UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

School of Telecommunications Engineering

Recommender System for Computer Components

Master's Thesis

Master's Degree in Telecommunication Engineering

AUTHOR: Herzog, Aljaz

Tutor: Fernández Diego, Marta

ACADEMIC YEAR: 2024/2025

Resumen

La rápida expansión del comercio electrónico ha transformado la experiencia de compra de los consumidores, ofreciendo acceso sin precedentes a productos. Sin embargo, esta vasta gama de opciones también ha introducido desafíos tanto para los usuarios, que se sienten abrumados por la cantidad de elecciones, como para las empresas, que buscan ofrecer recomendaciones personalizadas. Los sistemas de recomendación se han convertido en herramientas clave para abordar estos desafíos, permitiendo la entrega de sugerencias adaptadas a las preferencias de los usuarios. El dominio de los componentes para ordenadores presenta obstáculos únicos para los sistemas de recomendación, como la necesidad de compatibilidad técnica, el problema del arranque en frío y la complejidad de los requisitos de los usuarios para casos de uso altamente específicos, como juegos, edición de video o productividad general.

Esta tesis desarrolla un sistema de recomendación híbrido adaptado a los componentes para ordenadores, integrando los algoritmos de Filtrado Colaborativo (CF) y Filtrado Basado en Contenido (CBF). Se diseñó una arquitectura modular del sistema, que incluye un backend en Flask, una base de datos MongoDB y un frontend basado en React. Se implementaron cuatro algoritmos: un recomendador base utilizando similitud semántica, un modelo CBF basado en características de los artículos, un sistema CF basado en patrones de interacción usuario-artículo y un modelo híbrido que combina las fortalezas tanto de CBF como de CF. Se emplearon técnicas avanzadas, como incrustaciones de oraciones y codificación one-hot, para mejorar las capacidades de los modelos y garantizar su flexibilidad y escalabilidad.

La evaluación experimental reveló fortalezas y limitaciones distintas de cada algoritmo. Aunque el recomendador base fue efectivo para generar recomendaciones, resultó ser intensivo en cuanto a recursos computacionales y menos práctico para grandes conjuntos de datos. Por contra, los modelos CBF y CF demostraron un mejor rendimiento, siendo el enfoque CBF más eficaz al aprovechar los atributos de los artículos, mientras que el modelo CF mostró fortalezas al identificar patrones a partir de las interacciones de los usuarios. El sistema híbrido emergió como la solución más equilibrada, ofreciendo una combinación efectiva de precisión, diversidad y eficiencia computacional.

Los hallazgos validan que los algoritmos adaptados a los desafíos específicos de la recomendación de componentes para ordenadores mejoran significativamente la precisión y la satisfacción del usuario. Las técnicas de aprendizaje automático, especialmente en el modelo híbrido,

proporcionaron recomendaciones más personalizadas y adaptables. La preparación adecuada de los datos, que incluye el procesamiento de atributos e integración de múltiples fuentes de datos, jugó un papel fundamental en la mejora de la calidad general de las recomendaciones. El enfoque híbrido demostró su capacidad para escalar eficazmente mientras mantenía la precisión y la relevancia.

En conclusión, esta investigación contribuye al desarrollo de sistemas de recomendación específicos para dominios. Al abordar los requisitos únicos de las recomendaciones de componentes para ordenadores a través de un enfoque híbrido, la tesis resalta la importancia de integrar múltiples técnicas de recomendación y adaptarlas a las necesidades específicas de los usuarios. Estos hallazgos avanzan en el campo del comercio electrónico y establecen una base para futuros trabajos que apliquen redes neuronales y aprendizaje en tiempo real a dominios complejos, asegurando tanto la precisión como la adaptabilidad en las recomendaciones personalizadas.

Abstract

The rapid expansion of e-commerce has transformed consumer shopping experiences, offering unprecedented access to products. However, this vast array of options has also introduced challenges for users overwhelmed by choice and businesses seeking to provide personalized recommendations. Recommender systems have become critical tools for addressing these challenges, enabling the delivery of tailored suggestions based on user preferences. The domain of computer components presents unique obstacles for recommender systems, such as the need for technical compatibility, the cold start problem, and the complexity of user requirements in highly specific use cases like gaming, video editing, or general productivity.

This thesis develops a hybrid recommender system tailored for computer components by integrating Collaborative Filtering (CF) and Content-Based Filtering (CBF) algorithms. A modular system architecture was designed, featuring a Flask backend, a MongoDB database, and a React-based frontend. Four algorithms were implemented: a baseline recommender using semantic similarity, a CBF model using item features, a CF system based on user-item interaction patterns, and a hybrid model combining the strengths of both CBF and CF. Advanced techniques, such as sentence embeddings and one-hot encoding, were employed to enhance the models' capabilities and ensure flexibility and scalability.

Experimental evaluation revealed distinct strengths and limitations of each algorithm. While the baseline recommender was effective in generating recommendations, it was computationally

intensive and less practical for large datasets. In contrast, the CBF and CF models demonstrated faster performance, with the CBF approach excelling in leveraging item attributes and the CF model showing strengths in identifying patterns from user interactions. The hybrid system emerged as the most balanced solution, offering an effective combination of accuracy, diversity, and computational efficiency.

The findings validate that algorithms tailored to the specific challenges of recommending computer components significantly enhanced accuracy and user satisfaction. Machine learning techniques, particularly in the hybrid model, provided more personalized and adaptable recommendations. Proper data preparation, including the preprocessing of attributes and integration of multiple data sources, played a critical role in improving the overall quality of recommendations. The hybrid approach demonstrated its ability to scale effectively while maintaining precision and relevance.

In conclusion, this research offers contributions to the development of domain-specific recommender systems. By addressing the unique requirements of computer component recommendations through a hybrid approach, the thesis highlights the importance of integrating multiple recommendation techniques and adapting them to specific user needs. These findings advance the field of e-commerce and establish a foundation for future work in applying neural networks and real-time learning to complex domains, ensuring both accuracy and adaptability in personalized recommendations.

Keywords: Machine Learning; Recommender Systems; Collaborative Filtering; Content-Based Filtering; Computer Components; Hybrid Recommendation Approach; Full-Stack Web Architecture.

EXECUTIVE SUMMARY

The Master's thesis must develop in the text the following concepts, appropriately justified and discussed, focusing on the field of IT.

CONCEPT (ABET)	Done? (Y/N)	Where? (page numbers)
1. IDENTIFY:		
1.1. Problem statement and opportunity	Y	1
1.2. Constraints (standards, codes, needs, requirements & specifications)	Y	3 - 4
1.3. Setting of goals	Y	2 - 3
2. FORMULATE:		
2.1. Creative solution generation (analysis)	Y	28 - 37
2.2. Evaluation of multiple solutions and decision-making (synthesis)	Y	42 - 48
3. SOLVE:		
3.1. Fulfilment of goals	Y	48 - 52
3.2. Overall impact and significance (contributions and practical recommendations)	Y	52 - 54

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

Univerza v Mariboru

Fakulteta za elektrotehniko, računalništvo in informatiko

Aljaž Herzog

Recommender System for Computer Components

Master's Thesis

Valencia, January 2025

TABLE OF CONTENTS

INDEX OF FIGURES

INDEX OF TABLES

INDEX OF GRAPHS

(See index of figures.)

SYMBOLS AND ABBREVATIONS USED

CF – Collaborative Filtering

CBF – Content-Based Filtering

DBF – Demographic-Based Filtering

KBF – Knowledge-Based Filtering

PC – Personal Computer

NLP – Natural Language Processing

CNN – Convolutional Neural Network

RNN – Recurrent Neural Network

LLM – Large Language Model

MAE – Mean Absolute Error

CPU – Central Processing Unit

CBR – Case-Based Reasoning

AUC – Area Under the Curve

NCF – Neural Collaborative Filtering

MLP – Multi-Layer Perceptron

GMF – Generalized Matrix Factorization

MF – Matrix Factorization

AE – Autoencoder

PVR – Personalized Video Ranker

RMSE – Root Mean Square Error

API – Application Programming Interface

REST – Representational State Transfer

CORS – Cross-Origin Resource Sharing

JSON – JavaScript Object Notation

JWT – JSON Web Token

SQL – Structured Query Language

NoSQL – Not only SQL

UI – User Interface

DOM – Document Object Model

URL – Uniform Resource Locator

CSV – Comma-Separated Value

1 INTRODUCTION

1.1 Background

The rapid expansion of e-commerce has transformed how consumers shop and businesses operate. Online retailers and service providers generate vast amounts of user data, including browsing history, purchase behavior, and explicit feedback such as ratings and reviews. Effectively utilizing this data to provide personalized recommendations is a critical challenge, particularly in domains with complex product configurations such as computer components.

1.2 Problem statement

Consumers seeking computer components face difficulties in selecting compatible parts that match their technical requirements and budgets. Given the broad range of available products and the technical expertise required for optimal component selection, traditional search and filtering mechanisms often fall short in guiding users toward the best choices. Recommender systems play a crucial role in solving this issue by analyzing user preferences, identifying patterns, and generating personalized recommendations. However, existing approaches face significant challenges, including data sparsity, cold start issues, and the need for scalable solutions that can handle large and dynamic product catalogs. In the domain of computer component recommendations, factors such as hardware compatibility, performance requirements, and evolving user preferences further complicate the recommendation process [1].

1.3 Motivation and research challenges

The motivation for this thesis comes from the limitations of existing recommender systems in the computer components domain. Unlike general consumer goods, computer components must be carefully selected based on their compatibility with other parts, making traditional recommendation approaches insufficient. Collaborative Filtering (CF) may struggle due to sparse user interaction data, while Content-Based Filtering (CBF) may fail to capture the nuanced relationships between different components. Hybrid approaches combining multiple recommendation techniques have the potential to improve accuracy and usability. Additionally, recent advancements in machine learning, including deep learning and neural networks, provide new opportunities to enhance

recommendation quality. By leveraging these techniques, it is possible to design a hybrid recommender system tailored for computer components, improving both recommendation accuracy and user experience [2]. However, the design and implementation of effective recommender systems is not without its challenges. Several factors need to be carefully considered, including data sparsity, cold start problem, scalability, accuracy and diversity [3][4][5].

To address these challenges, various recommendation techniques and algorithms have been developed, including Collaborative Filtering (CF), Content-Based Filtering (CBF), Hybrid Filtering, Demographic-Based Filtering (DBF) and Knowledge-Based Filtering (KBF). In addition to these basic techniques, more advanced methods, such as deep learning and sentiment analysis, have been explored to improve the accuracy and effectiveness of recommender systems [6][7].

1.4 Research questions and hypotheses

This thesis aims to investigate the use of recommender systems in e-commerce, exploring the different techniques, algorithms, and challenges involved in their design and implementation. To further focus on this research, this thesis will address the following research questions and test hypotheses:

- **RQ1:** Which recommender system algorithms are most suitable for recommending computer components, considering their specific characteristics and user needs?
- **RQ2:** How do different algorithms compare in terms of accuracy, speed, and adaptability when applied to recommending computer components?
- **RQ3:** How do different data sources influence the quality of recommendations for computer components?

  In addition to these research questions, this thesis will explore the following hypotheses derived from each of the RQs:
- **H1:** By employing a suitable algorithm in the recommender system, the accuracy of recommendations and user satisfaction in the domain of computer components can be significantly improved.
- **H2:** Advanced machine learning techniques enhance the accuracy and personalization of the recommender system for computer components.
- **H3:** Different data sources influence the quality of recommendations for computer components, and the preparation of appropriate data leads to greater user satisfaction.

1.5 Expected contributions

The outcomes of this research are expected to contribute to both academic and practical advancements in recommender system design. By evaluating different recommendation algorithms and hybrid strategies, this thesis provides insights into the most effective methods for computer component recommendations. The integration of deep learning techniques is anticipated to further improve recommendation accuracy and personalization, potentially setting a foundation for future developments in AI-driven e-commerce platforms. From a business perspective, an improved recommender system can enhance customer satisfaction, increase engagement, and drive higher conversion rates for e-commerce platforms specializing in computer components. By addressing compatibility concerns and improving recommendation relevance, the system can also assist users in making more informed purchasing decisions.

1.6 Thesis sctructure

The thesis begins with an introduction that provides background on the role of recommender systems in e-commerce, highlights the challenges specific to computer component recommendations, and outlines the research questions and hypotheses. This is followed by a theoretical foundation chapter that explores existing recommendation approaches, algorithms, and their applications. The solution development chapter details the design, implementation, and integration of the proposed hybrid recommender system. Next, the experiments and analysis chapter evaluates algorithm performance, user satisfaction, and system effectiveness. Finally, the conclusion summarizes key findings, discusses contributions and limitations, and suggests directions for future research.

2 THEORETICAL FOUNDATIONS

2.1 Recommender Systems

Recommender systems are a class of software tools and techniques that have become essential in today's information-rich digital world. They are designed to predict and suggest items or content that a user might find interesting or valuable, acting as personalized information filters [8][9]. These systems achieve personalization by analyzing various types of user data, including past interactions with items, explicit feedback (ratings and reviews), implicit feedback (browsing history, purchase patterns and search queries), demographic information, and even contextual factors. By understanding user preferences and item characteristics, recommender systems can provide personalized recommendations that enhance user experience and offer significant benefits to businesses. One of the key benefits of recommender systems is their ability to reduce information overload. In today's digital world, users are constantly bombarded with an overwhelming amount of information, making it difficult to find relevant and interesting items. Recommender systems help users navigate through this vast sea of information and discover items that match their needs and preferences, leading to increased customer satisfaction, engagement, and loyalty [10][11]. They also play a crucial role in driving sales and business growth. By providing personalized recommendations, these systems can increase the likelihood of users making purchases or engaging with products and services. This can lead to increased revenue, improved customer retention, and enhanced brand loyalty.

Recommender systems are employed in a wide range of applications, including:

- **E-commerce:** suggesting products to customers based on their browsing and purchase history, ratings, and preferences.
- **Entertainment:** recommending movies, music, and TV shows on platforms like Netflix, Spotify, and YouTube, based on user listening and viewing habits.
- **Social media:** suggesting connections, groups, and content on platforms like Facebook, Twitter, and LinkedIn, based on user interactions and social graphs.
- **News and Content:** recommending articles, blogs, and news stories based on user reading history and expressed interests.
- **Personalized Advertising:** displaying targeted ads based on user preferences, demographics, and online behavior.

Recommending computer components presents unique challenges due to the technical complexity of the products and the specific requirements of users. Factors such as

compatibility with existing hardware, performance requirements, budget constraints, and the intended use case must be carefully considered. For example, a user building a gaming Personal Computer (PC) will have different needs and priorities than a user building a workstation for video editing or a home theater PC [12]. Traditional recommender systems often struggle with these nuances. CF, which relies on user-item interaction data, might not be effective due to the sparsity of data in this domain. Many users only purchase computer components infrequently, and new components are constantly being released, leading to the cold start problem. CBF, which relies on item attributes, can be challenging due to the complex relationships between components. For instance, recommending a compatible motherboard requires considering the CPU socket type, memory type, and form factor [12]. To address these challenges, recommender systems for computer components can use a variety of techniques:

- **Knowledge-based approaches:** these systems utilize expert rules and constraints to ensure compatibility and meet specific user requirements. This can involve incorporating knowledge about component specifications, compatibility rules, and performance benchmarks.
- **Natural Language Processing (NLP):** NLP techniques can be used to analyze user reviews and forum discussions to identify popular components, compatibility issues, and user preferences. This information can be used to improve the accuracy and personalization of recommendations.
- **Sentiment Analysis:** this technique can be used to analyze user reviews and opinions to understand their preferences and improve recommendation accuracy.
- **Deep Learning:** deep learning models, like Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), are used to analyze complex data and improve the performance of recommender systems [12].

