ISSN 2085-4552

Leveraging Content-Based Filtering for Personalized Game Recommendations: A Flutter-Based Mobile Application Development

Bryan Rezki Nugraha[1], Alexander Waworuntu[2]
[1,2] Department of Informatics, Universitas Multimedia Nusantara, Tangerang, Indonesia
[1]bryan.nugraha@student.umn.ac.id, [2]alex.wawo@umn.ac.id

Accepted 06 January 2025
Approved 16 January 2025

***Abstract*— The background of this study stems from the need for a recommendation system to assist users in finding games that match their interests. With the rapid growth of the gaming market, an increasing number of people engage in gaming activities. In 2022, the personal computer (PC) gaming market accounted for 37.9% of all gamers worldwide. One of the largest PC gaming platforms is Steam, developed by Valve Corporation, which boasts over 184 million active users. However, the overwhelming number of options can lead users to lose interest in purchasing games. Therefore, a recommendation system is required to help users find games that align with their preferences. The methods/theories employed in this study include data from the Steam Web API, SteamSpy API, and local JSON files. The Content-Based Filtering method, using the Cosine Similarity algorithm, was implemented to determine the similarity index between games and user preferences. Flutter was used for application development and to display the recommendation results to users. The results of this study show that the application was successfully developed, and the Content-Based Filtering method provided recommendations that met expectations. The highest cosine similarity factor achieved was 0.6454972244, indicating a fairly good level of accuracy. Application evaluation using the Technology Acceptance Model revealed positive reception, with a "Perceived Usefulness" score of 82.6% and a "Perceived Ease of Use" score of 86.2%, indicating that users found the application both useful and easy to use.**

***Index Terms*— *Terms*—Content-Based Filtering; Cosine Similarity; Flutter; similarity; Steam; SteamSpy API; Steam Web API**.

I. INTRODUCTION

One of the most popular activities in the digital era is gaming. With services like Steam, users can easily access a variety of games. Video games are a form of digital media-based activity where players aim to achieve predetermined objectives within the game [1]. According to Statista, the total revenue from the video game market is projected to reach 625.64 trillion USD by 2028 [2]. Additionally, data from 2022 reveals that the market for personal computer (PC) games is the second largest, following mobile games, accounting for 37.9% of all video game players worldwide [3].

One of the largest marketplaces for PC gaming is Steam, a game distribution platform developed by Valve Corporation, a U.S.-based company. Steam offers over 8,000 available games and boasts more than 184 million active users [4]. However, the abundance of game options presents a significant challenge for users, as too many choices can lead to decision fatigue. Research by Chernev, Böckenholt, and Goodman demonstrates that an excessive number of options can reduce consumer interest in making purchases [5]. This phenomenon highlights the need for an effective recommendation system to assist users in navigating Steam's extensive library and identifying games that align with their preferences.

Steam was chosen as the research object for several compelling reasons. As one of the largest and most influential game distribution platforms globally, its significant user base and vast library of games make it a prime candidate for studying recommendation systems. Additionally, Steam's robust Steamworks Web API and supplementary services like SteamSpy API provide access to valuable data on user activity and game information, enabling the development and testing of advanced algorithms. By addressing the issue of decision fatigue on a platform as prominent as Steam, the findings of this study have practical relevance and the potential to improve user satisfaction and engagement while supporting game developers in reaching their target audience.

Previously, several recommendation systems have been proposed to tackle similar challenges, including those based on Deep Learning [6], the K-Nearest Neighbor (KNN) algorithm [7], and matrix factorization techniques [8]. Each method has its strengths and limitations: Deep Learning achieves high accuracy but requires extensive user data for implementation, the KNN algorithm performs well with existing input, and matrix factorization can

identify personal preferences for approximately 33% of Steam users.

Among the various approaches, Content-Based Filtering has emerged as a widely used method for building recommendation systems. This technique has been successfully applied in other domains, such as movie recommendation systems based on genres [9] and property recommendation systems [10]. In the context of game recommendation systems, Content-Based Filtering utilizes user activity to generate personalized suggestions. A comparative study by the State University of Feira de Santana [11] indicates that this method is particularly effective with large and dense datasets, such as Steam's extensive game library and user base.

For these reasons, this study adopts Content-Based Filtering to develop a recommendation system that mitigates decision fatigue and assists users in finding games that match their preferences. By leveraging this approach, the study aims to address the challenges posed by Steam's vast game selection and improve the overall user experience on the platform.

