p-ISSN 2407-4489
e-ISSN 2528-7303

Web-Based Computational Thinking Assessment Using K-Nearest Neighbor and Rule-Based Adaptive Treatment

Muhammad Izzuddin Al Fatih[a], Erlangga Erlangga[b], Yuli Sopianti[c]

[a, b, c] *Computer Science Education Study Program, Universitas Pendidikan Indonesia, West Java, Indonesia*
email: [a]alfatih.cipondoh@upi.edu, [b]erlangga@upi.edu, [c]yulisopianti@upi.edu

Abstract

Conventional database-learning assessments generally provide aggregate scores without showing which computational thinking components require further support. This study developed a web-based adaptive assessment system for vocational high school students by integrating K-nearest neighbor classification, adaptive hints, and rule-based treatment. The research applied a development approach using the Smart Learning Environment Establishment Guideline through analysis, design, development, implementation, and evaluation stages. The KNN model was developed using 64 empirical records from students in XI RPL 2 and XI RPL 3. Four weighted computational thinking indicators—decomposition, pattern recognition, abstraction, and algorithmic thinking—were used as classification features. KNN, RandomOverSampler–KNN, and Synthetic Minority Over-sampling Technique (SMOTE)–KNN pipelines were evaluated using group-aware cross-validation. The selected SMOTE–KNN model used distance weighting with k = 9 and achieved 92.19% accuracy, 88.59% balanced accuracy, and 89.75% macro F1-score. The classification result adjusted adaptive hints, while explicit rules controlled indicator priority, difficulty progression, mastery points, and treatment transitions. All 24 black-box scenarios met the expected requirements. The novelty and main contribution of this study lie in integrating indicator-level CT diagnosis, data-driven classification, and transparent rule-based adaptation within a single web-based workflow for vocational database learning.

Keywords: adaptive assessment, black-box testing, computational thinking, K-nearest neighbor, rule-based system.

p-ISSN 2407-4489
e-ISSN 2528-7303

INTRODUCTION

Wing (2006) introduced computational thinking as a problem-solving approach that draws on concepts and practices from computer science and can be applied beyond programming activities. Shute et al. (2017) later reviewed computational thinking from educational perspectives and showed that the concept has been discussed through different definitions, instructional approaches, and assessment frameworks. For the present assessment, computational thinking is operationalized through four indicators: decomposition, pattern recognition, abstraction, and algorithmic thinking. These indicators are used to examine how vocational high school students approach database-related problems.

Database learning in vocational high schools requires students to understand Data Definition Language (DDL), Data Manipulation Language (DML), and Data Control Language (DCL). Learning these topics involves more than remembering SQL commands. Students must recognize relevant data, separate a problem into smaller operations, understand relationships among database elements, and arrange commands in a valid order. Previous studies have reported difficulties in teaching applied query languages when instruction relies heavily on static presentation media or procedural explanation (Putra & Sujatmiko, 2024). Declarative database languages may also be difficult for novice learners because the written command does not directly show the sequence of operations performed by the system (Shin, 2022). This relationship between database tasks and systematic problem solving makes CT relevant to database instruction, as also found in studies connecting CT with programming education (Agbo et al., 2019).

The way student performance is reported presents another issue. A total test score can summarize achievement, but it does not necessarily reveal how that score was formed. Students with equal totals may have different weaknesses; one may struggle with abstraction, while another may have difficulty constructing algorithms. Studies of vocational assessment have noted that conventional instruments do not always provide sufficient information about higher-order reasoning (Badriyah et al., 2018). Research on CT assessment has likewise identified the development of suitable instruments and diagnostic approaches as an area that still requires attention (Ria, 2023). An indicator-level profile is therefore needed so that assessment results can be used not only for grading but also for determining which component should receive earlier support.

