albahri *et al.*, 2023) is a systematic review that covers the ethical challenges to produce trustworthy and explainable Artificial Intelligence in healthcare.

This article states that:

- Responsible and Ethical use of sensitive data
- Implementation of regulations and ethical rules
- Data security, privacy and social acceptance
- Transparency and Reduction of Bias

Are crucial for the long-term sustainability in healthcare artificial intelligence.

In conclusion, the reviewed articles emphasize the importance of data privacy and ethical considerations in information handling that supply recommender systems. The implementation of methods like Homomorphic Encryption and the use of non-identifiable data are measures that serve as an example of what can be done to achieve a better privacy protection. Establishing ethical rules like transparency, user consent, explainability and having a good balance between privacy and personalization are key factors to build robust recommender systems.

2.6.3 RQ3: What is the motivation for users to adopt health applications to improve well-being and how can these applications provide guidance?

In the selected articles, three systematic reviews ((Cai *et al.*, 2022)(Ali *et al.*, 2023); (Pinto *et al.*, 2023)) discuss the fact that:

- user-friendly interfaces
- personalized recommendations
- The ability to induce behaviour change
- Noticeable health benefits
- Engaging content
- Aid in the decision-making and self-management process

Are key factors incorporated by health applications that successfully produce positive health outcomes, highlighting that these factors are what motivate users to adopt these applications.

(Valentine, D'Alfonso and Lederman, 2022) explores the advantages that a recommendation system for mental health can provide to its users. This article states that reducing the choice overload by giving select suggestions and establishing a digital therapeutic relationship could lead to greater user engagement and that aiding with self-management is essential for user motivation.

The application described in (Hinojosa-Nogueira *et al.*, 2023) concluded from questionnaires applied to the test subjects that quality and credibility of the information and aesthetics were the most valued aspects by users. This positive outcome resulted in an 88% recommendation rate by users.

(Sartori, Savi and Talpini, 2022) the data collection about the user experience of the application described in this article reported that improvements to their user interface and content were necessary for improving the user's adoption of the application.

(Caballero *et al.*, 2021) is an article about an application that notifies users in real time of the pollen count in their area to which they are allergic. The users of (Caballero *et al.*, 2021) reported that what they found most interesting in the application was the information about pollen levels, the frequent symptoms alert from other users with the same allergies and the alerts regarding the increasing pollen concentration.

(Coppens, De Pessemier and Martens, 2024) demonstrates that incorporating recommendation techniques like mood micro-profiling, content-based recommendations, and contextual post-filtering significantly boosts user engagement. The study used an experimental design where one group received fully personalized recommendations, while the control group only received partially personalized suggestions. The findings highlight that the personalization in the experimental group led to greater user satisfaction, engagement and a smaller dropout rate.

In summary, the reviewed articles concluded that the users' motivation to adopt health application relies on the application proven effectiveness, user friendly interface, engaging content, decision-making aid, self-management improvement and timely information.

2.6.4 RQ4: What type of recommendation system is more suitable for a health application that focuses on physical activity and dietary recommendations?

In 12 articles that described a recommender system or studied different types of systems, 8 concluded that hybrid type was the most appropriated.

The (Liu *et al.*, 2023) system uses a Multi-Layer Perceptron architecture to process contextual user data and predict a total workout distance, followed by a Multi-layer Bidirectional Long Short-Term Memory to predict the heart and speed sequences.

(Valentine, D'Alfonso and Lederman, 2022) and (Sartori, Savi and Talpini, 2022) propose hybrid recommender systems that incorporate content-base and collaborative filtering aided by context-aware information provided by wearables to process user data and supply recommendations.

(Caballero *et al.*, 2021) is a hybrid recommender system that uses the user's contextual information, such as location and physical activity, along with collaborative filtering from the symptoms of users in the same location with the same allergy to help with prevent worsening of the symptoms.

(Mustaqeem, Anwar and Majid, 2020) this system performs content-based filtering by utilizing patient attributes and medical records to form clusters and sub-clusters of cardiovascular diseases. After, it performs collaborative filtering utilizing the k-nearest neighbour approach to generate recommendations.

(Vairavasundaram *et al.*, 2022) is a hybrid recommender system that utilizes the random forest algorithm to classify users based on information gathered from fitness trackers (context-aware) and then provides recommendations utilizing Long Short-Term Memory technique. Additionally, if the probability of meeting goals is low, the system uses a Linear-Quadratic Regulator to generate an alternative activity plan.

(Ngoc *et al.*, 2020) is a systematic review of recommender systems in the healthcare domain, that focus on various types of recommender system, including food and physical activity recommendation. On the studies encompassed by this review, the most common type for both areas were hybrid systems, namely system that incorporated context-aware, content-base and collaborative filtering.

Contrastingly, the systematic review in (Cai *et al.*, 2022) states that a knowledge-based approach not only provided better recommendations for users but also was less exposed to drawbacks like cold-start and rating sparsity that are common in collaborative filtering (Jangde and Ramaiya, 2022).

Overall, in the reviewed articles, a hybrid approach was the most suitable option to elaborate recommender systems with less drawbacks, using various methodologies to compensate eventual flaws of the algorithms. The choice of the best recommendation system and what combination of approaches should be used relies deeply on the user needs and nature of the health application.

2.7 Discussion

This literature review provided valuable insights on health recommender systems. Various articles enabled the study and assessment of the potential of personalized healthcare and its various applications. However, most articles lacked a significant study on long-term effectiveness and a direct comparison with non-technological approaches, leaving doubt on the impact these solutions can really have in improving lasting well-being.

Another gap identified was the lack of technical solutions that addressed the issue of data protection, with only two articles focusing on concrete solutions to ensure user privacy. Even though all articles expressed data privacy concerns and ethical considerations, a significant number provided only a superficial analysis on these important topics. Despite this, (Albahri *et al.*, 2023) gives an extensive explanation on the best practices to develop user-centric, transparent and explainable artificial intelligence systems, which represents a positive indicative on raising awareness to this topic and providing guidance.

The topic of the user's motivation and adherence to health digital solutions provided another gap. Although some approaches that theoretically increase users' reception to personalized healthcare and raise engagement were studied, there was a lack of evidence on how these strategies can be amplified to raise the user's motivation on using health applications and a lack of user experience design study to keep the user engaged and committed.

From a constructive perspective, on the topic of what type of recommender systems are more suitable for a health application, the conclusion was that a hybrid approach is the most appropriate. In the articles that conclude this, the common reason was that due to the complexity of recommender systems and the flaws that each algorithm produces, a hybrid approach allows to mitigate these flaws and develop a robust recommender system.

Therefore, our project seeks to bridge these gaps by developing a recommender system that aggregates three modules: physical activity, nutrition and therapeutic exercises. With this development we strive to showcase how a recommender system can be implemented with considerations of user well-being and data protection, exemplify a user experience design study to tailor the application to be engaging, produce an algorithm that provides accurate recommendations and present an analytical review on the effectiveness of the solution. These approaches will be pivotal in producing a document that contributes to the evolution of digital health solutions.

2.8 Summary

The primary objective of this thesis is to document and evaluate the development of a health application with a recommender system applied to physical activity. This study is expected to represent a significant contribution to the field of health digital solutions and an example of an ethically correct development.

To that end, a literature review was conducted using the PRISMA method with the objective of bringing clarity to the questions surrounding the key concepts of a recommender system. Several gaps were identified: the absence of a long-term study on the effectiveness of health recommender systems, there is a need for more robust examples of technical implementations to ensure data protection, the lack of practical examples that illustrate ethical considerations, and an undervaluation in the design of the user experience.

Although the review identified significant gaps, such as lack of long-term effectiveness assessment, no comparisons with non-technological methods and a reduce number of concrete solutions to ensure user privacy, it also provided guidance on the key questions. Several articles offered comprehensive explanations on ensuring data protection, adopting a user-centric approach and maintaining transparent and explainable development. Others focussed on the flaws inherent in different types of recommender systems, leading to the conclusion that a hybrid recommender system produces the most reliable system. Additionally, it was possible to study various applications of personalized healthcare and their short-term impact.

This review will guide the process of development, highlighting gaps to address, consequently amplifying the thesis' contributions. From this review questions emerge that help conduct future work: what technical implementations should be used to ensure data protection, what user experience design produce the most engagement, what testing methodologies will allow us to ensure the accuracy of our system and mitigate the risk of misinformation, what efforts should be made to have a practical example of ethical considerations and promote transparency, and what algorithms should be used to develop an adequate hybrid recommender system.

In conclusion, this project will result in a system that provides tailored recommendations in three different health modules, something that wasn't found in the literature review, and for that reason, it will provide valuable insights to ethical health technology development.

3 System Requirements and Design

In this chapter, we will define functional and non-functional requirements as well as actors that outline what the hybrid recommender system must achieve to fulfil its intended functionalities and quality attributes. Additionally, this chapter presents the design of the system, detailing the key components and their interactions to achieve the system's objectives.

3.1 Actors

"An actor in use case modelling specifies a role played by a user or any other system that interacts with the subject."(Paradigm, 2024).

For this thesis three actors were identified:

- **User**: The person that interacts with the application and obtains exercise recommendations from it.
- **Recommender System**: The system is responsible for receiving and storing the user's information and use it to provide exercise recommendations.

3.2 Functional Requirements

“These are the requirements that the end user specifically demands as basic facilities that the system should offer.(…)”(GeeksforGeeks, 2024).

The core functionalities of this system will be represented using use case terminology to provide a representation of the system’s functional requirements. **Figure 2** illustrates the use case diagram that encompasses these functionalities.

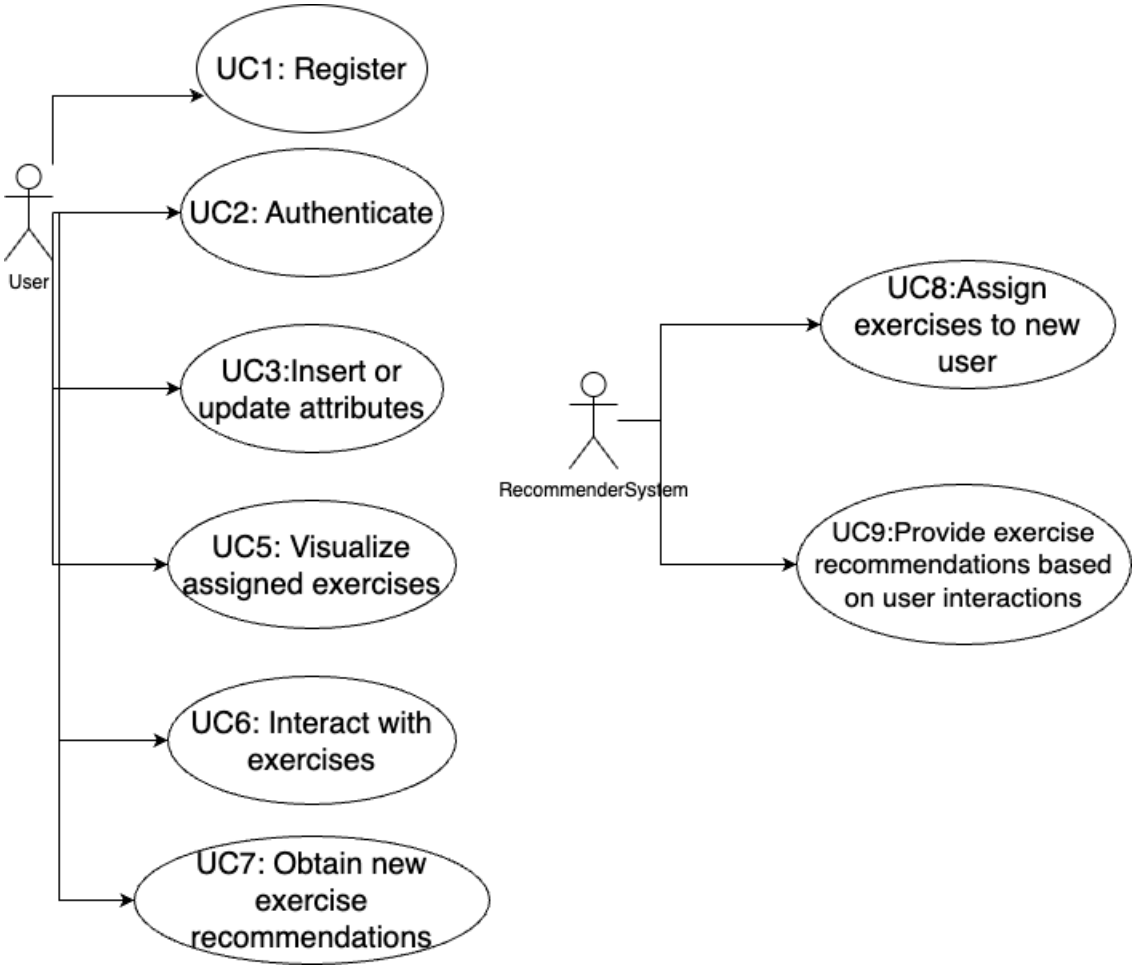

Figure 2: Use Case Diagram

- **UC1: The user must be able to register:**
  The user must be able to provide a username, email and password to register himself on the application.

- **UC2: The user must be able to authenticate:**
  After successful registration, a user must be able to authenticate in the application by providing their e-mail and password.

- **UC3: The user must be able to insert or update his attributes:**
  The system's ability to provide exercise recommendations is directly tied to the user's attributes like height, weight and fitness goals. To achieve this, the user must be capable to provide this information.

- **UC4: The user must be able to visualize assigned exercises:**
  After the user onboarding, the system will provide a set of exercises based on the user's attributes. The user must be able to visualize his new exercises on the application.

- **UC5: The user must be able to interact with exercises:**
  The application must allow users to engage with exercises by providing two interactions, liking and completing an exercise.

- **UC6: The user must be able to obtain new exercise recommendations:**
  After interacting with the system, the user should be able to request other additional exercise recommendations beyond the initial ones provided.

- **UC7: The recommender system must be able to assign exercises to a new user:**
  When a new user registers and hasn't yet interact with the exercises, the system must provide an initial set of exercises based on the information provided.

- **UC8: The recommender system must be able to provide exercise recommendations based on user interactions:**
  With interaction data available, the system should be able to recommend other exercises based on users' preferences.

3.3 Non-Functional Requirements

Non-Functional Requirements are defined as criteria that can be used to ensure broader quality attributes, providing guidelines on how the system should perform the functional requirements. In this section we will specify the criteria used(Scaled Agile, 2023).

3.3.1 NRF1: Security

When dealing with an information system, it is crucial to maintain the security of users' information. This includes implementing an authentication process to ensure that each user can only access their own information. Additionally, an authorization mechanism is crucial so that each user is only permitted to perform actions he is specifically allowed. Lastly, confidentiality must be preserved by encrypting user data and always protecting it from unauthorized access.

3.3.2 NRF2: Performance

Performance is critical for any information system because it directly impacts the user experience and system efficiency. The system must be able to process user requests and generate recommendations in a short amount of time and without errors.

3.3.3 NRF3: Usability

The usability of an application is directly tied to its success and adoption. The application must offer an intuitive and user-friendly interface, along with a responsive design that will allow users to use the application both on the desktop or on their mobile devices. Accessibility considerations should be integrated, enabling the application to be used by individuals with disabilities. Additionally, the application should support multiple languages, allowing users to interact with the application in their preferred language. Lastly, regular usability testing should be conducted to identify and address any potential issues.

Given that this system is a prototype, these three non-functional requirements will be the focus. In a more advanced stage of development, it will be important to consider other crucial non-functional requirements like **scalability, reliability** and **maintainability.**

3.4 System design

This section begins with the domain model, which provides a conceptual framework outlining the primary entities and their relationships. Following this, the system architecture is presented, illustrating how different components are integrated. Lastly, for each use case identified in the functional requirements section, it will be presented a sequence diagram that illustrates how the system's components interact.

3.4.1 Domain Model

The domain model lays the conceptual framework for the system domain, providing a high-level view of the entities, their attributes, and their relationships(IBM, 2021). The diagram presented in **Figure 3** captures the essential aspects of the domain and serves as a guide for the design and development processes.

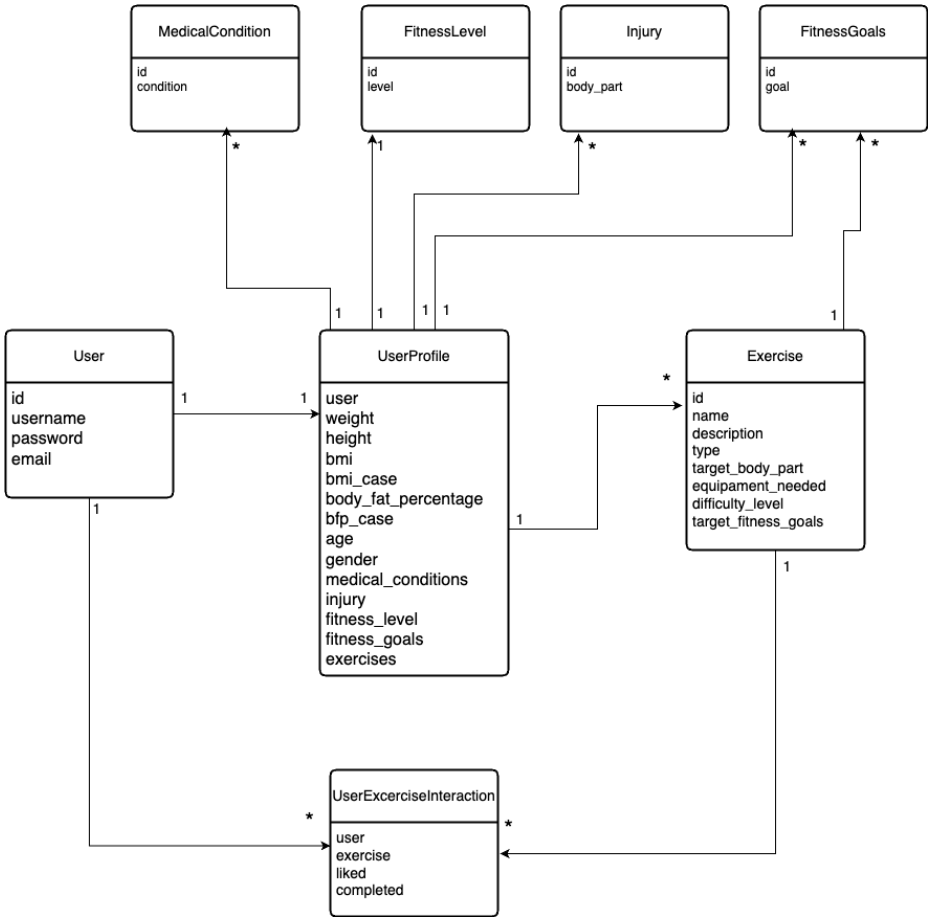

Figure 3: Domain Model

The **UserProfile** entity serves as the primary entity around which other entities are structured. This entity represents individual users of the system, storing relevant personal information that influences exercise recommendations, such as height or body mass index (BMI). It also captures specific user conditions through its relationships with other entities:

MedicalConditions represents the user's medical conditions that may affect their exercise routine.

Injury captures the body parts affected by injuries that need to be considered when recommending exercises.

FitnessLevel indicates the difficulty level of exercises that the user is capable of performing.

FitnessGoal reflects the goals the user is aiming to achieve.

The **Exercise** entity helps establish the relationship between the user and the exercises assigned to them. This entity stores detailed information about each available exercise's attributes, including the target body part and difficulty level. It is also linked to the **FitnessGoal** entity to indicate what each exercise is best suited to achieve.

The **UserExerciseInteraction** entity captures the interaction between users and the exercises they engage with, recording wether the user liked or completed an exercise. This entity is crucial for understanding user preferences and providing refined recommendations.

3.4.2 System Architecture

"Software architectural design is the process of applying various techniques and principle for the purpose of defining a (…) system in sufficient detail to permit its physical coding"(Mekni et al., 2017). In this section, the system architecture is outlined. It serves as the framework that defines the structure of the system and how its components interact with each other to meet both functional and non-functional requirements.

Figure 4 is a container-level diagram that depicts the monolithic architecture of the system. The decision to adopt a monolithic architecture for this system is based on the simplicity and reduced development time associated with this approach. Since the objective is to create a prototype where the primary goal is to implement a hybrid recommender system, the advantages of choosing a monolithic architecture outweigh the disadvantages. Monolithic systems are easier to test and deploy(Blinowsk, 2022).