Recent advances in NLP and knowledge graphs have enabled the development of more sophisticated recommender systems for this domain.

Prometheus Chatbot was developed with support from Lenovo. This system integrates a knowledge graph with a large language model (LLM) to interpret user requests in natural language and deliver personalized recommendations based on structured relational data. This approach ensures accurate comprehension and response to users' computer setup requirements, including compatibility constraints and performance goals. The chatbot can interpret complex rules and constraints related to computer components, ensuring that recommended configurations are compatible and meet user needs [12].

ResyDuo is a prototype system designed to aid Arduino developers. This system combines data models with collaborative filtering-based recommender systems to assist users in selecting appropriate hardware components and corresponding software libraries. This streamlines the development of Arduino projects by recommending relevant components and resources based on project requirements and user preferences. It uses a data model to represent and mine relevant data from the ProjectHub repository, a curated collection of Arduino projects. It then employs collaborative filtering to recommend hardware components based on project tags or existing hardware lists and suggests corresponding software libraries [13].

By combining knowledge-based approaches with collaborative filtering and content-based techniques, recommender systems can effectively address the challenges of recommending computer components. These systems can assist users in making informed decisions, ensuring compatibility, optimizing performance, and ultimately enhancing their computing experience.

Figure 1 illustrates the basic components and interactions within a recommender system:

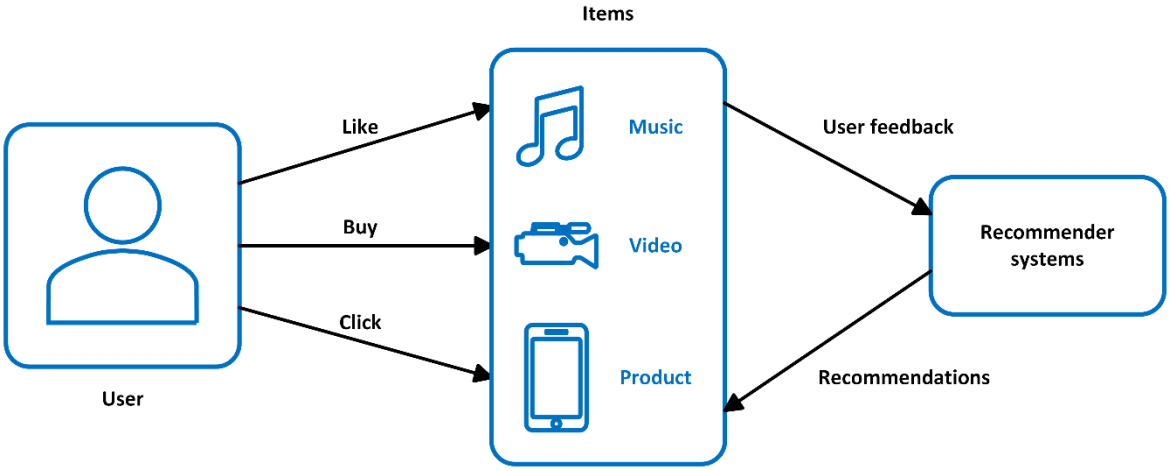


Figure 1: Recommendation process

The recommender system connects an individual user with a variety of items, including products, services, or content, to generate personalized recommendations. The user interacts with the system, providing explicit or implicit feedback through actions. This interaction data is processed by the system's core algorithm, which analyzes both user behavior and item attributes. Using advanced techniques like similarity measures, matrix factorization or deep learning models, the algorithm identifies patterns and predicts user

preferences. Based on this analysis, a personalized list of recommendations is presented to the user, tailored to their specific interests and needs.

Some of the challenges in Recommender Systems include:

- **Data Sparsity:** in many cases, user-item interaction data is sparse, meaning most users have only rated or interacted with a small fraction of the available items. This makes it difficult to accurately predict preferences and generate relevant recommendations. Data sparsity can be addressed through techniques such as dimensionality reduction, matrix factorization, and incorporating content-based features into collaborative models [3] .
- **Cold Start Problem:** recommending items to new users or new items with limited interaction data poses a significant challenge. The cold start problem can be tackled through hybrid filtering approaches, leveraging content information, or utilizing demographic data [4].
- **Scalability:** as the number of users and items grows, the computational complexity of recommendation algorithms can increase significantly. Recommender systems need to be scalable to handle large datasets and provide timely recommendations. Scalability can be achieved through efficient algorithms, distributed computing techniques, and incremental learning approaches [3].
- **Accuracy and Diversity:** balancing the accuracy of recommendations with their diversity is crucial to providing both relevant and novel suggestions. Accuracy can be improved through advanced algorithms and incorporating diverse data sources, while diversity can be enhanced through techniques such as serendipity, novelty, and exploration [8], [10].

Recommender systems are constantly evolving to address these challenges and provide more accurate, diverse, and personalized recommendations. The field is driven by ongoing research and innovation, exploring new techniques, such as deep learning, sentiment analysis, and natural language processing, to enhance the effectiveness of recommender systems in various domains.

2.2 Algorithms in recommender systems

2.2.1 **Collaborative Filtering (CF)**

CF is a widely used technique in recommender systems that uses the collective intelligence of users to provide personalized recommendations. It operates on the assumption that users who have agreed in the past will tend to agree in the future. This means that if two users have rated or purchased similar items in the past, the system can recommend items that one user has liked, to the other user [14][2].

Advantages of the CF are that it doesn't require the content of the items to be analyzed, making it applicable to a wide range of domains. The explicit ratings provided by users in CF systems allow for a more accurate assessment of the quality of items compared to content-based filtering. CF is based on user similarity rather than item similarity, which can lead to more effective recommendations, especially in sparse datasets [14].

Some of its disadvantages include the cold-start problem which means that it struggles to recommend items to new users or new items that have not been rated by any users. Another problem is also data sparsity, that can lead to inaccurate recommendations. Scalability can also be an issue here, as the number of users and items increases, the computational complexity of CF algorithms can become a bottleneck [2].

Several metrics are used to evaluate the performance of CF recommender systems. One commonly used metric is Mean Absolute Error (MAE), which measures the average absolute difference between the predicted ratings and the actual ratings. For example, authors in a specific study report an MAE of 0.6586 for their deep neural network-based collaborative filtering method on the MovieLens-1M dataset [4]. Another metric is Recall, which measures the proportion of relevant items that are recommended. Authors report a recall of 0.227 for their next basket recommendation model on the Ta-Feng dataset [15].

Cosine similarity and **Pearson correlation** are two common metrics used to measure the similarity between users or items in collaborative filtering. These metrics play a crucial role in identifying users or items with similar preferences, which forms the basis for generating recommendations. Cosine similarity measures the angle between two vectors in a multi-dimensional space. These vectors represent the ratings of users or items. Equation 1 represents, how cosine similarity is calculated, where R_{ik}represents the rating of item k given by user i and R_{jk} represents the rating of item k given by user j. The number of items that both users have rated (also known as co-rated items) is represented by n. The numerator is the dot product of the rating vectors of user i and user j. It essentially calculates the overlap or agreement between the two users' ratings. The denominator is the product of the Euclidean norms (magnitudes) of the rating vectors of both users [7].

$$(i,j) = \frac{\sum_{k=1}^{n} R_{ik} R_{jk}}{\sqrt{\sum_{k=1}^{n} R_{ik}^2} * \sqrt{\sum_{k=1}^{n} n\, R_{jk}^2}}$$

Equation 1: Cosine similarity

The resulting similarity value ranges from -1 to 1, where 1 indicates perfect similarity, 0 indicates no similarity, and -1 indicates perfect dissimilarity.

Pearson correlation measures the linear correlation between two vectors of ratings. It is calculated as the covariance of the vectors divided by the product of their standard deviations which is shown in Equation 2. $R_{i,k}$ represents the rating given by user i to item k, $\overline{R_i}$ is the average rating given by user i for all co-rated items and I_{ij} is the set of items that are rated by both users. Numerator Computes the covariance between the rating vectors of user i and user j, adjusted by their average ratings. The denominator normalizes the numerator, ensuring the similarity measure is independent of the scale of the ratings [7].

$$(i,j) = \sum_{k \epsilon I_{ij}} \left(R_{i,k} - \overline{R_i}\right)$$

Equation 2: Pearson correlation

CF is widely used in e-commerce to provide personalized product recommendations. Many large e-commerce platforms, such as Amazon and eBay, employ CF techniques to suggest products that users might be interested in based on their past behavior and the behavior of similar users [16].

CF can be effectively applied to recommend computer components. By analyzing the purchase history of users, the system can identify users with similar preferences and recommend parts that one user has purchased to another user. For example, if two users have both purchased a high-end graphics card and a powerful Central Processing Unit (CPU), the system can recommend other components, such as a compatible motherboard or a high-capacity power supply, that one user has purchased to the other user. Furthermore, the system can use user ratings and reviews to provide more accurate recommendations. For example, if a particular motherboard has received positive reviews from users who have purchased similar components, the system can recommend that motherboard to other users who are looking to build a similar computer.

2.2.2 **Content-Based Filtering (CBF)**

CBF is a recommendation technique that suggests items to a user, based on the characteristics of items the user has liked or purchased in the past. This method relies on analyzing the content of the items, such as product descriptions, keywords, or attributes, to create a profile of the user's preferences. The system then recommends items that are similar to the user's profile. For example, if a user has purchased several books by the same

author or in the same genre, the system can recommend other books by that author or in that genre. In the context of e-commerce, CBF can be used to recommend products that are similar to products the user has previously viewed, purchased, or rated positively [17]. The advantages of CBF are that it does not require the preferences of other users to be considered, making it less susceptible to the cold-start problem. The recommendations generated by CBF systems can be easily explained to the user, as they are based on the characteristics of the items the user has liked in the past. CBF can recommend new items that have not been rated by any users, as it relies on the content of the items rather than user ratings [17][16].

The effectiveness of CBF is limited by the ability to analyze the content of the items. In some domains, such as music or images, it can be difficult to extract meaningful features from the content. CBF systems also tend to recommend items that are very similar to the items the user has liked in the past, which can lead to a lack of diversity in the recommendations. Because of increasing numbers of items and users, the computational complexity of CBF algorithms can become a bottleneck [16][17].

Several metrics are used to evaluate the performance of CBF recommender systems. One commonly used metric is precision, which measures the proportion of recommended items that are relevant to the user. For example, the authors in one study report a precision of 0.297 for their Popular SimCat recommender algorithm on a dataset of travel agency tours [18].

CBF is widely used in e-commerce to provide personalized product recommendations. Many e-commerce platforms employ CBF techniques to suggest products that users might be interested in based on their past behavior and the characteristics of the products [16][17]. Same as CF, CBF can be applied to recommend computer components. System can recommend parts that are similar to parts the user has purchased in the past. For example, if a user has purchased a high-end graphics card, the system can recommend other high-end graphics cards or other components that are typically purchased with high-end graphics cards, such as a powerful CPU or a compatible motherboard.

2.2.3 **Knowledge-Based Filtering (KBF)**

KBF is a technique used in recommender systems that draws from research in Case-Based Reasoning (CBR). Unlike CF and machine learning approaches, KBF doesn't rely on a large dataset of user ratings or preferences. Instead, it uses knowledge about the users and products to reason about which products best meet the users' needs and requirements.

KBF uses knowledge about the products and users to make recommendations. For example, if a user is looking for a new computer, they might tell the system that they want a computer that is similar to their old one but with a better processor. The system would then use its knowledge of computer components to recommend a computer with a similar configuration but with a more powerful processor [19]. To make recommendations, KBF often integrates interaction data such as views, selections, and purchases. The system assigns different weights to these events to compute a preference score for each product. The Equation 3 is used to calculate the preference score for a user regarding a product. C_{up}^{v}, C_{up}^{s}, C_{up}^{b} are interaction scores for the user u viewing, selecting and buying the product p.X, Y and Z are weights assigned to viewing, selecting, and buying interactions, respectively. Denominators ensure that the contributions from different event types are on the same scale. Final score C_{up} represents the overall preference score for user u on item p.

$$C_{up} = X\frac{C_{up}^{v}}{\sqrt{\sum_{p=1}^{n}\left(C_{up}^{v}\right)^{2}}} + Y\frac{C_{up}^{s}}{\sqrt{\sum_{p=1}^{n}\left(C_{up}^{s}\right)^{2}}} + Z\frac{C_{up}^{b}}{\sqrt{\sum_{p=1}^{n}\left(C_{up}^{b}\right)^{2}}}$$

Equation 3: Preference score

Advantages of KBF are that there is no ramp-up required, meaning it can be used with limited user data. KBF is sensitive to preference changes, it can identify niches precisely and it gathers detailed qualitative preference feedback [20]. Main two disadvantages of KBF are that it requires knowledge engineering, and its suggestion ability is static [19].

KBF is used in e-commerce, but it's not as widely used as collaborative filtering or machine learning approaches. This is because it requires a significant investment in knowledge engineering. However, KBF can be a valuable tool for recommending products to users, especially when there is limited user data available. It can be applied to recommending computer components by creating a knowledge base of computer components and their properties. This knowledge base can then be used to reason about which components are compatible with each other and which components best meet the users' needs and requirements.

2.2.4 **Demographic-Based Filtering**

DBF is a type of recommender system that uses demographic information about users to predict their preferences for items and make recommendations. This information can include age, gender, location, occupation, education level, and other personal attributes.

By analyzing the preferences of users with similar demographic characteristics, the system can identify items that are likely to be of interest to a target user [21].

DBF is relatively easy to set up and does not require complex algorithms or large datasets. It can be used to make recommendations for new users who have not yet interacted with the system, as it does not rely on past behavior or ratings. It can also provide a degree of personalization by tailoring recommendations to the specific demographic characteristics of each user [21].

DBF may not be as accurate as other filtering methods, as it does not consider the individual preferences of users. It can lead to stereotyping and bias, as it assumes that users with similar demographic characteristics have the same preferences. It also requires access to demographic data about users, which may not always be available or reliable [21].

Authors in one study used demographic information to improve the performance of a product recommender system on microblogs. Their system achieved a precision of 0.688 [22]. Other authors used demographic information to improve the performance of a collaborative filtering recommender system for an e-clothing store. Their system achieved an F1-measure of 0.922 and AUC of 0.895 [21].

DBF is used to some extent in e-commerce, but it is often combined with other filtering methods to improve accuracy and personalization. For example, Amazon.com uses a hybrid recommender system that combines collaborative filtering, content-based filtering, and demographic-based filtering [21]. DBF could be used to recommend computer components by considering the demographic characteristics of users, such as their occupation, education level, and interests. For example, the system could recommend higher-end computer components to users who are professionals in the technology industry or students in computer science.

Table 1 presents the collected knowledge and compares the analyzed algorithms.

Table 1: Comparison of algorithms

Algorithm	**Advantages**	**Disadvantages**	**Other**
Collaborative Filtering (CF)	Simple to implement, can provide accurate recommendations.	It suffers from a cold start problem,	It can be used in conjunction

		requires large amount of user data.	with other algorithms.
Content-Based Filtering (CBF)	Does not suffer from cold start problem, can provide personalized recommendations.	It can be difficult to implement, requires detailed product information.	It can be used to recommend new products.
Knowledge-Based Filtering (KBF)	It can provide very accurate recommendations, does not require a large amount of user data.	It can be difficult to implement, requires expert knowledge.	It can be used to recommend complex products.
Demographic-Based Filtering (DBF)	Simple to implement, can be used to target specific demographics.	It can be inaccurate, can lead to biased recommendations.	It can be used in conjunction with other algorithms.

