An Evaluation of Testcase Recommendation Systems through Feedback Model

Minh Nguyen-Doan-Nhat[1,2], Khoa Vu-Dang[1,2], Tien Vu-Van[1,2], Huy Tran[1,2], Thanh-Van Le[1,2], Hoang-Anh Pham[1,2,*], and Nguyen Huynh-Tuong[3]

[1] Ho Chi Minh City University of Technology (HCMUT), Vietnam
[2] Vietnam National University Ho Chi Minh City (VNU-HCM), Vietnam
[3] Industrial University of Ho Chi Minh City (IUH), Vietnam
Email: `minh.nguyendoannhat@hcmut.edu.vn`, `khoa.vukhoa3004@hcmut.edu.vn`, `vvtien@hcmut.edu.vn`, `tranhuy@hcmut.edu.vn`, `ltvan@hcmut.edu.vn`, `anhpham@hcmut.edu.vn`, `htnguyen@iuh.edu.vn`
[*] Corresponding author

Abstract. This study proposes integrating a Feedback Model into the Testcase Recommendation System (TRS) to support learners in improving their programming assignments. This study also suggests two types of feedback: direct feedback, represented by numeric ratings, and indirect feedback, conveyed through textual reviews. Methodologies are designed to evaluate the Feedback Model-integrated system deployed in real-world settings, followed by learner data collection for a comprehensive evaluation analysis. The research findings highlight the suitability of direct feedback for measuring learner satisfaction and the efficacy of indirect feedback in identifying system improvements.

Keywords: Testcase Recommendation Systems · Feedback Model

1 Introduction

A Learning Management System (LMS) in higher education connects teachers and learners, offering management tools and support for managers, lecturers, and learners themselves [1]. It aims to track learner progress and performance through various modules like lecture videos and social media [22], [14]. In data science research, LMS features automate evaluation, analyze learning styles, and gauge learner satisfaction through data collection tools [6]. Assignments are common for assessing learner knowledge and research skills, typically involving understanding subject lectures and self-researching for solutions. Programming exercises are often evaluated by comparing code outputs against expected results, known as testcases, with higher scores for more correct testcases. Assignments align with subject learning outcomes, requiring learners to achieve certain scores. Due to the complexity of assignment criteria and the large number of learners, an automatic grading system (AGS) is used to assess submitted code efficiently, reducing grading time and ensuring quality assessment.

A recommendation system (RS) acts as a digital guide, suggesting items to users faced with numerous choices, widely used in e-commerce and digital platforms like Netflix [9], Amazon [11], Google, and YouTube [12]. RS predicts user ratings for items and generates tailored recommendations by analyzing their interactions with previous items. While research in e-learning recommender systems is relatively new, recent studies show promise. The authors in [23] propose an ontology-based framework for e-learning RS, while the authors in [19] advocate for ontology in these systems. A model is developed to predict learning styles in e-learning [8], and the authors in [20] suggest personalized RS combining ontology and collaborative filtering. A method of merging collaborative and content-based filtering is also proposed in e-learning RS [4].

Recommendation systems generally employ three methodologies: Content-based filtering (CBF), Collaborative filtering (CF), and a Hybrid approach, which merges CBF and CF for improved accuracy and relevance. CBF suggests items by analyzing user and item attributes, focusing on similarities between consumed and suggested items. CF assumes users with similar behaviors share preferences, using similarity measures to recommend items or users [12]. Memory-based CF identifies relevant neighbors for recommendations [10], while model-based CF learns from the user-item rating matrix, commonly using Matrix Factorization (MF) for scalability and accuracy [5]. A hybrid approach combines CBF and CF for enhanced predictive accuracy. In the model-based approach, parameters are trained using the user-item matrix to generate recommendations [5].

Previous studies have specifically explored recommendation systems applications in the context of software testing. The authors in [7][18] have examined the test scripts utilized by automation teams and proposed generating new testcases based on the structural similarity of the source code. Similarly, in [13], the author developed a recommender system aimed at identifying an optimal set of tests to accompany a code change. Moreover, in [3], the authors implemented an item-based collaborative filtering recommender system, leveraging user interaction data and application modification history to prioritize testcase development. These endeavors predominantly revolve around suggesting testcases akin to those already existing for software testing purposes.