II. THEORY

A. *Flutter*

Flutter is a cross-platform framework used to develop high-performance mobile applications. It was launched by Google in 2016 [12]. Flutter enables the creation of high-performance applications akin to native apps, thanks to its high-performance rendering engine. In Flutter's architecture, C/C++ code is compiled using the NDK on Android and LLVM on iOS, while Dart code is compiled Ahead Of Time (AOT) [13].

B. *Dart*

Dart is a programming language designed with principles of ease of use, familiarity for most programmers, and scalability. Dart was created to provide tools specifically tailored to meet the needs of modern software and hardware [14]. It is an Object-Oriented Programming (OOP) language developed and maintained by Google. Dart has also been utilized to develop large-scale web applications [15].

C. *Content-Based Filtering*

The Content-Based Filtering (CBF) algorithm is one of the most successful recommendation algorithms, using correlations between content as its foundation. CBF relies on an item's information, represented by attributes, which are compared with other items to calculate similarity [16]. An example of its implementation is a recommendation system that compares a user profile with the content of each document in a collection. The content of a document can be represented by several keywords that reflect the user's profile [17]. One of the methods to calculate similarity between textual data is through Cosine Similarity.

D. *Cosine Similarity*

Cosine Similarity is a commonly used metric to measure the degree of similarity between two vectors, calculated based on the cosine of the angle between them [18]. This method is also useful for measuring the similarity between two documents based on matching terms [19]. The formula for Cosine Similarity is as follows:

$$\text{cosine\_similarity}\ (\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\| \|\mathbf{b}\|} \qquad (1)$$

Explanation:

- a: Represents vector a
- b: Represents vector b
- Cosine Similarity (a,b): The similarity value between vectors a and b, calculated based on the cosine of the angle between them.

E. Technology Acceptance Model

With the growing technological demands in the 1970s and increasing system adoption failures across various organizations, predicting system usability became a popular field among researchers. In 1985, Fred Davis proposed the Technology Acceptance Model (TAM), based on the Theory of Reasoned Action (TRA), to explain individual behavior in adopting technology [20].

The Technology Acceptance Model suggests that Perceived Ease of Use and Perceived Usefulness are significant predictors of application usage, which play a major role in evaluating the effectiveness and usability of a technological system [21].

III. METHOD

A. *Problem Identification*

Before designing and developing the application, the primary step is identifying existing problems. This process was conducted through market research and by gathering information from previous studies on game recommendation systems.

B. *Literature Review*

The literature review involved collecting and analyzing related studies from various written sources, such as journals, articles, and research reports that address similar topics.

C. *Application UI Prototyping*

The initial stage of mobile application development involved creating a user interface (UI) prototype. The prototype was designed using Figma and served as the foundation for the UI of the developed application.

D. Integration of Steamworks API

In this phase, the Steamworks Web API was integrated into the application. This included implementing the login system via Steam using the OpenID service and extracting user data through the API's GET function. The Uri.parse() method was used to retrieve specific user data, which was utilized for the recommendation system.

E. Development of Core Application Functions

At this stage, the core functions designed during the prototyping phase were developed and implemented into the application. These functions were built using the tools and components provided by the Flutter framework.

F. Integration of Content-Based Filtering

The Content-Based Filtering algorithm was implemented to generate personalized recommendations by comparing the similarity between games previously played and other available games on the Steam platform. The tag data from the Steamworks Web API was utilized for this purpose. The integration process followed these steps (Figure 1):

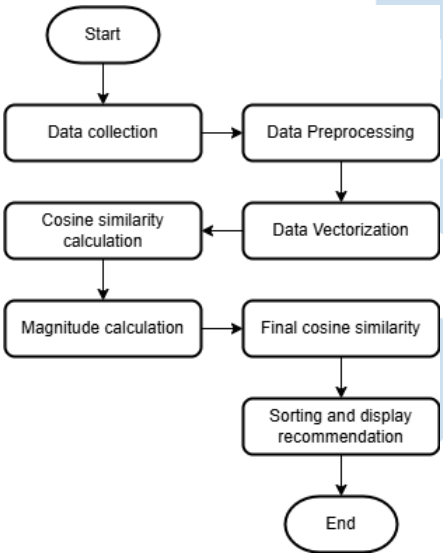

Fig. 1. Content-Based Filtering Process

1) Data Collection: The application retrieved data from Steam Web API, SteamSpy API, and a JSON dataset containing information on 55,000 Steam games.
2) Data Preprocessing: Symbols were removed from the data, and all tags were converted to lowercase for uniformity.
3) Vectorization: The preprocessed tags were vectorized into binary vectors, which were used for cosine similarity calculations.
4) Cosine Similarity Calculation: The dot product was calculated between Vector A: Representing the user profile, containing tags from the games played by the user, and Vector B: Representing tags from a specific game being compared.
5) Magnitude Calculation: The magnitudes of both vectors were calculated by summing the squared values of their components.
6) Final Cosine Similarity: The cosine similarity value was computed by dividing the dot product by the square root of the magnitudes of both vectors.
7) Sorting and Displaying Results: The system iterated through steps 4 to 6 for all games in the dataset, sorted the results based on the cosine similarity values, and displayed the top 10 games with the highest similarity scores.