Machine learning offers one possible way to process a profile containing several indicator scores. K-nearest neighbor (KNN) assigns a class by comparing a new case with labeled cases that occupy nearby positions in the feature space (Cunningham & Delany, 2021). Because the prediction is based on neighboring records, the relationship between a new profile and the available examples remains relatively understandable. KNN has also been applied to educational data for student-performance prediction and classification (Wiyono et al., 2020; Milala et al., 2023. In the present study, decomposition, pattern-recognition, abstraction, and algorithmic-thinking scores become the input features, while Low, Medium, and High are treated as supervised target categories.

The target categories do not originate from an independent diagnosis produced by the algorithm. They are established through an operational rubric that combines the weighted total score with minimum requirements for individual indicators. This rule is intended to prevent a high score in several components from hiding a substantial weakness in another component. KNN is consequently used to study whether the four-score profile can reproduce the predefined categories and to provide a classification component that can later be retrained when additional empirical records become available.

Classification alone, however, does not produce an adaptive learning process. An adaptive system must use information about a learner to change the assistance, content, or sequence presented during learning (Brusilovsky, 2001). Recent research has also examined opportunities for using artificial intelligence to support student assessment in computing education (dos Santos & Junior, 2024). Data-driven estimation of learners' knowledge states can help determine the support required by different

p-ISSN 2407-4489
e-ISSN 2528-7303

learners, particularly in programming-related environments (Delianidi et al., 2025). Adaptive learning systems may also apply explicit score-update rules and mastery thresholds to determine when learners are ready to progress to subsequent material (Matayoshi et al., 2022). In the system developed in this study, KNN determines the overall CT category and adjusts the amount of information provided in a hint. Treatment progression remains rule based: the rules select the weakest indicator, determine the starting level, update mastery points, and control transitions between levels and indicators. Separating these functions keeps treatment decisions explicit while retaining a data-driven classification component.

The treatment questions use database problems as their learning context. Problem-based learning places a contextual problem at the beginning of learning and requires learners to connect prior knowledge with the information needed to formulate a solution (Savery, 2006). Previous research has also associated PBL activities with the development of students' computational thinking skills (Khatun et al., 2023). Accordingly, DDL, DML, and DCL items are presented through situations involving database design, data operations, query construction, and access control. PBL provides the problem context, whereas KNN and the rule-based controller determine how the system classifies the profile and organizes the treatment.

Recent reviews indicate that several challenges remain in computational thinking assessment. Tan et al. (2024), in a scoping review of machine-learning applications for CT assessment, found that existing studies generally relied on relatively small datasets, were frequently situated in programming or computing-related contexts, and did not comprehensively assess CT competencies. Similarly, Forsström and Bond (2024) reported that CT assessment in primary and secondary education still shows a strong emphasis on programming skills, with less attention given to broader cognitive processes of computational thinking. These findings suggest a continuing need for assessment approaches that represent multiple CT components while also supporting individualized interpretation of assessment results.

Taken together, previous studies support the use of CT assessment, machine-learning classification, adaptive learning, and problem-based activities, but these components are generally investigated in separate methodological contexts. Among the studies reviewed for this research, an approach that integrates four indicator-level CT scores, empirical KNN classification, classification-informed adaptive hints, and explicit rule-based mastery progression within a single web-based workflow for vocational database learning was not identified. The novelty of this study therefore lies in the integration of data-driven CT classification with transparent rule-based adaptive treatment rather than in proposing a new classification algorithm. Accordingly, this study aims to develop, implement, and functionally evaluate a web-based CT assessment system integrating KNN and rule-based adaptive treatment. The investigation covers classification-model evaluation, system implementation, and black-box functional testing; it does not examine whether the system improves students' learning outcomes.

METHOD

Research Design and Procedure

This study employed a Research and Development approach to design, implement, and functionally evaluate a web-based computational thinking assessment system. The development process followed the Smart Learning Environment Establishment Guideline (SLEEG), which integrates the Plan–Do–Check–Act cycle with the Analyze, Design, Development, Implementation, and Evaluation stages (Rosmansyah et al., 2023). The study was conducted at a vocational high school in Bandung during the 2025/2026 academic year.