Inspired by LMS and RS, the Testcase Recommendation System (TRS) [21] aims to assist learners in testing their programming assignments. TRS suggests testcases based on learners' current abilities, focusing on areas where they make mistakes. Testcases are pre-prepared by teachers and selected by the system to match learners' abilities, helping them pass with effort. TRS emphasizes providing learners with a manageable set of testcases to prevent being overwhelmed by numerous errors in assignment submissions.

Feedback is integral to recommendation systems, providing insights that guide the selection of relevant items for users. Typically conveyed through ratings, feedback offers users a standardized way to express their opinions about various entities [2]. These ratings serve as crucial indicators of item relevance, with higher scores indicating greater relevance within the pool of candidate items.

Memory-based collaborative filtering methods, such as those discussed in [15], commonly leverage user ratings to personalize recommendations, utilizing user or item neighborhoods to predict user preferences for new items.

While ratings are fundamental for recommendations, they also serve as a tool to evaluate the system's performance [17]. Assessing user feedback can be challenging, especially when considering both required and optional feedback forms. In scenarios where users rate items, their ratings can serve as a measure of utility, reflecting the value derived from recommendations. However, capturing and interpreting user feedback presents additional complexities, particularly in optional forms such as textual reviews. Integrating textual reviews with numeric ratings enables a deeper understanding of user preferences, though processing these reviews requires sophisticated text analysis techniques [16].

In addition to numeric ratings, user-generated reviews have emerged as valuable sources of feedback [24]. Integrating textual reviews with numeric ratings provides richer insights into user intentions and preferences, contributing to the evaluation of recommendation systems. This study aims to explore the significance of textual reviews in conjunction with numeric ratings for evaluating recommender systems, particularly focusing on processing implicit ratings embedded within textual reviews.

The main contributions of this research are summarized as follows. Firstly, it enhances the existing TRS approach by integrating a feedback model. Secondly, it implements and deploys the enhanced system, collecting learners's feedback. Lastly, it evaluates the effectiveness of our TRS through analytical research, examining both required numeric ratings and optional textual reviews from learners.

The subsequent sections of this paper follow the following structure: Section II provides a detailed explanation of our proposed approach, which includes the TRS with Feedback Model, the design of feedback content, and the design of evaluation content. Section III presents the implementation and results of the proposed approach, delving into deploying the system and collecting real-world data from learners, followed by insights gained from the findings. Finally, Section IV concludes with concluding remarks, summarizing our discoveries and suggesting potential avenues for future research.

2 Proposed Approach

2.1 The TRS with Feedback Model

The proposed approach aims to integrate the previous TRS with a Feedback Model to gather learner satisfaction. In the existing TRS [21], learners begin engaging with the system by solving sample testcases. After successfully completing these sample testcases, learners are provided with recommended testcases. The ***Recommendation System*** component is responsible for suggesting testcases that align with the learner's abilities. The recommendation process involves learners receiving a set of testcases, solving them until all are completed correctly, and then receiving a new set of testcases, repeating the recommendation cycle. The recommendation process stops when the maximum number of daily recommendations is reached, and learners continue to receive recommendations the next day.

This study proposes integrating the ***Feedback Model*** into the existing TRS. Fig. 1 illustrates the proposed system with the ***Feedback Model*** component. Collected feedback consists of learners' opinions and comments regarding recommended testcases. Therefore, feedback is solicited after learners have completed a set of testcases. To prevent excessive distraction for learners, the proposal suggests that feedback appears after all testcases in a recommendation set have been completed correctly (after the ***Check*** block). As a result, feedback regarding partially completed submissions will not be recorded. Instead, feedback information will be related to the entire testcases rather than individual testcase.