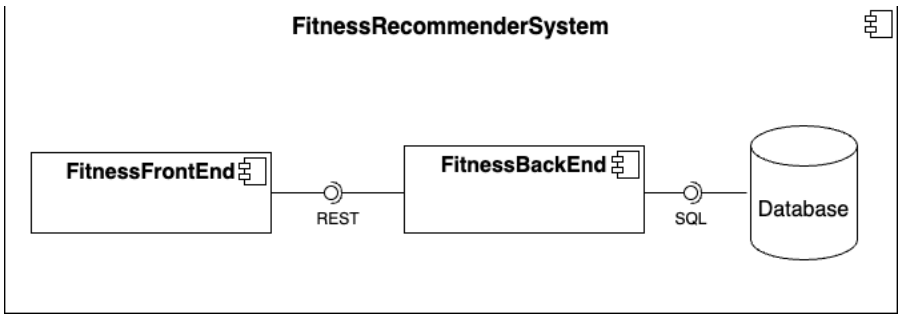

Figure 4: FitnessRecommenderSystem container diagram

FitnessFrontEnd is the client-side application that provides the user interface for the recommender system. It is responsible for displaying exercise recommendations, collecting user inputs, and communicating with the server-side application via REST calls to retrieve data and submit user information. Its primary goal is to offer an intuitive and responsive user experience.

FitnessBackEnd is the server-side application that handles business logic, data processing and communication with the database. It is also the module where the content-based and collaborative filtering algorithms are implemented, generating exercise recommendation based on user's preferences and past interactions.

Database is the storage component of the system, where all data is persisted and securely stored.

However, it must be noted that this type of architecture possesses challenges in a real-world scenario as the application grows. It becomes increasingly difficult to maintain and evolve the code base, due to the tight coupling changes in one module can result in a cascade of errors. Scalability is also limited, and deployment times can increase significantly. Ideally, this system would evolve into a microservices architecture, where the *FitnessBackEnd* would be decoupled, resulting in multiple advantages such as making it more fault-tolerant and easier to maintain and scale(Dragoni *et al.*, 2017).

Figure 5 is a component-level diagram of *FitnessBackEnd* it presents a view on the several components our server-side application has and how they interact with each other.

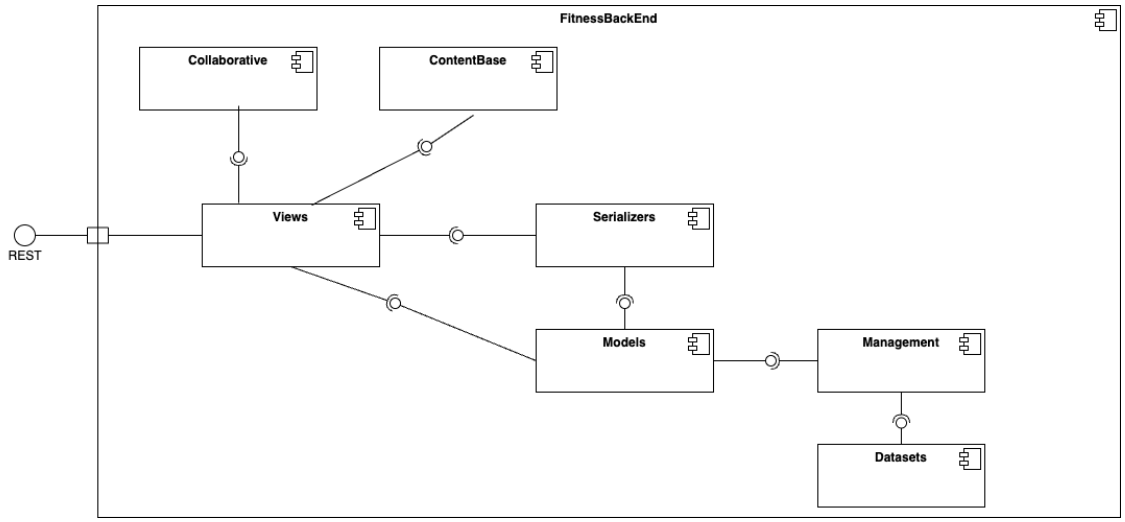

Figure 5: FitnessBackEnd component diagram

Models component define the structure for data models used within the system. These models define how entities are stored, accessed, and manipulated within the database.

Dataset component represents the external data sources that are utilized by the system.

Management component includes custom commands that perform essential tasks such as importing users and exercises.

Serializers component are responsible for converting complex data types into JSON and vice-versa, ensuring that data is validated and structured correctly.

Views component handles incoming HTTP requests and processes them using business logic encapsulated in **Serializers** and **Models**, it is the entry point for the client-side application to interact with the server-side, enabling functionalities like user profile management or exercise recommendation requests.

ContentBase component is where the content-based filtering algorithm is implemented. This algorithm generates recommendations based on users' attributes and is used for new users to address the cold-start problem of recommender systems.

Collaborative component implements the collaborative filtering algorithm, which generates recommendations based on the past interactions of users.

These two algorithms are used in the **Views** component to answer to exercise recommendation requests.

3.4.3 Functional Requirements Design

This section will detail the sequence diagrams that represent the system behavior to fulfill use cases. These diagrams visually represent how various components of the system interact to achieve the desired functionalities, giving a better understanding of the system's internal processes.

3.4.3.1 UC1

The sequence diagram in **Figure 6** represents the first use case, where a user registers in the application. The process begins with the user providing the necessary information, such as username, email, and password, through the client-side application. Once the user submits this information, the client-side application sends a create request to the server's UserViewSet. This view handles the request by invoking the CreateUserSerializer, which validates the input data. Upon successful validation, the serializer proceeds to create a new user by calling the UserModel and a corresponding user profile by calling the UserProfileModel. Both models interact with the database to store the new user and user profile. After successful completion, a response is sent back to the client-side application, confirming the user's registration.

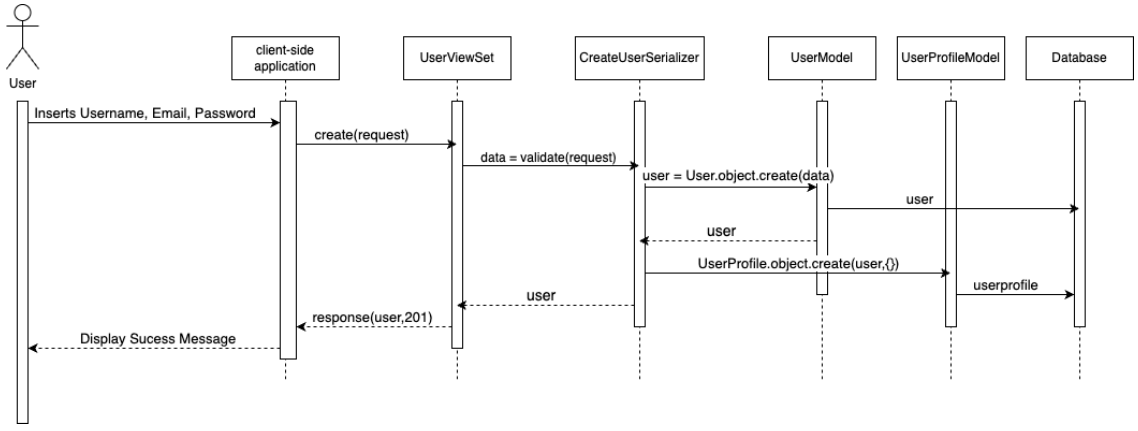

Figure 6: Sequence Diagram Use Case 1

3.4.3.2 UC2

The sequence diagrams in **Figure 7** and **Figure 8** represent the second use case, the user's authentication process. Figure 7 represents the login process, which begins when the user provides their credentials through the client-side application. The client-side application sends a login request to the server's TokenObtainPairView. This view verifies the user's credentials using the Authentication System. Upon successful validation, the view returns a response to the client-side application containing the generated access token and refresh token.

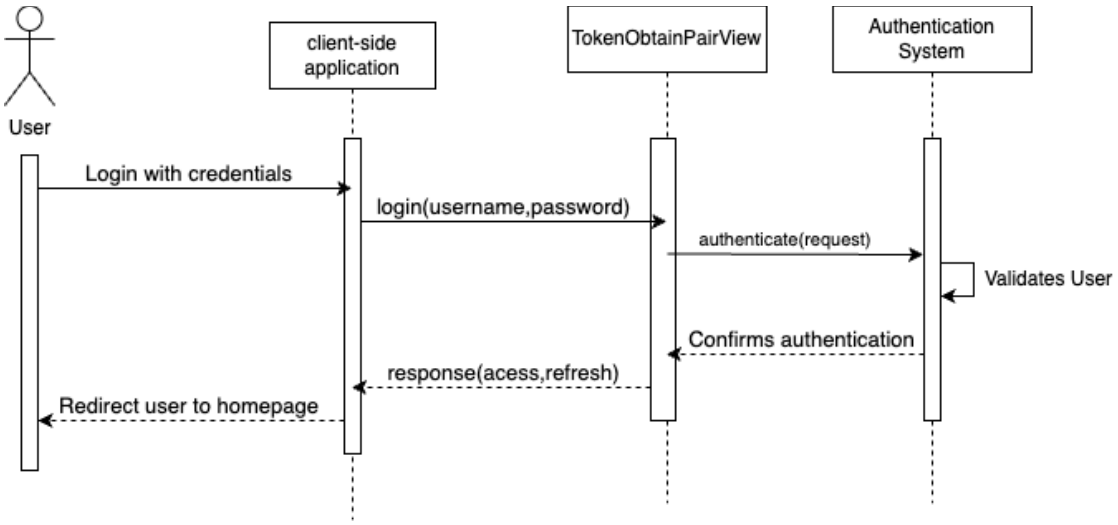

Figure 7: Sequence Diagram 1 Use Case 2

Figure 8 illustrates the token refresh process. Periodically, the client-side application sends a refresh token request to the TokenRefreshView. The view validates the refresh token, and if successful, returns a new access token to the client-side application. The token is then stored by the client-side application to ensure the session remains valid without requiring the user to log in again.

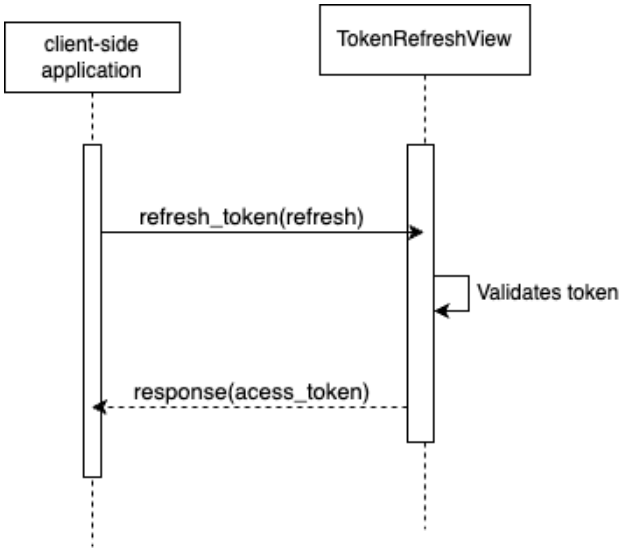

Figure 8: Sequence Diagram 2 Use Case 2

3.4.3.3 UC3

The sequence diagram in **Figure 9** represents the third use case, where the user can insert or update their attributes in the system. The process begins when the user inputs their new or updated attributes through the client-side application. The client-side application then sends an update request to the server's UserViewSet. The view processes this request by invoking the UpdateUserSerializer, which validates the user's input data.

Once the data is validated, the UserProfileModel is called to save the updated profile data. After the successful operation, a response is sent back to the client-side application with a confirmation of success.

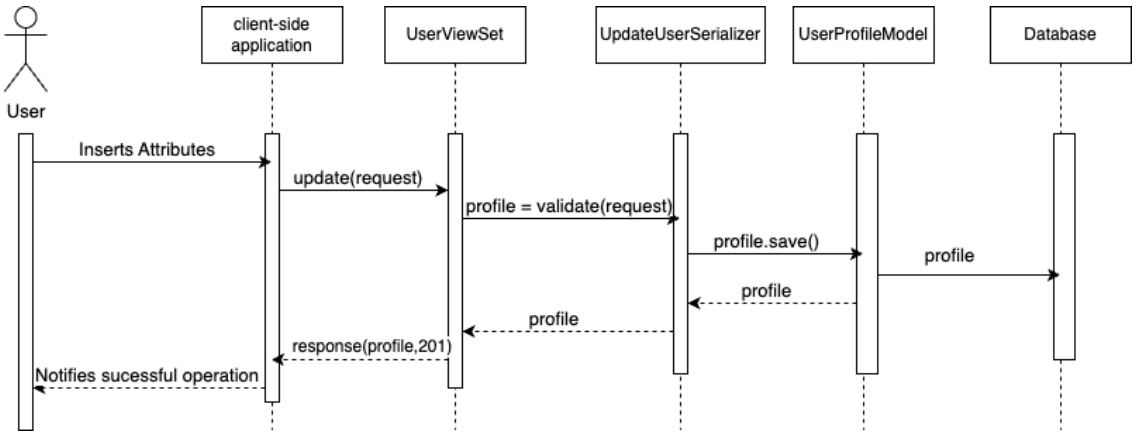

Figure 9: Sequence Diagram Use Case 3

3.4.3.4 UC4

The sequence diagram in **Figure 10** represents the fourth use case, where the user wants to visualize their assigned exercises. The process begins when the user selects the option to view their exercises via the client-side application. A request is then sent to the server, where the

UserViewSet is responsible for handling it. The view retrieves the user's profile from the database and accesses the exercises associated with this profile.

Once the exercises are retrieved, the view uses the ExerciseSerializer to format the exercises into the appropriate format and generates a response for the client-side application, returning the list of exercises.

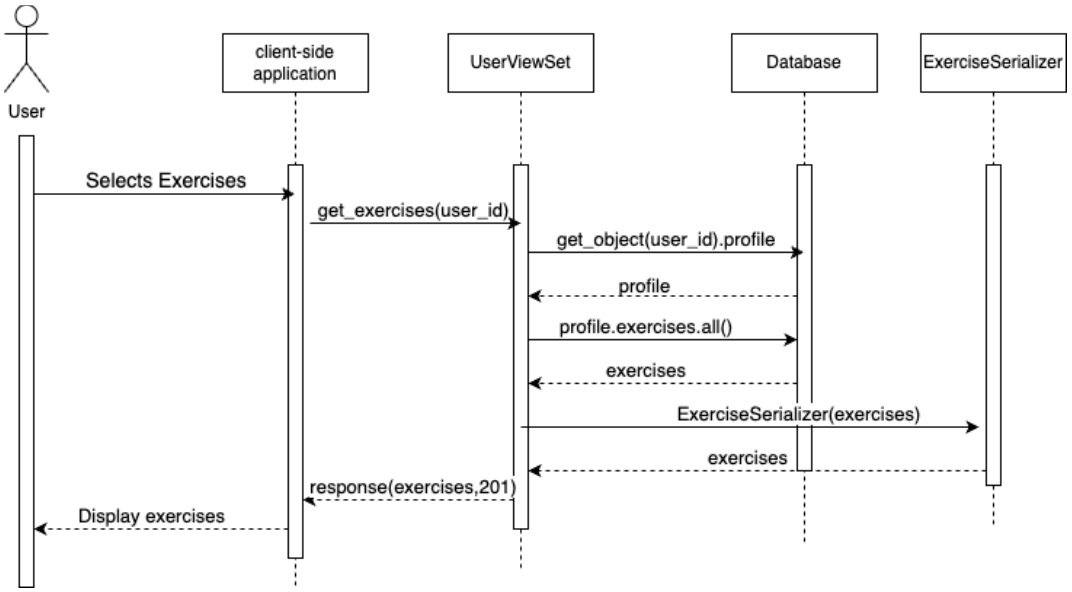

Figure 10: Sequence Diagram Use Case 4

3.4.3.5 UC5

The sequence diagram in **Figure 11** represents the fifth use case, where the user can interact with exercises by either liking or completing them. The process starts when the user performs an interaction through the client-side application. This sends a request to the server, handled by the UserExerciseInteractionViewSet.

The view then processes this request by using the UserExerciseInteractionSerializer, which validates the interaction details. Upon validation, the UserExerciseInteractionModel updates the exercise interaction in the database by saving the new interaction state. Once the interaction is successfully saved, the view returns a response to the client-side application, confirming the updated state of the exercise interaction.

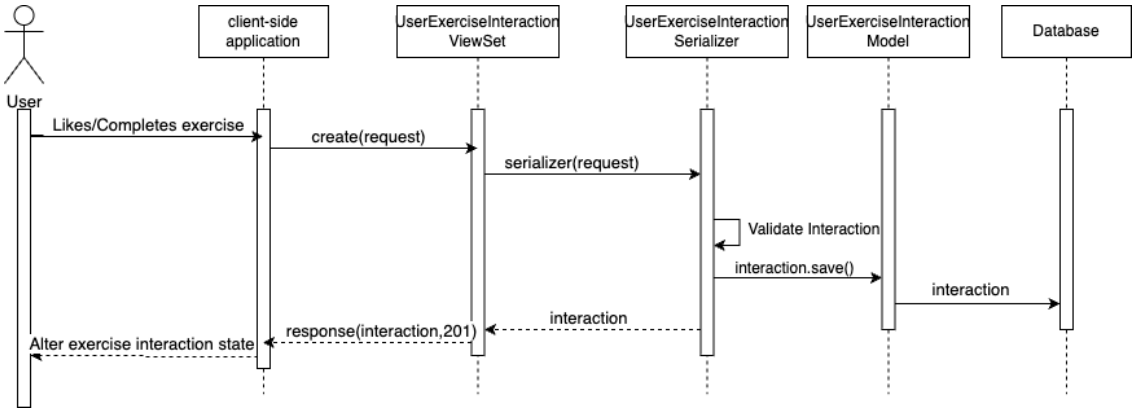

Figure 11: Sequence Diagram Use Case 5

3.4.3.6 UC6

The sequence diagram in **Figure 12** represents the sixth use case, where the user requests new exercise recommendations. The process begins when the user initiates a request for recommendations through the client-side application. This request is received by the RecommendationView, which processes the request and retrieves the collaborative filtering algorithm from the cache. Next, the Exercise model is queried to fetch all available exercises, followed by querying the UserExerciseInteraction model to filter out exercises the user has already interacted with.

The collaborative filtering algorithm then iterates over the remaining exercises, using the algorithm to predict and score each exercise based on the user's profile. The top recommendations are then serialized into a response and sent back to the client-side application, where the user can view the recommended exercises.

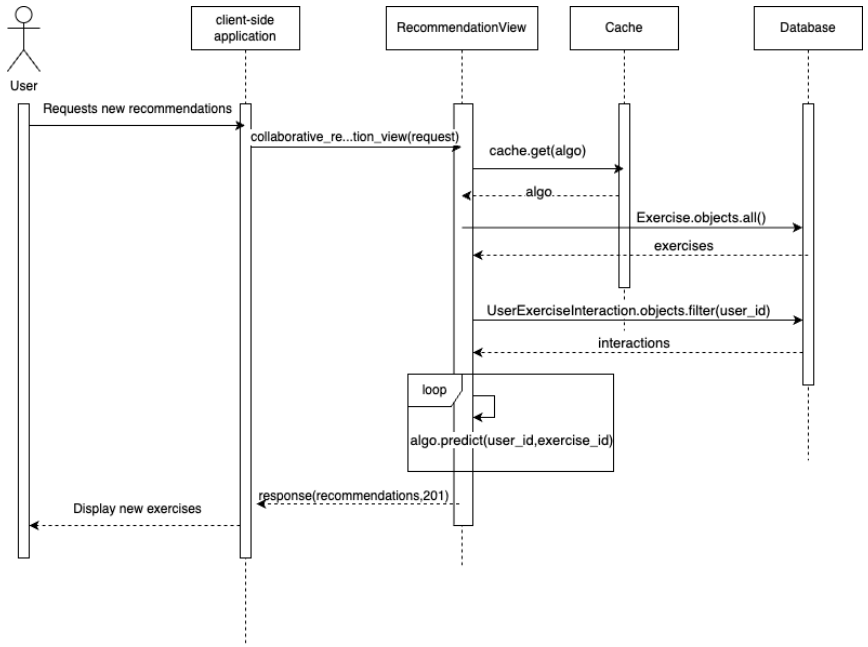

Figure 12: Sequence Diagram Use Case 6

3.4.3.7 UC7

The sequence diagrams in **Figure 13** and **Figure 14** represent the seventh use case, where the recommender system assigns exercises to a new user based on content-based filtering.

In Figure 11, the process begins during system initialization, where the system's AppConfig invokes a function that preprocesses the necessary data. The user profile data is retrieved from the database and then transformed into feature vectors by the ContentBase component. These preprocessed user features are stored in the cache for efficient access when new recommendations are needed.