During the analysis stage, the learning problems, database-learning materials, computational thinking indicators, user characteristics, infrastructure, and system requirements were identified through

p-ISSN 2407-4489
e-ISSN 2528-7303

literature and field studies. The design stage included the preparation of assessment instruments, application architecture, database structure, user interface, KNN classification process, adaptive hints, and rule-based treatment. During development, the classification pipeline and web application were implemented using Python, Streamlit, scikit-learn, imbalanced-learn, joblib, and Supabase. The implementation stage involved integrating the serialized model into the web application and applying the system to the implementation class. Finally, the evaluation stage included classification-model evaluation, black-box testing, and iterative system revision. The development procedure is summarized in Figure 1.

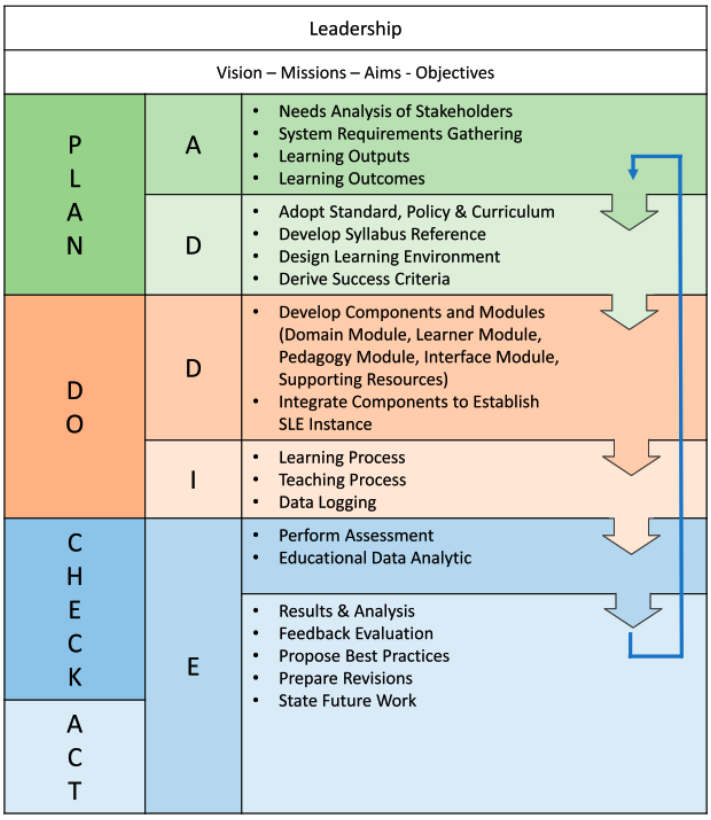


Fig 1. Smart Learning Environment Establishment Guideline framework (Rosmansyah et al., 2023).

Research Data and Instruments

The empirical dataset for model development consisted of 64 eleventh-grade students from XI RPL 2 and XI RPL 3 who completed the computational thinking pretest. The assessment instruments comprised 32 pretest items, 80 adaptive-treatment items, and 32 posttest items covering Data Definition Language, Data Manipulation Language, and Data Control Language. The items represented four computational thinking indicators: decomposition, pattern recognition, abstraction, and algorithmic thinking. Each pretest indicator consisted of eight items with easy, medium, and hard difficulty levels. Correct responses were weighted 1, 2, and 3 according to the respective difficulty levels. Thus, the maximum score for each indicator was 14, while the maximum overall score was 56. Before implementation, the items were examined through validity, reliability, difficulty-level, and discrimination-index analyses.

The weighted score for each indicator was calculated using Equation 1.

$$S_c = \sum_{i=1}^{n_c} w_i x_i$$

p-ISSN 2407-4489
e-ISSN 2528-7303

where (S_c) represents the score for computational thinking indicator (c), (n_c) is the number of items representing the indicator, (w_i) is the difficulty weight of item (i), and (x_i) equals 1 for a correct response and 0 for an incorrect response. The overall score was obtained by summing the scores for decomposition, pattern recognition, abstraction, and algorithmic thinking.