According to the results, we conclude that KBF is the best single algorithm for a recommender system that recommends computer components. This is because KBF can provide very accurate recommendations, does not require a large amount of user data, and can be used to recommend complex products. However, KBF can be difficult to implement and requires expert knowledge. In addition to the advantages and disadvantages listed in the table, KBF is particularly well-suited for recommending computer components because it can consider the specific needs of the user. For example, a user who is building a gaming PC will have different needs than a user who is building a PC for work. KBF can use this information to recommend the appropriate parts for each user.

To enhance recommendation accuracy and adaptability, we can use hybrid recommendation systems. They are a combination of two or more recommendation algorithms, typically CF and CBF, but they can also include other algorithms such as KBF which we identified as best algorithm for our problem. This approach combines the strengths of each algorithm to overcome their individual limitations [23]. Computer components have a wide range of features and specifications. Hybrid algorithms can better

capture this diversity by considering both user preferences and item characteristics. They can better handle the cold-start problem by using CBF to recommend items to new users and CF to recommend new items to existing users. They can also fix data sparsity problems by using content-based filtering to recommend items even when there are few user interactions [24][23][11].

A hybrid recommender system could use the following approach, which is shown in Figure 2:

1. **User profiling**:
    - Gather user data, such as past purchases, browsing history, explicit preferences (e.g., gaming, video editing), and product reviews.
2. **Item profiling**:
    - Collect detailed item data, such as specifications (e.g., CPU clock speed, GPU memory), features, compatibility constraints (e.g., motherboard socket type), and customer reviews.
3. **KBF**:
    - Use explicit rules and structured knowledge to ensure compatibility and address technical requirements. For example:
        - Recommend only motherboards compatible with the selected CPU socket.
        - Filter GPUs that meet power supply constraints for the user's setup.
        - Match component suggestions with the user’s stated budget and performance needs.
4. **CF**:
    - Identify similar users based on their interactions with items (e.g., purchases, ratings) and recommend components that have been popular among those users.
5. **CBF**:
    - Identify similar items based on their characteristics (e.g., recommend GPUs with similar performance benchmarks or CPUs with matching core/thread counts).
6. **Recommendation generation**:
    - Combine the results of knowledge-based filtering, collaborative filtering, and content-based filtering to generate recommendations.

- Use a weighting system to prioritize compatibility (KBF) while still considering user preferences (CF and CBF).

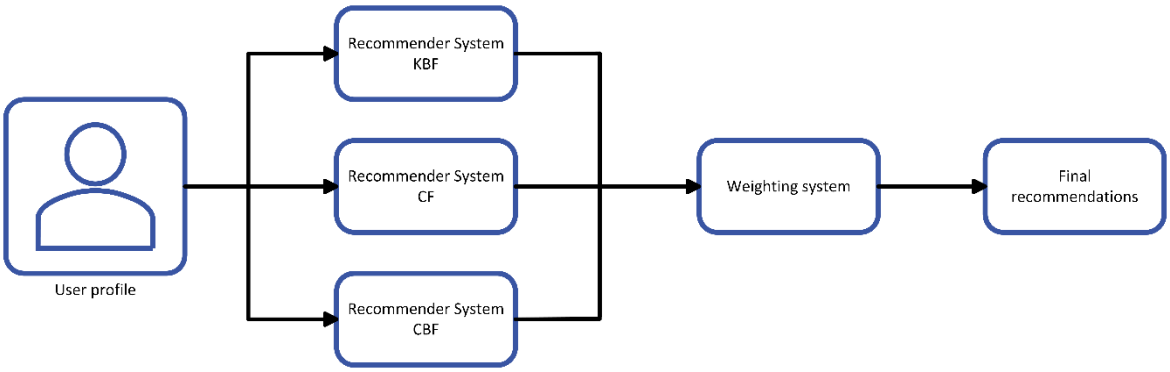


Figure 2: Proposed hybrid approach

In this approach the KBF ensures technical compatibility, a critical factor for computer components. CF tailors recommendations based on community trends, and CBF personalizes them further by matching item characteristics to user preferences. The system can handle a variety of user types, from novices (who rely heavily on knowledge-based rules) to advanced users (who benefit from collaborative and content-based suggestions).

2.3 Recommender Systems with Neural Networks

Traditional recommender systems have proven effective but often face limitations described in section **Napaka! Vira sklicevanja ni bilo mogoče najti.**. The use of neural n etworks has opened new possibilities for improving recommender systems by using their ability to learn intricate patterns and representations from data. Neural networks excel at capturing non-linear and non-trivial relationships between users and items, overcoming the limitations of traditional linear models. By learning complex interactions and dependencies, neural networks can significantly enhance the accuracy and personalization of recommender systems.

2.3.1 CF with neural networks

The application of neural networks to CF has led to the emergence of Neural Collaborative Filtering (NCF). NCF is a general framework that uses neural network architecture to model user-item interactions. The key idea is to replace the traditional inner product or cosine similarity with a neural network that can learn more complex and non-linear relationships between users and items [25]. One of the common approaches within NCF is to use a Multi-

Layer Perceptron (MLP) to learn the interaction function. The MLP takes the concatenated user and item latent vectors as input, and outputs a prediction score. The multiple layers and non-linear activation functions of the MLP enable it to capture intricate patterns in user-item interactions, leading to improved accuracy [26]. Another approach within NCF is Generalized Matrix Factorization (GMF). GMF can be seen as a neural network interpretation of matrix factorization (MF), a popular CF technique. GMF uses an element-wise product of user and item embeddings, followed by a linear transformation, to predict the interaction score. While GMF is conceptually simpler than MLP, it can still benefit from the flexibility of neural networks and learn more effective latent representations [26]. To combine the strengths of both GMF and MLP, some NCF models fuse the two approaches. This fusion allows the model to use the linearity of MF and the non-linearity of MLP, resulting in a more expressive and accurate model. Several studies have provided evidence that using neural networks can improve the accuracy and personalization of CF algorithms. Authors demonstrated that NCF models outperform traditional MF methods on benchmark datasets such as MovieLens. The authors attributed the improvement to the ability of neural networks to learn more complex interaction functions and adapt to specific user preferences [26]. Neural networks can adapt to specific users by learning user-specific parameters or representations. This can be achieved through techniques such as attention mechanisms, which allow the model to focus on the most relevant parts of the user's interaction history. By capturing user-specific patterns, neural networks can provide more personalized recommendations [25].

2.3.2 CBF with neural networks

Neural networks can also improve CBF algorithms. They can provide several advantages over traditional CBF approaches. They can capture complex, nonlinear relationships between item attributes, enabling them to learn intricate patterns and dependencies that may be missed by linear models. This leads to more accurate and nuanced recommendations. Neural networks excel at automatically learning relevant features from raw data, reducing the need for manual feature engineering. This not only saves time and effort but also allows the model to discover hidden patterns and representations that might not be apparent to human engineers. They can also adapt to individual user preferences by learning user-specific representations. This enables the model to tailor recommendations to each user's unique tastes and needs, leading to a more personalized experience [15].

Different neural network architectures can be employed to improve CBF algorithms:

- **Multilayer Perceptrons:** MLPs are a basic type of neural network that can be used to model nonlinear interactions between item attributes. They can be used to learn a mapping from item attributes to user preferences, enabling the model to predict the relevance of an item to a specific user [15].
- **Autoencoders (AEs):** AEs are unsupervised neural networks that can learn compressed representations of item attributes. These compressed representations can then be used as input to a traditional CBF algorithm, leading to improved accuracy and efficiency [15].
- **Convolutional Neural Networks:** CNNs are particularly well-suited for processing data with spatial or temporal structure, such as images or text. In CBF, CNNs can be used to extract features from item descriptions or user reviews, enabling the model to better understand the content of items and user preferences [15].
- **Recurrent Neural Networks:** RNNs are designed to process sequential data, such as user browsing history or purchase sequences. In CBF, RNNs can be used to model the temporal dynamics of user preferences, enabling the model to adapt to changes in user interests over time [15].

Several studies have demonstrated the effectiveness of neural networks in improving CBF algorithms. In one study a deep learning model was used to learn representations of items and user profiles from textual data. The results showed significant improvements in recommendation accuracy compared to traditional CBF methods [27]. In another study, a collaborative deep learning model was proposed that jointly learns representations of item content and user-item interactions. This model was shown to outperform traditional CBF methods in terms of personalization and recommendation accuracy [28].

2.3.3 Hybrid approach with neural networks

Neural networks can be seamlessly integrated into various hybrid architectures to improve different aspects of the recommendation process. Neural networks can combine features from different data sources, such as user demographics, item attributes, and interaction history, to learn a unified representation that captures complex relationships. This can lead to more accurate and personalized recommendations [29]. Using a certain approach, one recommendation technique pre-filters the candidate items, and a neural network further refines the recommendations based on learned user preferences. This can improve the efficiency and scalability of the system [30]. Neural networks can be used to generate additional features from auxiliary information, such as textual reviews or images, which can

then be incorporated into a hybrid system. This can enhance the system's ability to capture user preferences and item characteristics [31].

The integration of neural networks into hybrid recommender systems offers several benefits. For instance, neural networks can learn complex non-linear relationships between users and items, leading to more accurate predictions and recommendations [11]. By learning user-specific preferences and adapting to individual needs, neural networks enable highly personalized recommendations. They can continuously learn and adapt to new data and changing user preferences, ensuring the system remains relevant and accurate over time [18][32].

Several studies have demonstrated the effectiveness of neural networks in enhancing hybrid recommender systems. One study showed that a hybrid system combining collaborative filtering with a neural network improved recommendation accuracy by up to 10% compared to traditional collaborative filtering methods. Similarly, another demonstrated that a hybrid system using a neural network to augment features from user reviews improved personalization and user satisfaction [33][16].

Neural networks excel in adapting to specific user preferences. By learning user-specific features and interaction patterns, they can tailor recommendations to individual needs and preferences. This adaptability is crucial in e-commerce, where users have diverse tastes and requirements.

2.3.4 Benefits of neural networks and proposed hybrid approach

Neural networks excel at capturing intricate patterns and non-linear relationships in data, enabling recommender systems to generate highly personalized recommendations. By analyzing user interactions, such as past purchases, browsing history, and ratings, neural networks can learn individual preferences and predict future behavior with greater accuracy. This personalized approach enhances user satisfaction and engagement by presenting them with items that are truly relevant to their interests and needs [32].

The accuracy of recommender systems is crucial for providing relevant and useful suggestions. Neural networks can significantly improve accuracy by effectively handling data sparsity and capturing complex interactions between users and items. CF methods often struggle with sparse datasets, where users have rated or interacted with only a small fraction of the available items. Neural networks can overcome this limitation by learning

latent representations of users and items, enabling them to make accurate predictions even with limited data [29].

User preferences and behaviors change over time, and recommender systems need to adapt to these changes to maintain their effectiveness. Neural networks can continuously learn and update user models, ensuring that recommendations remain relevant and personalized. This adaptive capability is essential for capturing evolving trends, incorporating new items into the recommendation pool, and responding to shifts in user interests [31].

While the hybrid approach described in offers a robust foundation for a recommender system, incorporating neural networks can significantly enhance its capabilities and performance. Instead of relying on manually engineered features for items and users, neural networks can automatically learn complex and intricate representations from raw data. A CNN can analyze product images to extract visual features, while a RNN can process user browsing sequences to capture temporal dynamics. Neural collaborative filtering models can capture non-linear relationships between users and items more effectively than traditional methods. A deep learning model can learn latent factors that represent user preferences and item characteristics in a high-dimensional space. Neural networks can improve CBF by learning complex similarity metrics between items. A Siamese network can be trained to compare the feature vectors of two items and predict their similarity. Neural networks can also be used to combine the outputs of different filtering components in a more sophisticated way. A MLP can learn to weight the recommendations from KBF, CF and CBF based on user and item characteristics. Neural networks can adapt to changes in user preferences and item popularity over time. For example, a recurrent neural network can track user interactions and update its recommendations accordingly. The new approach using neural networks is shown in Figure 3.

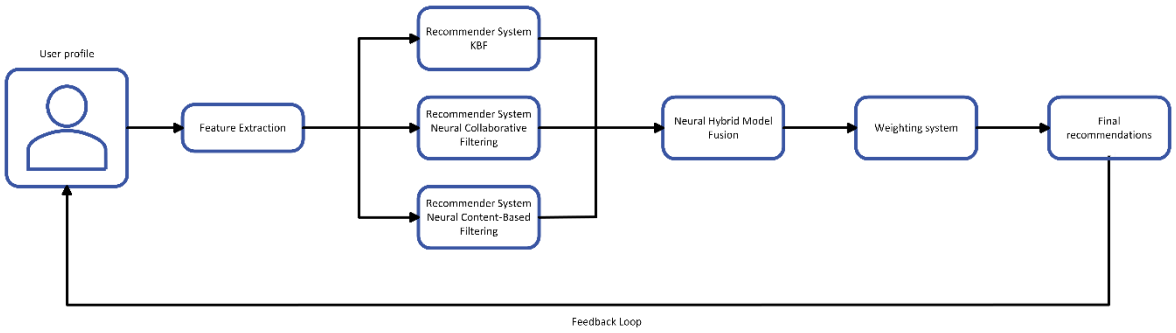


Figure 3: Proposed hybrid approach with neural networks

Neural networks have emerged as a powerful tool for improving recommender systems in e-commerce. Their ability to capture non-linear relationships, extract effective representations, and adapt to user preferences has led to significant advancements in personalization, accuracy, and user adaptation. By integrating neural networks into collaborative filtering, content-based filtering, and hybrid methods, recommender systems can provide more relevant and tailored suggestions, enhancing the overall shopping experience for users.

2.4 Recommender Systems used in e-commerce

2.4.1 Netflix

Netflix uses a variety of algorithms and data sources to provide recommendations. Its recommender system is not just one algorithm but a collection of algorithms for different use cases. Its algorithms are used to personalize the homepage for each member, helping them find something to watch. The homepage is the main presentation of recommendations, where 2 of every 3 hours streamed on Netflix are discovered [34][35].
The Personalized Video Ranker (PVR) algorithm is designed to rank an entire video catalog for each member profile in a highly personalized manner, determining the ordering of videos in various rows, such as genre-specific categories. PVR achieves optimal performance by blending personalized signals with elements of impersonalized popularity. Similarly, the Top N Video Ranker focuses on generating the most relevant personalized recommendations for the "Top Picks" row, aiming to identify the best options from the entire catalog. The Trending Ranker uses short-term temporal trends, which are often strong predictors of user behavior, to identify videos likely to be watched. For the "Continue Watching" row, a specialized ranker predicts user intent, distinguishing between videos members plan to resume or rewatch and those abandoned due to diminished interest. The Video-Video Similarity (Sims) algorithm underpins the "Because You Watched" rows by generating a ranked list of related videos for each item in the catalog. Unlike other approaches, Sims operates as an impersonalized algorithm, relying solely on video-level similarity metrics. The outputs of these algorithms feed into the Page Generation Algorithm, which integrates relevance and diversity considerations to construct personalized recommendation pages for each member. This system ensures that individual

rows are tailored while maintaining a cohesive and engaging page structure. Evidence algorithms enhances the user experience by determining the most relevant metadata to display, such as predicted ratings, synopses, awards, and cast information. These algorithms aim to help members assess whether a video aligns with their interests. Additionally, Netflix's search algorithms extend the recommendation framework to query results, suggesting videos even when initial searches fail. Netflix employs a robust experimental framework, including A/B testing and offline experimentation, to evaluate and refine its recommendation algorithms. Offline experimentation provides metrics to measure how well algorithm variants align with previous user engagement patterns, ensuring continuous improvement and adaptation to user needs. Through this multifaceted system, Netflix delivers a dynamic and highly personalized viewing experience [35].