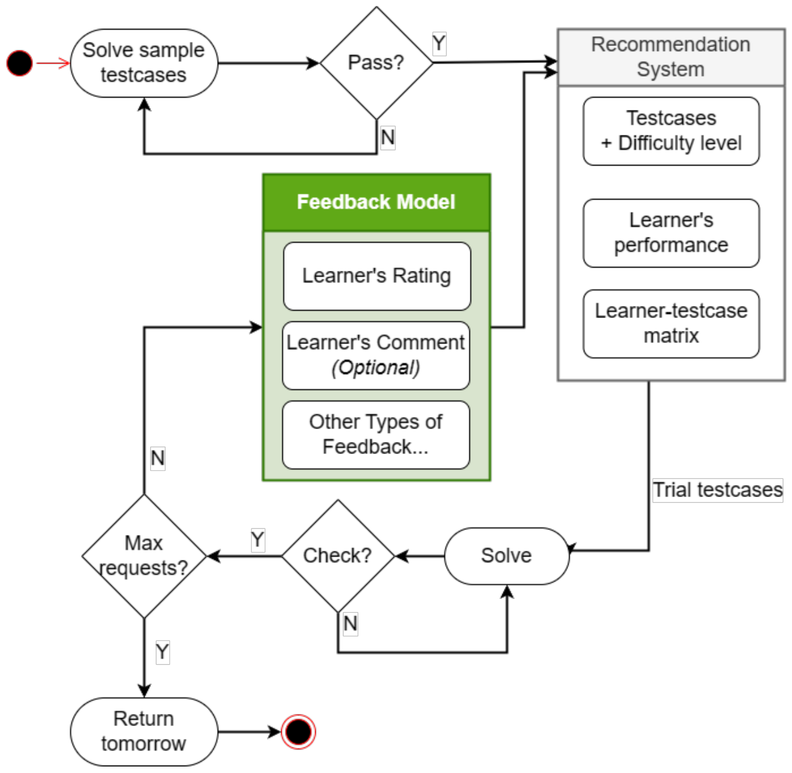


Fig. 1: Overview about the TRS with Feedback Model

Feedback is collected through a feedback form containing several questions for learners. Feedback forms need to have a certain level of mandatory requirement for learners to ensure the collection of opinions. The proposed form will be displayed before the next set of recommended testcases is sent to learners. Learners must complete the feedback form before receiving a new set of recommendations.

2.2 Designing Feedback Content

The ***Feedback Model*** can include various types of feedback. In this study, the focus is on analyzing two types of feedback: ***Rating*** and ***Comment***.

Rating is a form of feedback representing learners' satisfaction with the recommended testcases. ***Rating*** is characterized as direct feedback because it directly indicates satisfaction. Additionally, ***Rating*** requires every learner to respond before receiving a new set of testcases. This is necessary to capture user satisfaction. Therefore, ***Rating*** also has a mandatory nature. This study considers ***Rating*** as direct feedback.

On the other hand, ***Comment*** is a textual review and optional to learners. ***Comment*** can use to evaluate the reliability of the ***Rating***. ***Comment*** is optional to demonstrate the learner's initiative. Learners who respond to ***Comment*** express their opinions about the system. ***Comment*** can be collected directly or indirectly. In this study, the proposal involves using an indirect method and asking users open-ended questions. Here, ***Comment*** is considered indirect feedback.

2.3 Designing Evaluation Content

This section proposes and analyzes factors to evaluate systems integrated with the ***Feedback Model***. The first factor considered is the *Complexity of the recommendation process*. Complexity is estimated by the number of steps required to generate a recommendation set. Adding the ***Feedback Model*** may increase the number of steps compared to the previous system. To quantify this impact, it is necessary to assess the severity of the steps taken by users. This can be achieved by collecting feedback from users about the ease or difficulty of the steps to evaluate the impact of feedback on user experience.

Evaluating system usage is the next factor of interest. The system needs to be designed to log information each time it makes recommendations and the user responds to them. Additionally, responses to feedback should also be recorded. Therefore, system usage can be determined by the average number of times users engage with the system, completing tasks, and responding to feedback forms.

User feedback responsiveness also needs to be considered. This is an important indicator for evaluating user engagement when providing feedback. It can be measured by calculating the ratio of the number of learners who responded to optional questions to the total number of form submissions.

Finally, in evaluating the *user satisfaction level with the system*, consideration needs to be given to the percentage of users rating satisfaction on a specific scale compared to the total number of form submissions. This helps us understand the level of acceptance and satisfaction of users with the system.

This study evaluates the TRS based on direct and indirect feedback. Direct feedback may utilize various scales to indicate user satisfaction and effectiveness. We suggest employing a five-level Likert scale for the Rating component to standardize the evaluation process.