The Low, Medium, and High categories were determined using an operational rubric combining the weighted total score with minimum-indicator constraints. This approach reduced the masking effect in which a high total score could conceal a substantial weakness in one indicator. The operational criteria and empirical data distribution are shown in Table 1.

Table 1. Operational Criteria and Distribution of Computational Thinking Categories

Category	Total-score criterion	Indicator constraint and adjustment	Number of records
Low	Total score ≤ 24	A total score of 25–39 was also classified as Low when at least one indicator score was below 4	7
Medium	Total score 25–39	A total score ≥ 40 was reduced to Medium when at least one indicator score was below 7	13
High	Total score ≥ 40	All four indicator scores had to be at least 7	44
Total	—	—	64

The resulting categories served as supervised target labels for KNN development. KNN was therefore used to learn the relationship between the four-dimensional score profiles and the predefined operational categories rather than to determine the category criteria independently.

The use of 64 empirical profiles should be considered when interpreting the classification results. Although group-aware cross-validation and fold-specific resampling were used to reduce optimistic evaluation, the dataset represents students from only one vocational-school context and contains relatively few Low and Medium profiles. Consequently, the reported performance should be interpreted as an initial evaluation of the classification pipeline rather than evidence of broad generalizability.

A further limitation is that the target categories were generated from the predefined scoring rubric rather than from an independent external diagnosis. Accordingly, model performance measures agreement with these operational categories and does not independently validate the underlying Low, Medium, and High constructs.

KNN Model and Adaptive System Development

The KNN model used decomposition, pattern-recognition, abstraction, and algorithmic-thinking scores as input features. The target categories were encoded as 0 for Low, 1 for Medium, and 2 for High. Before modeling, the data were checked for missing values, duplicate student identifiers, invalid indicator scores, incorrect total scores, and label inconsistencies. Feature scaling was not applied because all four features had the same range of 0–14. Because the empirical dataset was imbalanced, three candidate pipelines were evaluated: KNN without resampling, RandomOverSampler–KNN, and Synthetic Minority Over-sampling Technique (SMOTE)–KNN. SMOTE generates synthetic minority-class observations through interpolation between neighboring minority records (Chawla et al., 2002). In this study, SMOTE used one nearest minority neighbor because the Low category contained only seven empirical records.

p-ISSN 2407-4489
e-ISSN 2528-7303

The candidate values of *k* were 3, 5, 7, 9, and 11. Uniform and distance-based weighting were examined using Euclidean distance. Each candidate pipeline was evaluated using the same five outer folds generated through StratifiedGroupKFold. Records with identical combinations of the four indicator scores were assigned to the same group to prevent identical profiles from appearing in both the training and validation portions. Within each candidate pipeline, three inner group-aware folds were used to optimize the KNN hyperparameters. Oversampling was placed inside an imbalanced-learn pipeline and was therefore performed only on the training portion of each fold. The validation portions remained entirely composed of empirical student records. Macro F1-score was used as the main evaluation metric, supported by balanced accuracy, overall accuracy, per-class recall, and cross-fold stability. The model-development configuration is summarized in Table 2.

Table 2. KNN Model-Development Configuration

Component	Configuration
Empirical records	64 students from XI RPL 2 and XI RPL 3
Input features	Decomposition, pattern recognition, abstraction, and algorithmic thinking
Target classes	Low, Medium, and High
Candidate pipelines	KNN, RandomOverSampler–KNN, and SMOTE–KNN
Candidate (k)	3, 5, 7, 9, and 11
Neighbor weighting	Uniform and distance
Distance metric	Euclidean
Outer evaluation	Five-fold StratifiedGroupKFold
Inner optimization	Three-fold StratifiedGroupKFold
Primary metric	Macro F1-score

After the candidate pipelines were compared, the selected pipeline was refitted using all 64 empirical records and exported as a .pkl file using joblib. The serialized pipeline was subsequently loaded by the Streamlit application to classify new student profiles.