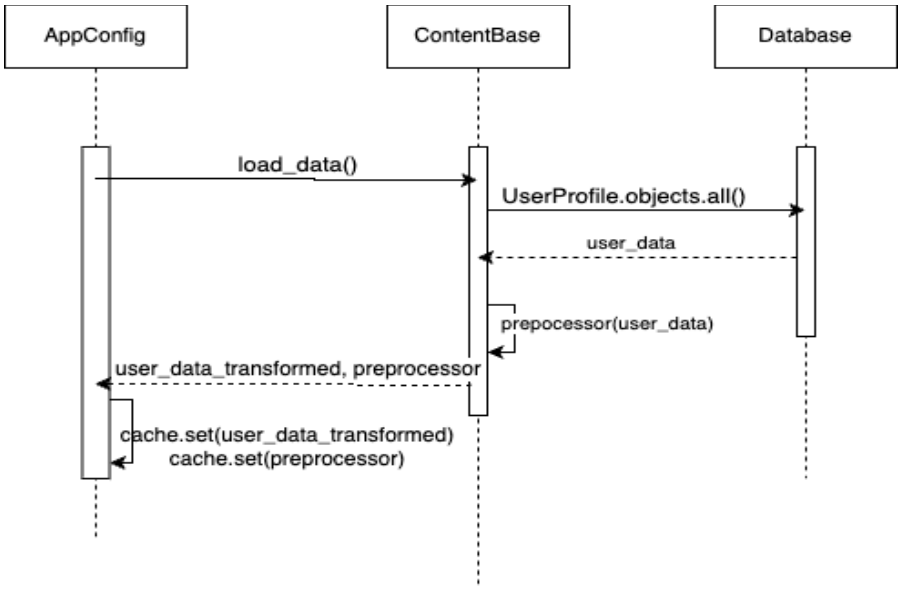

Figure 13: Sequence Diagrama 1 Use Case 7

In **Figure 14**, the process is triggered when the new user inputs their attributes. The client-side application automatically sends a request for recommendations to the RecommendationView, which retrieves the preprocessed data and the scaler from the cache. The ContentBase component then preprocesses the new user's attributes to match the format required to compute the cosine similarity between the new user and the stored profiles. The most similar user is identified, and their exercises are assigned to the new user. The client-side application then displays the recommended exercises to the user, completing the process.

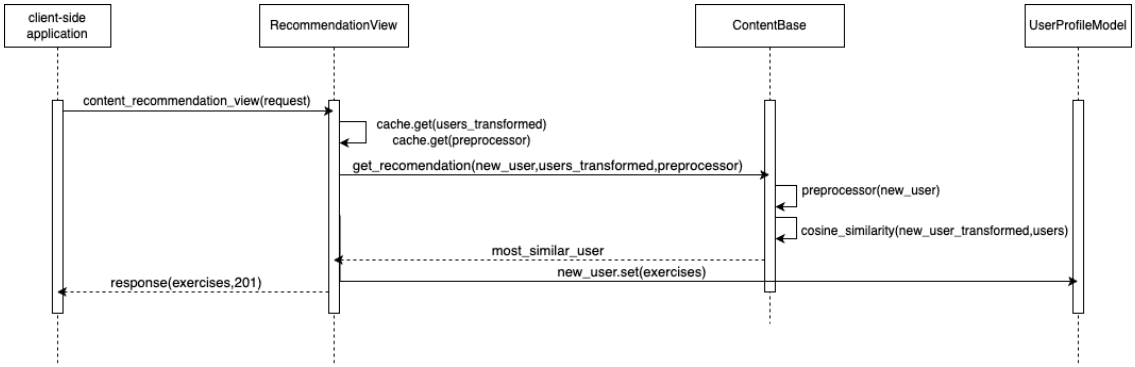

Figure 14: Sequence Diagram 2 Use Case 7

3.4.3.8 UC8

The sequence diagram in **Figure 15** represents the first phase of the eight use case, where the recommender system provides exercise recommendations based on collaborative filtering. The process begins with the system's initialization component AppConfig invoking a function from the Collaborative component that retrieves all interactions from the database. The model is then trained using the interaction data, implementing the SVD algorithm for this purpose. Once the model is trained, it is stored in cache to be user later.

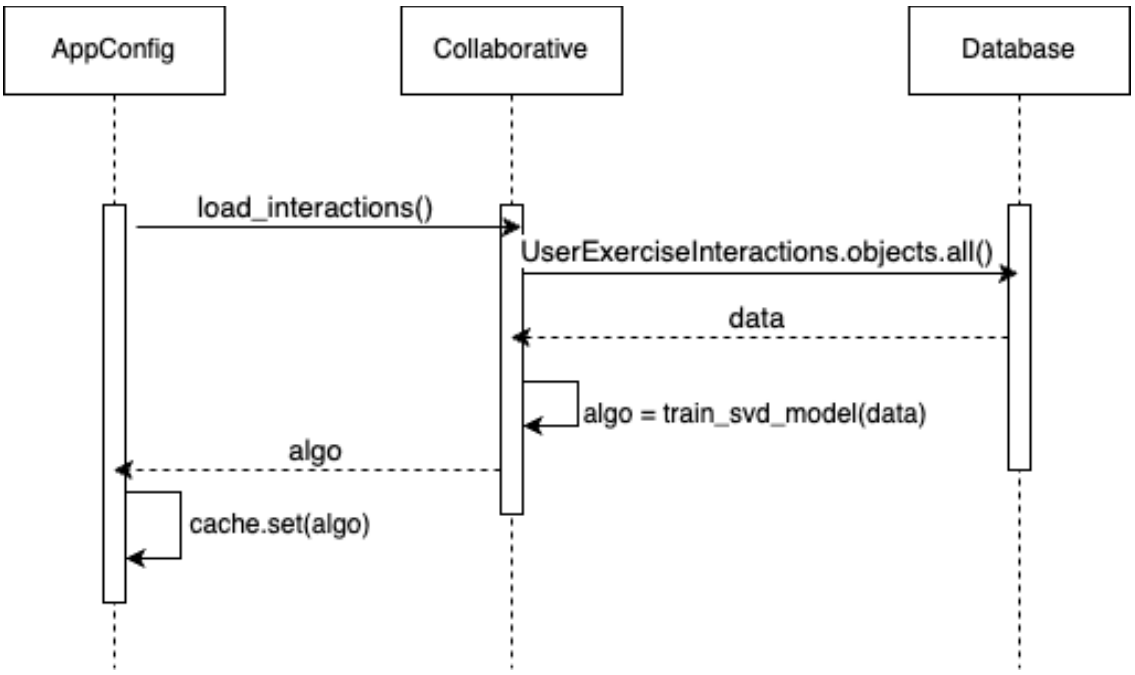

Figure 15: Sequence Diagram Use Case 8

The second phase of this use case is the same as the process described in Use Case 6. The responsible view retrieves the cached algorithm and predicts new exercises for the user based on their previous interactions.

4 Implementation

In this chapter, the focus will be on discussing the development and implementation of an application that integrates a hybrid recommender system. Built with Python and Django, the system combines content-based and collaborative filtering methods to offer exercise recommendations to users. The key components of a hybrid recommender system, such as datasets, models, and algorithms, as well as their implementation, will be explained in detail to showcase how they work together to deliver the desired outcome.

The chapter begins by presenting the hybrid recommender system and the algorithms it encompasses. Next, it explores the core building blocks of the system, including the datasets and the models derived from them. Finally, the chapter delves into the server-side application, explaining how these components are implemented and interact to provide exercise recommendations to users.

4.1 Hybrid Recommender System

For this system, a **switching hybrid recommender system** was implemented, this type of hybrid approach switches between different recommendation algorithms depending on the situation. This strategy allows the system to adapt to the strengths of each algorithm while compensating for their weaknesses.

In the context of this application, first, the content-based filtering algorithm is applied for a new user. This method leverages the new user attributes to determine who is the must similar user in the system and assign their exercises. However, content-based approaches are susceptible to overspecialization, were recommendations become too narrow, focusing only on exercises of similar users have been assigned, limiting the diversity of recommendations (Bourgais *et al.*, 2022).

Once the user has interacted with the exercises, the system uses the collaborative filtering. This approach is more effective in providing diverse recommendations based on user interactions. However, it struggles with the cold-start problem, where it cannot generate meaningful recommendations for users with no interaction history.

By switching between these two methods, the system can mitigate the overspecialization in content-based filtering and the cold-start problem in collaborative filtering, ensuring a more comprehensive recommender system(Prugel-Bennett and Ghazanfar, 2010).

Figure 16 is a flowchart that represents the decision process on how the system decides to apply content-based filtering or collaborative filtering.

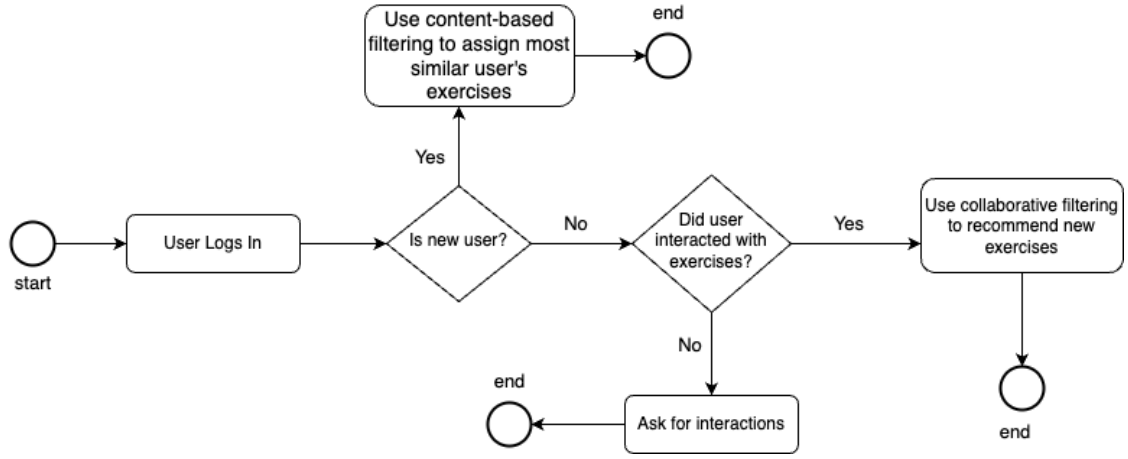

Figure 16: Switching Hybrid Recommender System Flowchart

4.1.1 Content-Based Filtering

Content-based filtering is a recommendation approach that, traditionally, uses the item's attributes to generate recommendations and is particularly useful to address the cold start problem, where new users lack historical data (Murel and Kavlakoglu, 2024). However, in the context of a fitness application, the focus is on the user's attributes, such as fitness goals and medical conditions, rather than item attributes. This approach allows for personalized recommendations early in a user's engagement with the system, ensuring that the recommended exercises align with their specific needs and preferences.

For this system, **cosine similarity** was chosen as the similarity measure, as shown in **Equation 1**, which represents the dot product between two vectors. This method is particularly effective in a fitness application because it measures the similarity in orientation of the vectors (representing users), while ignoring magnitude differences. In the context of fitness attributes like height and weight, cosine similarity allows the system to compare users meaningfully, even when their attributes differ significantly in magnitude.

Additionally, cosine similarity is well-suited for high-dimensional spaces, where user profiles may contain numerous attributes such as fitness goals, medical conditions, and injuries. In these cases, cosine similarity is effective in handling multi-dimensional data and focusing on how users align across these diverse factors. This is important in a fitness recommendation system, where numerous user characteristics must be considered to generate accurate and personalized exercise suggestions (Januzaj and Luma, 2022).

Equation 1: Cosine Similarity Calculation

$$similarity(A,B) = \cos(\theta) = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

4.1.2 Collaborative filtering

Collaborative filtering is a technique in recommender systems that leverages users' interactions with items to generate recommendations. This technique is effective in uncovering complex patterns in user behavior that are not captured through content analysis.

For this system, a latent factor model was implemented, the objective of these models is to decompose the user-item interaction matrix into two lower dimensional matrices that represent users and items in the latent space. In the latent space, each user and item are represented by a vector and these vectors are composed of latent factors (hidden factors), these factors could be something like "interest in strength training" or more abstract patterns. The closer a user's vector is to an exercise vector in this space, the more likely the user is to prefer that exercise(Tiu, 2020).

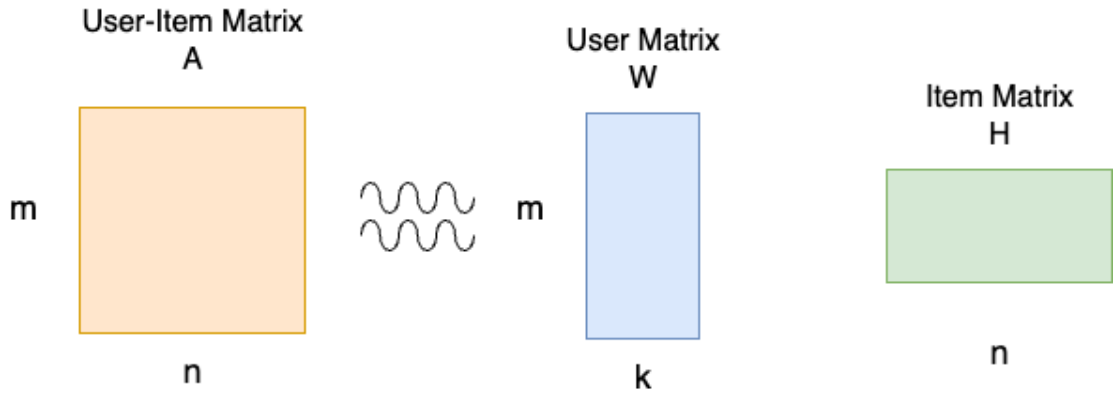

Figure 17:Matrix Factorization

Represented in **Figure 17** is the decomposition process explained previously, also known as **matrix factorization**, where A represents the original user-item interactions, W is the matrix representing user factors and H is the matrix representing item factors, this translates to the mathematical equation represented in **Equation 2.**

Equation 2: Matrix Factorization mathematical equation

$$A \approx W \times H^T$$

Singular Value Decomposition (SVD) was selected as the matrix factorization technique for this recommender system. SVD was chosen because due to its efficiency in handling sparse data, ability to reduce the user-item interaction matrix while retaining the most significant patterns and achieving approximation to the original matrix using fewer dimensions, making it computationally efficient.

The goal of matrix factorization using SVD is to minimize the difference between the actual interaction matrix (A in Figure 13) and the product of the user matrix (W in Figure 13) and the item matrix (H in Figure 13). This is achieved by the function represented in **Equation 3**.

Equation 3: Regularized Cost Function Formula

$$\min_{W,H} \sum_{(u,i)\epsilon R} (A_{ui} - W_u \times H_i)^2 + \lambda(\|W_u\|^2 + \|H_i\|^2)$$

In this formula, A_{ui} represents the original interaction between user u and item i, W_u represents the latent vector of user u, H_i represents the latent vector of item i and λ is a regularization parameter that is used to control the complexity of the model and prevent overfitting. By minimizing the difference between the original matrix and the approximation, the model learns the best representations of users and exercises, enabling the system to predict how much a user might like a particular exercise that they haven't interacted with yet.

This minimization process is essential for the matrix factorization to accurately capture the relationships in the data, and using SVD is an efficient way to achieve this (Koren, Bell and Volinsky, 2009).

4.2 Datasets and Models

This section will detail the building blocks of the recommender system, the datasets and the models. The datasets provide the raw information, directly influencing the quality of recommendations, and, for that reason, the data must undergo a preprocessing phase to clean and standardize it, making it ready for use by recommendation algorithms (Tatyaso and Khaiyum, 2023). In this system two datasets were used, a user dataset that contains demographic and physical information of 5000 users, and an exercise dataset with 2918 exercises and their attributes.

These datasets were transformed into structured models, the UserProfile and Exercise models were derived from their respective datasets. A third model, UserExerciseInteraction, was developed to capture users' interactions with the exercises. This data modelling process structures the raw data into interconnected entities that the recommender system can use. Through this approach, the system can map users to their preferences and interactions, forming the foundation for generating recommendations (IBM, 2006).

The connections between these elements are illustrated in the class diagram presented in **Figure 18**.

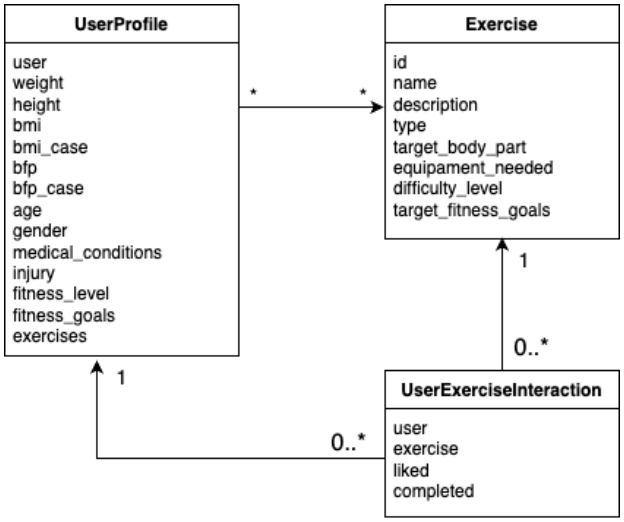

Figure 18: Core Models Class Diagram

4.2.1 User Dataset and UserProfile model

The user dataset used for this system was sourced from Kaggle, a platform for data science and machine learning datasets (Mahmoud, 2024).

This dataset was the foundation to create users, it contains demographic information of 5000 users, including physical attributes, such as **height**, **weight**, **body mass index** (BMI), **body fat percentage** (BFP), and also **age**, **gender**, **BMI_case** and **BFP_case**, the last two correspond to the classification of users' BMI and BFP numerical values.

The dataset also included a target variable named **Exercise Recommendation Plan**, where each user is assigned a plan numbered one to seven. However, the data source does not define these plans, and for that reason they did not have any direct use. Despite this, the plan numbering is correlated with the users' BMI values. **Figure 19** shows a bar chart that represents the correlation between the target variable and BMI values, this chart was retrieved from an analysis about this dataset available in Kaggle(Xash, 2024). This correlation provided a basis for clustering users into different groups.

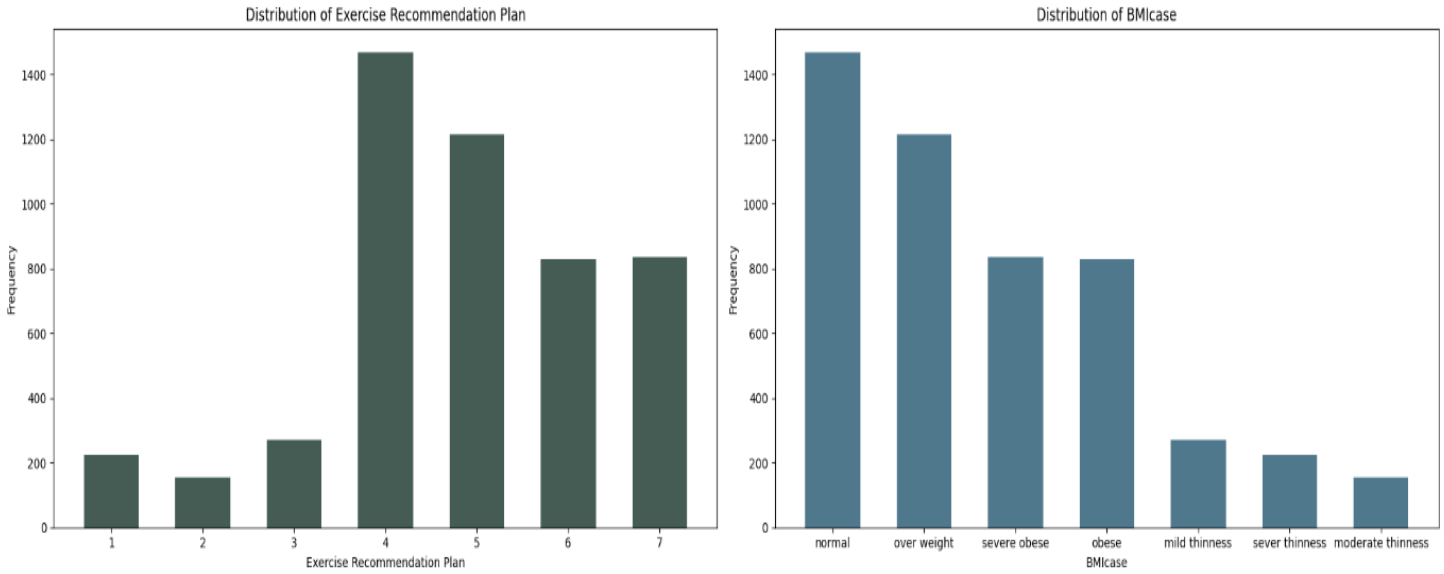

Figure 19: BMI correlation Bar Chart

In real-world datasets, it is common for data not to follow a normal distribution due to the variability among individuals. This is especially true in contexts like fitness and health, where physical attributes can show significant deviations from the norm. Recognizing this, the numerical variables of the user dataset were tested to confirm non-normality (Bono *et al.*, 2017).

To verify if the numerical variables followed a normal distribution, the Shapiro-Wilk test was used, this test indicates if the null hypothesis can be rejected, if so, it is an indication that the data does not follow a normal distribution(King and Eckersley, 2019) .

This test results in two values for each numerical variable, a statistic value that ranges between 0 and 1, a value close to 1 suggests that the data is likely to be normally distributed, and a P-value, if this value is superior to 0.05 it also suggests normality in the data.

Table 3 presents the results of the Shapiro-Wilk Test.