Regarding indirect feedback, a method of mapping values from ***Comment*** to the values of the ***Rating*** is proposed. Table 1 summarizes the mapping from indirect feedback to some proposed properties. Four factors and their combinations are considered: *Praise, No Praise, Criticism, No Criticism.* When users express neither praise nor criticism about the system, this factor is mapped to the corresponding level of *Neutral.* When users only provide criticism without praising the system, it is mapped to *Complaining.* When users only praise without criticism, it is mapped to *Satisfied.* Finally, when users offer both praise and criticism, they are providing *Suggestive* feedback to the system. Then, each property of *Suggestive*, *Complaining*, *Satisfied*, *Neutral* is also mapped to specific corresponding values in direct feedback.

Table 1: Mapping of indirect feedback to properties

	Praise	**No Praise**
Criticism	Suggestive	Complaining
No Criticism	Satisfied	Neutral

3 Implementation and Results

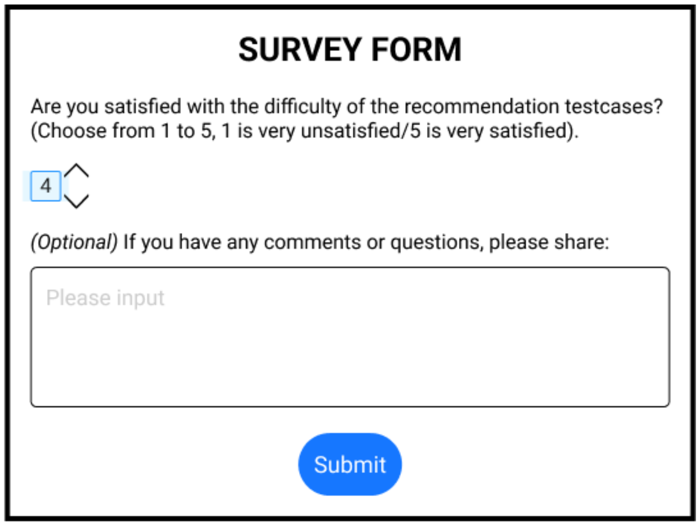
3.1 Implementation

Testcase Recommendation System (TRS) has been deployed for use in the Programming Fundamentals (PF) course. The TRS allows learners to submit their code and receive graded feedback and recommended testcases to improve their code further. Additionally, the system integrates a feedback model through surveys to gather learner insights.

Upon completion of the recommended testcases, learners are asked to complete an evaluation form to receive new recommended testcases, as shown in Fig. 2. The survey form prompts learners to rate their satisfaction with the recommended testcases set and offer comments for the system. These two questions correspond respectively to ***Rating*** and ***Comment***.

Regarding ***Rating***, a typical five-level Likert scale ranges from 1 to 5, where higher values indicate greater satisfaction. As for ***Comment***, feedback is provided to 4 instructors of programming courses, who categorize comments into one of four properties: *Suggestive*, *Complaining*, *Satisfied*, *Neutral.* The factor with the highest votes is used as the final factor. In cases where a comment has an equal number of selections for each factor (e.g., 2 instructors choose Suggestive and 2 choose Complaining, or 4 instructors choose 4 different factors), these comments are further discussed.

After obtaining corresponding factors for indirect feedback, the following mapping is proposed for each factor to a value in the Likert scale: *Complaining* is mapped to 1 and 2, *Neutral* is mapped to 3, *Suggestive* is mapped to 4, and *Satisfied* is mapped to 5.



SURVEY FORM

Are you satisfied with the difficulty of the recommendation testcases?
(Choose from 1 to 5, 1 is very unsatisfied/5 is very satisfied).

4

(Optional) If you have any comments or questions, please share:

Please input

Submit

Fig. 2: Survey Form

3.2 Results

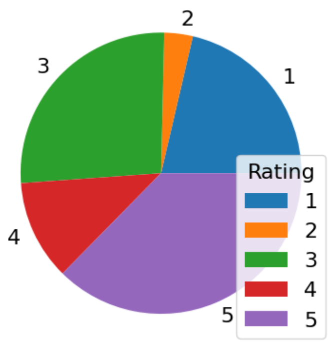
Regarding **evaluation for TRS with Feedback Model**, the *System Usage Level* is recorded in Table 2 that shows the statistical results after implementing the assignment. The statistics offer valuable insights into the engagement and performance of learners in two distinct PF courses. Among 1576 learners, 978 (a significant proportion of 62%) actively participated by submitting code over five days. This resulted in a total of 3887 submissions, accompanied by 2115 recommendations and 487 survey forms, indicating an average of approximately 3.97 submissions per learner. Notably, the initial recommendation does not mandate users to complete a survey; the system only starts collecting surveys from the second recommendation onward. Additionally, to avoid disrupting learners during their practice, the system randomly determines whether they need to complete a survey, resulting in fewer survey forms than recommendations.