Main take-aways from Netflix recommender system is that deep learning models can outperform simpler models, but they may also intensify the offline-online metric problem. Integrating deep learning toolboxes streamlines experimentation with both deep and non-deep learning approaches. Sequential models are valuable for session-based recommendations, while the bag-of-items approach can be effective for certain tasks. Additionally, many recommendation problems share the challenge of addressing diverse moods, needs, and contexts [35][36].

2.4.2 Amazon

Amazon's recommender system stands as a pioneering example of their implementation and evolution. Since the late 1990s, Amazon has been a leader in developing innovative approaches to product recommendations, shaping how customers discover and engage with items they might not have otherwise encountered [37].

Its recommender system primarily relies on a technique called item-to-item collaborative filtering. This algorithm focuses on finding similar items that customers tend to purchase together. By analyzing past purchase history, the system identifies items that are frequently co-purchased and builds a "similar-items table." When a customer views a product or adds it to their shopping cart, the system quickly looks up similar items from this pre-computed table and recommends them to the customer. This approach has several advantages over traditional user-based collaborative filtering, which finds similar customers based on their purchase history and then recommends items those customers have purchased. Item-to-

item collaborative filtering is more scalable, as the bulk of the computation is done offline to build the similar-items table. This allows the system to generate recommendations in real-time, even with millions of customers and products [37][38].

Amazon's recommender system demonstrates adaptability to user needs in several ways. The system considers the timing of past purchases to make more relevant recommendations. Recent purchases are given more weight than older purchases, and items bought sequentially, like books in a series, are used to recommend the next item in the sequence. The system recognizes that some purchases indicate long-term interests, while others are more transient. This allows the system to recommend items that are relevant to both types of interests. It also strives to provide a diverse set of recommendations, introducing customers to new products they might not have considered before. This is particularly important given the breadth of Amazon's catalog, which spans numerous product categories [37].

While Amazon's recommender system primarily relies on its own vast trove of customer purchase history, there are indications that it may also incorporate external data sources. For example, the system can identify compatible products, such as memory cards that work with a particular digital camera, even without explicit compatibility information in its knowledge base. This suggests that the system may be able to learn and infer relationships between products from external sources like customer reviews or product descriptions.

Amazon's system goes beyond item-to-item collaborative filtering by incorporating features such as customer ratings and reviews, which offer direct customer-to-customer recommendations and significantly influence item selection. It also provides personalized recommendations tailored to returning customers based on their browsing and purchase history. Additionally, the system suggests complementary items based on the contents of a customer's shopping cart, mimicking the impulse buy strategy commonly seen in supermarket checkout lines [38].

2.4.3 Comparison and application to our recommender system

To create an effective recommender system for computer components, combining insights from Amazon and Netflix, presented in Table 22.4.3 Comparison and application to our recommender system , offers a comprehensive approach. A hybrid system that integrates CF, CBF and KBF recommendations can address the diverse needs of customers building or upgrading their systems. Personalization should be central, tailoring suggestions based on

user profiles, past purchases, browsing history, and expressed preferences, such as gaming, content creation, or budget considerations. Item-to-item collaborative filtering, inspired by Amazon, can be used to recommend compatible or complementary components, ensuring users build functional and balanced systems. Additionally, incorporating sequential modeling, similar to Netflix's session-based recommendations, can capture the order in which users browse or purchase components, revealing patterns in assembling complete systems over time. CBF should analyze product descriptions, specifications, and reviews to suggest components with similar features or functionalities. Compatibility can be ensured by using expert knowledge and inferred relationships, guiding users toward suitable parts that work seamlessly together. Recommendations should also include explanations, inspired by Netflix's evidence algorithms, to build trust and provide users with context about why certain components are suggested. The system should blend personalized suggestions with popular or trending products, introducing users to new or highly rated components while maintaining relevance. Integrating user feedback through reviews and ratings can enhance credibility and help refine recommendations. Finally, regular experimentation and A/B testing, as employed by Netflix, will allow continuous improvement and adaptation to evolving user needs, resulting in a highly personalized, efficient, and user-friendly experience for computer component shoppers.

Table 2: Comparison of Amazon's and Netflix's Recommender System

Feature	**Netflix**	**Amazon**
Primary Algorithm	Hybrid approach, including Personalized Video Ranker, Top N Video Ranker, Trending Ranker, Continue Watching Ranker, Video-Video Similarity and Page Generation Algorithm.	Item-to-item collaborative filtering.
Data Sources	Primarily user viewing history, including ratings, watch time, and search queries.	Primarily user purchase history, potentially incorporating external data like customer reviews and product descriptions.

Personalization	Highly personalized, tailoring recommendations to individual user profiles and preferences.	Personalized, but with a strong emphasis on item-to-item relationships and real-time recommendations.
Real-Time Processing	Combines offline pre-computation with real-time data for homepage generation and search suggestions.	Uses offline pre-computation (similar-items table) for real-time recommendations.
Key Focus	Predicting user engagement and satisfaction, considering diverse moods and contexts.	Scalability and real-time performance, leveraging pre-computed similarity tables.
Adaptability	Adapts to user behavior by considering temporal trends, distinguishing between resumed and abandoned videos, and maintaining diversity in recommendations.	Adapts to user needs by considering the timing of past purchases, recognizing long-term interests, and providing diverse recommendations.
Additional Features	Evidence algorithms for displaying relevant metadata, search algorithms for suggesting videos in query results.	Customer ratings and reviews, personalized recommendations for returning customers, complementary item suggestions based on shopping cart contents.

2.5 Evaluating Recommender Systems

The effectiveness of a recommender system rests on its ability to accurately predict user preferences and provide recommendations that align with individual tastes. Therefore, the evaluation of these systems is a critical process that not only assesses their performance but also informs its improvement and adaptation to diverse user needs and behaviors. Evaluation is essential for several reasons. Firstly, it provides a measure of the system's effectiveness in meeting its objectives, whether it is to increase user satisfaction, drive sales, or promote the discovery of new items. By quantifying the system's performance,

evaluation helps identify areas for improvement and guides the selection of appropriate algorithms and parameterizations [39]. Evaluation also allows for the comparison of different recommender system algorithms and techniques, enabling informed decision-making about the most suitable approach for a given scenario. This is particularly important in e-commerce applications, where the choice of algorithm can significantly impact business outcomes. Evaluation helps ensure that recommender systems are fair, unbiased, and do not disproportionately favor certain items or groups of users. By examining the system's recommendations across various user groups and item categories, evaluation can reveal potential biases and inform the development of mitigation strategies [1].

Accuracy metrics are essential for evaluating how effectively a recommender system predicts user preferences. These metrics quantify the discrepancy between predicted ratings and actual user ratings, with smaller values signifying better performance. The MAE represents the average absolute difference between predicted and actual ratings, while the Root Mean Squared Error (RMSE) is a similar measure that assigns greater importance to larger errors. Beyond these rating-based metrics, precision evaluates the proportion of recommended items that are relevant to the user, and recall assesses the proportion of relevant items successfully recommended. To provide a balanced assessment of both precision and recall, the F1-measure combines them as a harmonic mean, offering a comprehensive view of accuracy. [40]

Trust metrics are designed to evaluate the trustworthiness of users or items within a recommender system, playing a crucial role in applications where trust significantly influences decision-making, such as e-commerce platforms or social networks. Trust propagation algorithms infer trust relationships between users who are not directly connected by analyzing the trust values of their mutual connections within a social network. Direct trust metrics rely on explicit trust statements made by users about others or specific items, which can be expressed either in binary terms, such as "trust" or "don't trust," or as weighted values on a scale. In contrast, indirect trust metrics derive trust levels from implicit indicators, such as user ratings or purchase histories. For instance, if two users frequently give high ratings to the same movies, it may suggest mutual trust in each other's preferences. [1]

Performance metrics evaluate the efficiency and scalability of a recommender system, ensuring it can manage large datasets and deliver recommendations promptly. Time complexity refers to the time required for the system to generate recommendations,

usually described relative to the size of the dataset, such as the number of users and items. Similarly, space complexity assesses the memory required to store data and execute computations, also expressed in terms of dataset size. These metrics are critical for determining whether a recommender system can operate effectively under real-world conditions with extensive data and high user demand [41].

Recommender systems can use trust and performance metrics to adapt to different user types and provide more personalized recommendations. If a user is known to be a cold start user (i.e., a new user with few or no ratings), the system can rely more on trust metrics to generate recommendations. This is because trust metrics can be computed even when little or no rating data is available. Conversely, if a user is known to be a heavy rater (i.e., a user who has provided many ratings), the system can rely more on performance metrics to ensure that recommendations are generated quickly. This is because heavy raters typically have large amounts of rating data, which can make the recommendation process computationally expensive [1].

Evaluation methods for recommender systems are crucial for understanding how effectively they can predict user preferences and provide relevant suggestions. These methods can be broadly categorized into two main types, each with its own strengths and weaknesses as depicted in the Figure 4.

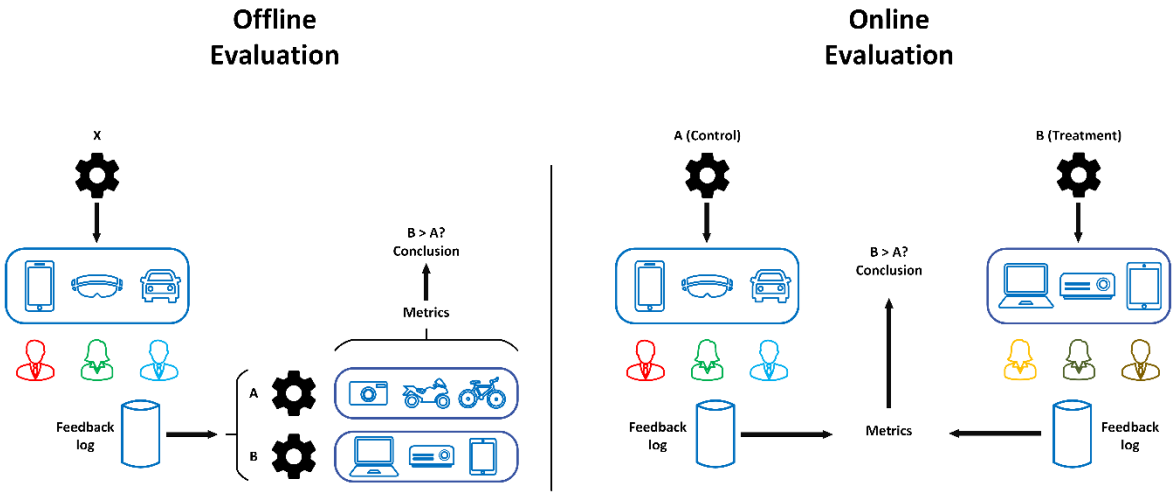


Figure 4: Offline and Online Evaluation

Offline Evaluation relies on past data. It utilizes a pre-existing dataset (represented by the 'X' in the Figure) containing historical user information such as ratings, purchase history, and preferences. This data is static, meaning it doesn't change during the evaluation

process. Different recommendation algorithms (A and B) are then applied to this dataset. Each algorithm generates a set of recommendations (illustrated by the different items). The effectiveness of each algorithm is then assessed using various metrics (e.g., precision, recall) to determine which one provides better recommendations according to those metrics. This comparison helps in selecting the most suitable algorithm for the recommender system. The feedback log in this context represents the historical data itself, providing insights into past user behavior and preferences.

Online Evaluation uses real-world testing. This method involves deploying a live recommender system and observing how real users interact with it. In the image, this is represented by two groups of users interacting with different versions of the system (A - Control, B - Treatment). Users are typically divided into groups. One group interacts with the existing system (A), while the other group experiences a modified version or a completely new system (B). This allows for direct comparison of user behavior and preferences. Metrics are collected based on user interactions with the system, such as click-through rates, conversion rates, time spent on pages, and user feedback. In online evaluation, the feedback log is actively updated with real-time user interactions and feedback, providing valuable insights into the system's performance in a live environment. While offline evaluation is quicker and easier to conduct, it relies on past data that may not accurately reflect current user preferences. Online evaluation, with its real-world testing, provides more accurate insights into how the system performs in practice. Online evaluation is more complex, time-consuming, and requires more resources to set up and run compared to offline evaluation. Offline evaluation is primarily used for comparing different algorithms and selecting the best one. Online evaluation focuses on assessing the actual performance of the deployed system and identifying areas for improvement [40][41].

The best evaluation method for recommender systems in e-commerce is a combination of offline evaluation, online evaluation, and user studies. Offline evaluation can be used to compare different algorithms and identify promising candidates. Online evaluation can then be used to test these candidates with real users and gather data on their performance. Finally, user studies can provide valuable feedback on user satisfaction and identify areas for improvement.

3 SOLUTION DEVELOPMENT

3.1 System Architecture Design

The recommender system is built using Flask framework [42] for API management, a MongoDB database [43] for data persistence and a React-based frontend [44] for user interaction. This architecture facilitates scalability, maintainability and flexibility in system operations. Figure 5 illustrates system architecture design. The frontend uses JavaScript and React for creating a responsive user interface that collects user data (such as the products he viewed) and displays recommendations. The backend acts as a bridge between the client and the database. It handles API requests, authentication, recommendation logic, and data processing. The database is powered by MongoDB, which stores structured data for users, components, and recommendations. Its flexible schema allows for efficient management of diverse data types.

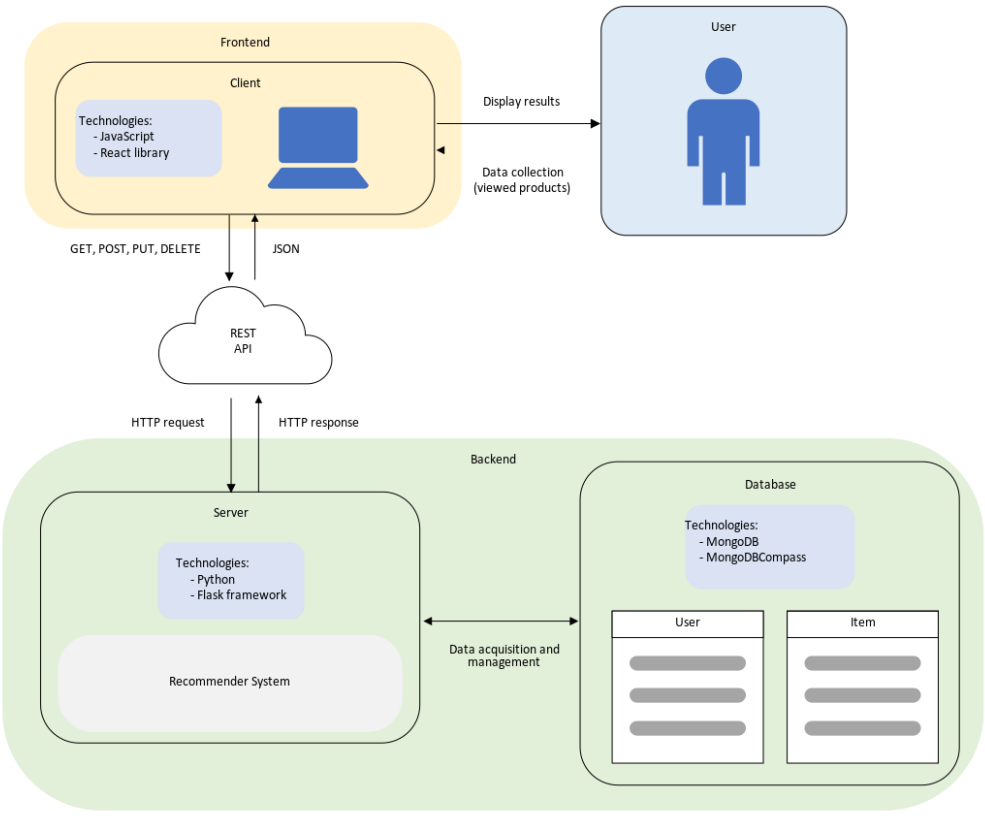


Figure 5: Application architecture

Figure 6 focuses on RESTful communication between the client, server, and database. The client makes HTTP requests (GET, POST, PUT, DELETE) [45] to the REST API. The server processes these requests, executes queries on the database, and sends JSON responses

back to the client. CORS is enabled to ensure smooth communication across different domains.