Table 3: Shapiro-Wilk test results

Variable	Statistic value	P-value
Age	0.9539083175356056	4.5353805154461776e-37
Height	0.9619622927355375	2.460157418696456e-34
Weight	0.9570941032907577	4.8882606757533835e-36
BMI	0.9628564973308305	5.272653916747998e-34
BFP	0.9755290911448872	1.8041043550839236e-28

Based on the test results, although the statistic value might suggest that the data is close to being normally distributed, the significantly low p-value leads to reject the null hypothesis of normality, therefore, the data does not follow a normal distribution.

Given that the sample size could affect the effectiveness of this test, a Q-Q plot visualization was also implemented, it is a scatter plot that compares the quantiles of the data against the quantiles of a theoretical distribution (NIST, no date).

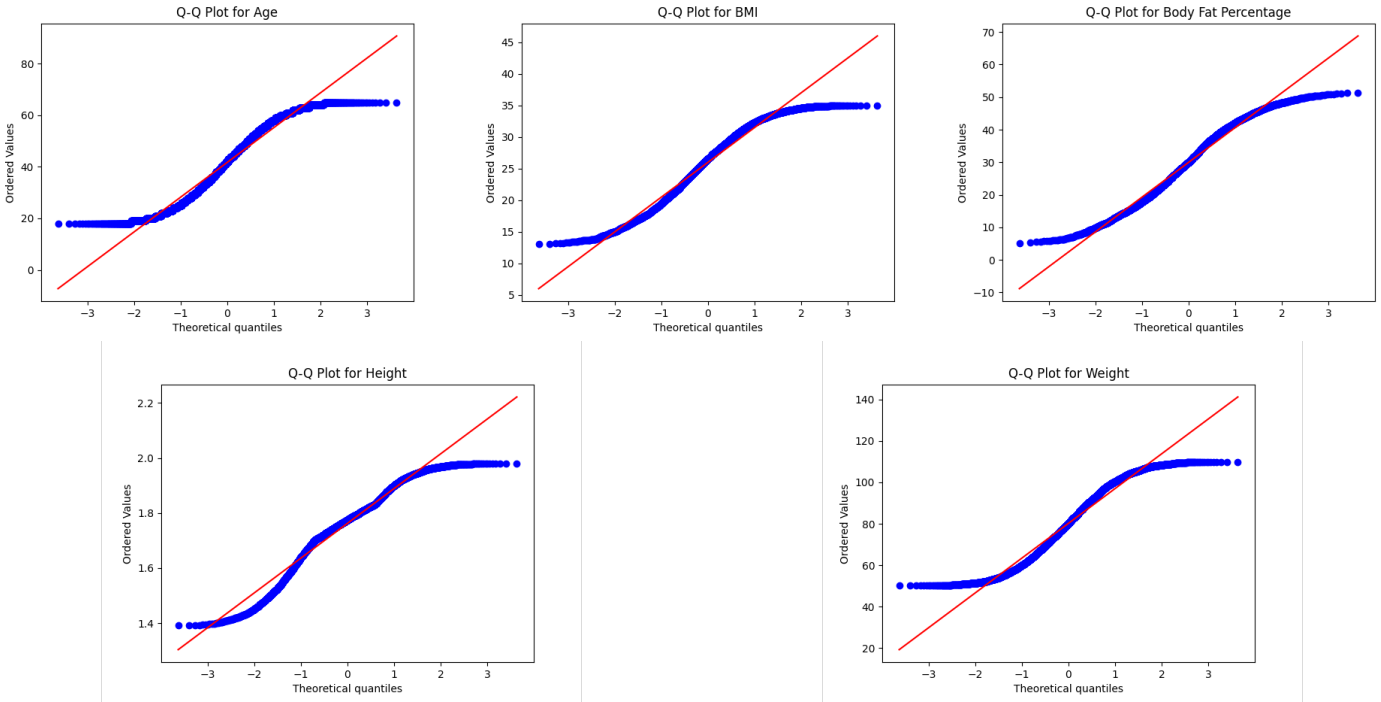

Figure 20: Q-Q Plot Height and Weight

Figure 20 represents the result of applying a Q-Q plot visualization to the numerical values.

As it can be seen by the plots and the values presented by the Shapiro-Wilk test, the data does not follow a normal distribution as expected.

Although this analysis was positive and provided confidence in the dataset, there were two categorical variables that needed to be verify for consistency. BFP_case and BMI_case represent a classification for the user's BMI and BFP value.

For that objective, a boxplot was used, this type of plot displays the distribution of data as quartiles (Cruz, 2022).

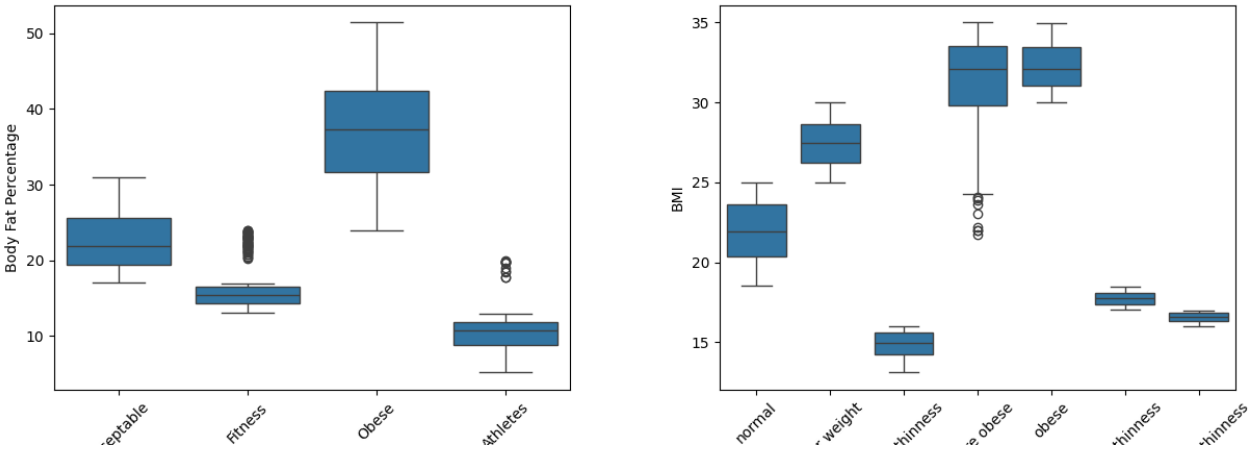

Figure 21: BMI and BFP classification Boxplot

Figure 21 represents the data distribution for the numerical value of BMI on the y axis and the corresponding classification on the x axis. It reveals an overlap between the severe obese and obese classifications, indicating inconsistencies in these categorical variables. Due to this inconsistency, it was decided to recalculate and reclassify both the numerical values and their respective categorical classifications to ensure a more accurate data representation.

For the BFP values and classifications, the American Council on Exercise (ACE) values were used(Dorwart, 2024).

Table 4 presents the threshold for each range of BFP values, and the corresponding classification used in the system.

Table 4: BFP classification

Gender	BFP	Classification
Male	6% - 13%	Athlete
	14% - 17%	Fitness
	18% - 24%	Normal
	Above 25%	Obesity
Female	14%-20%	Athlete
	21%-24%	Fitness
	25%-31%	Normal
	Above 31%	Obesity

For the BMI values and classifications, the World Health Organization (WHO) standards were applied (World Health Organization, 2010).

Table 5 presents the thresholds for each range of BMI values and the corresponding classification.

Table 5: BMI classification

BMI	Classification
Below 16.0	Severe thinness
16.0 - 16.9	Moderate thinness
17.0 - 18.4	Mild thinness
18.5 - 24.9	Normal
25.0 - 29.9	Overweight
30.0 – 34.9	Obesity class I
35.0 – 39.9	Obesity class II
Above 40.0	Obesity class III

In the WHO classification, the first three values are aggregated under one category “Underweight”. However, in this dataset, users were divided into seven distinct groups based on their BMI values to better differentiate users' physical conditions. This allowed for a more comprehensive approach to the recommendations. The last two classifications were merged, as it was determined that this group of individuals would not require distinction in the initial exercise recommendations due to similar needs at this stage.

This grouping into seven clusters was the basis for creating static exercise plans for each group. These predefined plans were later used to assign initial exercises to each user in the dataset.

In **Equation 4** and **Equation 5** is represented the formula used to recalculate the BFP and BMI values. These equations were retrieved from the same sources as the classifications.

Equation 4: Body Mass Index Calculation

$$BMI = weight \,/\, (height \ast\ast 2)$$

Equation 5: Body Fat Percentage Calculation

$$Male:$$
$$BFP = (1.20 * BMI) + (0.23 * age) - 16.2$$
$$Female:$$
$$BFP = (1.20 * BMI) + (0.23 * age) - 5.4$$

To gain a more comprehensive knowledge about the user, it was determined that it would be beneficial to capture more attributes beyond the demographic ones. These attributes include **fitness goals**, **fitness level**, **injuries**, and **medical conditions**. By incorporating these attributes, the system can provide precise and safer recommendations.

For this system, four fitness goals were considered, **Weight Loss**, **Flexibility Training**, **Muscle Strength,** and **Cardiovascular training**. By offering the ability to choose diverse fitness goals, the system can deliver personalized exercises that align with specific objectives.

The fitness level attribute is categorized in three levels, namely **Beginner**, **Intermediate** and **Expert**.

Injury attribute corresponds to the 16 body parts that are targeted by exercises. With this information, the user can pick the body part he has suffered an injury, which then can be used to filter exercises, if necessary.

In this system, four medical conditions are considered **Cardiovascular**, **Metabolic**, **Neurological** and **Respiratory**. Although these conditions are broad, they encompass a wide range of possible conditions and allows for safer recommendations as well.

In summary, the resulting UserProfile model is as shown in **Table 6**. This set of attributes ensures that the system can provide tailored and safe recommendations.

Table 6: UserProfile Model Attributes

Model	Attributes	
UserProfile	user	weight
	height	age
	bmi	bmi_case
	bfp	bfp_case
	gender	medical_conditions
	injury	fitness_level
	fitness_goals	exercises

The UserProfile model consists of numerical variables with different scales and of categorical variables. As mentioned on section 5.1.1, the cosine similarity calculation will focus on similarity between users, therefore, before applying the similarity measure, there is a need to apply a data preprocessing step, known as feature scaling or attribute scaling.

This technique involves adjusting the scale of different attributes to ensure that each attribute contributes equally to the calculations. In the context of a fitness recommender system, attributes like height and BMI have different scales and without any scaling the features with the largest scale would dominate the similarity measure.

For this purpose, **standardization**, or **z-score normalization** was utilized. The result of this process is to rescale attributes so that the mean and standard deviation are 0 and 1, respectively, which is useful for algorithms that use distance measurements, like cosine similarity. Although standardization is often applied when data is normally distributed, it is also effective for managing outliers. This is relevant in this system's context given that maintaining the natural variance in user data is beneficial(Liu, 2022).

In addition to scaling numerical features, another necessary step in the preprocessing process is handling categorical data. Categorical variables like gender or fitness goals are not directly compatible with algorithms such as cosine similarity, so they must be converted into a numerical format.

This technique transforms each category within a variable into a new binary feature, where the presence of a specific category is indicated by a 1 and its absence by a 0. For example, a user's categorical attribute like gender is transformed into two categories (male, female), this transformation ensures that categorical attributes can be properly compared when using cosine similarity (Echeburúa, 2024).

In conclusion, the user dataset served as the foundation for creating the UserProfile model, ensuring that the essential user attributes were captured. After conducting analysis and applying preprocessing steps, such as feature scaling and classification transformations, the resulting model is structured to implement cosine similarity. This allows the system to identify the most similar user to a new user.

4.2.2 Exercise Dataset and Exercise model

The exercise dataset used in this system is composed by 2918 exercises and was also retrieved from Kaggle (Pandit, 2022), it served as the foundation for building the items (Exercise model) used in the recommender system.

Each exercise in the dataset is identified by a **Title**. Some exercises are accompanied by a **Description** that provides brief instructions on how the exercise is performed. The exercises are also classified with a **Level** of difficulty, there are three, **Beginner**, **Intermediate**, and **Expert**.

The exercises are categorized into seven distinct **Types**, each representing a training focus. These types include strength, flexibility, cardiovascular, plyometrics, strongman, powerlifting, and Olympic weightlifting. The last three categories, strongman, powerlifting, and Olympic weightlifting included a significant number of beginner-level exercises, which did not align with the vision of the experts who helped design this system. To ensure consistency with this vision, all exercises in these categories were reclassified to "Expert".

Associated with each exercise is one of sixteen distinct **Body Part**, indicating the primary area of the body that the exercise targets. **Table 7** represents all 16 body parts.

Table 7: Target Body Parts

Body Part	
Abdominals	Abductors
Biceps	Calves
Chest	Forearms
Glutes	Hamstrings
Lats	Lower Back
Middle Back	Neck
Quadriceps	Shoulders
Traps	Triceps

For each exercise there is a **Equipament** attribute that indicates the type of equipament required to perform each exercise. **Table 8** lists all the equipaments specified in the dataset.

Table 8 : Exercise Equipament

Bands
Barbell
Dumbell
Cable
Machine
Body Only
Medicine Ball
Exercise Ball
Foam Roll
E-Z Curl Bar

Lastly, the dataset also included **Rating** and **Rating description** attributes for each exercise. However, these attributes were not used in the development of the item model due to lack of explanation regarding their meaning and the fact that many exercises lacked a rating.

The Exercise model serves as a representation of the “item” in the recommender system. It is composed of several attributes from the exercise dataset, such as Title, Description, Body Part, Equipment, and Level.

In addition to the attributes from the exercise dataset, a **Target Fitness Goals** attribute was considered beneficial to this model. As shown in **Table 9**, each exercise type can correspond to more than one fitness goal, this attribute allowed to align the exercise type with corresponding fitness goals.

Table 9: Exercise Type Fitness Goals

Exercise Type	Fitness Goal
Strength	Weight Loss/ Muscle Strength
Stretching	Flexibility Training
Cardio	Weight Loss/ Cardiovascular Training
Plyometrics	Weight Loss/Muscle Strength/Cardiovascular Training
Olympic Weightlifting/ Powerlifting / Strongman	Muscle Strength

In conclusion, apart from the Type attribute, no additional cleaning or preprocessing steps were required for the dataset. The model serves as a target variable within the recommendation process, but it is not directly involved in any computations, leading to the final structure represented in **Table 10**.

Table 10:Exercise Model Attributes

Model	Attributes
Exercise	id
	name
	description
	type
	target_body_part
	equipament_needed
	target_fitness_goals
	difficulty_level

4.2.3 UserExerciseInteraction Model

The UserExerciseInteraction model was design to capture the interactions between users and exercises within the system. This model includes four key attributes, **User** and **Exercise**, these two attributes are identifiers and link the interaction to a specific user and exercise, **Liked** and **Completed**, boolean attributes that capture the user's explicit feedback, resulting in the model represented in **Table 11**.

Table 11: UserExerciseInteraction Model Attributes

Model	Attributes
UserExerciseInteraction	user
	exercise
	liked
	completed

This model is essential for the collaborative filtering process, as it is used to create the user-item matrix.

To compute this model and use it in the collaborative filtering process, there was a need for a data preprocessing step. This step involved transforming the explicit feedback captured by the UserExerciseInteraction model into quantitative ratings.

Table 12: User Exercise Interaction Value

Interaction	Value
Liked && Completed	2
Liked	1
Completed	0.5
No interaction	-1

Table 12 represents the mapping between the user interactions and the quantitative value assigned. This scale is designed to capture the user's engagement level with each exercise. This transformation is essential to make the users' interactions are suitable for input into the collaborative filtering algorithm.

In conclusion, these datasets and models form the backbone of the recommender system, enabling both content-based and collaborative filtering methods to generate exercise recommendations. Through effective preprocessing and modelling, the system can process new users and interactions to enhance its recommendations over time.

4.3 Server-side application

The server-side application was developed using the Django Framework, a high-level Python web framework. The choice of Django was influenced by its compatibility with the Python ecosystem, which facilitated easy integration with libraries such as Scikit-learn and Surprise, both of which were used to implement the recommender algorithms. Django's built-in functionalities, including its ORM and security features, accelerate the development process and ensure that performance and security considerations are effectively addressed. This makes Django a robust choice for building a recommender system application (Django Overview, 2024).

The application utilizes SQLite3 as its backend, a lightweight, relational database management system (RDBMS) that is self-contained and serverless. It stores data in tables that can be related to each other through foreign keys, supporting the relational model's principles. This database is ideal for development of small-scale applications due to its simplicity and ease of setup. Being the default database in Django, SQLite3 simplifies the initial configuration process and allows for the development to focus on core functionalities like the recommendation algorithms, making it ideal for a system like the one presented in this thesis, which is not in a development phase that requires scaling concerns (SQLite, 2023).

Figure 22 represents the system's architecture, including all key components that interact to provide the functionality of the recommendation system. Each module plays a specific role, from managing user profiles and exercise data to handling requests, processing data and generating recommendations. This modular structure ensures that each part of the system works together efficiently, forming the foundation of the recommendation process.

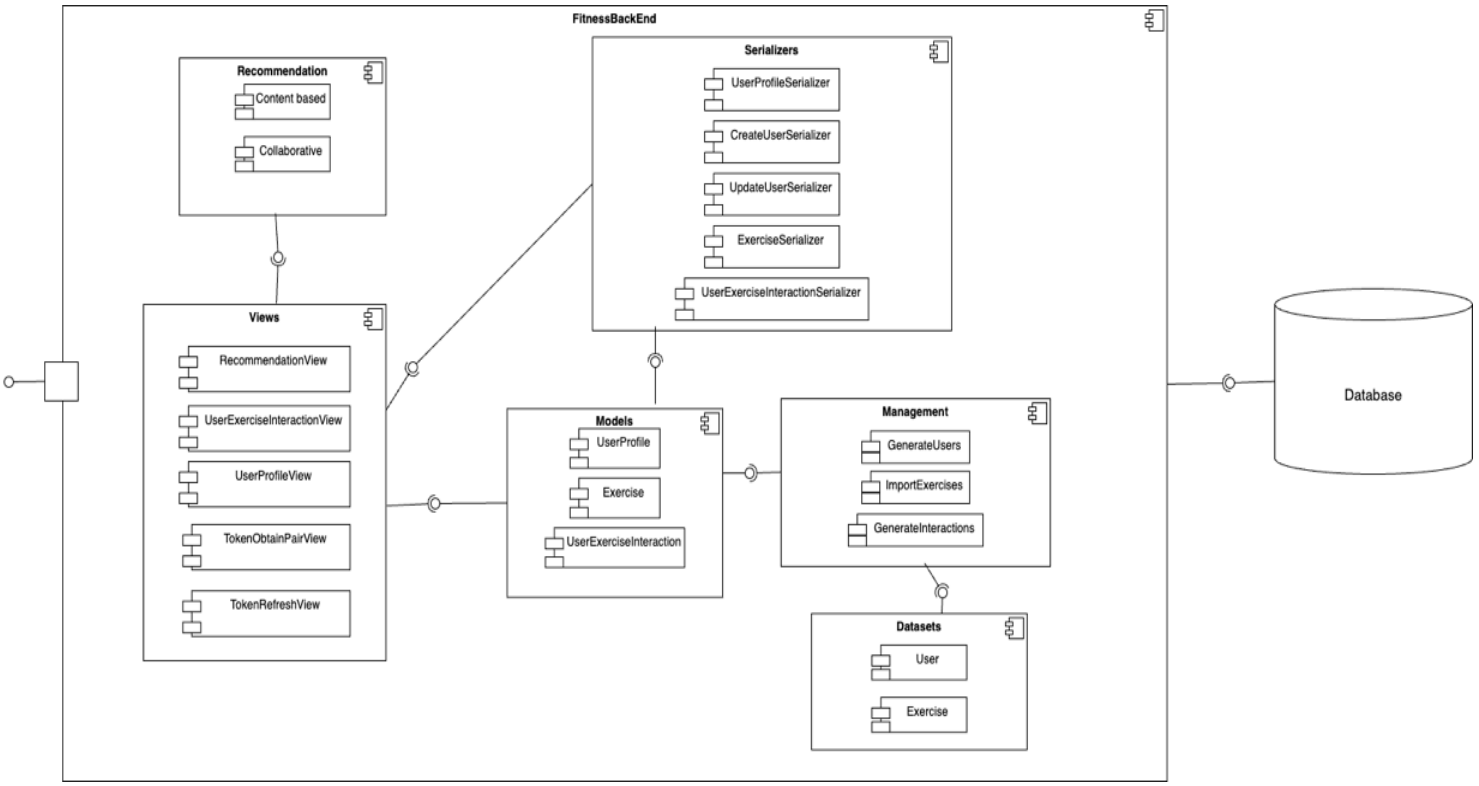

Figure 22: FitnessBackEnd Detailed Container Diagram

Figure 23 represents the system workflow, it is organized into three distinct phases that guide the process from data initialization to generating exercise recommendations.

The first phase is **Data Initialization**, this phase focus on importing and generating data. The system loads datasets for users and exercises and generates synthetic data to fill the users' attributes and create interactions.