Table 2: Statistical results after assignment implementation

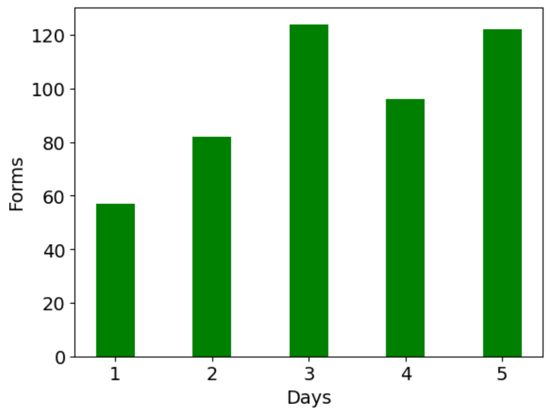
Course	Programming Fundamentals
Total learners	1576
Learners used TRS	978
Days with submissions	5
Submissions	3887
Recommendations	2115
Survey Forms	487
Average submissions per learner	3.97

Fig. 3a shows a distribution across the ratings from 1 to 5. There are 104 ratings with a score of 1, 16 ratings with a score of 2, 129 ratings with a score of 3, 56 ratings with a score of 4, and 182 ratings with a score of 5. The number of 5 score ratings is the highest among 5 values. Furthermore, out of the total 487 ratings, there are a total of 49% (238) ratings that are higher than 3. This result indicates that many learners have provided positive feedback on the course, with high satisfaction.

Meanwhile, Fig. 3b shows the number of survey forms collected over five consecutive days, from day 1 to 5. There are 57 survey forms on day 1, 82 on day 2, 124 on day 3, 96 on day 4, and 122 on day 5. Out of the 487 survey forms collected, a subset of 122 forms contained responses to the optional question. This result indicates that approximately 25% of the survey participants are willing to provide optional comments.



(a) Rating scores



(b) Survey forms by day

Fig. 3: Distribution of rating scores and collected survey forms

Regarding Evaluation for feedback contents, we analyze both direct and indirect feedback to provide insights into the usage of each type of feedback in the TRS. The collected feedback consists of 487 responses for the **Rating** aspect and 122 responses for the **Comment** aspect. Table 3 records the distribution of 487 ratings by each property: **Complaining**, **Neutral**, **Suggestive** and **Satisfied**. The majority of rating scores fall into the **Satisfied** category, which includes rating scores of 5, accounting for 37.37%. This result indicates a high level of satisfaction among the TRS users.

On the other hand, the **Complaining** category constitutes 24.64% of the ratings. This statistic suggests that users frequently express their dissatisfaction and frustration with the system. It is crucial to carefully consider and address the issues raised in this category to improve user experience and system performance. Furthermore, the **Neutral** category represents 26.49% of the ratings, indicating a moderate level of satisfaction or indifference among the students. This category highlights the need for further investigation into the factors contributing to user neutrality and potential areas for improvement. Additionally, the **Suggestive** property accounts for 11.50% of the ratings, indicating that users actively provide constructive feedback and recommendations to enhance the TRS.

Table 3: Distribution of ***Rating*** by each property

Comment property	**Rating score**	**No. ratings**	**Percentage**
Complaining	1+2	120	24.64%
Neutral	3	129	26.49%
Suggestive	4	56	11.50%
Satisfied	5	182	37.37%

Table 4 records the proportion of indirect feedback categorized into four properties: Suggestive, Satisfied, Neutral, and Complaining. To evaluate them, the semantic content was analyzed, and the feedback was classified. The columns *Full - No. comments* and *Full - Percentage* record each property's number and percentage for the entire 122 feedback. During the selection process by four instructors, 16 conflicts occurred when the selected properties had equal votes. These conflicting comments were removed, and the data was then recorded in the columns *Remove conflicts - No. comments* and *Remove conflicts - Percentage.* As can be seen, removing conflicts helped the properties *Satisfied*, *Suggestive*, and *Neutral* to have values closer to the corresponding properties in direct feedback in Table 3. However, the *Complaining* property increased the gap compared to direct feedback.