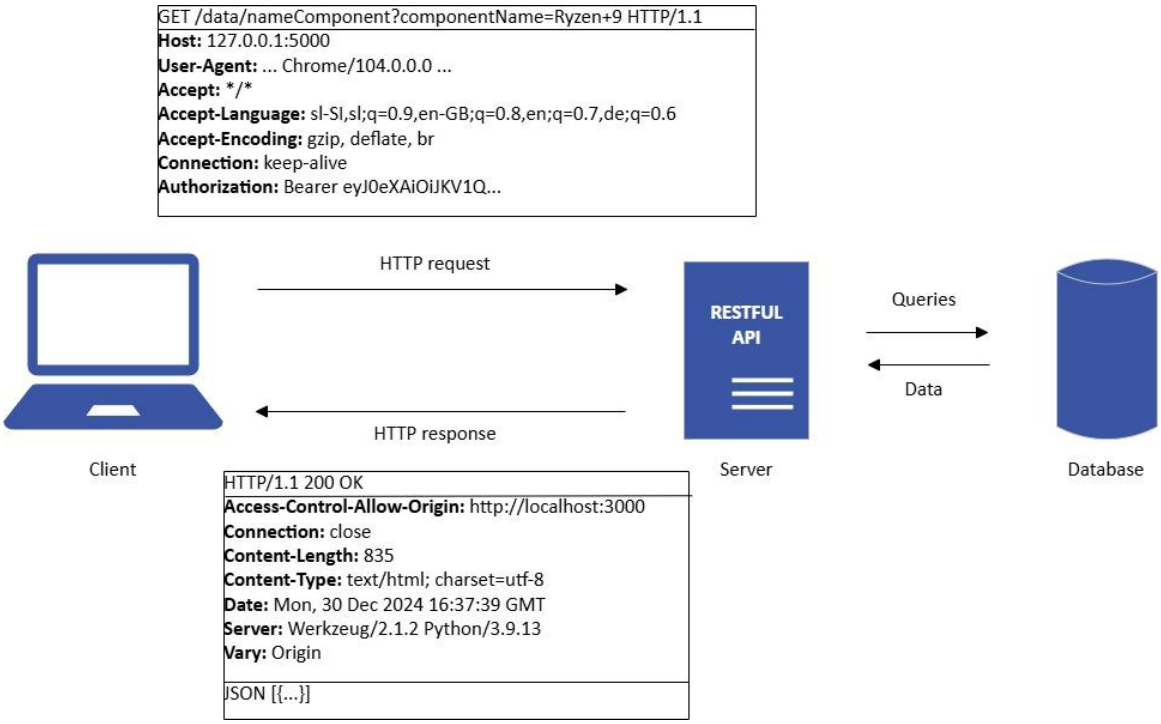


Figure 6: REST

3.1.1 Flask backend

Flask serves as the web server and API layer, offering RESTful endpoints for user authentication, data processing, recommendation generation, and component management. The API layer is organized into multiple blueprint modules, each responsible for specific domains. These modules communicate with the MongoDB database, which stores structured information about users, components, and recommendations. For authentication and session management we use JSON Web Tokens (JWTs) [46] to ensure secure and user-specific interactions. The system's recommendation logic integrates several machine-learning techniques, including basic recommender, CF for user-item similarity, CBF based on item features, and a hybrid method that combines the strengths of both approaches. Each recommendation strategy is developed as an independent service, enhancing the modularity and ease of updating algorithms without disrupting the entire system:

1. **Basic Recommender**:

- It uses the SentenceTransformer model to encode component data into vectors, enabling cosine similarity calculations for ranking recommendations.
- Calculates similarity indices between components and generates recommendations by selecting the top similar items to the user's selected component.
- Supports dynamic adaptation to user preferences by enabling personalized component filtering and recommendation adjustments based on the user's profile and the database.

2. **CBF**:
   - Processes data using feature engineering and one-hot encoding to convert categorical attributes into numerical formats.
   - Uses Random Forest Regressor for predicting ratings based on user preferences.
   - Filters out components already rated by the user and ranks the remaining items based on predicted scores.

3. **CF**:
   - Constructs a user-item matrix from user ratings.
   - Calculates similarities using cosine similarity to identify like-minded users.
   - Recommends items highly rated by similar users but not yet rated by the target user.

4. **Hybrid Method**:
   - Combines the outputs of CBF and CF using weighted scores.
   - Assigns a 70% weight to content-based recommendations and 30% to collaborative recommendations, ensuring a balanced and comprehensive suggestion list.

The preprocessing module prepares global and user-specific datasets by performing data cleaning to remove invalid or inconsistent entries, feature extraction to convert specifications like ports, formats, and prices into numerical or categorical features, and normalization to scale numerical attributes such as price to a uniform range for improved model accuracy. To maintain an up-to-date database of components, the system uses web scraping techniques to retrieve product attributes.

3.1.2 MongoDB database

MongoDB is a NoSQL database known for its flexibility, scalability, and ease of integration with modern application stacks. Unlike relational databases, MongoDB stores data in a

document-oriented format using JSON-like structures, making it ideal for dynamic and evolving data schemas. It supports features like horizontal scaling, high availability, and built-in replication, which align well with the needs of modern web applications. We chose this type of database because its document-oriented approach allows us to store data in a flexible, schema-less format. This means we can easily adapt our database structure to changes in application requirements without undergoing extensive schema migrations. As the user base and the number of components grow, MongoDB ensures we maintain performance without significant infrastructure changes. Flask backend integrates seamlessly with MongoDB using pymongo library [47].

3.1.2.1 React frontend

React is a widely adopted open-source JavaScript library used for building user interfaces. Developed by Facebook, React emphasizes the creation of reusable UI components and the efficient management of the user interface state using a virtual DOM. React applications are component-based, enabling developers to modularize the UI into manageable pieces of code. React offers several key features that make it a popular choice for building user interfaces. It employs component-based architecture, where UI elements are encapsulated into independent, reusable components, making development modular and maintainable. Our system's frontend is built entirely using React, employing modern JavaScript features and practices. The architecture is structured into logical components that map to specific functionalities of the system. Each component represents a distinct piece of the user interface, such as navigation, user authentication, and data visualization. The application is built using several essential React components that work together to deliver its functionality. At its core is the **App Component**, which serves as the root of the application. It initializes the app, incorporates the header, and sets up routing between different pages using the react-chrome-extension-router library. This component also manages global state through the useJWTToken hook, enabling actions like storing, removing, and utilizing tokens.

The routing system connects various pages, each serving a distinct purpose. The **Home page** provides basic first page interface for users, while the **Login** and **Register pages** handle user authentication by securely interacting with backend endpoints. These pages utilize axios [48] for API calls and hash passwords with js-sha256 before transmission for added security. The **Admin Page** empowers administrators to manage system components

and user data through various backend endpoints, while the **Profile page** allows users to update personal information or delete their account.

A key part of the application is the recommendation system, which includes dedicated pages for different algorithms. The **Recommend page** fetches and displays user-specific recommendations from the backend. Specialized pages such as **RecommendCBF**, **RecommendCF**, and **RecommendHybrid** implement CBF, CF and hybrid recommendation algorithms, respectively, by interacting with their corresponding backend endpoints.

Additional components enhance the application's functionality and user experience. The **ComponentCard** dynamically renders cards for various items, including processors, coolers, and motherboards, based on the item type. The **BadgeAv** component visually indicates the availability of components using color-coded badges, making it easier for users to identify stock status at a glance. React components communicate with the backend using axios, a promise-based HTTP client. Authorization is handled using JWT tokens, which are stored in local storage and retrieved using the useJWTToken custom hook.

```
axios({
    method: "POST",
    url: "http://localhost:5000/auth/login",
    data: form
})
.then((response) => {
    storeToken(response.data.access_token);
    goTo(Home);
})
.catch((error) => console.log(error.response));
```

Figure 7: User authentication code

The code snippet in Figure 7 demonstrates a login process using the axios library to make a POST request to the backend at the endpoint http://localhost:5000/auth/login. It sends user credentials as form data in the request body. Upon receiving a successful response from the server, the access_token provided in the response is stored using a storeToken function, and the user is redirected to the home page using the goTo(Home) function. If the request fails, the error details are logged to the console through the catch block, enabling debugging and error handling. This code is designed to facilitate user authentication by securely transmitting data and managing authentication tokens effectively.

```
axios({
    method: "GET",
    url: "http://127.0.0.1:5000/data/recommended",
})
.then((response) => setComponents(response.data))
.catch((error) => console.log(error.response));
```

Figure 8: Code snippet for Fetching Recommendations

The code snippet in Figure 8 illustrates a GET request made using the axios library to retrieve recommended data from the backend API at http://127.0.0.1:5000/data/recommended. Upon receiving a successful response, the then block is executed to update the application's state with the received data by calling the setComponents function. If the request fails, the catch block handles the error by logging the error response to the console for debugging purposes. This code is designed to fetch recommendation data efficiently while ensuring errors are managed appropriately.

State management is achieved using React's built-in hooks useState and useEffect. This ensures an intuitive flow for fetching and displaying dynamic data.

The application uses react-bootstrap [49] for styling and layout. Components like Container, Row, and Card provide a responsive and visually appealing design while maintaining consistency across the application.

The React-based frontend also functions as a browser extension, designed to streamline the process of collecting product information and adding it to a user's profile. Using Manifest Version 3, the extension integrates with the user's browser to monitor and capture URLs from supported websites, such as those matching https://anni.si/*. This ensures only relevant product URLs are collected. The captured URLs are temporarily stored and can be retrieved or processed through interactions with the extension popup. When users interact with the extension, the collected URLs are sent to the backend to add the corresponding products to their profile automatically. This automation significantly simplifies the process for users, eliminating the need for manual input while ensuring a personalized and efficient experience.

3.2 Recommender System implementation

In our research we integrated four different algorithms:

1. **Base Recommender Module**: this serves as the foundation for encoding item attributes, calculating cosine similarity, and implementing item-based suggestions using embedding model SentenceTransformer [50].

2. **Content-Based Filtering:** this approach uses item features and user preferences to train models (Random Forest) and predicts recommendations based on feature similarity.
3. **Collaborative Filtering:** this algorithm focuses on user-user similarities and item-user interactions to predict recommendations for a specific user.
4. **Hybrid Recommendation System**: the hybrid approach combines the strengths of collaborative filtering and content-based filtering, integrating their scores to deliver more accurate recommendations.

3.2.1 Base recommender system

The base recommender module uses an embedding-based similarity approach to generate item recommendations. The process begins with retrieving item data from a database. For a given target item, identified by its name, the algorithm determines its category and fetches all items within the same category. Next, the attributes of these items are combined into a single textual representation. This text is then converted into high-dimensional numerical vectors, or embeddings, using a pre-trained SentenceTransformer model (all-MiniLM-L6-v2) [51]. These embeddings are designed to capture the semantic meaning of the textual data.

The algorithm proceeds by calculating the cosine similarity between the embedding of the target item and the embeddings of all other items within its category. Cosine similarity quantifies how closely two items are related in a vector space, with scores ranging from -1 (completely dissimilar) to 1 (identical). Based on these similarity scores, the items are ranked, and the most similar items, excluding the target item itself, are returned as recommendations.

Despite its efficiency, this method has several limitations that contribute to its relatively lower accuracy compared to other algorithms. It relies on pre-trained SentenceTransformer embeddings, which lack the domain-specific context required to distinguish nuanced differences among computer components. Additionally, this approach does not incorporate user-specific information, such as past purchases or ratings, resulting in a lack of personalization. The algorithm also focuses solely on textual data, ignoring critical numerical and categorical features, which are often essential for accurate recommendations. This approach does not involve training on historical data, as seen in

machine learning-based methods, which limits its ability to learn and adapt to user behavior over time.

3.2.2 Content-based recommender system

The CBF algorithm recommends items by utilizing both item attributes and user-specific preferences. This method builds a machine learning model tailored to each user, predicting how well they might like new items based on the features of items they have previously interacted with. The process begins with data preparation, where the algorithm accesses user-specific data stored as CSV files in a designated folder. Each file contains details such as item attributes (e.g., type, brand, and technical specifications) and user-assigned ratings. Additionally, a global dataset of all available components serves as a catalog for potential recommendations. The algorithm ensures the completeness of the data, particularly the presence of a rating column, and skips any component types with insufficient data.

During feature engineering, non-numeric attributes like brand and type are converted into dummy variables through one-hot encoding, ensuring that all attributes are represented numerically. The global dataset is then aligned with the user dataset to ensure compatibility, with any missing features filled with zeros. Once the data is prepared, the user's dataset is split into features (X) and target values (y), where the features represent the item attributes, and the target values correspond to user ratings. A Random Forest Regressor [52] is trained on this data to learn the relationship between item attributes and user ratings. Random Forest is chosen for its ability to handle mixed data types and its robustness against overfitting. The model is trained and validated by dividing the dataset into training (80%) and testing (20%) subsets.

After training, the model predicts ratings for items in the global dataset that the user has not interacted with, generating a list of unrated items ranked by predicted ratings. The top N items (default: 5) are selected as recommendations. To handle the cold start problem, the algorithm skips component types where the user dataset is too small, as insufficient data prevents meaningful training. The generated recommendations for each component type are saved in CSV files and are also stored in the database.

The strengths of this algorithm include its ability to provide personalized recommendations by building individual models for each user, its explainability through Random Forest feature importance, and its flexibility in handling structured datasets with both categorical and numerical attributes. While it struggles with sparse user data, the algorithm can still

recommend new items based on their attributes, partially addressing the cold start problem. The algorithm may suffer from over-specialization, offering recommendations limited to items similar to those the user has already rated, leading to a lack of diversity. Its effectiveness is heavily dependent on the size and quality of the user dataset, and it cannot capture broader trends, such as popular items among other users. Moreover, the need to train a separate model for each user and component type introduces computational overhead, which can be resource-intensive for large user bases.

3.2.3 Collaborative-filtering recommender system

The CF algorithm generates recommendations by analyzing user interactions, in our case ratings, to uncover patterns of shared preferences among users. This approach focuses on relationships between users and uses similarity-based methods to predict preferences for items a user has not yet interacted with. The process begins with data retrieval, where user information and interaction data are loaded. Usernames are extracted from the database, while item details for various component types are obtained from pre-defined CSV files. Additionally, individual user files containing interaction data, such as ratings, are read from the dataset folder. This data is then organized into a complete dataset of available components and user-specific data categorized by component type.

A user-item matrix is created, where rows represent users, columns represent items, and the cell values capture user-item interactions, such as ratings. For items a user has not interacted with, a default value of 0 is assigned, representing the absence of a rating. This matrix serves as the basis for similarity calculations. Pairwise cosine similarity is computed between users based on their respective rows in the user-item matrix. Cosine similarity measures how closely aligned two users are in their preferences, regardless of the magnitude of their ratings. The result is a user similarity matrix, where each cell indicates the similarity score between two users.