Once the data is initialized, the system is ready to be configured. The **System Configuration** phase prepares the system to handle future recommendation tasks. This involves preparing the data for content-based filtering and training the collaborative filtering model.

The final phase, the Recommendation Phase, is when the system is ready to generate exercise recommendations. It processes incoming requests from users and provides exercise suggestions based on the pre-configured algorithms.

These three phases ensure that the system is able to move from data preparation to delivering recommendations.

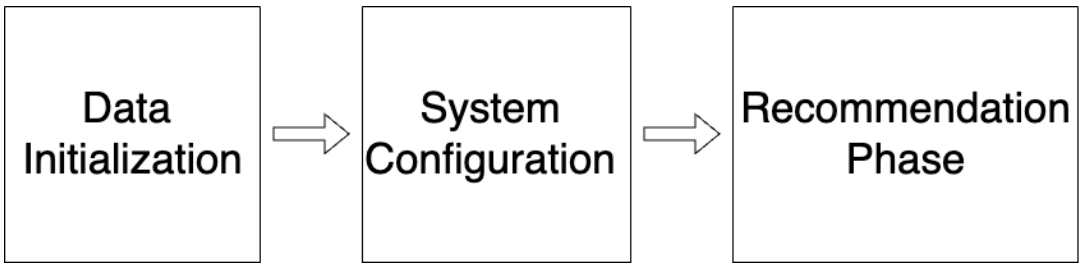

Figure 23: System Workflow

In the description of each phase, a sequence diagrams will be presented to support the explanation. The database is not explicitly represented in the diagrams because the models interact with the database using Django Object-Relational Mapping (ORM). Django ORM allows each model to map to database tables, enabling operations like data insertion, retrieval, and updates using Python code. This abstracts the need for SQL queries by handling database interactions at the model level (Django Models, 2024).

4.3.1 Data Initialization

The data initialization phase has achieved using three Management commands that were designed to generate and import the necessary data for the system. The first command, **GenerateUsers**, imports users from the user dataset and generates synthetic information for the missing attributes, such as fitness levels and goals. The second command, **ImportExercises**, imports exercises from the exercise dataset and assigns each exercise with the corresponding fitness goals. The third command, **GenerateInteractions**, produced synthetic user-exercise interactions, ensuring that the system could provide recommendations based on user preferences.

Figure 24 represents the sequence diagram of the first command, **GenerateUsers**.

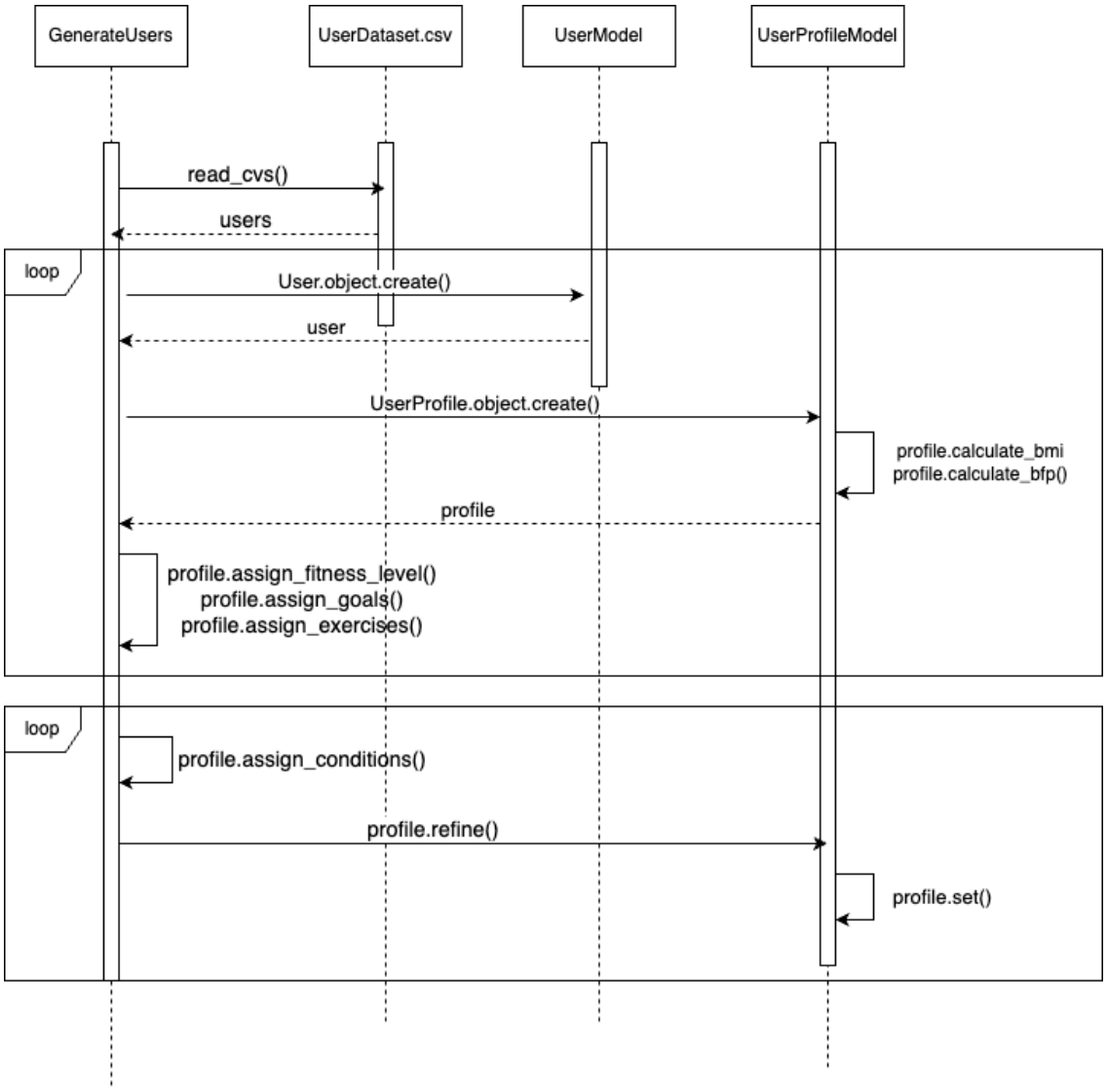

Figure 24: GenerateUsers command Sequence Diagram

The **GenerateUsers** custom command accomplished several necessary steps. The command starts by reading the User dataset and iterating over its rows. For each row a new User object is created using username, email and password. After that, a new UserProfile object is also created and initialized with the weight, height, gender, and age values. With the attributes of weight, height and gender the values and classifications for BMI and BFP are calculated, following the equations presented in Figure 22 and Figure 23, as well as the classifications presented in Table 5 and Table 4, from section 4.2.1.

As mentioned in section 4.2.1, besides the physical attributes, there were attributes added to the UserProfile model so that there was more information about each user, resulting in safer recommendations. These attributes included fitness level, fitness goals, medical conditions and injuries. Since the original dataset didn't contain this information about the users, it was decided that this data would be created, with the objective of having a diverse user population.

Code snippet 1 represents the assignment of a fitness level, fitness goals and an initial set of exercises.

```
def generate_users(self):
    data = pd.read_csv('../final_dataset_BFP.csv')
    fitness_level_ids = list(FitnessLevel.objects.values_list('id', flat=True))

    for index,row in data.iterrows():
        profile, created = UserProfile.objects.update_or_create(…)

        cluster_id = bmi_to_plan_id.get(profile.bmi_case)
        cluster_details = plan_dict.get(cluster_id)

        fitness_level = self.assign_fitness_level(cluster_id,fitness_level_ids)
        profile.fitness_level.set([fitness_level])

        if fitness_level.level.lower() == 'beginner':
            exercises = Exercise.objects.filter(name__in=cluster_details['beginner_exercises'])
        else:
            exercises = Exercise.objects.filter(name__in=cluster_details['exercises'])

        profile.exercises.set(exercises)

        goals = FitnessGoal.objects.filter(id__in=cluster_details['goals'])
        if len(goals) <= 3:
            profile.fitness_goals.set(goals)
        else:
            num_goals =np.random.choice(range(1,5))
            profile.fitness_goals.set(np.random.choice(goals,num_goals,replace=False))
```

Code Snippet 1: User Conditions Assignment

First, the **bmi_to_plan_id** dictionary is used to map the user's BMI classification to a corresponding cluster ID. This mapping ensures that users are assigned to a predefined exercise plan appropriated to their physical condition. These plans were designed with the input of health experts, with the objective of having initial plans for each cluster.

The plans vary depending on the user's BMI cluster. For example, extreme BMI cases, such as severe thinness or obese class II, have more restrictive exercise options, only targeting big muscle groups. These users are limited to the beginner level option and have fewer fitness goals available to ensure a safe progress.

After determining the user's BMI case, a fitness level is assigned to a user using the **assign_fitness_level()** function shown in **Code Snippet 2**. This function uses the BMI cluster to assign a fitness level. Users in clusters 1,2,6 and 7 (extreme cases) are limited to the "Beginner" set of exercises for precautionary reasons. For the remaining clusters a random selection is used, having a higher probability of being "Beginner" or "Intermediate" so that it reflects a real-world population.

```
def assign_fitness_level(cluster_id,fitness_level_ids):
    if cluster_id in [1,2,6,7]:
        return FitnessLevel.objects.get(level="Beginner")
    else:
        return
FitnessLevel.objects.get(pk=np.random.choice(fitness_level_ids,p=[0.4,0.4,0.2]))
```

Code Snippet 2: Assign Fitness Level Function

The last step in the **generate_users()** function is assigning fitness goals to users. This process has distinct approaches depending on the cluster the user belongs to. If a user's cluster provides only a limited number of fitness goals, the system assigns all available goals to that user. However, for users in clusters with a broader range of fitness goals, the system applies a random selection, choosing between one and four goals. This randomization introduces diversity and better simulates a diverse user base with differing fitness goals.

The final step in the GenerateUser command is handled by the **assign_conditions()** function, represented in **Code snippet 3**. This function uses the list of user profiles and divides them into four groups. Half of the users will not have any conditions assigned to them. The remaining half is further divided into three groups, one group will only have injuries, another will have only medical conditions, and the last group will have both injuries and medical conditions. This approach ensures that the user base includes a diverse range of profiles with different health conditions, contributing to the overall diversity of the user population.

```
  def assign_conditions():

      user_profiles = list(UserProfile.objects.all())
      total_users = len(user_profiles)
      no_condition_size = total_users // 2
      remaining_users = total_users - no_condition_size
      group_size = remaining_users // 3

      injury_only_users = user_profiles[no_condition_size:no_condition_size + group_size]
      condition_only_users = user_profiles[no_condition_size + group_size:no_condition_size
+ 2 * group_size]
      both_conditions_and_injuries_users = user_profiles[no_condition_size + 2 *
group_size:]

      # injuries only
      for profile in injury_only_users:
         injury_num = np.random.randint(1, 4)
         selected_injuries = random.sample(all_injuries,injury_num)
         profile.injury.set(selected_injuries)
         profile.refine_for_injuries()


      # medical conditions only
      for profile in condition_only_users:
          …
      # medical conditions and injuries
      for profile in both_conditions_and_injuries_users:
          …
```

Code Snippet 3: User Conditions Assignment 2

Lastly, after assigning conditions and goals to users, the initial exercise set must be refined to address these specific attributes. In the UserProfile model, functions were implemented to filter exercises according to these conditions. Once the conditions are assigned, each instance of the user profile invokes these filtering functions to ensure that the selected exercises align with the user's individual needs, creating a reliable user database.

Figure 25 represents the sequence diagram of the second command, **ImportExercises**.

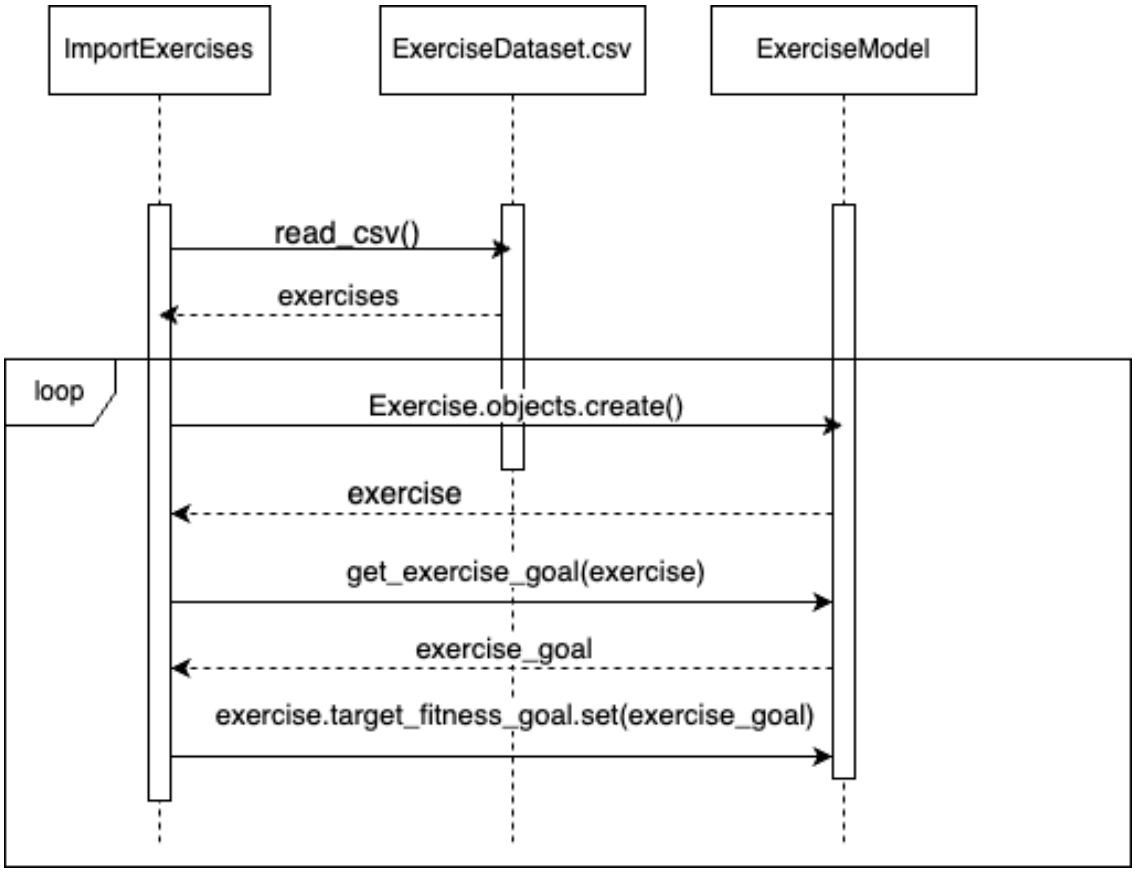

Figure 25: ImportExercises command Sequence Diagram

The **ImportExercises** command starts by reading the CSV file and iterating through each row in the dataset. For each row, a new Exercise object is created, and the corresponding fitness goals are assigned to the new object.

This command populates the database with the target variable for both recommendation algorithms.

Figure 26 represents the sequence diagram of the third command, **GenerateInteractions**.

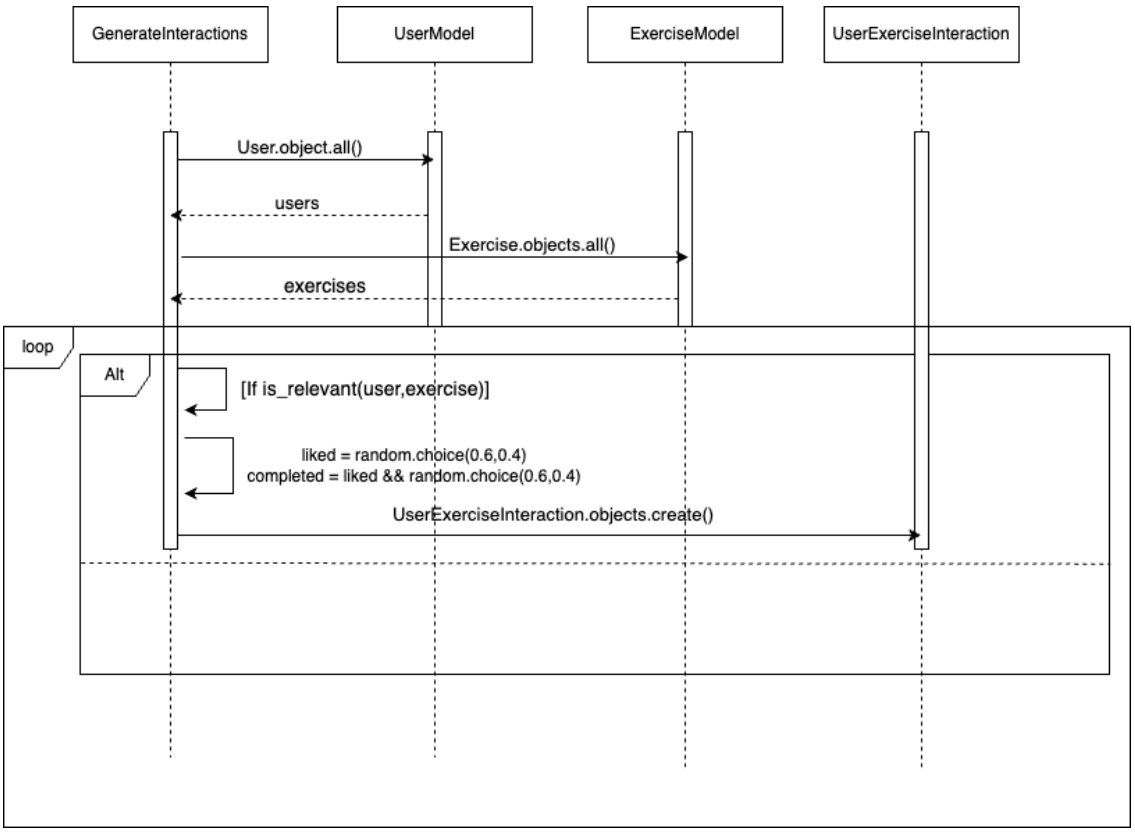

Figure 26: GenerateInteractions Sequence Diagram

The **GenerateInteractions** command was designed to create synthetic user-exercise interactions, so that it would be possible to test the collaborative filtering approach.

This command starts by fetching all users and all exercises. After, the loop iterates over exercises that were not assigned to a user initially, determining whether the exercise is suitable through the **is_relevant()** function.

As shown in **Code Snippet 4**, this function applies a series of verifications to assess the suitability of each exercise for the user. First, it ensures the exercise is at or below the user's fitness level. If this condition is met, the function checks whether the exercise's target body part aligns with the user's cluster's primary muscle groups. This is particularly important for extreme BMI cases, where users should avoid exercises targeting smaller muscle groups. Following these checks, the alignment between the exercise's target fitness goals and the user's own goals is evaluated. Lastly, the function ensures that the exercise doesn't conflict with any of the user's injuries or medical conditions. This function was designed with the help of health experts, to ensured accurate synthetic interactions.

```
   def is_relevant(self, exercise,
user_goals,user_level,user_injury,user_medical_condition,user_bmi_case):

      if not self.is_level_appropriate(user_level, exercise):
         return False

      if not self.is_exercise_compatible_with_cluster(user_bmi_case,exercise):
         return False

      if not self.are_goals_aligned(user_goals, exercise):
         return False

      if self.is_injury_conflict(user_injury, exercise):
         return False

      if not self.is_condition_compatible(user_medical_condition, exercise):
         return False

      return True
```

Code Snippet 4:Is Relevant Function

Code Snippet 5 represents the loop shown on **Figure 34**. If all criteria are satisfied, the exercise is considered relevant, and an interaction is generated with a slight positive bias, it is expected that the user is more likely to engage positively with suitable exercises.

```
class Command(BaseCommand):
   def handle(self, *args, **options):
      users = User.objects.filter(is_superuser=False)
      exercises = set(Exercise.objects.all())

      for user in users:
         profile = UserProfile.objects.get(user=user)
          …

         for exercise in exercises:
 if self.is_relevant(exercise,user_goals,user_level,user_injury,user_medical_condition):

               liked = np.random.choice([True, False], p=[0.6, 0.4])
               completed = liked and np.random.choice([True, False], p=[0.6, 0.4])
               UserExerciseInteraction.objects.update_or_create(
                  user=user,
                  exercise=exercise,
                  defaults={'liked': liked, 'completed': completed )
```

Code Snippet 5: Generate Interaction Function

In conclusion, the data initialization phase is a crucial step in preparing the system for generating exercise recommendations, ensuring all necessary data, such as, users, exercises, and interactions are accurately prepared. Through the execution of three management commands, the system builds the database required for generating exercise recommendations. This setup effectively prepares the application for the next phases, configuration and recommendation, where the recommender algorithms can operate efficiently based on the initialized data.