In the application, pretest responses were converted into four weighted indicator scores and passed to the KNN pipeline to predict the overall category. The classification result adjusted the detail of adaptive hints, while the treatment sequence was controlled by explicit rule-based logic. The indicators were ordered from the lowest to the highest score. Tied scores were resolved using the predetermined order of decomposition, pattern recognition, abstraction, and algorithmic thinking.

An indicator score below 7 initiated treatment at the easy level, a score from 7 to 9 initiated treatment at the medium level, and a score above 9 initiated treatment at the hard level. Correct responses increased the mastery point by one, while incorrect responses reduced it by one without allowing a negative value. The mastery targets were three points for the easy level, five points for the medium level, and two points for the hard level. Once the hard-level target was achieved, the system proceeded to the next indicator. Students continued to the posttest after all indicators had been completed.

p-ISSN 2407-4489
e-ISSN 2528-7303

Functional Testing and Data Analysis

Classification performance was analyzed using accuracy, balanced accuracy, macro precision, macro recall, macro F1-score, per-class recall, and a confusion matrix. The comparison of candidate pipelines and the performance of the selected model are reported in the Results and Discussion section. The application was evaluated using black-box testing, which examined observable system responses without assessing the internal program structure. Each test case contained the tested module, user action, expected output, actual output, and status. A total of 24 scenarios covered navigation, student identity validation, pretest administration, score calculation, KNN classification, diagnostic results, adaptive hints, rule-based treatment, posttest administration, session recovery, result presentation, and database storage.

A scenario was considered successful when the actual system output corresponded to the predefined expected output. The functional success percentage was calculated using Equation 2.

$$P = \frac{N_{passed}}{N_{tested}} \times 100\%$$

where (P) is the functional success percentage, (N_{\mathrm{passed}}) is the number of successful scenarios, and (N_{\mathrm{tested}}) is the total number of scenarios. The detailed black-box testing results are presented in the Results and Discussion section.

RESULTS AND DISCUSSION

System Implementation Results

The development process produced a web-based computational thinking assessment system deployed through Streamlit Cloud. The system implemented five sequential stages: student identification, pretest, adaptive treatment, posttest, and final-result presentation. A navigation stepper displayed the student's current position, while the instructional expander presented directions according to the active stage. At the initial stage, students entered their identification data and reviewed the curriculum information covering Data Definition Language, Data Manipulation Language, and Data Control Language. The pretest interface presented 32 items and automatically updated the completion progress as students answered each question. Submission was prevented when one or more items remained unanswered.

After pretest submission, the system calculated the weighted decomposition, pattern-recognition, abstraction, and algorithmic-thinking scores. The interface displayed the four indicator scores, total score, KNN-predicted ability category, and indicator selected as the initial treatment focus. These functions enabled the system to present both an overall classification and an indicator-level diagnostic profile rather than only a single aggregate score. Figure 2(a) shows the student-identity interface and curriculum information. Figure 2(b) presents the pretest interface and response-progress indicator, while Figure 2(c) displays the diagnostic results generated after the pretest. The diagnostic interface separated the four computational thinking scores and presented the KNN classification and initial treatment focus.

KNN Classification Results

Three classification pipelines were compared using the same group-aware outer folds: KNN without resampling, RandomOverSampler–KNN, and SMOTE–KNN. The comparison focused primarily on macro F1-score because the empirical dataset contained an unequal class distribution of 7 Low, 13 Medium, and 44 High records. Accuracy alone could provide an overly favorable interpretation toward the majority class. The comparative results are presented in Table 3.