G. Testing and Debuging

Once development was completed, the application underwent comprehensive testing to ensure its functionality met expectations. Testing focused on evaluating functionality, reliability, performance, and security. Any identified bugs were addressed through debugging processes to ensure smooth application performance.

H. Evaluation using Technology Acceptance Model (TAM)

The developed application was distributed to research participants, who evaluated it using a **survey** based on the Technology Acceptance Model (TAM). The survey included questions addressing two key factors, which are Perceived Ease of Use, and Perceived Usefulness. The TAM framework, proposed by **Fred Davis** and **Richard Bagozzi** [20], was used to measure technology acceptance. The collected results were analyzed to determine the application's acceptance level among users

IV. Results and Discussions

I. Application Interface

Figure 2 shows the application's login interface, where users log in using Steam to extract their gaming data.

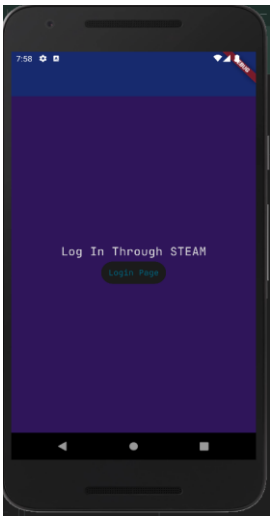

Fig. 2. Login Page Interface

Figure 3 shows the main page of the application, where users can choose to request game recommendations or view their profiles.

Fig. 3. Home Page Interface

Figure 4 illustrates the recommendation page, where users can request recommendations based on games they own or have played in the past two weeks.

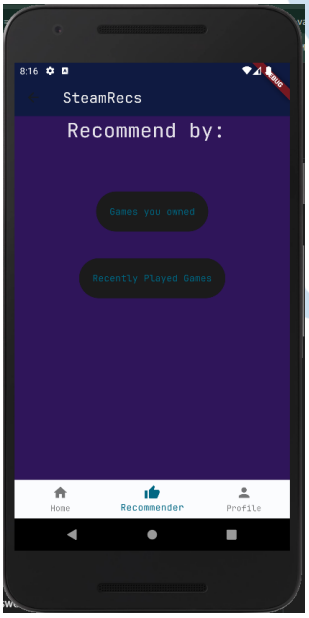

Fig. 4. Recommender Page Interface

Figure 5 displays the results page, where the system shows recommended games along with their **cosine** similarity scores.

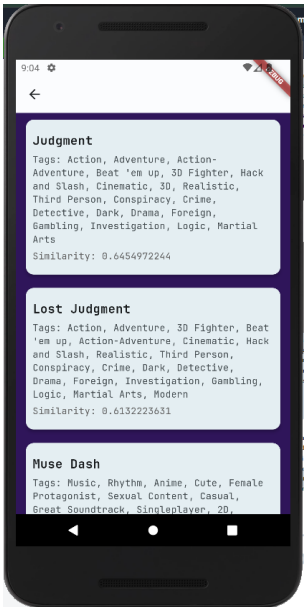

Fig. 5. Recommendation Result Interface

J. *Implementation of Steam Web API*

The Steam Web API was implemented to support the login system and retrieve user data. The login system utilized the WebView plugin, which redirected users to the Steam login page via OpenID. Once the Steam account was linked, users were directed to the main application interface.

The Steam Web API was also used to extract critical user data, including UserID, the user's Steam Library, and general game data from Steam. This data was obtained through the GET function, returning a JSON file.

K. *Data Acquisition*

Before providing game recommendations, necessary data was collected:

1) App ID Data: Retrieved via the Steamworks Web API for detailed game information.
2) Alternative Data Source: Due to limitations of the Steam Web API, the SteamSpy API was used to supplement the required game data.
3) Game Dataset: A JSON file containing data for 55,000 Steam games sourced from Kaggle was utilized to overcome API rate limitations.

L. *Content-Based Filtering Implementation*

After gathering the necessary data, the recommendation process commenced. The first step was pre-processing, which involved filtering games based on playtime, removing symbols and special characters, converting tags to lowercase, and preventing duplicates by converting the data into **Sets**. This ensured uniformity and efficiency in subsequent processes.