4.3.2 System Configuration

The system configuration phase is an important step that ensures that the recommender system is ready to provide exercise recommendation to users. **Figure 27** represents the sequence of steps necessary to prepare the system and optimize performance. This process relies on the default caching system provided by Django. Django's cache is an in-memory caching system that is suitable for small-scale applications, as it provides ease of setup, improved performance and memory efficiency (Django Cache, 2024).

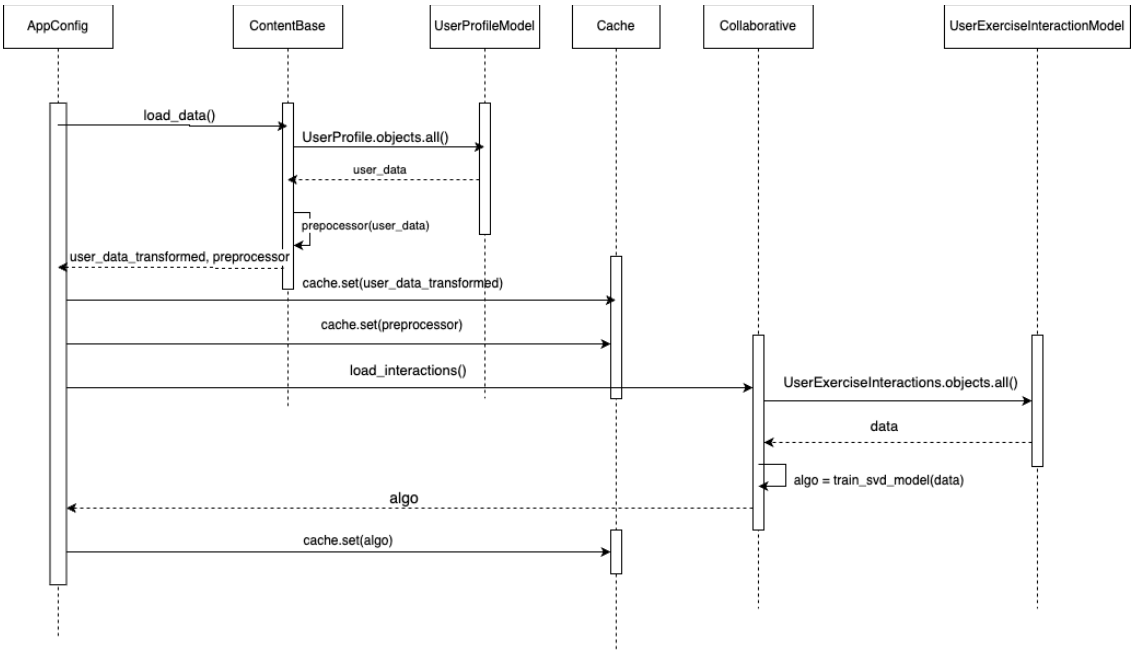

Figure 27: System Configuration Sequence Diagram

During this phase, the system invokes two functions. The **load_data()** function implements two methods from the **Scikit-learn** library to prepare the user data, **StandardScaler** and **OneHotEncoder**. This library built on NumPy, SciPy, and matplotlib, provided the necessary tools for data preprocessing steps and the cosine similarity algorithm (Scikit-Learn, 2024).

```
import pandas as pd
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer

numerical_features = ['weight', 'height', 'bmi', 'body_fat_percentage', 'age']
categorical_features = ['gender', 'bmi_case',
'bfp_case','medical_conditions','injury','fitness_level','fitness_goals']

def load_data():
   user_profiles = UserProfile.objects.all()

   user_data = pd.DataFrame([{…} for user in user_profiles])

   # Preprocess the data
   preprocessor = ColumnTransformer(
      transformers=[
         ('num', StandardScaler(), numerical_features),
         ('cat', OneHotEncoder(), categorical_features)  ])

   # Fit and transform the user data
   user_features_transformed = preprocessor.fit_transform(user_data)

  return user_features_transformed, preprocessor
```

Code Snippet 6: load_data Function

Code Snippet 6 presents the load_data() function. This function begins by extracting the user profiles from the database and transforming them into Pandas DataFrame (pandas, 2024). This transformation is necessary because DataFrames provide a structured way to manipulate data, making it easier to apply preprocessing techniques.

Next, the **ColumnTransformer** class is used to apply both StandardScaler and OneHotEncoder. This class applies transformers to the columns of a Pandas DataFrame simultaneously. The StandardScaler is applied to the numerical features to standardize them and the OneHotEncoder handles the categorical features, transforming them into binary features.

The output of this function includes the **preprocessor** object, this object applies the necessary transformation to the existing user profiles, and it can be reused to preprocess a new user. Additionally, the function returns the **users_features_transformed** object, which contains the transformed user profiles ready for use in cosine similarity calculations.

```
import pandas as pd
from surprise import Dataset, Reader

def load_interactions():
    interactions = UserExerciseInteraction.objects.select_related('user', 'exercise').all()
    for interaction in interactions:
        rating = (
            2 if interaction.liked and interaction.completed else
            1 if interaction.liked and not interaction.completed else
            0.5 if not interaction.liked and interaction.completed else
            -1
        )

    interaction_df = pd.DataFrame(interactions)

    # Load the data into the Surprise format
    reader = Reader(rating_scale=(-1, 2))
    surprise_data = Dataset.load_from_df(interaction_df[['user_id', 'exercise_id', 'rating']],
reader)

    # Train and evaluate the model
    algo = train_svd_model(surprise_data)

   return algo
```

Code Snippet 7: load_interactions Function

The second function the system invoked is the **load_interaction()** function. As shown in **Code Snippet 7**, this function transforms the user interactions into a suitable format to be used by the **Surprise** library. The Surprise library is another Python library built on SciPy and designed for building recommender systems. Surprise's implementation of SVD and dataset handling provided an effective way to handle the matrix factorization required for collaborative filtering. The library's ease of use and flexibility in model training and evaluation were key reasons for selecting it as the tool to implement this approach (Hug, 2024).

The function starts by querying the database to fetch all user-exercise interactions. It then iterates through each interaction and calculates the rating according to Table 9 of section 4.2.3. This list is then converted into a Pandas Dataframe, to be easily interpreted by the Surprise library method.

Next, the **Reader** class is used to define the rating scale within the dataset, informing the recommendation algorithm that ratings range from -1 one to 2. After, the Dataframe is converted into a Surprise dataset using the **Dataset.load_from_df()** function, resulting in the **surprise_data** object that is ready to be used to train the SVD model.

As shown in **Code Snippet 8**, the load_interactions() function has a nested function called **train_svd_model()**.

```
from surprise import SVD
from surprise.model_selection import train_test_split

def train_svd_model(data):
    # Split data into train and test sets
    trainset, testset = train_test_split(data, test_size=0.2)

    # Use the SVD algorithm
    algo = SVD()

    # Train the algorithm on the trainset
    algo.fit(trainset)

    (…)
    return algo
```

Code Snippet 8: train_svd_model Function

The **train_svd_model()** function begins by splitting the preprocessed data into training and test sets using the **train_test_split()** method from the Surprise library. This split ensures that the model is trained on one portion of the data while its performance is evaluated on the other. Even though there is no strict rule for the split ratio, an 80/20 split is a common practice in machine learning, a test size of 20% provides a good balance between having enough data to train the model while retaining enough data to validate its performance (Joseph, 2022).

Once the data is split, the SVD algorithm is instantiated with **algo = SVD()**. In this instantiation, the default regularization term is used, which has a value of $\lambda = 0.02$ (Hug, 2015). Following the instantiation, the model is trained on the training set using **algo.fit(trainset)**, where it learns to approximate the original matrix by minimizing the error between the predicated and actual interactions.

The function returns the trained SVD model, **algo**, which can be used to generate recommendation for users.

In conclusion, by saving the outcomes of the load_data() and load_interactions() functions in the Cache, the system can quickly access them during user requests, reducing computation time and enhancing overall performance for both content-based and collaborative recommendations.

4.3.3 Recommendation Phase

The Recommendation Phase is the core of the hybrid recommender system, where it actively generates exercise suggestions based on the user's profile and interaction data. The sequence diagram represented in **Figure 28** illustrates how the system switches between content-based and collaborative filtering to provide exercise recommendations

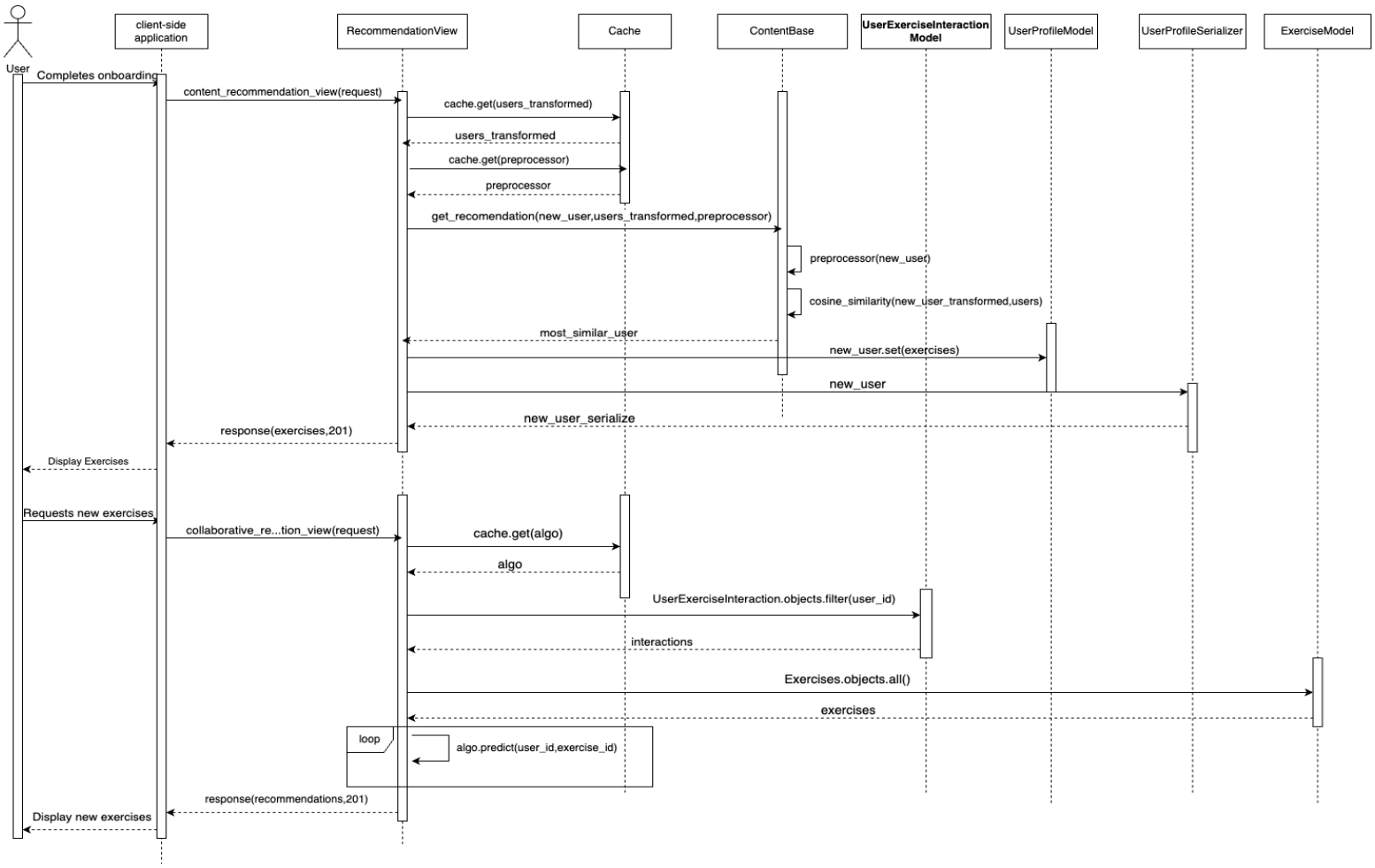

Figure 28: Recommendation Phase Sequence Diagram

In an initial phase, when the user is new to the system, the content-based approach is used. After the user inserts their information, the client-side application sends a request that is handled by the RecommendationView, more specifically, the **content_recommendation_view()** function.

Code Snippet 9 represents this function. Initially, the function retrieves the necessary data and preprocessor from the cache, which are essential for executing the content-based recommendation algorithm. It identifies the most similar user and assigns that user's exercises to the new user. The function then performs a series refinement function to ensure that only appropriate exercises are recommended based on the user's conditions and preferences. Finally, it returns the refined exercise recommendations as a JSON response.

```
@api_view(['GET'])
@permission_classes([IsAuthenticated])
def content_recommendation_view(request, user_id):
    data = cache.get('content_base_data')
    preprocessor = cache.get('recommender_preprocessor')
    try:
        user_profile = UserProfile.objects.get(user__id=user_id)
        recommendation = get_recommendation(data,user_profile,preprocessor)
        most_similar_index = int(recommendation[1])
        most_similar_user_profile = UserProfile.objects.all()[most_similar_index]
        serializer = UserProfileSerializer(most_similar_user_profile)

        new_user_exercises = most_similar_user_profile.exercises.all()
        user_profile.exercises.set(new_user_exercises)

        if user_profile.medical_conditions.exists(): user_profile.refine_for_conditions()
        if user_profile.injury.exists(): user_profile.refine_for_injuries()
        user_profile.refine_for_goals()

        return JsonResponse({'recommended_user_profile': serializer.data})
    except Exception as e:
        return JsonResponse({'An error occured': str(e)}, status=500)
```

Code Snippet 9: content_recommendation_view Function

To execute the content-based recommendation algorithm, this function invokes another function, **get_recommendation()**. As shown in **Code Snippet 10**, this function is responsible for applying the cosine similarity calculation to a new user, with the goal of finding the most similar user based on the new user's attributes.

The preprocessor object is reused to apply the same transformations to the new user's Panda DataFrame, ensuring the new user's data is scaled and encoded in the same way as the existing users.

Next, the function calculates the cosine similarity between the transformed features of the new user and the features of all existing users. The output of this calculation is a one-dimension array where each element represents the cosine similarity score between the new user and an existing user. This array is then sorted in ascending order using Numpy's **argsort()** function. To identify the most similar users, the array is reversed with **[::-1]**, resulting in the most similar user listed first.

```
import numpy as np
import pandas as pd
from sklearn.metrics.pairwise import cosine_similarity


def get_recommendation(user_features_transformed, new_user,preprocessor):

    # Create a DataFrame for the new user
    new_user_data = pd.DataFrame([{…}])

    # Transform the new user data using the same preprocessor
    new_user_features = preprocessor.transform(new_user_data)

    # Compute cosine similarity between the new user and existing users
    similarity_scores = cosine_similarity(new_user_features, user_features_transformed)

    # Get indices of the most similar users
    most_similar_user_index = np.argsort(similarity_scores[0])[::-1]


    return most_similar_user_index
```

Code Snippet 10: get_recommendation Function

With the conclusion of this process, a new user is assigned exercises and can now experiment and interact with the system to express his preferences by interacting with the exercises.

After the user interacted with the exercises, the system switches to the collaborative filtering approach to provide new recommendations. This process starts by the user requesting new exercises in the client-side application. This action initiates a request that is once again handled by the RecommendationView, more specifically the **collaborative_recommendation_view()** function.

```
@api_view(['GET'])
@permission_classes([IsAuthenticated])
def collaborative_recommendation_view(request, user_id):
   algo = cache.get('collaborative_algorithm')
   all_exercises = Exercise.objects.values_list('id', flat=True)

   user_exercises =
set(UserExerciseInteraction.objects.filter(user_id=user_id).values_list('exercise_id',
flat=True))

   non_interacted_exercises = set(all_exercises) - user_exercises
   recommendations=[]

   # Generate predictions for non-interacted exercises
   for exercise_id in non_interacted_exercises:
         predicted_rating = algo.predict(user_id, exercise_id).est
         recommendations.append((exercise_id, predicted_rating))

   recommendations.sort(key=lambda x: x[1], reverse=True)

   top_exercise_ids = [exercise_id for exercise_id, _ in recommendations[:5]]

   recommended_exercises = Exercise.objects.filter(id__in=top_exercise_ids)

   serializer = ExerciseSerializer(recommended_exercises, many=True)

   return JsonResponse({'exercises':serializer.data})
```

Code Snippet 11: collaborative_recommendation_view Function

Code Snippet 11 represents how the view is implemented. It starts by retrieving the trained SVD model from the cache and identifies all exercises the user has not yet interacted with. The function then uses the model to predict which exercises the user is most likely to engage with positively, ultimately returning the top 5 recommended exercises.

In conclusion, the three phases form the core functionalities of the hybrid recommender system workflow. The Data initialization phase sets the foundation by generating and importing the necessary information. The System Configuration phase ensures the system is optimized and ready to provide recommendations. Finally, the Recommendation phase brings all these elements together to deliver exercise recommendations.

4.3.4 Authentication

Authentication is a critical component of any system, ensuring that only authenticated users have access to specific resources. Authentication serves as a layer of security that helps validate a user's identity, granting access to protected views.

When a user logs in, the system verifies their credentials and, if valid, issues an authentication token. This token is then included in request to prove the user's identity, allowing access to restricted areas of the system or performing requests that require authentication.

For this system, **Simple-JWT** was implemented as the authentication method. Simple-JWT is a plugin for DRF that provides JSON Web Tokens (JWT) for handling authentication (Sanders, 2020).

As shown in **Code Snippet 12**, this process revolves around two endpoints. The first endpoint, "login/" uses the **TokenObtainPairView** to issue a pair of tokens (an access token and a refresh token) when a user successfully logs in. This view uses Django's built-in User model (as mentioned earlier, a part of the authentication system) to check the credentials against existing users.

The second endpoint, "login/refresh/" uses the **TokenRefreshView**, which handles the refreshing process. When a token expires, the refresh token is sent to this endpoint to obtain a new access token without having to log in again.

The combine use of access tokens and refresh tokens offer a good balance between security and better user experience, this mechanism reduces the risk if an attacker gains access to a token.

```
from rest_framework_simplejwt.views import TokenObtainPairView, TokenRefreshView

    path('login/', TokenObtainPairView.as_view(), name='token_obtain_pair'),
    path('login/refresh/', TokenRefreshView.as_view(), name='token_refresh')
```

Code Snippet 12: Authentication Endpoints

4.3.5 Unit and API testing

To ensure the system's reliability, key functions were tested using unit tests. These unit tests were implemented using Django's test suite (Django Tests, 2024). These tests focus on critical actions, such as the is_relevant() function presented in section 4.3.1. Testing functions like these in isolation is essential, as they play a determining role in the system's outcomes. By isolating each function and verifying its correctness, we ensure that the individual components behave as expected, minimizing the risk of errors when they interact with other parts of the system.

```
  def test_is_level_appropriate(self):
    self.assertTrue(self.command.is_level_appropriate(self.user_profile.fitness_level,
self.exercise1))

  def test_are_goals_aligned(self):
      self.assertTrue(self.command.are_goals_aligned(self.user_profile.fitness_goals.all(),
self.exercise1))

  def test_is_injury_conflict(self):
    self.assertTrue(self.command.is_injury_conflict(self.user_profile.injury.all(),
self.exercise1))

  def test_is_condition_compatible(self):
      self.assertFalse(self.command.is_condition_compatible(self.user_profile.medical_con
ditions.all(), self.exercise1))

  def test_is_exercise_compatible_with_cluster(self):
      self.assertFalse(self.command.is_exercise_compatible_with_cluster(self.user_profile.
bmi_case,self.exercise2))

  def test_all_conditions_pass(self):
    self.assertTrue(self.command.is_relevant(
      self.exercise1,
      self.user_profile.fitness_goals.all(),
      self.user_profile.fitness_level.all(),
      self.user_profile.injury.all(),
      self.user_profile.medical_conditions.all(),
      self.user_profile.bmi_case
    ))
```

Code Snippet 13: is_relevant Test Case

Code Snippet 13 illustrates the set of unit tests implement to verify If the behavior of the is_relevant() function. These tests ensure that each function operates correctly in isolation before being integrated with other components.

In addition to unit testing, API testing was conducted using **Postman** as a form of integration testing. This type of testing is essential for ensuring that different parts of the application work together as expected when accessed through the API endpoints. By testing the endpoints, integration testing evaluates the interaction between various components such as views, models, the serializers and the recommendation algorithms. This helps identify any issues in the data flow and business logic within the system.

API testing plays an important role in verifying that the system provides correct responses under different conditions, such as handling various types of user input. It ensures that the application's endpoints are not only reachable but also accurately execute the intended functionality. This helps confirm the overall performance, reliability, and stability of the system, providing confidence that it will deliver recommendations to users (Postman, 2024).

4.4 Summary

This chapter covered the development and implementation steps of an application that integrates a switching hybrid recommender system using Python, the Django framework and the Scikit-learn and Surprise libraries. The application was structured into three primary phases, Data Initialization, System Configuration, and the Recommendation phase. These three phases combined created a process that begins with importing and generating data, preparing it for the recommendation algorithms and then providing exercise recommendations to users. Each step in this process contributes to building an efficient recommendation system that leverages both content-based and collaborative filtering.