p-ISSN 2407-4489
e-ISSN 2528-7303

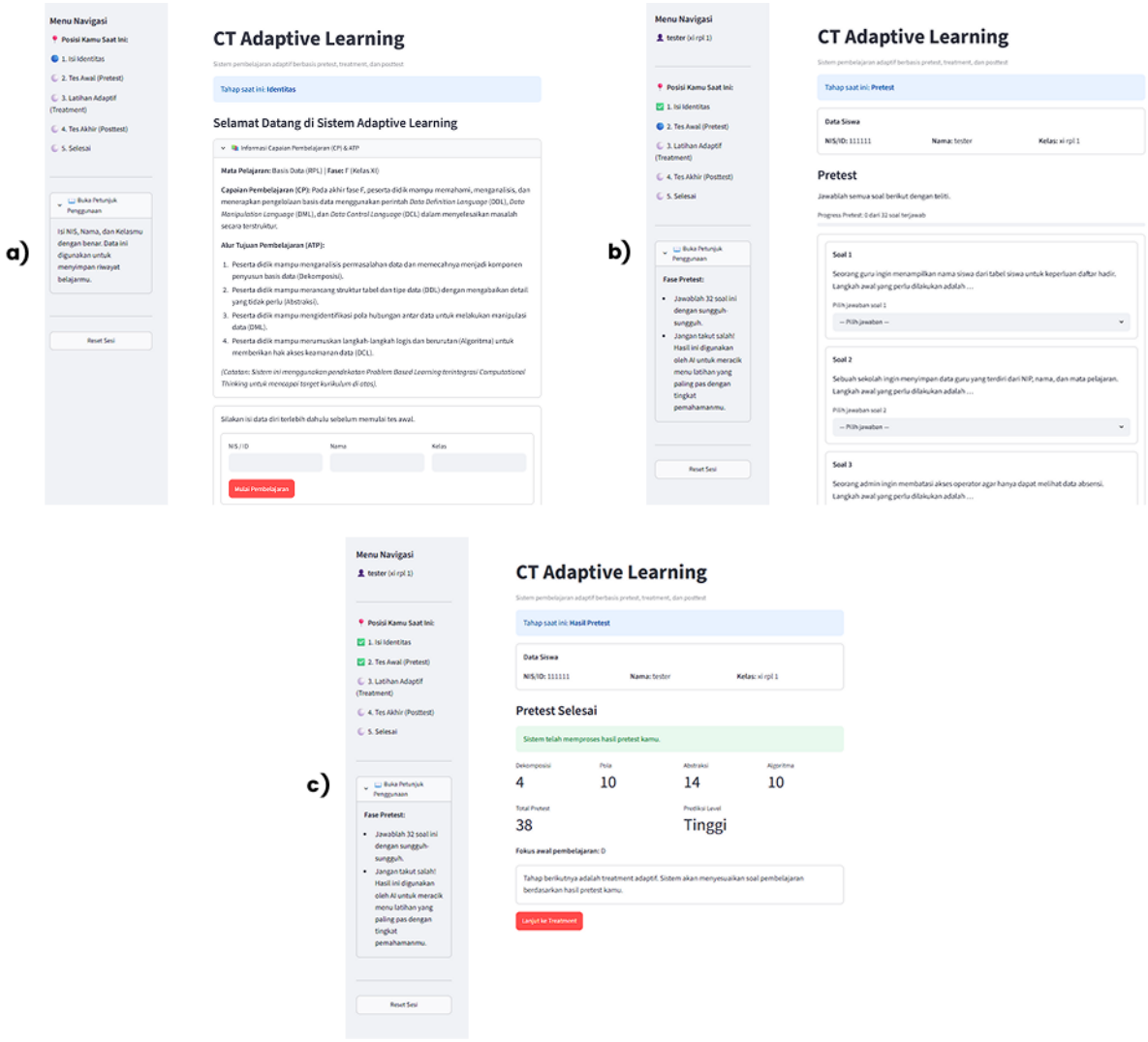


Fig 2. Main assessment interfaces: (a) student identity and curriculum information; (b) computational thinking pretest; and (c) indicator-level diagnostic results and KNN classification.

KNN without resampling achieved an accuracy of 85.94%. However, its balanced accuracy was only 68.13%, and the recall values for the Low and Medium categories were 42.86% and 61.54%, respectively. The perfect recall for the High category indicates that the model was more successful in recognizing the majority class than the two smaller classes.