Next, the tags were vectorized into binary vectors, enabling the system to perform similarity calculations. The Cosine Similarity metric was then used to measure the similarity between the User Profile Vector (A**)**, representing tags derived from the user's game library, and the Game Vector (B), which contained tags for each game in the dataset.

After calculating the cosine similarity for all games in the dataset, the system sorted the results based on the highest similarity values. The top 10 games with the highest cosine similarity scores were then displayed to the user as recommendations.

The system was tested using the researcher's Steam account, where the highest cosine similarity score of 0.6454972244 was obtained for the game *"Judgment"*. Tags associated with *Judgment*, such as *Action, Adventure, Beat'em Up, Hack and Slash*, closely aligned with the tags in the user profile, demonstrating the accuracy of the recommendation system.

M. Application Evaluation

The application evaluation was conducted using the Technology Acceptance Model (TAM)**.** Upon completion of the application, it was distributed to participants along with a survey created using Google Forms. The survey included questions designed in accordance with TAM principles (Table 1).

TABLE I. SURVEY QUESTIONS BASED ON TECHNOLOGY ACCEPTANCE MODEL (TAM)

No.	Question	Response Scale
1	Name (Initials allowed)	Text input
2	Is the application useful to you?	1 (Not Useful) - 5 (Very Useful)
3	Does the application simplify finding games you want to play?	1 (Strongly Disagree) - 5 (Strongly Agree)
4	Can you easily obtain useful information from the application?	1 (Strongly Disagree) - 5 (Strongly Agree)
5	Does the application help you better understand recommendation systems?	1 (Strongly Disagree) - 5 (Strongly Agree)
6	Does the app interface facilitate your interaction with the recommendation system?	1 (Strongly Disagree) - 5 (Strongly Agree)
7	Does interacting with the app make using recommendation systems easier?	1 (Strongly Disagree) - 5 (Strongly Agree)

Questions 2 to 4 focused on the perceived usefulness of the application, assessing how beneficial the application was for users. Meanwhile, questions 5 to 7 measured the perceived ease of use, evaluating how easy it was for users to interact with and utilize the application. The survey received responses from 30 participants, adhering to the sampling method suggested by Sugiyono [22].

TABLE II. PERCEIVED USEFULNESS

Perceived Usefulness	1	2	3	4	5
Question 2	0	0	3	15	12
Question 3	0	0	8	12	10
Question 4	0	1	6	14	9

The results of the survey are presented in Table II for the "perceived usefulness" aspect and Table III for the "perceived ease of use" aspect. These tables summarize participants' responses on a scale of 1 (Strongly Disagree/Not Useful) to 5 (Strongly Agree/Very Useful).

TABLE III. PERCEIVED EASE OF USE

Perceived Usefulness	1	2	3	4	5
Question 5	0	0	5	11	14
Question 6	0	0	5	12	13
Question 7	0	0	3	13	14

The survey results provided insights into the participants' opinions on the application. To calculate the percentage of perceived usefulness and perceived ease of use, the following formula was applied:

$$\frac{\sum_{i=1}^{p}(x_i \times y_i)}{(p \times s \times j)} \times 100\% \quad (2)$$

Explanation:

- i: Scale value of the question.
- p: Total number of scale points in the question.
- x_i: Specific scale value for a given question.
- y_i: Total number of responses corresponding to the scale value x_i
- s: Total number of survey participants (sample size).
- j: Total number of questions related to the evaluation factor being measured.

Based on the results, Formula 3 was used to calculate the percentage for perceived usefulness, while Formula 4 was applied to compute the percentage for perceived ease of use. The evaluation demonstrated the application's effectiveness and usability from the perspective of the participants.

$$\text{perceived usefulness} = \frac{(0\times1)+(1\times2)+(17\times3)+(41\times4)+(31\times5)}{5\times30\times3} \times 100\% = 82.6\% \quad (3)$$

$$\text{perceived ease of use} = \frac{(0\times1)+(1\times2)+(17\times3)+(41\times4)+(31\times5)}{5\times30\times3} \times 100\% = 86.2\% \quad (4).$$

V. CONSLUSION

The study concluded that the design and development of the application were successfully completed, with the application running smoothly on nearly all devices used during testing. The recommendation system, which employed the Content-Based Filtering method, delivered satisfactory results, achieving the highest Cosine Similarity score of 0.6454972244. Additionally, the evaluation using

the Technology Acceptance Model (TAM) demonstrated positive reception from users, with a perceived usefulness score of 82.6% and a perceived ease of use score of 86.2%**,** indicating that the majority of participants found the application effective and user-friendly.