To generate recommendations, the algorithm identifies users most similar to the target user (excluding the target user). It aggregates items rated highly by these similar users but not yet rated by the target user. Predicted ratings for these items are calculated using a weighted average, where the ratings are weighted by the similarity scores of the users who provided them. Items are ranked based on their predicted ratings, and the top N items (default: 5) are selected as recommendations. The recommendations are then saved both as CSV files for each user and in the database.

$$Predictedrating_{item} = \sum (SimilarityScore_{user} \times Rating_{item})$$

Equation 4: Predicted rating

Equation 4 predicts a target user's rating for an item by using a weighted average of ratings from other users, scaled by their similarity scores. Higher similarity scores indicate stronger alignment with the target user, giving those users' ratings more influence. The prediction sums these weighted contributions, ensuring that users with preferences most similar to the target user have the greatest impact. This approach enhances the accuracy and personalization of recommendations by prioritizing relevant input.

The CF algorithm offers several strengths. It provides highly personalized recommendations based on user interactions and shared preferences, and it is not reliant on item metadata, making it effective even when item attributes are unavailable or inconsistent. Additionally, it can recommend items that are dissimilar in features but popular among similar users, enabling cross-domain suggestions. The algorithm struggles with the cold start problem for new users who lack sufficient interaction data, and sparsity in the user-item matrix can make similarity calculations unreliable. Popular items with high interaction volumes may dominate recommendations, reducing diversity, and the computational complexity of calculating pairwise similarities and aggregating ratings can be resource-intensive for large datasets.

3.2.4 Hybrid recommender system

The Hybrid Recommender System combines the strengths of CBF and CF to deliver more accurate and diverse recommendations. By merging these two approaches, the algorithm addresses their individual weaknesses and uses their complementary strengths. The workflow begins with retrieving recommendations generated by both systems. The CBF system provides predictions based on item attributes and user preferences, assigning each item a content-based score (cbf_score), while the CF system identifies recommendations by analyzing user interactions and assigns collaborative filtering scores (cf_score). These recommendations are retrieved from their respective collections in the database and merged using the item title as an identifier. Items present in both lists are combined, while

items exclusive to one list are included with a score of zero for the missing recommendation type.

A hybrid score is calculated for each item by combining the cbf_score and cf_score using weighted averages. Weights prioritize content-based filtering (70%) over collaborative filtering (30%), but these can be adjusted to emphasize one approach depending on the application. Items are then ranked based on their hybrid scores, and the top items are selected as the final recommendations. These results are stored in the database, replacing any previous hybrid recommendations for the user.

This hybrid approach offers significant advantages. By integrating CBF and CF, the system provides personalized recommendations based on item attributes while also introducing diversity through insights from similar users. It addresses the cold start problem by allowing CF to recommend popular items for new users and enabling CBF to suggest items using metadata when interaction data is unavailable. The system also reduces over-specialization by counterbalancing CBF's narrow focus with CF's broader perspective, resulting in improved accuracy and variety in recommendations. The flexibility to adjust the weighting of the two methods allows the system to adapt to different domains and user behaviors.

It is also more computationally complex than standalone methods and requires careful calibration of merging and weighting to achieve optimal performance. Furthermore, its effectiveness depends on the availability of both interaction data for CF and item attribute data for CBF. If either dataset is incomplete, the system's performance may degrade.

3.3.3 Tool interface

The developed recommender system is designed as a web-based tool that can also be accessed as a Chrome extension. The tool provides multiple recommendation methods, filtering options, and user interaction features to enhance the component selection process. The tool's interface consists of several main sections. The navigation bar shown in Figure 9 provides access to different sections of the tool, including profile management, stored components, recommendations, and admin functionalities.

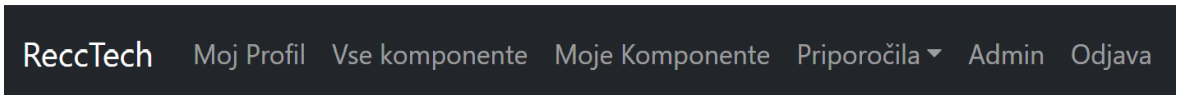


Figure 9: Navigation bar

Figure 10 shows that users can select different recommendation types from a dropdown menu, including content-based filtering (CBF), collaborative filtering (CF), and a hybrid approach.

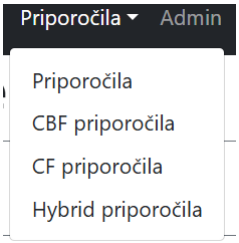


Figure 10: Recommendation algorithms

Users can store and manage their own components within the system. This allows the recommendation algorithms to provide personalized suggestions based on the stored items. The interface in Figure 11 includes My Components Page, which displays all of the user's saved components, with options to search, filter and add new components using URL from the website.

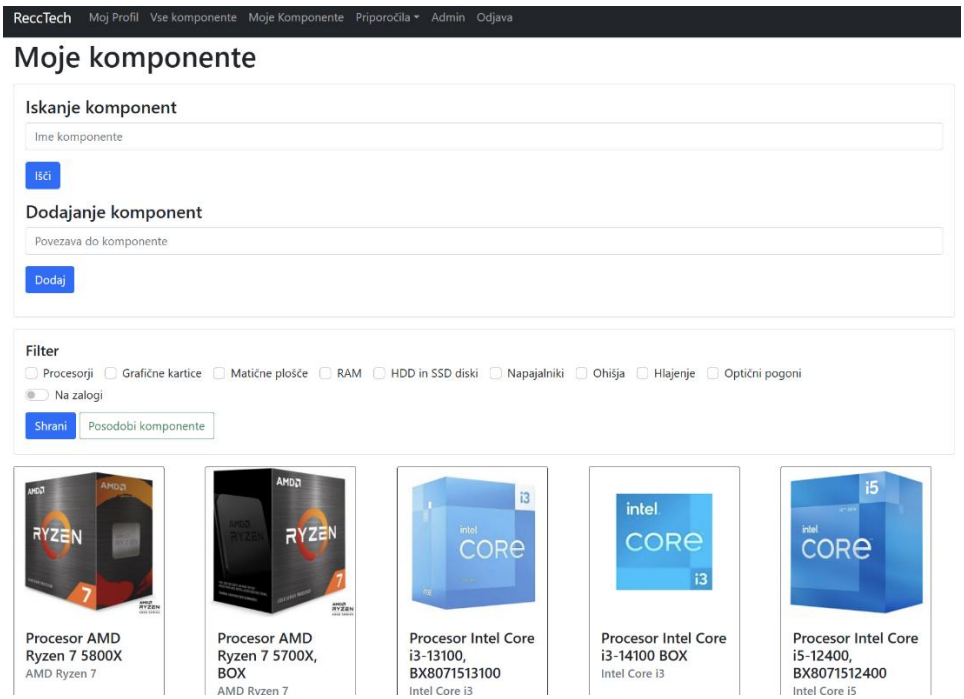


Figure 11: User's components

A product card shown in Figure 12 displays detailed information about a specific component, including specifications, availability, pricing, and interactive features such as rating, copying, or comparing items.

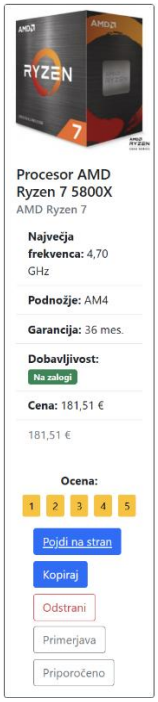


Figure 12: Component card

The tool allows users to filter stored and recommended components based on categories using different filters shown in Figure 13. They can also apply availability filters to only display in-stock items.

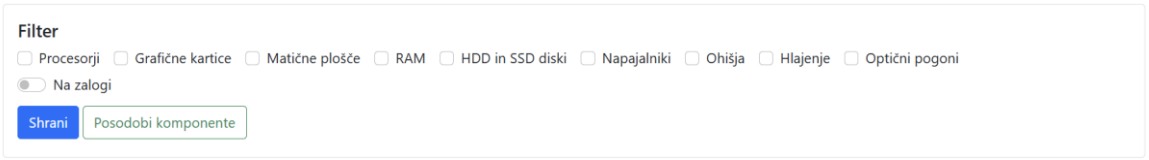


Figure 13: Filtering options

Figure 14 shows that the final recommendations are displayed in an intuitive grid format, showing relevant product details such as storage capacity, price, and availability.

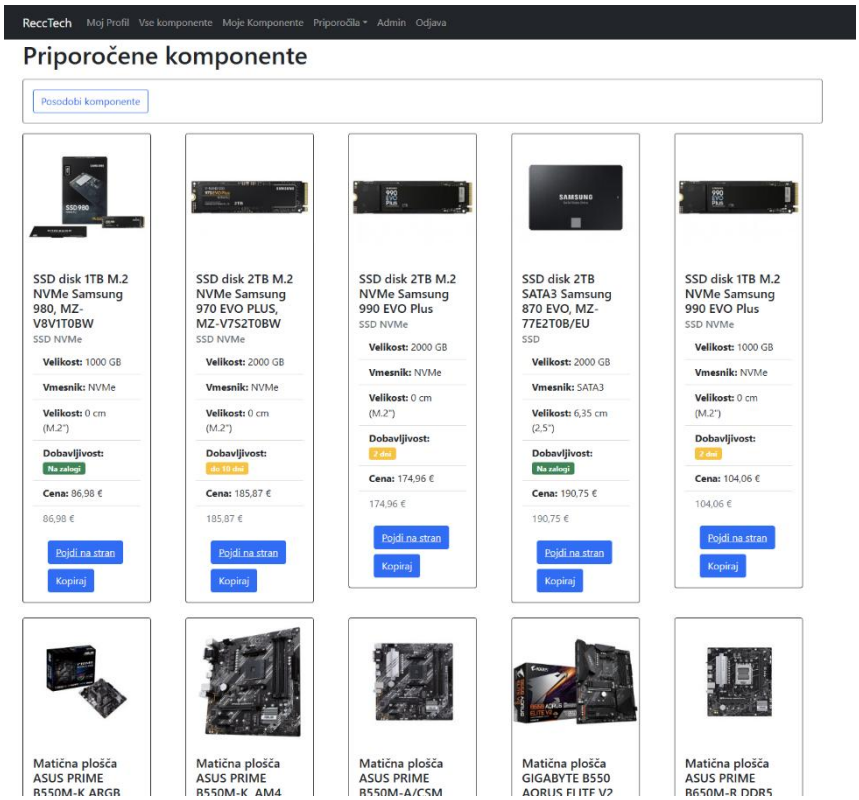


Figure 14: Recommended components

In addition to the web interface, the tool is available as a Chrome extension shown in Figure 15. This enables users to access recommendations and manage components directly from their browser without opening the full website.

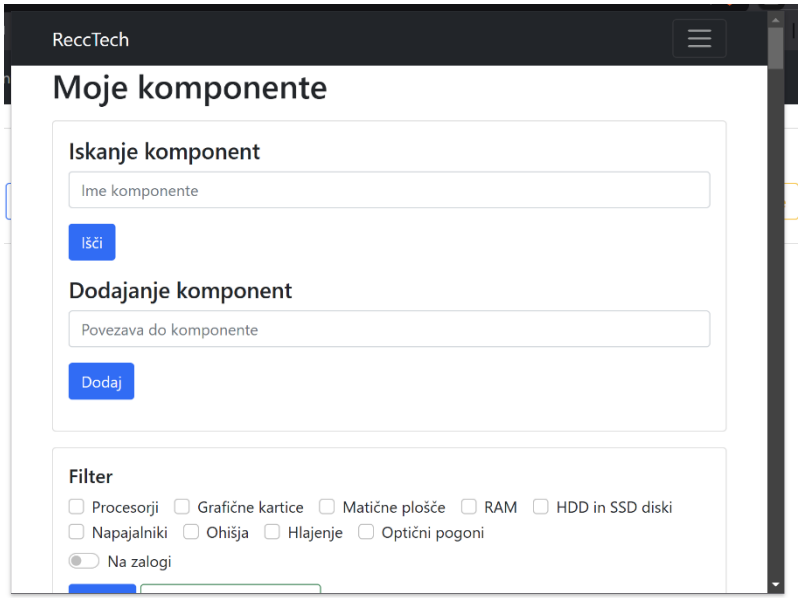


Figure 15: Extension

4 EXPERIMENTAL RESULTS

4.1 Experimental setup

The objective of this experiment is to evaluate the performance and recommendation speed of four distinct algorithms implemented in our system. These algorithms will be assessed under two different user configuration setups. The primary focus is on measuring the time taken to generate recommendations for a specific user under varying data and user scenarios.

The experiment involves a set of predefined user profiles, each representing a distinct configuration of components. These configurations are as follows:

1. Setup 1:
    - User A: 4 processors, 4 motherboards, 4 graphics cards, 3 RAM modules.
    - User B: 5 processors, 3 motherboards, 3 RAM modules.
    - User C: 4 graphics cards, 3 RAM modules.
2. Setup 2:
    - User A: 6 processors, 4 motherboards, 4 RAM modules, 4 drives, 4 graphics cards, 2 cases.
    - User B: same configuration as in Setup 1.
    - User C: Same configuration as in Setup 1.
    - User D: 3 graphics cards, 3 processors.
    - User E: 2 drives, 3 motherboards.

The experiment evaluates the recommendation speed for User A in both setups. The rationale for focusing on User A is their diverse and expanded configuration in Setup 2, which allows for testing the system's scalability and ability to handle a higher number of different component types and users. The following implemented algorithms will be assessed: **Basic Recommender, Content-Based Filtering, Collaborative Filtering and Hybrid Recommender System.**

User profiles and their respective configurations will be preloaded into the system. Component details are preprocessed and stored in a structured format. We will repeat each test scenario 10 times to account for variability in system performance and record execution times for recommendations in each run. We will then calculate the average execution time across the repeated test. The primary metric is the average time taken to generate recommendations. The timing also included database search operations to

provide a realistic performance evaluation. Outputs from print() statements were disabled during the experiment to avoid skewing the timing results.

4.2 Performance of the algorithms

The results of the experiment are summarized in Table 3 and illustrated in Figure 16, highlighting the performance of four different recommendation algorithms across two setups. The analysis is based on the average time taken to generate recommendations, measured in seconds.

Table 3: Speed of different algorithms

	Setup 1	Setup 2
Basic Recommender	37,22 s	64,07 s
CBF Recommender	1,22 s	1,96 s
CF Recommender	0,74 s	1,26 s
Hybrid Recommender	0,91 s	1,42 s

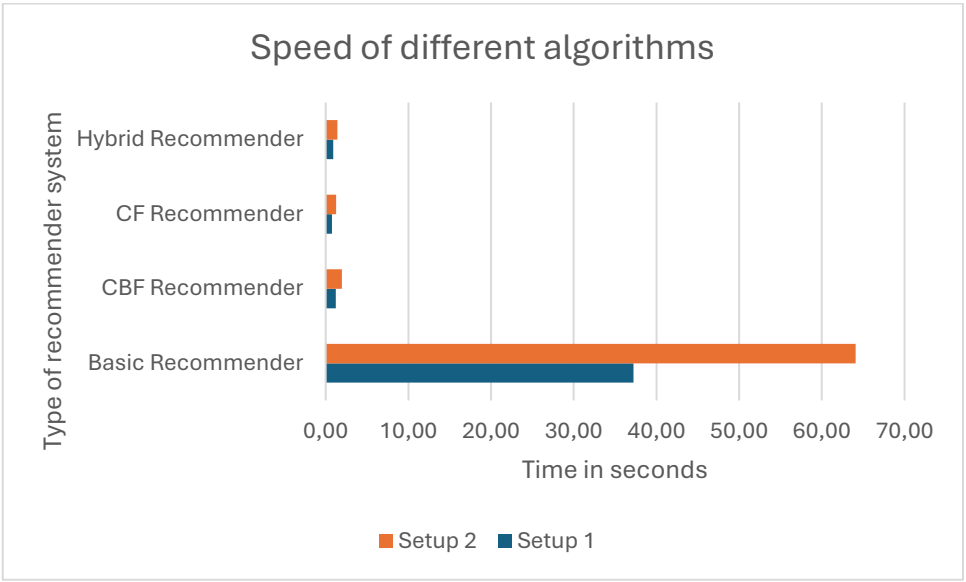


Figure 16: Speed of different algorithms

The Basic Recommender demonstrated significantly slower performance compared to the other algorithms in both setups. In Setup 1, it required 37,22 seconds, while in Setup 2, the time increased to 64,07 seconds. The primary reason for this performance is the utilization of SentenceTransformers, which involves computationally expensive processes such as encoding and cosine similarity calculations for large datasets. As the number of components and types increases in Setup 2, the computational load grows, resulting in higher execution times.