Additionally, this chapter also covered the authentication and testing mechanisms implemented to secure and validate the system’s functionality. Together, these components form a robust framework for delivering exercise recommendations to users.

5 System Evaluation

The system evaluation focusses on verifying the functionality and effectiveness of the recommendation algorithms, as well as critical functions from Models, Management commands and Views.

Given the absence of real-world user data, the evaluation of the recommendation algorithms was conducted using synthetic user profiles and interactions. For the content-based algorithm 20 new profiles were created, consisting of 10 males and 10 females, each with varying physical and categorical attributes. For the physical attributes, **Mean Absolute Error** (MAE) and **Mean Squared Error** (MSE) were used, and for the categorical attributes **Precision**, **Recall** and **F1-score** were used.

The collaborative filtering model was tested using **Precison@K** and **Recall@K** to measure the relevance of recommended items, while **Root Mean Squared Error** (RMSE) and MAE were used to evaluate prediction accuracy based on user-item interactions.

Lastly, **Unit tests** were developed to target critical functions in the system, such as functions responsible for calculating user attributes determining exercise suitability for generating interactions. **API endpoint testing** was also conducted using Postman to validate API behavior. These tests ensured that the system responded appropriately to different input scenarios.

5.1 Content-based Filtering Algorithm Evaluation

For the content-based approach, five metrics were used to evaluate the algorithm's effectiveness. MAE and MSE were applied to measure the similarity between a new user and a recommended user regarding physical attributes such as height and weight. These metrics provide insight into how closely the algorithm matched numerical features. In addition, Precision, Recall, and F1-score were employed to assess how well the algorithm matched categorical attributes, including fitness goals and recommended exercises. These metrics were important for determining how effectively the system aligned users based on preferences and assigned exercises.

The Mean Absolute Error (MAE) is a measure used to evaluate how much the predictions deviate from the ground truth. It calculates the average absolute difference between the predicted and actual values. MAE is less sensitive to large deviations (outliers) because it treats all errors equally, without amplifying the impact of significant outliers as in other metrics such as MSE. This makes MAE useful to identify the presence of smaller and consistent errors.

Equation 6: Mean Absolute Error

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|x_i - x|$$

The MAE formula is represented in **Equation 6**, where x_i is the predicted value, x is the actual value and n the number of observations (Burch, 2023).

The Mean Squared Error (MSE) is another measure that evaluates how much the predictions deviate from the ground truth, but this measure accentuates larger errors, making it useful to emphasize large differences between the new user physical attributes and those of the recommended user. As shown in **Figure 36**, it is the average of the squares of the differences between predicted and actual values.

Equation 7: Mean Squared Error

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2$$

In the formula represented in **Equation 7**, x_i are the actual values, y_i are the predicted values and n the total number of predictions (Hossain, Choi and Chen, 2024).

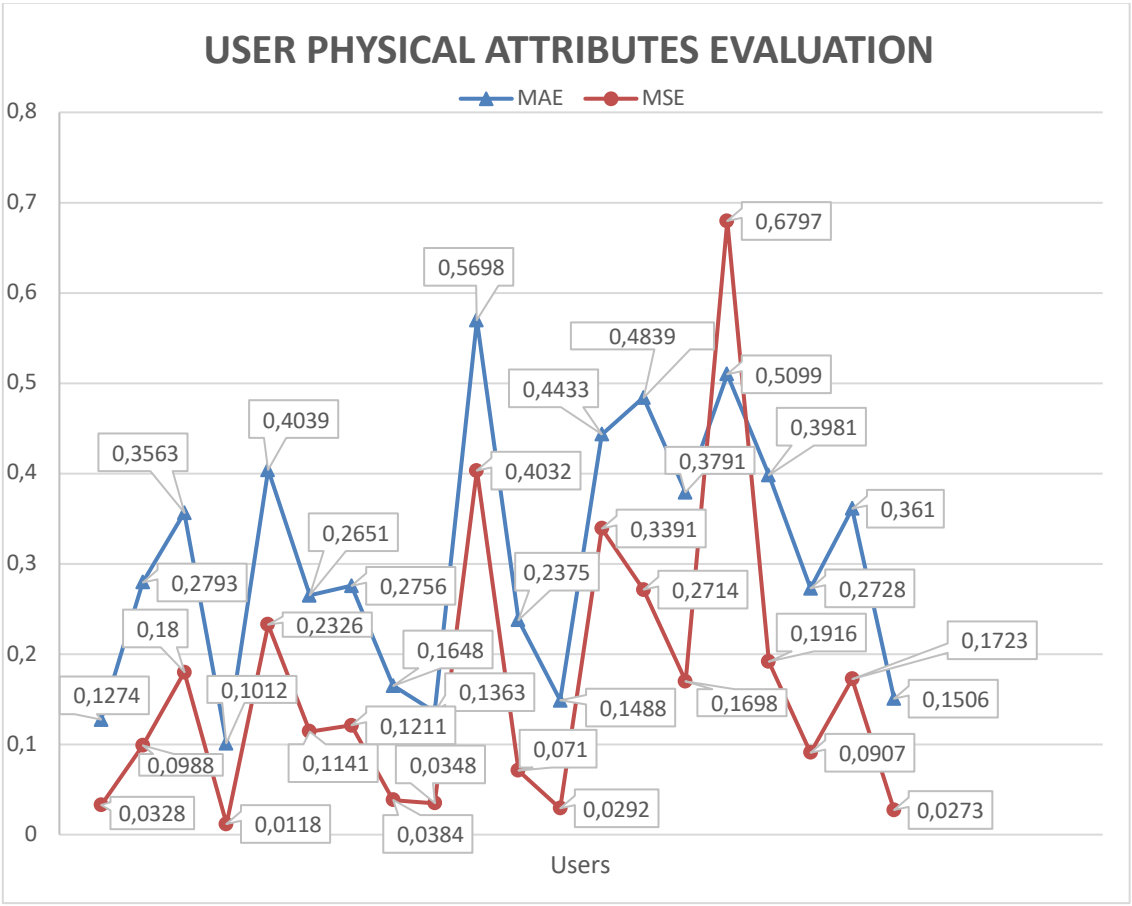

Figure 29: User Physical Attributes Evaluation Chart

In **Figure 29**, a line chart represents the results of applying the MAE and MSE metrics to assess the effectiveness of the cosine similarity calculation. This chart shows the values of these two metrics for 20 new user profiles, helping to illustrate how well the algorithm identified similar users based on their physical attributes. The data used in the evaluation was normalized, meaning that values like height and weight are scaled between 0 and 1 to ensure a consistent comparison across different features with varying units.

Nine users had low MSE (<0.1) values paired with moderate MAE (>0.1), indicating small but consistent discrepancies. Six users resulted in moderate MSE and MAE values, which translates to noticeable but not extreme mismatches. Three users had high MSE (>0.3) and MAE, indicating users for which the algorithm struggled to find more precise users. Two users had low values for both metrics, showing that the algorithm perform well in these cases.

Although most users showed moderate to high MAE values, the MSE values tended to be moderate to low for most users. In the context of a fitness application, where physical attributes can vary significantly across individuals, this can be considered a relatively good performance. The MSE values suggest that while there are discrepancies between users' physical attributes, they are generally not large enough to significantly impact the overall similarity calculation.

As mentioned earlier, to evaluate the effectiveness of the content-based filtering approach for categorical attributes, three metrics were implemented, Precision, Recall and F1-score. These metrics were selected because they offer complementary insights into how well the similarity calculations align a new user with a most similar user based on categorical features.

Precision measures the proportion of correct positive predictions out of the total positive predictions made by the algorithm, this metric is useful to evaluate the system's ability to avoid making incorrect matches.

Equation 8: Precision

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

The formula for the precision calculation is represented in **Equation 8**. True Positive is the number of categories the new user and the similar user have in common; False Positive is the number of categories the algorithm predicted would match with the new user but aren't present.

Recall measures the proportion of actual positives that were identified by the algorithm. This metric evaluates how well the system identified all relevant categorical attributes. The formula to calculate the recall in presented **Equation 9**. This formula is identic to the precision formula, but instead of using False Positives, the recall measure uses False Negatives, that is the number of relevant matches that were missed by the system.

Equation 9: Recall

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

F1-score combines both Precision and Recall into one performance metric, balancing both measures and giving an overall evaluation on the system's effectiveness. The formula for the calculation of F1-score is presented in **Equation 10**.

Equation 10: F1-Score

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

For all metrics, macro-averaging was implemented. This approach ensures that each category is treated equally, providing a balanced assessment of the system's ability to match users based on categorical features (Kundu, 2022).

In **Figure 30**, a line chart represents the application of the discussed metrics to assess if the algorithm matched a new user with the expected set of exercises.

The evaluation showed that Recall is consistently 1 for all users. This consistency is expected because all users were assigned plans from a static set of exercise plans and underwent the same refinement. However, Precision and F1-score registered lower values for some users, particularly one user registered the low value of 0.2592 for precision and 0.4118 for F1-score. These lower scores suggest that while the system recalled the relevant exercises, it also

included several irrelevant ones, reducing the overall recommendation quality. Users with more specific conditions experience this over-prediction more frequently.

Overall, while there is room for improvement, particularly in reducing irrelevant recommendations for users with specific needs, the results indicate that the system effectively recommended relevant exercises.

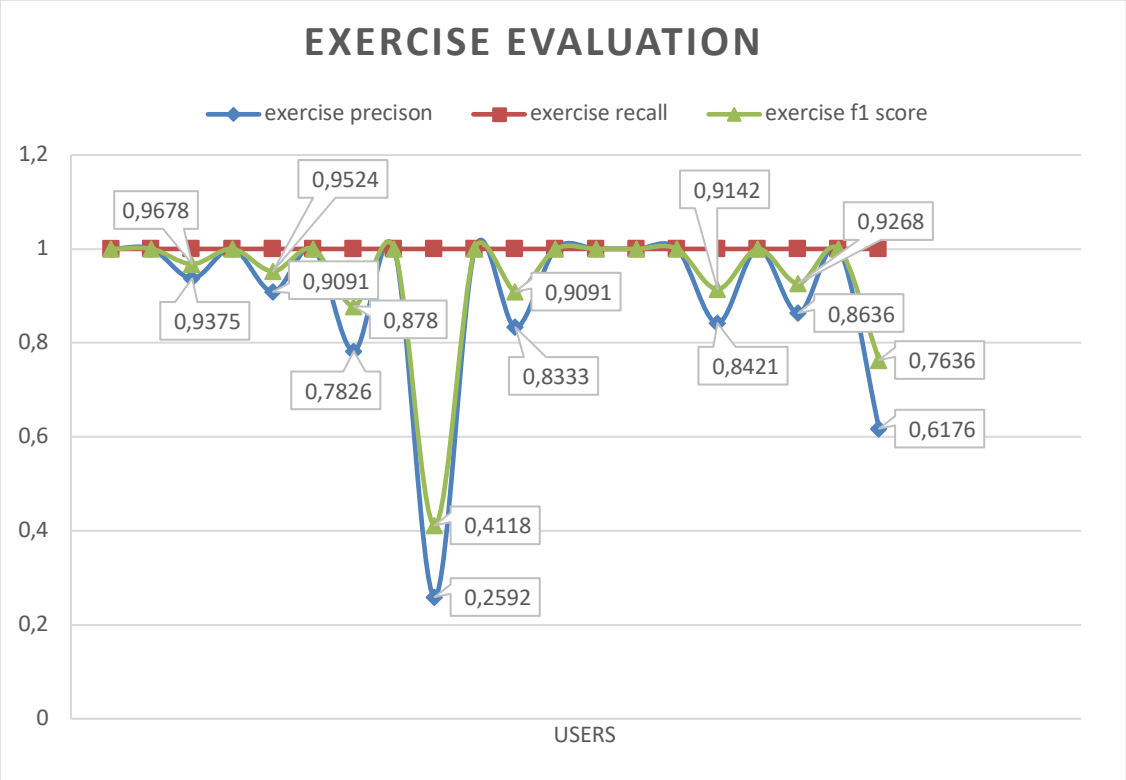

Figure 30: Exercise Evaluation Chart

Figure 31 represents the line chart that resulted from the application of Precision, Recall and F1-score for the remaining user categorical attributes, such as medical conditions and fitness goals. This evaluation also demonstrates a strong performance, with high values for all metrics across most users. However, the same users that registered the lowest values for the exercise evaluation, stand out again in this evaluation, with the lowest F1-score with the value of 0,6667, indicating that the system is predicating irrelevant attributes more often to them.

Overall, the system preforms well in identifying similar users based on categorical attributes, but users with more diverse need experience some decline in Precision.

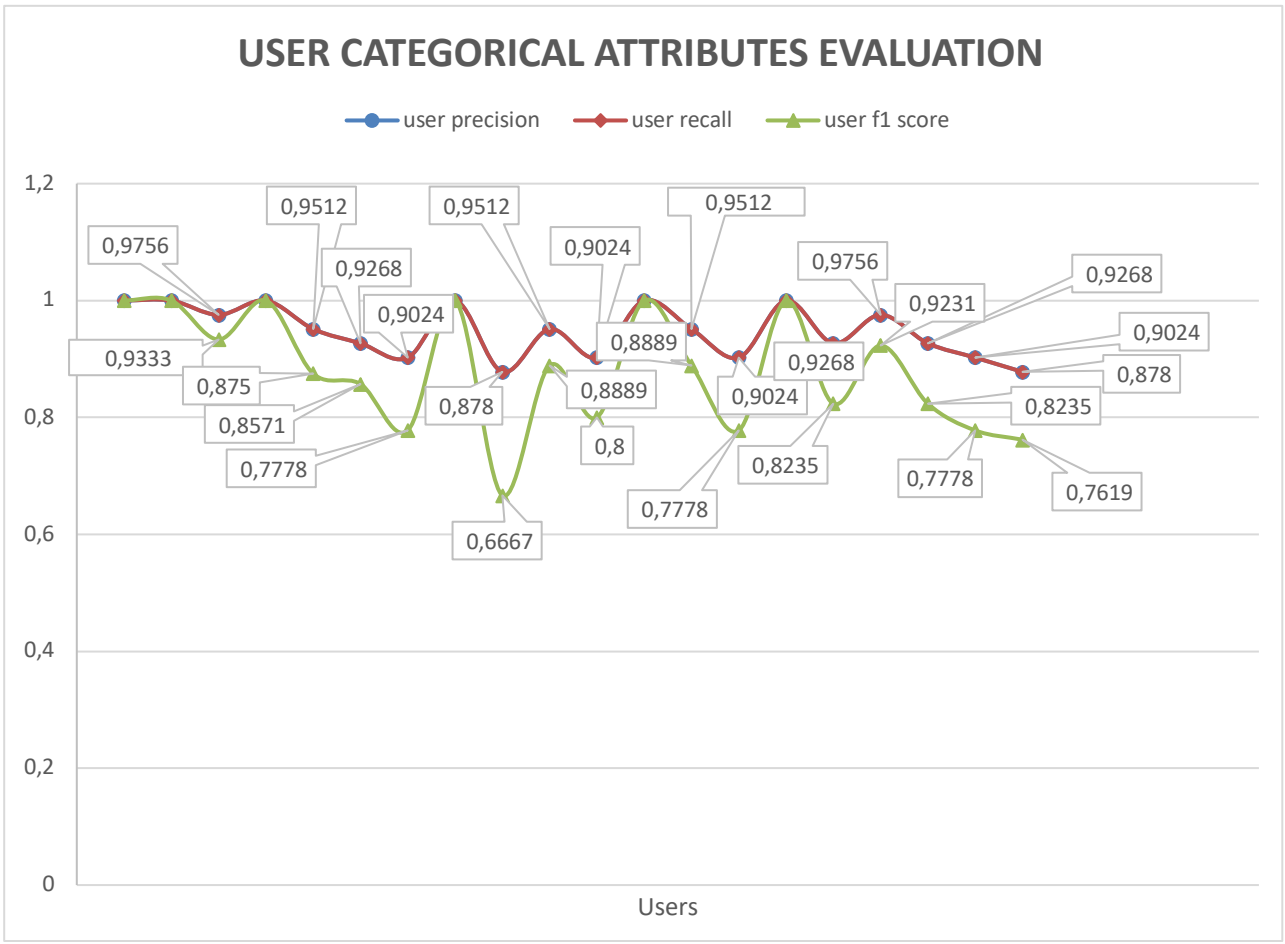

Figure 31: User Categorical Attributes Evaluation Chart

In conclusion, the evaluation shows that the content-based filtering algorithm performs effectively overall in matching users based on attributes. However, the system struggles with users who have a higher number of specific conditions, often leading to irrelevant recommendations. This highlights the need for a more diverse user base, so that when users with unique combinations appear, there are better matching options available.

5.2 Collaborative Filtering Algorithm Evaluation

The collaborative filtering algorithm was evaluated using five metrics, including **Precision@K**, **Recall@K**, F1-Score@K, **MAE** and **Root Mean Squared Error** (RMSE). These metrics were chosen to assed both the quality of the recommendations and the accuracy of the rating predictions. Precision@K, Recall@K and F1-Score@K measure how well the system recommends relevant items within the top K results, while MAE and RSME provide insights into how closely the predicted ratings match the actual user ratings.

Precision@K, Recall@K and F1-Score@K are variations of the standard Precision, Recall and F1-Score metrics, adjusted to account for the top **K** items recommended to a user.

Precision@K evaluates the proportion of relevant items within the top K recommendations, providing insights into how accurate the recommendations are when the system suggests a limited number of items. **Equation 11** represents the formula of this variation

Equation 11: Precision@K

$$Precision@K = \frac{Number\ of\ Relevant\ items\ in\ Top\ K}{K}$$

Recall@K measures how many of the total relevant items are included in the top K recommendations, capturing the number of relevant items in a specific number of recommendations. **Equation 12** represents the formula of this variation.

Equation 12: Recall@K

$$Recall@K = \frac{Number\ of\ Relevant\ item\ in\ Top\ K}{Total\ Number\ of\ relevant\ items}$$

F1-Score@K represents the same type of measure than F1-Score, it is the combination of Precison@K and Recall@K, it accounts for both how accurate the recommendations are and how many relevant items are retrieved. As shown in **Equation 13**, the formula is identical as well.

Equation 13: F1-Score@K

$$F1 - Score@K = 2 \times \frac{Precision@K \times Recall@K}{Precision@K + Recall@K}$$

Root Mean Squared Error (RMSE) is used to evaluate the accuracy of rating predictions. RMSE is derived from MSE and adjusts the error back to the same scale as the original ratings. Both metrics emphasize larger errors, but RMSE is easier to interpret since it is expressed in the same units as the original ratings. As shown in **Equation 14**, the formula for this metric is the same as for MSE, but with the added square root calculation (Olumide, 2023).

Equation 14: Root Mean Squared Error

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2}$$

Table 13 presents the evaluation results of the collaborative filtering algorithm across different values of K, using the described metrics.

Across all K values, Precision@K, Recall@K, and F1-Score@K remain stable. Precision@K ranges from 0.7246 to 0.7339, indicating that the proportion of relevant recommendations is consistently high. Similarly, Recall@K stays within the range of 0.6290 and 0.6464, showing that the system retrieves a substantial portion of relevant items in the top K recommendations.

The F1-Score@K varies slightly between 0.6719 and 0.6857 at K=20, possibly indicating this is the optimal point for balancing accuracy and coverage.

The MAE and RMSE values are also consistent, with MAE ranging from 0.9763 to 0.9851 and RMSE ranging from 1.2100 to 1.2219. These results indicate that the rating predictions made by the system are fairly accurate, with average deviations from the actual ratings being less than 1 point on the rating scale. RMSE, is slightly higher than MAE, highlighting that there are some larger prediction errors, but they remain within a reasonable range.

Table 13: Collaborative Filtering Algorithm Evaluation Results

K	Precision@K	Recall@k	MAE	RMSE	F1-Score@k
5	0.7339	0.6290	0.9772	1.2128	0.6719
10	0.7252	0.6464	0.9816	1.2157	0.6773
15	0.7265	0.6377	0.9763	1.2107	0.6742
20	0.7246	0.6408	0.9851	1.2219	0.6857
25	0.7287	0.6464	0.9819	1.2188	0.6751
30	0.7311	0.6361	0.9788	1.2100	0.6764

In conclusion, the collaborative filtering algorithm demonstrates good performance across all evaluation metrics, delivering relevant recommendations with reasonable accuracy. Precision@K, Recall@K, and F1-Score@K show the system provides relevant items. In terms of rating predictions, MAE and RMSE indicate decent accuracy, with errors typically below 1 point. However, these values also reflect that there is clear room for improvement, refining recommendations and making rating predictions more precise could enhance the system overall.

5.3 Discussion

The system evaluation highlights both strengths and areas needing improvement. The content-based algorithm generally performed well in matching users based on physical and categorical attributes. The MSE values indicate that, in most cases, the differences between new users and their matched users were not large, suggesting that the cosine similarity calculation was effective. However, with users that have a more unique combination of conditions, the precision values were lower, indicating that the system can recommend irrelevant exercises or mismatch user attributes. This suggests a limitation in the system's ability to handle outlier cases.