For future research, it is recommended to explore alternative methods to access dynamic data without being constrained by API call limitations, enabling the use of more accurate and comprehensive data from Steam services. Additionally, the recommendation process could be expedited by optimizing the algorithm further or adopting alternative methods that can deliver equivalent or superior results with reduced processing time.

REFERENCES

[1] R. Ramadan and Y. Widyani, "Game development life cycle guidelines," in *2013 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*. IEEE, 2013, pp. 95–100.

[2] J. Clement, "Video game market revenue worldwide from 2018 to 2028," Dec. 2023. [Online]. Available: https://www.statista.com/statistics/1344668/revenue-video-game-worldwide/

[3] ——, "Leading devices used to play games worldwide 2022," Jan. 2023. [Online]. Available: https://www.statista.com/statistics/533047/leading-devices-play-games/

[4] G. Sergey, "All the data and stats about steam games," 2016. [Online]. Available: http://steamspy.com/

[5] A. Chernev, U. Bockenholt, and J. Goodman, "Choice overload: A conceptual review and meta-analysis," *Journal of Consumer Psychology*, vol. 25, no. 2, pp. 333–358, 2015.

[6] D. Wang, M. Moh, and T.-S. Moh, "Using deep learning and steam user data for better video game recommendations: Proceedings of the 2020 ACM southeast conference," Apr. 2020. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/3374135.3385283

[7] G. Cheuque, I. P. U. Catolica Santiago, J. Guzmán, D. Parra, G. Tech, and O. Metrics, "Recommender systems for online video game platforms: The case of steam: Companion proceedings of the 2019 world wide web conference," May 2019. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/3308560.3316457

[8] S. Bayram, "Game recommendation system for steam platform," Jan. 1970. [Online]. Available: https://openaccess.mef.edu.tr/xmlui/handle/20.500.11779/1721

[9] S. Reddy, S. Nalluri, S. Kunisetti, S. Ashok, and B. Venkatesh, "Content-based movie recommendation system using genre correlation," Jan. 1970. [Online]. Available: https://link.springer.com/chapter/10.1007/978-981-13-1927-3_42

[10] T. Badriyah, S. Azvy, W. Yuwono, and I. Syarif, "Recommendation system for property search using content-based filtering method," Mar. 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8350801

[11] R. Glauber and A. Loula, "Collaborative filtering vs. content-based filtering: Differences and similarities," Dec. 2019. [Online]. Available: https://arxiv.org/abs/1912.08932

[12] A. Tashildar, N. Shah, R. Gala, T. Giri, and P. Chavhan, "Application development using flutter," *International Research Journal of Modernization in Engineering Technology and Science*, vol. 2, no. 8, pp. 1262–1266, 2020.

[13] W. Wu, "React native vs flutter, cross-platforms mobile application frameworks," 2018.

[14] A. Hassan, "Java and dart programming languages: Conceptual comparison," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 17, no. 2, pp. 845–849, 2020.

[15] G. I. Arb and K. Al-Majdi, "A freights status management system based on dart and flutter programming language," in *Journal of Physics: Conference Series*, vol. 1530, no. 1. IOP Publishing, 2020, p. 012020.

[16] M. Al-Shamri, B. Amiri, A. Biswas, W. Carrer-Neto, F. Colace, S. Choi, M. Everett, L. Freeman, O. Kwon, Y. Li, et al., "Content-based filtering for recommendation systems using multiattribute networks," Aug. 2017. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0957417417305468

[17] R. van Meteren and M. van Someren, "Using content-based filtering for ... - ics-forth." [Online]. Available: http://users.ics.forth.gr/~potamias/mlnia/paper_6.pdf

[18] A. R. Lahitani, A. E. Permanasari, and N. A. Setiawan, "Cosine similarity to determine similarity measure: Study case in online essay assessment," Sep. 2016. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7577578/

[19] D. Gunawan, C. A. Sembiring, and M. A. Budiman, "The implementation of cosine similarity to calculate text relevance between two documents," Mar. 2018. [Online]. Available: https://iopscience.iop.org/article/10.1088/1742-6596/978/1/012120/meta

[20] M. Chuttur, "Overview of the technology acceptance model: Origins, developments and future directions," Jun. 2013. [Online]. Available: https://aisel.aisnet.org/sprouts_all/290

[21] M. Masrom, "Technology acceptance model and E-learning," May 2007. [Online]. Available: https://www.researchgate.net/publication/228851659_Technology_acceptance_model_and_E-learning

[22] Sugiyono, *Metode penelitian pendidikan: pendekatan kuantitatif, kualitatif, dan RD*, revised edition ed. Bandung: Alfabeta, 2016.