The investigation of the 16 conflicting comments revealed that 11 (68.75%) were written as *"ok"*. Instructors assessed *"ok"* with two votes for *Satisfied* and two votes for *Neutral*, leading to conflict. Through discussions with instructors, it was found that short comments like *"ok"* or *"good"* couldn't determine the property due to individual perceptions of natural language. Conversely, longer comments were received, such as *"I hope the interface of this website can be similar to the one in the LMS with a show difference section so that we can easily identify mistakes"*. This comment received all four votes as *Suggestive.*

Table 4: Distribution of ***Comment*** by each property

Comment property	**Full No. comments**	**Full Percentage**	**Remove conflicts No. comments**	**Remove conflicts Percentage**
Complaining	27	30.33%	35	33.02%
Neutral	20	16.19%	20	18.87%
Suggestive	6	4.92%	6	5.66%
Satisfied	59	48.36%	45	42.45%

The above analytical results show that direct feedback quickly reflects learners' satisfaction levels. Meanwhile, indirect feedback requires additional processing by instructors and is more challenging to evaluate due to varying perceptions of natural language. When indirect feedback is relatively lengthy, instructors find assessing and providing more meaningful information to the system easier. Thus, surveying through ***Rating*** is advisable for learner satisfaction. On the other hand, when considering areas for system improvement, ***Comment*** can be used to gather helpful feedback from learners.

4 Conclusion

This paper presents an integration of a feedback model into the Testcase Recommendation System, aiming to assist learners in testing their programming assignments. A survey was conducted to collect user satisfaction through numeric ratings for direct feedback and textual reviews for indirect feedback. Additionally, the evaluation of the integrated feedback model within the testcase recommendation system was designed. Subsequently, the effectiveness of both direct and indirect feedback methods was assessed through a comprehensive evaluation of the TRS. The study findings suggest that direct feedback is suitable for capturing learner satisfaction, while indirect feedback is apt for identifying areas for system improvement.

In future works, several avenues can be explored to enhance the TRS. Firstly, additional questions can be proposed within the survey to serve as evaluation metrics for the TRS. Secondly, there is a need to investigate methods for determining whether a textual review is considered valuable for observation. Developing criteria to identify high-quality textual reviews could improve the accuracy of feedback analysis. Lastly, further studies can focus on applying multiple models to evaluate the TRS based on the feedback model proposed in this research. With feedback information collected from learners, comparisons of the effectiveness of various models can be made, leading to enhanced recommendation systems in the future.

Acknowledgment

This research is funded by Vietnam National University Ho Chi Minh City (VNU-HCM) under grant number DS2022-20-07. The authors also acknowledge Ho Chi Minh City University of Technology (HCMUT), VNU-HCM for supporting this study.

References

1. Adzharuddin, N.: Learning Management System (LMS) among University Students: Does It Work? International Journal of e-Education, e-Business, e-Management and e-Learning **3**(233), 248–252 (2013). https://doi.org/10.7763/IJEEEE.2013.V3.233
2. Ahmed, M., et al.: Rating-Based Recommender System Based on Textual Reviews Using IoT Smart Devices. Mobile Information Systems **2022**, 1–18 (2022). https://doi.org/10.1155/2022/2854741
3. Azizi, M., Do, H.: A Collaborative Filtering Recommender System for Test Case Prioritization in Web Applications. In: Proceedings of the 33rd Annual ACM Symposium on Applied Computing. p. 1560–1567. ACM (2018). https://doi.org/10.1145/3167132.3167299
4. Benhamdi, S., Babouri, A., Chiky, R.: Personalized recommender system for e-Learning environment. Education and Information Technologies **22**(4), 1455–1477 (2017). https://doi.org//10.1007/s10639-016-9504-y
5. Bennett, J., Lanning, S.: The Netflix Prize. Proceedings of KDD cup and workshop **2007**, 35 (2007)
6. Graf, S., Kinshuk: Analysing the Behaviour of Students in Learning Management Systems with Respect to Learning Styles, pp. 53–73. Springer Berlin Heidelberg (2008). https://doi.org/10.1007/978-3-540-76361_3
7. John, S.B., Gaur, D., Siddiqui, A.: Test case Recommendation for regression with Named Entity Recognition for test step prediction. In: Proceedings of 2021 4th Biennial International Conference on Nascent Technologies in Engineering (ICNTE). pp. 1–6 (2021). https://doi.org/10.1109/ICNTE51185.2021.9487774
8. Kolekar, S.V., Sanjeevi, S.G., Bormane, D.S.: Learning style recognition using Artificial Neural Network for adaptive user interface in e-learning. In: Proceedings of 2010 IEEE International Conference on Computational Intelligence and Computing Research. pp. 1–5 (2010). https://doi.org/10.1109/ICCIC.2010.5705768
9. Koren, Y.: Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. p. 426–434. Association for Computing Machinery (2008). https://doi.org/10.1145/1401890.1401944
10. Lee, J., Sun, M., Lebanon, G.: A Comparative Study of Collaborative Filtering Algorithms. CoRR **abs/1205.3193** (2012)
11. Linden, G., Smith, B., York, J.: Amazon.com recommendations: item-to-item collaborative filtering. IEEE Internet Computing **7**(1), 76–80 (2003). https://doi.org/10.1109/MIC.2003.1167344