The collaborative filtering algorithm presented consistency in delivering relevant recommendations, as shown by the stable metrics across the different values of K. The rating prediction metrics also suggested accurate predictions, however, there is a slight tendency towards larger errors, as indicated by the RMSE being higher than the MAE. This performance implies that while the collaborative filtering method is robust, there's still room for improvement, especially in reducing larger prediction errors to increase overall accuracy.

Overall, while the system demonstrates a solid foundation for providing recommendations, enhancing the system's ability to handle outlier cases and prediction accuracy could improve its reliability. The content-based approach could benefit from a more diverse user base and the collaborative filtering could be enhanced through further parameter tuning, for example, adjusting the regularization parameter.

6 Conclusion

This document has explored the development and evaluation of a fitness application incorporating a switching hybrid recommender system focused on exercise recommendations.

The literature review initially identified key gaps in existing health recommender system, particularly regarding studies of the long-term effectiveness of these systems, concrete data privacy measures, user motivation assessment and the need for a hybrid approach to overcome algorithm's limitations. The conclusion of that review guided the system's objective, to design and implement a hybrid recommender system that provides accurate exercise recommendations to enhance health and fitness.

The system and design chapter outlined both functional and non-functional requirements of the system, as well as the chosen architecture and the sequence diagrams to achieve the core functionalities. These functionalities included user registration, data input, exercise interaction and recommendations. The non-functional requirements gave considerations for security, performance and usability. This chapter also introduced the decision to use a monolithic architecture for simplicity and rapid development. This approach integrated all components, such as data models, views, serializers and recommendation algorithms into a unified system, suitable for a prototype demonstration.

The implementation chapter detailed the process of implementing a switching hybrid recommender system using content-based and collaborative filtering methods. The implementation involved transforming user and exercise datasets into structured models and configuring a server-side application with Django for data processing and recommendation generation.

Lastly, the system evaluation involved testing the recommendation algorithms using various metrics, including Mean Absolute Error, Mean Squared Error, Root Mean Squared Error

Precision, Recall, F1-Score and their @K variations. The content-based approach demonstrated a solid performance in matching users based on their physical and categorical attributes, however it revealed limitations with users that had more complex combinations of conditions. The collaborative filtering approach also performed well in delivering relevant recommendations, although, its accuracy could be improved with parameter tuning. Overall, these results provided confidence in the system and a path for future work.

6.1 Objectives

The main objective of the work detailed in this document was to demonstrate how can the use of hybrid techniques in a recommender system be used to improve health and fitness outcomes.

This objective was accomplished through the successful integration of content-based and collaborative filtering techniques. The switching hybrid system effectively addressed key challenges, such as the cold start problem for new users, and refined its recommendations based on users' interactions, ensuring more accurate exercise suggestions.

The evaluation of the system showcased its capabilities, particularly because it was tested with users possessing diverse profiles and characteristics. The content-based approach proved to have an effective performance in recommending exercises to new users based on their attributes. Additionally, the collaborative filtering method also proved to be efficient in enhancing the relevance of suggestions by learning from user-exercise interactions. With the involvement of health experts, it was possible to an application that demonstrated how these techniques can provide tailored recommendations, supporting users in improving their health and fitness outcomes.

Another objective was to adopt a user-centric approach, this objective was achieved by incorporating factors like fitness levels, goals, injuries and medical conditions. These factors ensured that the system promoted safe and appropriate exercise recommendations. Extensive testing validated the capability of a hybrid approach in considering these factors to produce recommendations.

The objective of studying exercises that could be performed without equipment or with basic equipment was partly accomplished. The exercise dataset included a "equipment_needed" attribute that was implemented in the exercise model. Although this attribute was not directly used in the recommendation process, it remains part of the model and can still indirectly influence recommendations through user interactions. Additionally, this attribute provides a valuable foundation for future enhancements, potentially allowing the system to better meet this objective.

The objectives of incorporating dietary orientation, a therapeutic component, and implementing a client-side application were not achieved, due to delays in the development of the recommender system.

6.2 Future Development

To further enhance this system there are several steps that could be considered.

Initially, creating a client-side application would enable the system to be tested extensively in real-world scenarios, facilitating the development of a more diverse user base.

Next, the transition from a monolithic architecture to a modular, microservices architecture allow the system to scale more efficiently and handle high demand more effectively.

Additionally, switching from a relational database to a non-relational database, such as Redis, would improve performance and reliability. Redis, is designed for horizontal scaling and can persist cached data to disk, which prevents the need to preprocess data every time the system restarts. It was noted during testing that restarting the system sometimes would lead to different recommendations, and a solution like Redis would eliminate inconsistencies.

Another enhancement involves further testing on how parameter tuning would improve the recommendation algorithms, leading to more accurate recommendations.

Implementing database encryption is a key strategy to enhance data security. By encrypting sensitive data stored in the database with methods such as Advanced Encryption Standard would safeguard information from unauthorized access.

Lastly, expanding the system's models to include dietary and therapeutic recommendations would fulfill the project's initial objectives, providing a more comprehensive health and fitness application.

7 Bibliography

NIST (no date) *1.3.3.24. Quantile-Quantile Plot*. Available at: https://www.itl.nist.gov/div898/handbook/eda/section3/qqplot.htm (Accessed: 26 August 2024).

World Health Organization (2010) *A healthy lifestyle - WHO recommendations*. Available at: https://www.who.int/europe/news-room/fact-sheets/item/a-healthy-lifestyle---who-recommendations (Accessed: 26 August 2024).

SQLite (2023) *About SQLite*. Available at: https://www.sqlite.org/about.html (Accessed: 3 September 2024).

Albahri, A.S. *et al.* (2023) 'A systematic review of trustworthy and explainable artificial intelligence in healthcare', *Information Fusion*, 96, pp. 156–191. Available at: https://doi.org/10.1016/J.INFFUS.2023.03.008.

Ali, O. *et al.* (2023) 'A systematic literature review of artificial intelligence in the healthcare sector: Benefits, challenges, methodologies, and functionalities', *Journal of Innovation & Knowledge*, 8(1), p. 100333. Available at: https://doi.org/10.1016/J.JIK.2023.100333.

Blinowsk, G. (2022) *IEEE Xplore Full-Text PDF:* Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9717259 (Accessed: 22 August 2024).

Dorwart, L. (2024) *Body Fat Percentage: Charting Averages in Men and Women*. Available at: https://www.msn.com/en-us/health/wellness/body-fat-percentage-charting-averages-in-men-and-women/ar-BB1iVeuN (Accessed: 26 August 2024).

Bono, R. *et al.* (2017) 'Non-normal distributions commonly used in health, education, and social sciences: A systematic review', *Frontiers in Psychology*, 8(SEP), p. 267072. Available at: https://doi.org/10.3389/FPSYG.2017.01602/BIBTEX.

Bourgais, M. *et al.* (2022) 'Avoiding the Overspecialization of Recommender Systems in Tourism with Semantic Trajectories, Initial Thoughts', *Procedia Computer Science*, 207, pp. 1933–1942. Available at: https://doi.org/10.1016/J.PROCS.2022.09.252.

Caballero, P. *et al.* (2021) 'Paving the way to collaborative context-aware mobile applications: a case study on preventing worsening of allergy symptoms', 80, pp. 21101–21133. Available at: https://doi.org/10.1007/s11042-021-10759-6.

Cai, Y. *et al.* (2022) 'Health Recommender Systems Development, Usage, and Evaluation from 2010 to 2022: A Scoping Review', *International journal of environmental research and public health*, 19(22). Available at: https://doi.org/10.3390/IJERPH192215115.

Cruz, J. (2022) *caixa de bigodes (boxplot) — Documentação Bioestatística*. Available at: https://sweet.ua.pt/pedrocruz/bioestatistica/ed-boxplot.html#gsc.tab=0 (Accessed: 26 August 2024).

Coppens, I., De Pessemier, T. and Martens, L. (2023) 'Connecting physical activity with context and motivation: a user study to define variables to integrate into mobile health recommenders', *User Modeling and User-Adapted Interaction*, 34(1), pp. 147–181. Available at: https://doi.org/10.1007/S11257-023-09368-9/FIGURES/8.

Coppens, I., De Pessemier, T. and Martens, L. (2024) 'Exploring the added effect of three recommender system techniques in mobile health interventions for physical activity: a longitudinal randomized controlled trial', *User Modeling and User-Adapted Interaction*, pp. 1–56. Available at: https://doi.org/10.1007/S11257-024-09407-Z/FIGURES/18.

Liu, C. (2022) *Data Transformation: Standardization vs Normalization - KDnuggets*. Available at: https://www.kdnuggets.com/2020/04/data-transformation-standardization-normalization.html (Accessed: 31 August 2024).

Diário da República (2021) 'Diário da República, 2.ª série PARTE E Artigo 2.º'.

Django Overview (2024) *Django overview | Django*. Available at: https://www.djangoproject.com/start/overview/ (Accessed: 3 September 2024).

Django Cache (2024) *Django's cache framework | Django documentation | Django*. Available at: https://docs.djangoproject.com/en/5.1/topics/cache/ (Accessed: 5 September 2024).

Dragoni, N. *et al.* (2017) *(PDF) Microservices: yesterday, today, and tomorrow*. Available at: https://www.researchgate.net/publication/315664446_Microservices_yesterday_today_and_tomorrow (Accessed: 22 August 2024).

European Parliament (2016) '(General Data Protection Regulation) (Text with EEA relevance)'.

Kundu, R. (2022) *F1 Score in Machine Learning: Intro & Calculation*. Available at: https://www.v7labs.com/blog/f1-score-guide (Accessed: 9 September 2024).

Fernandez, M. and Bellogín, A. (2020) 'Recommender Systems and Misinformation: The Problem or the Solution?' Available at: https://www.buzzfeednews.com/ (Accessed: 5 January 2024).

Park, S. *et al.* (2019) *From treatment to prevention: The evolution of digital healthcare*. Available at: https://www.nature.com/articles/d42473-019-00274-6 (Accessed: 5 January 2024).

GeeksforGeeks (2024) *Functional vs Non Functional Requirements - GeeksforGeeks*. Available at: https://www.geeksforgeeks.org/functional-vs-non-functional-requirements/ (Accessed: 20 August 2024).

Gough (2021) *Health/fitness clubs membership worldwide by region 2009-2019 | Statista*. Available at: https://www.statista.com/statistics/273069/members-of-health-clubs-worldwide-by-region/ (Accessed: 17 October 2023).

Pandit, N. (2022) *Gym Exercise Dataset*. Available at: https://www.kaggle.com/datasets/niharika41298/gym-exercise-data/data (Accessed: 27 August 2024).

Hinojosa-Nogueira, D. *et al.* (2023) 'Stance4Health Nutritional APP: A Path to Personalized Smart Nutrition', *Nutrients 2023, Vol. 15, Page 276*, 15(2), p. 276. Available at: https://doi.org/10.3390/NU15020276.

IBM (2006) *Guideline: Data Model*. Available at: https://home.iscte-iul.pt/~hro/RUPSmallProjects/core.base_rup/guidances/guidelines/data_model_80FB2539.html (Accessed: 12 September 2024).

IBM (2021) *Domain models - IBM Documentation*. Available at: https://www.ibm.com/docs/en/ida/9.1.2?topic=types-domain-models (Accessed: 21 August 2024).

Jangde, M.P. and Ramaiya, M. (2022) 'Health Recommender Systems: A Systematic Review'.

Januzaj, Y. and Luma, A. (2022) 'Cosine Similarity – A Computing Approach to Match Similarity Between Higher Education Programs and Job Market Demands Based on Maximum Number of Common Words', *International Journal of Emerging Technologies in Learning*, 17(12), pp. 258–268. Available at: https://doi.org/10.3991/IJET.V17I12.30375.

Joseph, V.R. (2022) 'Optimal ratio for data splitting', *Statistical Analysis and Data Mining*, 15(4), pp. 531–538. Available at: https://doi.org/10.1002/SAM.11583.

King, A.P. and Eckersley, R.J. (2019) 'Inferential Statistics IV: Choosing a Hypothesis Test', *Statistics for Biomedical Engineers and Scientists*, pp. 147–171. Available at: https://doi.org/10.1016/B978-0-08-102939-8.00016-5.

Koren, Y., Bell, R. and Volinsky, C. (2009) '42 COVER FE ATURE'.

Liu, X. *et al.* (2023) 'Privacy-Preserving Personalized Fitness Recommender System P3FitRec: A Multi-level Deep Learning Approach', *ACM Transactions on Knowledge Discovery from Data*, 17(6). Available at: https://doi.org/10.1145/3572899.

Mahmoud, M. (2024) *fitness exercises using BFP & BMI*. Available at: https://www.kaggle.com/datasets/mustafa20635/fitness-exercises-using-bfp-and-bmi/data?select=final_dataset_BFP+.csv (Accessed: 26 August 2024).

Hug, N. (2015) *Matrix Factorization-based algorithms — Surprise 1 documentation*. Available at: https://surprise.readthedocs.io/en/stable/matrix_factorization.html (Accessed: 3 September 2024).

Burch, D. (2023) *Mean Absolute Error In Machine Learning: What You Need To Know - Arize AI*. Available at: https://arize.com/blog-course/mean-absolute-error-in-machine-learning-what-you-need-to-know/ (Accessed: 9 September 2024).

Hossain, S., Choi, Y. and Chen, M. (2024) *Mean Squared Error (MSE): How to calculate and interpret MSE for regression ML tasks · Testing with Kolena*. Available at: https://docs.kolena.com/metrics/mean-squared-error/ (Accessed: 9 September 2024).

Mekni, M. *et al.* (2017) 'Software Architectural Design in Agile Environments', *Journal of Computer and Communications*, 6(1), pp. 171–189. Available at: https://doi.org/10.4236/JCC.2018.61018.

Django Models (2024) *Models | Django documentation | Django*. Available at: https://docs.djangoproject.com/en/5.1/topics/db/models/#querysets-still-return-the-model-that-was-requested (Accessed: 4 September 2024).

Mustaqeem, A., Anwar, S.M. and Majid, M. (2020) 'A modular cluster based collaborative recommender system for cardiac patients', *Artificial Intelligence in Medicine*, 102, p. 101761. Available at: https://doi.org/10.1016/J.ARTMED.2019.101761.

Ngoc, T. *et al.* (2020) 'Recommender systems in the healthcare domain: state-of-the-art and research issues'. Available at: https://doi.org/10.1007/s10844-020-00633-6.

Orue-saiz, I., Kazarez, M. and Mendez-zorrilla, A. (2021) 'Systematic review of nutritional recommendation systems', *Applied Sciences (Switzerland)*. MDPI. Available at: https://doi.org/10.3390/app112412069.

Page, M.J. *et al.* (2021) 'The PRISMA 2020 statement: an updated guideline for reporting systematic reviews', *BMJ*, 372. Available at: https://doi.org/10.1136/BMJ.N71.

pandas (2024) *pandas - Python Data Analysis Library*. Available at: https://pandas.pydata.org/ (Accessed: 26 August 2024).

Paradigm, V. (2024) *Use Case Analysis: How to Identify Actors?* Available at: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/how-to-identify-actors/ (Accessed: 20 August 2024).

Prugel-Bennett and Ghazanfar (2010) *(PDF) Building Switching Hybrid Recommender System Using Machine Learning Classifiers and Collaborative Filtering*. Available at: https://www.researchgate.net/publication/45692369_Building_Switching_Hybrid_RecommenderSystem_Using_Machine_Learning_Classifiers_andCollaborative_Filtering (Accessed: 11 September 2024).

Petersen, C. *et al.* (2022) 'AMIA Position Paper AMIA's code of professional and ethical conduct 2022'. Available at: https://doi.org/10.1093/jamia/ocac192.

Pinto, A. *et al.* (2023) 'Recommendation systems to promote behavior change in patients with diabetes mellitus type 2: A systematic review', *Expert Systems with Applications*. Elsevier Ltd. Available at: https://doi.org/10.1016/j.eswa.2023.120726.

Principal (2018) *Fitness. Ginásios querem captar mais meio milhão de portugueses*. Available at: https://www.dinheirovivo.pt/empresas/fitness-ginasios-querem-captar-mais-meio-milhao-de-portugueses-12801170.html (Accessed: 17 October 2023).

Olumide, S. (2023) *Root Mean Square Error (RMSE) In AI: What You Need To Know - Arize AI*. Available at: https://arize.com/blog-course/root-mean-square-error-rmse-what-you-need-to-know/ (Accessed: 10 September 2024).

Samad, S. *et al.* (2022) 'Smartphone apps for tracking food consumption and recommendations: Evaluating artificial intelligence-based functionalities, features and quality of current apps', *Intelligent Systems with Applications*, 15, p. 200103. Available at: https://doi.org/10.1016/J.ISWA.2022.200103.

Sartori, F., Savi, M. and Talpini, J. (2022) 'Tailoring mHealth Apps on Users to Support Behavior Change Interventions: Conceptual and Computational Considerations', *Applied Sciences (Switzerland)*, 12(8). Available at: https://doi.org/10.3390/app12083782.

Scaled Agile (2023) *Nonfunctional Requirements - Scaled Agile Framework*. Available at: https://scaledagileframework.com/nonfunctional-requirements/ (Accessed: 20 August 2024).

Scikit-Learn (2024) *scikit-learn: machine learning in Python — scikit-learn 1.5.1 documentation*. Available at: https://scikit-learn.org/stable/index.html (Accessed: 2 September 2024).

Sengan, S. *et al.* (2021) 'A Secure Recommendation System for Providing Context-Aware Physical Activity Classification for Users', *Security and Communication Networks*, 2021. Available at: https://doi.org/10.1155/2021/4136909.

Sequeira (2024) *Como a atividade física beneficia a saúde – Revista Atletismo*. Available at: https://revistaatletismo.com/como-a-atividade-fisica-beneficia-a-saude/ (Accessed: 17 October 2023).

Sanders, D. (2020) *Simple JWT — Simple JWT 5.2.2.post30+gfaf92e8 documentation*. Available at: https://django-rest-framework-simplejwt.readthedocs.io/en/latest/index.html (Accessed: 5 September 2024).

Sun, Y. *et al.* (2023) 'When Recommender Systems Snoop into Social Media, Users Trust them Less for Health Advice', *Conference on Human Factors in Computing Systems - Proceedings* [Preprint]. Available at: https://doi.org/10.1145/3544548.3581123.

Hug, N. (2024) *Surprise · A Python scikit for recommender systems.* Available at: https://surpriselib.com/ (Accessed: 2 September 2024).

Tatyaso, P.R. and Khaiyum, Dr.S. (2023) 'Data Pre-processing Techniques Of The Performance Of Recommendation Systems', *International Journal of Engineering Research & Technology*, 11(6). Available at: https://doi.org/10.17577/NCRTCA-PID-030.

Tiu, E. (2020) *Understanding Latent Space in Machine Learning | by Ekin Tiu | Towards Data Science*. Available at: https://towardsdatascience.com/understanding-latent-space-in-machine-learning-de5a7c687d8d (Accessed: 2 September 2024).

Django Tests (2024) *Unit tests | Django documentation | Django*. Available at: https://docs.djangoproject.com/en/5.1/internals/contributing/writing-code/unit-tests/ (Accessed: 14 September 2024).

Vairavasundaram, S. *et al.* (2022) 'Dynamic Physical Activity Recommendation Delivered through a Mobile Fitness App: A Deep Learning Approach', *Axioms*, 11(7). Available at: https://doi.org/10.3390/axioms11070346.

Valentine, L., D'Alfonso, S. and Lederman, R. (2022) 'Recommender systems for mental health apps: advantages and ethical challenges', *AI & SOCIETY*, 38(4), pp. 1627–1638. Available at: https://doi.org/10.1007/S00146-021-01322-W.

Wellhub Editorial Team (2024) *A importância da atividade física para a saúde física e mental*. Available at: https://gympass.com/pt-br/blog/estilo-de-vida/atividade-fisica-para-saude-fisica-e-mental/ (Accessed: 17 October 2023).

Postman (2024) *What is API Testing? A Guide to Testing APIs | Postman*. Available at: https://www.postman.com/api-platform/api-testing/ (Accessed: 10 September 2024).

Murel, J. and Kavlakoglu, E. (2024) *What is content-based filtering? | IBM*. Available at: https://www.ibm.com/topics/content-based-filtering (Accessed: 29 August 2024).

Echeburúa, A. (2024) *What Is One Hot Encoding and How to Implement It in Python | DataCamp*. Available at: https://www.datacamp.com/tutorial/one-hot-encoding-python-tutorial (Accessed: 2 September 2024).

Wieczorek, M. *et al.* (2023) 'The ethics of self-tracking. A comprehensive review of the literature', *Ethics and Behavior*, 33(4), pp. 239–271. Available at: https://doi.org/10.1080/10508422.2022.2082969.

Xash (2024) *Fitness_Excercise_Recommendations*. Available at: https://www.kaggle.com/code/xash707/fitness-excercise-recommendations (Accessed: 27 August 2024).