The CBF showed fast and consistent performance, taking 1,22 seconds in Setup 1 and slightly more at 1,96 seconds in Setup 2. The increase in time can be attributed to the larger variety of component types and additional users in Setup 2, but the algorithm's optimized structure kept the increase minimal.

The CF Recommender exhibited the fastest response times among all algorithms, with 0,74 seconds in Setup 1 and 1,26 seconds in Setup 2. Its speed advantage arises from its reliance on pre-computed similarity matrices, which efficiently handle additional users and data types.

The Hybrid Recommender, combining the strengths of the CBF and CF approaches, achieved moderate execution times of 0,91 seconds in Setup 1 and 1,42 seconds in Setup 2. Its performance is slightly slower than CF but remains competitive due to its weighted combination of algorithms, balancing accuracy and efficiency.

Setup 2 introduces additional components and new users (Users D and E), which increase the complexity of the recommendation process.

All algorithms experienced an increase in execution time in Setup 2, but the extent varied significantly. The smaller relative increase for the Hybrid Recommender suggests that its balanced approach can handle scaling more gracefully compared to standalone methods like the Basic or CF Recommenders

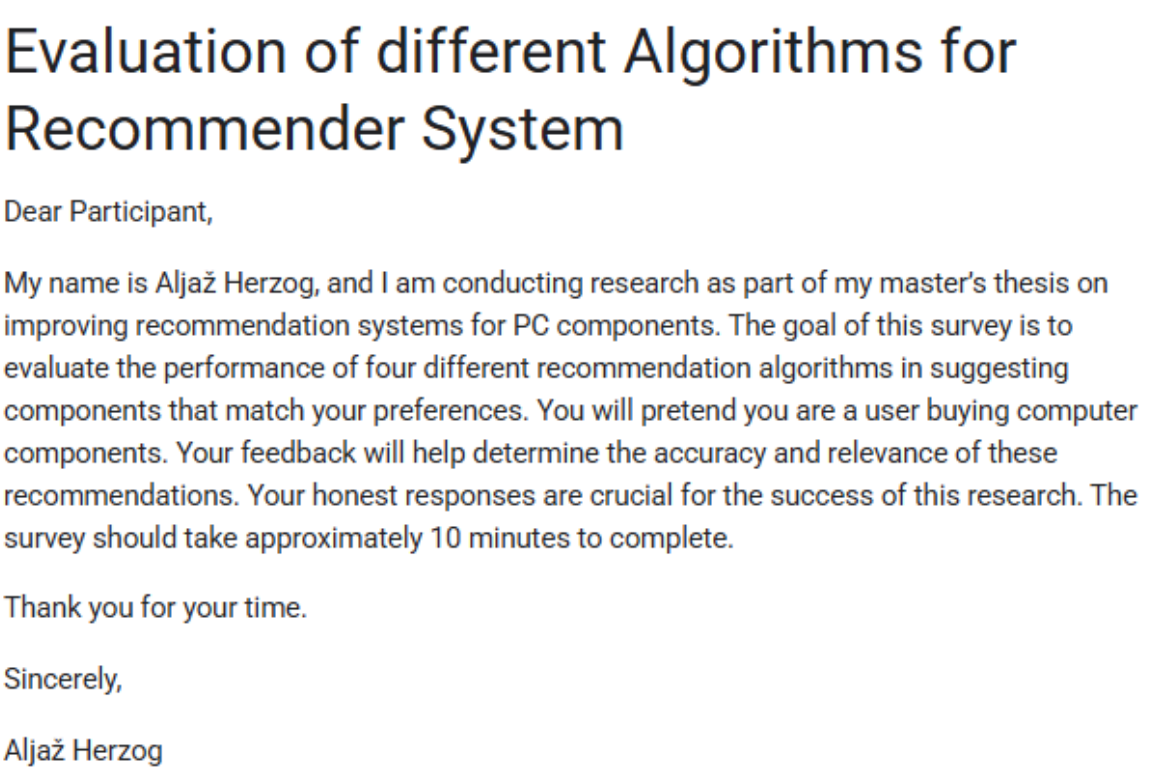
While the Basic Recommender demonstrates the ability to generate recommendations, its reliance on resource-intensive operations like SentenceTransformers makes it unsuitable for scenarios involving larger datasets or diverse user profiles. CBF and CF are highly efficient and scalable, with CF being the fastest and CBF offering good performance while leveraging content-based features. The Hybrid approach strikes a balance between speed and accuracy, showing potential as the best compromise for applications requiring both precision and efficiency.

Future considerations are that incorporating lighter embedding models or pre-computed embeddings could dramatically improve the Basic Recommender's performance. Further experiments with larger datasets and more diverse user configurations can validate the robustness of these algorithms. Adjusting the weights of the Hybrid Recommender's components may further optimize its balance between speed and recommendation quality

4.3 User Experience

The evaluation of implemented recommender systems was done through a user survey. These systems include Basic Recommender, CBF and CF. The Hybrid Recommender was excluded as it produced identical recommendations to the CBF system due to a limited user dataset, which impacted the CF performance negatively. The survey aimed to measure how effectively these systems could recommend PC components that matched user preferences. The components included processors, motherboards, RAM modules, storage drives and graphics cards. Participants were guided through a process where they assessed recommendations generated by the three systems.

They received the introduction visible in Figure 17:



Evaluation of different Algorithms for Recommender System

Dear Participant,

My name is Aljaž Herzog, and I am conducting research as part of my master's thesis on improving recommendation systems for PC components. The goal of this survey is to evaluate the performance of four different recommendation algorithms in suggesting components that match your preferences. You will pretend you are a user buying computer components. Your feedback will help determine the accuracy and relevance of these recommendations. Your honest responses are crucial for the success of this research. The survey should take approximately 10 minutes to complete.

Thank you for your time.

Sincerely,

Aljaž Herzog

Figure 17: Introduction for survey

Each participant reviewed a table of five previously rated products, along with their own ratings. Ratings reflected preferences, where 1 indicated dislike and 5 indicated strong preference. Participants were then presented with three recommendation tables, each generated by one of the systems under evaluation. For each table, participants provided:

- An overall relevance score for the recommendations, ranging from 1 (poor) to 10 (excellent).
- An indication of whether they would consider purchasing at least one of the recommended items (Yes/No).

The survey provided a structured mechanism for collecting user feedback on three recommendation systems for PC components. It also served as a valuable tool for understanding user preferences and assessing the potential of different algorithms in real-world scenarios.

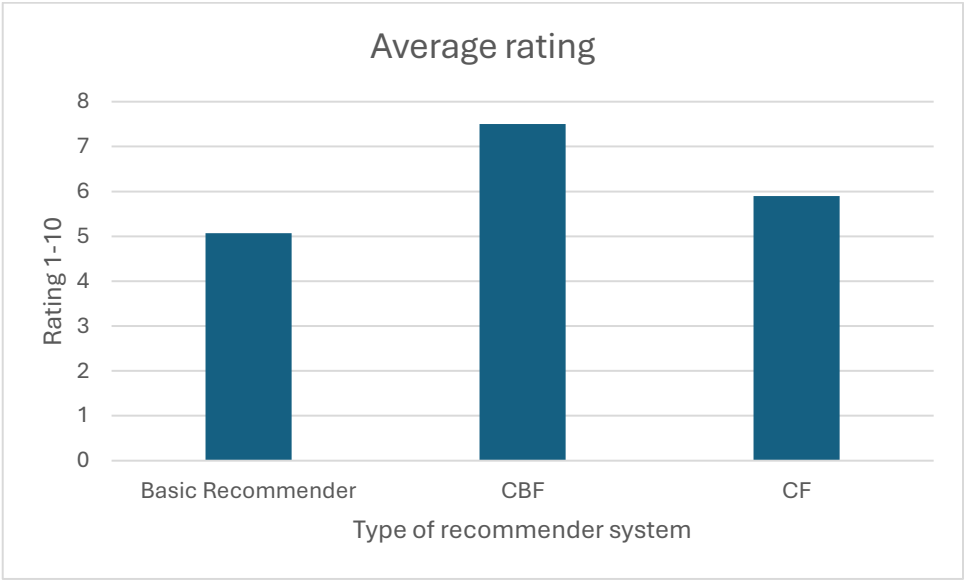


Figure 18: Average rating of algorithms

The bar chart in Figure 18 illustrates the differences in average user ratings for the three algorithms. The ratings were calculated by averaging the scores given by participants across all recommendation instances. The CBF approach significantly outperformed the others, achieving the highest average rating of 7,5. This suggests that participants found the recommendations from the CBF system to be the most relevant and aligned with their preferences. The CF system followed with a moderately higher score of 5,9, showing some effectiveness but still falling short compared to the CBF system. The Basic Recommender,

which employs simpler recommendation logic, received the lowest average rating of 5,07, indicating that its recommendations were the least satisfactory overall. The higher score for CBF suggests that using item metadata and tailoring recommendations to user preferences based on content was well-received by participants. The relatively lower performance of CF could be attributed to the limited user dataset, which likely impacted on its ability to make accurate recommendations. The Basic Recommender's performance highlights the drawbacks of relying on static or rule-based recommendation methods in scenarios where user preferences vary widely.

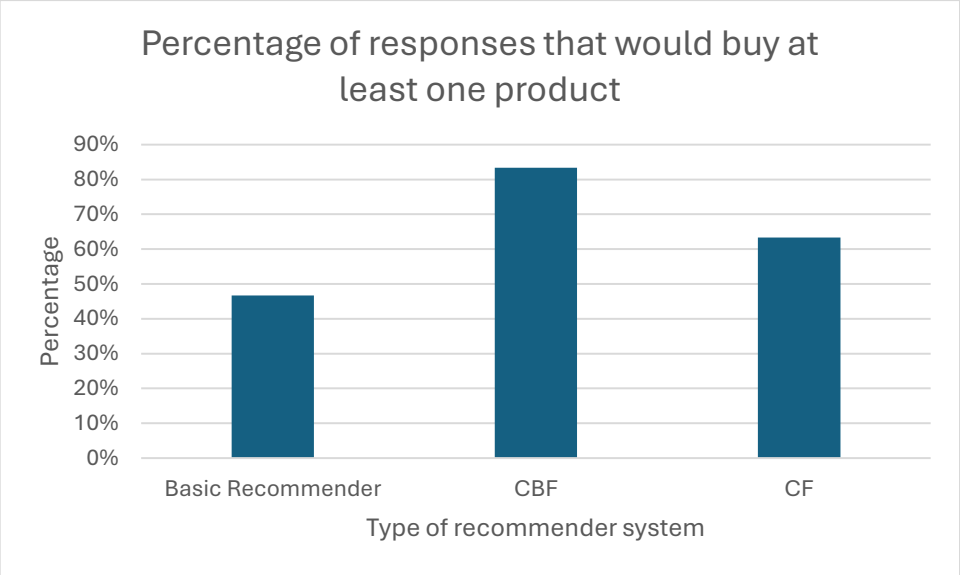


Figure 19: Percentage of responses that would buy at least one recommended product

The survey also assessed the willingness of participants to purchase at least one product recommended by each algorithm. This was measured as the percentage of participants who responded "Yes" to the question regarding their purchase intent. Figure 19 highlights clear differences in participants' willingness to purchase items recommended by the three algorithms. CBF emerged as the most effective system in generating actionable recommendations, with 83% of participants indicating they would buy at least one recommended product. This suggests that the algorithm's reliance on item features effectively aligned with user preferences. CF followed with 63%, demonstrating moderate success in generating recommendations that participants found convincing enough to consider purchasing. The Basic Recommender achieved the lowest percentage, with only 47% of participants wanting to buy an item from its recommendations. This outcome

reflects the limitations of simpler, rule-based approaches in capturing and satisfying diverse user needs.

The high willingness to purchase recommendations from CBF reinforces its ability to provide relevant, personalized suggestions based on item attributes. While CF showed promise, its effectiveness likely suffered due to the limited dataset, which restricted its ability to fully use user interaction patterns. The Basic Recommender's lower performance underscores its limitations in adapting to complex and personalized user preferences.

The user survey provided valuable insights into the effectiveness of the implemented recommender systems: Basic Recommender, CBF and CF. By evaluating these systems based on their relevance and potential to drive purchasing decisions, the survey highlighted key strengths and limitations of each approach. The results demonstrated that the CBF system performed significantly better in both relevance ratings and purchase intent, showcasing its ability to provide tailored recommendations that align with user preferences. The CF system displayed moderate performance but was hindered by the limited dataset, which reduced its ability to use collaborative data. The Basic Recommender, while simpler and easier to implement, struggled to meet the complex needs of diverse users. These findings underscore the importance of leveraging robust item metadata, as seen in the success of CBF, and highlight the need for larger and more comprehensive user datasets to enhance the potential of CF systems. The limitations of the Basic Recommender further emphasize the necessity of moving beyond static and rule-based systems in favor of more sophisticated, data-driven approaches. This evaluation provides a foundation for refining and optimizing recommendation systems for PC components. Future efforts could focus on integrating hybrid models that combine the strengths of CBF and CF to achieve even greater accuracy and relevance, especially as user data availability increases.

5 DISCUSSION

5.1 Research questions

RQ1: Which recommendation system algorithms are best suited for recommending computer components according to their specific characteristics and user needs?

The best single algorithm for recommending computer components based on their specific characteristics and user needs is KBF. This approach is particularly effective because it ensures compatibility between components, addresses technical requirements, and does not rely on large amounts of user interaction data, making it well-suited for the computer hardware domain. However, the overall best solution is a hybrid recommender system that combines multiple approaches, including CF, CBF and KBF. The systems use the strengths of each algorithm while mitigating their individual weaknesses. KBF ensures that recommended components are compatible and meet user needs. CBF provides recommendations based on technical specifications and product attributes. CF enhances personalization by identifying patterns in user behavior and preferences. By integrating these methods, the hybrid approach offers more accurate, diverse, and user-friendly recommendations while effectively handling challenges like the cold start problem and data sparsity.

RQ2: How do different algorithms compare in terms of accuracy, speed, and adaptability?

The experiments revealed significant variations in the performance of the four algorithms tested: Basic Recommender, CBF, CF and the Hybrid Recommender. User survey indicated that the CBF approach achieved the highest relevance score (7,5 out of 10) and had the highest purchase intent (83%). This suggests superior accuracy in tailoring recommendations to user preferences. The CF system followed with a moderately high relevance score (5,9) and purchase intent (63%), whereas the Basic Recommender performed the worst, with a relevance score of 5,07 and purchase intent of 47%.