12. Mehta, R., Rana, K.: A review on matrix factorization techniques in recommender systems. In: Proceedings of 2nd International Conference on Communication Systems, Computing and IT Applications (CSCITA). pp. 269–274 (2017). https://doi.org/10.1109/CSCITA.2017.8066567
13. Nandan, S.: Test case recommendation system. International Journal of Advance Research, Ideas and Innovations in Technology (IJARIIT) **5**(2), 841–845 (2019)
14. Otchie, W., Pedaste, M.: Social Media as a Learning Management System: Is it a Tool for Achieving the Goal of "Education for All"? US-China Education Review A **9**, 79–90 (2019). https://doi.org/10.17265/2161-623X/2019.02.003
15. Roy, D., Dutta, M.: A systematic review and research perspective on recommender systems. Journal of Big Data **9**, 1–36 (2022). https://doi.org/10.1186/s40537-022-00592-5
16. Shalom, O.S., Uziel, G., Kantor, A.: A generative model for review-based recommendations. In: Proceedings of the 13th ACM Conference on Recommender Systems. p. 353–357. ACM (2019). https://doi.org/10.1145/3298689.3347061
17. Shani, G., Gunawardana, A.: Evaluating Recommendation Systems, pp. 257–297. Springer US (2011). https://doi.org/10.1007/978-0-387-85820-3_8
18. Shimmi, S., Rahimi, M.: Leveraging Code-Test Co-evolution Patterns for Automated Test Case Recommendation. In: Proceedings of 2022 IEEE/ACM International Conference on Automation of Software Test (AST). pp. 65–76 (2022). https://doi.org/10.1145/3524481.3527222
19. Shishehchi, S., Banihashem, S.Y., Zin, N.A.M.: A proposed semantic recommendation system for e-learning: A rule and ontology based e-learning recommendation system. In: Proceedings of 2010 International Symposium on Information Technology. vol. 1, pp. 1–5 (2010). https://doi.org/10.1109/ITSIM.2010.5561329
20. Tarus, J., Zhendong, N., Khadidja, B.: E-Learning Recommender System Based on Collaborative Filtering and Ontology. International Journal of Computer and Information Engineering **11**(2), 256–261 (2017)
21. Vu-Van, T., Tran, H., Le, T.V., Pham, H.A., Huynh-Tuong, N.: An Adaptive Testcase Recommendation System to Engage Students in Learning: A Practice Study in Fundamental Programming Courses. International Journal of Advanced Computer Science and Applications (IJACSA) **14**(6), 1101–1109 (2023). https://doi.org/10.14569/IJACSA.2023.01406118
22. Yildirim, S., Temur, N., Kocaman, A., Goktas, Y.: What makes a good LMS: An analytical approach to assessment of LMSs. Proceedings of the 8th International Conference on Information Technology Based Higher Education and Training pp. 125–130 (2004). https://doi.org/10.1109/ITHET.2004.1358150
23. Yu, Z., et al.: Ontology-Based Semantic Recommendation for Context-Aware E-Learning. In: Indulska, J., et al. (eds.) Ubiquitous Intelligence and Computing. pp. 898–907. Springer Berlin Heidelberg (2007). https://doi.org/10.1007/978-3-540-73549-6_88
24. Zhang, Y., Chen, X.: Explainable Recommendation: A Survey and New Perspectives. Found. Trends Inf. Retr. **14**(1), 1–101 (2020). https://doi.org/10.1561/1500000066