In terms of speed, the CF algorithm was the fastest, with execution times of 0,74 seconds (Setup 1) and 1,26 seconds (Setup 2). The CBF algorithm performed slightly slower but remained efficient, with times of 1,22 seconds and 1,96 seconds. The Hybrid Recommender balanced speed and accuracy with times of 0,91 seconds and 1.42 seconds. The Basic Recommender demonstrated the poorest performance, with times of 37,22 seconds and 64,07 seconds due to its computationally expensive SentenceTransformers.

The Hybrid Recommender exhibited the greatest adaptability by effectively combining the strengths of CBF and CF. The Basic Recommender lacked adaptability, as its reliance on static computations rendered it ineffective in scaling to larger datasets and diverse configurations.

In conclusion, the CF algorithm is the best choice for applications requiring speed, while the CBF system excels in accuracy. The Hybrid Recommender strikes a balance, making it suitable for scenarios demanding both precision and efficiency. The Basic Recommender is inadequate for complex applications due to its slow performance and low accuracy.

RQ3: How do different data sources affect the quality of recommendations?

The study explored the effects of varied data configurations on recommendation quality. In Setup 2, the introduction of additional components and new user profiles increased the complexity of recommendations. The algorithms generally showed increased execution times, but the Hybrid Recommender handled this growth more gracefully, suggesting its potential for scalability.

The inclusion of Users D and E in Setup 2 allowed testing of adaptability to new profiles. The Hybrid and CBF algorithms managed this diversity effectively, whereas the Basic Recommender struggled to provide meaningful suggestions.

Diverse and well-prepared datasets improve the quality of recommendations. Algorithms leveraging comprehensive item metadata (CBF) or collaborative user interactions (CF) benefit the most. The Hybrid Recommender can effectively balance these advantages, underscoring the importance of robust data preparation.

5.2 Hypotheses

H1: Using an appropriate algorithm in the recommender system can significantly improve the accuracy of recommendations and user satisfaction.

The results confirm this hypothesis. The CBF and Hybrid Recommenders demonstrated high accuracy and user satisfaction, as evidenced by their relevance scores and purchase intent. Conversely, the Basic Recommender, with its simpler logic, failed to meet user expectations effectively.

H2: Advanced machine learning techniques improve the accuracy and personalization of the recommender system.

This hypothesis is supported by the superior performance of the CBF and Hybrid algorithms. Their ability to incorporate advanced techniques like item metadata and combined methodologies significantly enhanced the personalization and accuracy of recommendations compared to the Basic Recommender.

H3: Different data sources affect the quality of recommendations; preparing appropriate data leads to greater user satisfaction.

The hypothesis is validated by the findings. Larger and more diverse datasets (Setup 2) allowed algorithms to better adapt to user needs, with the CBF and Hybrid systems benefiting most from well-prepared item metadata and collaborative patterns. The Basic Recommender's poor performance highlights the limitations of static approaches when handling complex data.

6 CONCLUSIONS AND FUTURE WORK

The primary aim of this thesis was to explore and implement a recommendation system tailored to computer components, addressing challenges in e-commerce with a focus on personalization and accuracy. The project aimed to answer key research questions about the suitability, accuracy, and adaptability of various recommendation algorithms in this domain. To achieve these goals, we implemented a complete system comprising a React-based frontend, a Flask backend, and a MongoDB database for data storage. Additionally, four algorithms were developed and integrated into the system: a basic recommender, CBF, CF and a hybrid approach combining CBF and CF.

The four algorithms provided distinct methodologies for generating recommendations, and the hybrid approach successfully combined the strengths of CBF and CF to deliver a well-rounded recommendation experience. Each algorithm's strengths and limitations were evaluated, providing valuable insights into their suitability for the domain of computer components.

Evaluation metrics revealed that among the individual algorithms, the CBF approach demonstrated the highest accuracy for our dataset, outperforming CF and the basic recommender system. The hybrid approach matched the accuracy of CBF in our experiments, primarily because the collaborative filtering component lacked sufficient user interaction data to influence the recommendations significantly. This limitation underscored the importance of larger user datasets for improving collaborative filtering and, consequently, hybrid systems.

To enhance the system and address its current limitations, the following improvements are proposed. Increasing the user base and interaction data to bolster the collaborative filtering component. More diverse user data will enable the hybrid system to effectively use both CBF and CF, enhancing recommendation accuracy and diversity. Experimenting with advanced techniques, such as neural collaborative filtering and deep learning, to capture complex relationships between users and items. Implementing adaptive weighting for the hybrid system to dynamically adjust the contribution of CBF and CF based on the availability of data. Incorporating contextual factors, such as user location, time of access, and specific use cases (e.g., gaming or video editing), to refine recommendations further.

References

[1] P. Massa and P. Avesani, “Trust Metrics in Recommender Systems,” in *Computing with Social Trust*, J. Golbeck, Ed., London: Springer London, 2009, pp. 259–285. doi: 10.1007/978-1-84800-356-9_10.

[2] L. Lü, M. Medo, C. H. Yeung, Y.-C. Zhang, Z.-K. Zhang, and T. Zhou, “Recommender systems,” *Phys Rep*, vol. 519, no. 1, pp. 1–49, 2012, doi: https://doi.org/10.1016/j.physrep.2012.02.006.

[3] A. Da’u and N. Salim, “Recommendation system based on deep learning methods: a systematic review and new directions,” *Artif Intell Rev*, vol. 53, no. 4, pp. 2709–2748, 2020, doi: 10.1007/s10462-019-09744-1.

[4] R. J. K. Almahmood and A. Tekerek, “Issues and Solutions in Deep Learning-Enabled Recommendation Systems within the E-Commerce Field,” Nov. 01, 2022, *MDPI*. doi: 10.3390/app122111256.

[5] S. Zhang, L. Yao, A. Sun, and Y. Tay, “Deep Learning Based Recommender System: A Survey and New Perspectives,” *ACM Comput. Surv.*, vol. 52, no. 1, Feb. 2019, doi: 10.1145/3285029.

[6] L. Peska, “Using the Context of User Feedback in Recommender Systems,” in *Proceedings 11th Doctoral Workshop on Mathematical and Engineering Methods in Computer Science, MEMICS 2016, Telč, Czech Republic, 21st-23rd October 2016*, J. Bouda, L. Hol\’{\i}k, J. Kofron, J. Strejcek, and A. Rambousek, Eds., in EPTCS, vol. 233. 2016, pp. 1–12. doi: 10.4204/EPTCS.233.1.

[7] H. Li, S. Zhang, and X. Wang, “A personalization recommendation algorithm for e-commerce,” *Journal of Software*, vol. 8, no. 1, pp. 176–183, 2013, doi: 10.4304/jsw.8.1.176-183.

[8] G. Jawaheer, P. Weller, and P. Kostkova, “Modeling User Preferences in Recommender Systems: A Classification Framework for Explicit and Implicit User Feedback,” *ACM Trans. Interact. Intell. Syst.*, vol. 4, no. 2, Jun. 2014, doi: 10.1145/2512208.

[9] W. Wu, L. He, and J. Yang, “Evaluating recommender systems,” in *Seventh International Conference on Digital Information Management (ICDIM 2012)*, 2012, pp. 56–61. doi: 10.1109/ICDIM.2012.6360092.

[10] P. M. Alamdari, N. J. Navimipour, M. Hosseinzadeh, A. A. Safaei, and A. Darwesh, "A Systematic Study on the Recommender Systems in the E-Commerce," *IEEE Access*, vol. 8, pp. 115694–115716, 2020, doi: 10.1109/ACCESS.2020.3002803.

[11] Y. Guo, M. Wang, and X. Li, "An interactive personalized recommendation system using the hybrid algorithm model," *Symmetry (Basel)*, vol. 9, no. 10, Oct. 2017, doi: 10.3390/sym9100216.

[12] Y. Wang, S. Chen, and K. Jin, "Prometheus Chatbot: Knowledge Graph Collaborative Large Language Model for Computer Components Recommendation," *arXiv preprint arXiv:2407.19643*, 2024.

[13] J. Di Rocco and C. Di Sipio, "ResyDuo: Combining data models and CF-based recommender systems to develop Arduino projects," in *2023 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, 2023, pp. 539–548.

[14] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, "Recommender systems survey," *Knowl Based Syst*, vol. 46, pp. 109–132, 2013, doi: https://doi.org/10.1016/j.knosys.2013.03.012.

[15] H. Wang, N. Wang, and D.-Y. Yeung, "Collaborative Deep Learning for Recommender Systems," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, in KDD '15. New York, NY, USA: Association for Computing Machinery, 2015, pp. 1235–1244. doi: 10.1145/2783258.2783273.

[16] F. Karimova, "A Survey of e-Commerce Recommender Systems," *European Scientific Journal, ESJ*, vol. 12, no. 34, p. 75, Dec. 2016, doi: 10.19044/esj.2016.v12n34p75.

[17] S. Sivapalan, A. Sadeghian, H. Rahnama, and A. M. Madni, "Recommender systems in e-commerce," in *2014 World Automation Congress (WAC)*, 2014, pp. 179–184. doi: 10.1109/WAC.2014.6935763.

[18] R. Mu, "A Survey of Recommender Systems Based on Deep Learning," *IEEE Access*, vol. 6, pp. 69009–69022, 2018, doi: 10.1109/ACCESS.2018.2880197.

[19] R. Burke, "Knowledge-based recommender systems," *Encyclopedia of library and information systems*, vol. 69, no. Supplement 32, pp. 175–186, 2000.

[20] B. Prasad, "A knowledge-based product recommendation system for e-commerce," *International Journal of Intelligent Information and Database Systems*, vol. 1, no. 1, pp. 18–36, 2007.

[21] S. Khodabandehlou, S. A. Hashemi Golpayegani, and M. Zivari Rahman, "An effective recommender system based on personality traits, demographics and behavior of customers in time context," *Data Technologies and Applications*, vol. 55, no. 1, pp. 149–174, 2021.

[22] X. W. Zhao, Y. Guo, Y. He, H. Jiang, Y. Wu, and X. Li, "We know what you want to buy: a demographic-based system for product recommendation on microblogs," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 1935–1944.

[23] G. Geetha, M. Safa, C. Fancy, and D. Saranya, "A Hybrid Approach using Collaborative filtering and Content based Filtering for Recommender System," *J Phys Conf Ser*, vol. 1000, no. 1, p. 012101, 2018, doi: 10.1088/1742-6596/1000/1/012101.

[24] S. P. Sahu, A. Nautiyal, and M. Prasad, "Machine Learning Algorithms for Recommender System - a comparative analysis," *International Journal of Computer Applications Technology and Research*, vol. 6, no. 2, pp. 97–100, Feb. 2017, doi: 10.7753/ijcatr0602.1005.

[25] W. Chen, F. Cai, H. Chen, and M. De Rijke, "Joint neural collaborative filtering for recommender systems," *ACM Transactions on Information Systems (TOIS)*, vol. 37, no. 4, pp. 1–30, 2019.

[26] M. Fu, H. Qu, Z. Yi, L. Lu, and Y. Liu, "A novel deep learning-based collaborative filtering model for recommendation system," *IEEE Trans Cybern*, vol. 49, no. 3, pp. 1084–1096, 2018.

[27] C. Musto, T. Franza, G. Semeraro, M. De Gemmis, and P. Lops, "Deep content-based recommender systems exploiting recurrent neural networks and linked open data," in *Adjunct Publication of the 26th conference on user modeling, adaptation and personalization*, 2018, pp. 239–244.

[28] I. Portugal, P. Alencar, and D. Cowan, "The use of machine learning algorithms in recommender systems: A systematic review," *Expert Syst Appl*, vol. 97, pp. 205–227, 2018, doi: https://doi.org/10.1016/j.eswa.2017.12.020.

[29] Y. Afoudi, M. Lazaar, and M. Al Achhab, "Hybrid recommendation system combined content-based filtering and collaborative prediction using artificial neural network," *Simul Model Pract Theory*, vol. 113, p. 102375, 2021.

[30] C. Christakou, S. Vrettos, and A. Stafylopatis, “A hybrid movie recommender system based on neural networks,” *International Journal on Artificial Intelligence Tools*, vol. 16, no. 05, pp. 771–792, 2007.

[31] T. K. Paradarami, N. D. Bastian, and J. L. Wightman, “A hybrid recommender system using artificial neural networks,” *Expert Syst Appl*, vol. 83, pp. 300–313, 2017.

[32] M. Gridach, “Hybrid deep neural networks for recommender systems,” *Neurocomputing*, vol. 413, pp. 23–30, 2020.

[33] K. Wang, T. Zhang, T. Xue, Y. Lu, and S.-G. Na, “E-commerce personalized recommendation analysis by deeply-learned clustering,” *J Vis Commun Image Represent*, vol. 71, p. 102735, 2020, doi: https://doi.org/10.1016/j.jvcir.2019.102735.

[34] X. Amatriain and J. Basilico, “Recommender systems in industry: A netflix case study,” in *Recommender systems handbook*, Springer, 2015, pp. 385–419.

[35] C. A. Gomez-Uribe and N. Hunt, “The netflix recommender system: Algorithms, business value, and innovation,” *ACM Transactions on Management Information Systems (TMIS)*, vol. 6, no. 4, pp. 1–19, 2015.

[36] H. Steck, L. Baltrunas, E. Elahi, D. Liang, Y. Raimond, and J. Basilico, “Deep learning for recommender systems: A Netflix case study,” *AI Mag*, vol. 42, no. 3, pp. 7–18, 2021.

[37] B. Smith and G. Linden, “Two decades of recommender systems at Amazon. com,” *IEEE Internet Comput*, vol. 21, no. 3, pp. 12–18, 2017.

[38] G. Linden, B. Smith, and J. York, “Amazon. com recommendations: Item-to-item collaborative filtering,” *IEEE Internet Comput*, vol. 7, no. 1, pp. 76–80, 2003.

[39] F. H. Del Olmo and E. Gaudioso, “Evaluation of recommender systems: A new approach,” *Expert Syst Appl*, vol. 35, no. 3, pp. 790–804, 2008.

[40] T. Silveira, M. Zhang, X. Lin, Y. Liu, and S. Ma, “How good your recommender system is? A survey on evaluations in recommendation,” *International Journal of Machine Learning and Cybernetics*, vol. 10, pp. 813–831, 2019.

[41] W. Wu, L. He, and J. Yang, “Evaluating recommender systems,” in *Seventh International Conference on Digital Information Management (ICDIM 2012)*, 2012, pp. 56–61.

[42] “Welcome to Flask — Flask Documentation (3.1.x).” [Online]. Available: https://flask.palletsprojects.com/en/stable/

[43] “MongoDB: The Developer Data Platform.” [Online]. Available: https://www.mongodb.com/

[44] “React.” [Online]. Available: https://react.dev/

[45] "HTTP request methods - HTTP MDN," Nov. 2024. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods

[46] auth0.com, "JWT.IO - JSON Web Tokens Introduction." [Online]. Available: http://jwt.io/

[47] "PyMongo 4.10.1 documentation." [Online]. Available: https://pymongo.readthedocs.io/en/stable/

[48] "react-axios," Jun. 2022. [Online]. Available: https://www.npmjs.com/package/react-axios

[49] "React Bootstrap React Bootstrap." [Online]. Available: https://react-bootstrap.netlify.app/

[50] "SentenceTransformers Documentation — Sentence Transformers documentation." [Online]. Available: https://sbert.net/

[51] "sentence-transformers/all-MiniLM-L6-v2 · Hugging Face," Jan. 2024. [Online]. Available: https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2

[52] "RandomForestRegressor." [Online]. Available: https://scikit-learn/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html