Table 3. Performance Comparison of Candidate Classification Pipelines

Pipeline	Accuracy (%)	Balanced accuracy (%)	Macro F1 (%)	Recall Low (%)	Recall Medium (%)	Recall High (%)
KNN	85.94	68.13	73.43	42.86	61.54	100.00
RandomOverSampler–KNN	89.06	88.89	87.53	85.71	92.31	88.64
SMOTE–KNN	92.19	88.59	89.75	85.71	84.62	95.45

KNN without resampling achieved an accuracy of 85.94%. However, its balanced accuracy was only 68.13%, and the recall values for the Low and Medium categories were 42.86% and 61.54%,

p-ISSN 2407-4489
e-ISSN 2528-7303

respectively. The perfect recall for the High category indicates that the model was more successful in recognizing the majority class than the two smaller classes.

RandomOverSampler–KNN improved the balanced accuracy to 88.89% and the macro F1-score to 87.53%. It produced the highest Medium-category recall of 92.31%, although its recall for the High category decreased to 88.64%. SMOTE–KNN obtained the highest overall accuracy and macro F1-score, reaching 92.19% and 89.75%, respectively. Its balanced accuracy of 88.59% was slightly below that of RandomOverSampler–KNN, but it produced a more balanced combination of overall performance and class-specific recognition. The pipeline correctly recognized 85.71% of Low profiles, 84.62% of Medium profiles, and 95.45% of High profiles.

Based on the primary selection criterion, SMOTE–KNN was selected as the final classification pipeline. During final hyperparameter optimization, distance-weighted KNN with $k = 9$ and $k = 11$ produced the same mean cross-validated macro F1-score of 0.9276. The smaller value, $k = 9$, was retained for the final model. The selected pipeline used Euclidean distance and distance-based neighbor weighting. The class-level performance of the selected pipeline is presented in Table 4.

Table 4. Class-Level Performance of the Selected SMOTE–KNN Pipeline

Category	Precision (%)	Recall (%)	F1-score (%)	Support
Low	100.00	85.71	92.31	7
Medium	78.57	84.62	81.48	13
High	95.45	95.45	95.45	44
Macro average	91.34	88.59	89.75	64

Six of the seven Low profiles were classified correctly, while one was classified as Medium. Eleven of the thirteen Medium profiles were classified correctly, and two were classified as High. Forty-two of the forty-four High profiles were classified correctly, while two were classified as Medium. In total, 59 of the 64 empirical profiles were correctly classified.

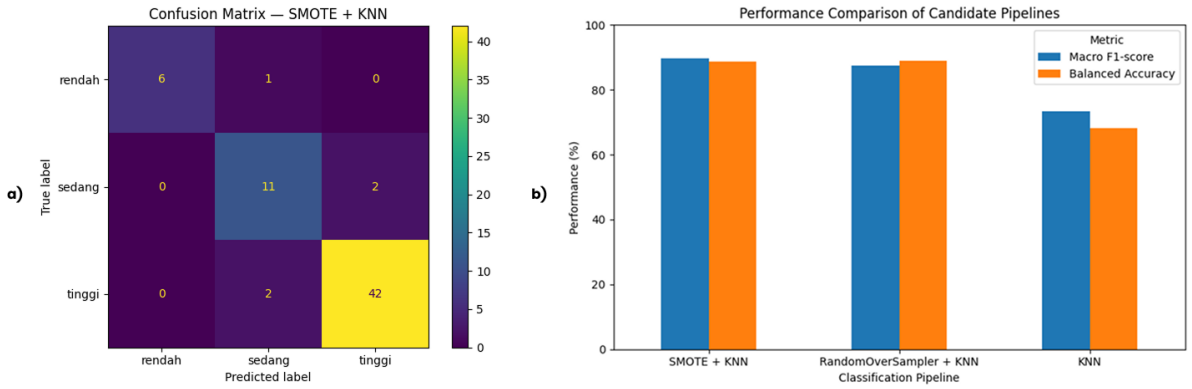


Fig 3. Classification-model performance: (a) comparison of macro F1-score and balanced accuracy across candidate pipelines; and (b) cross-validated confusion matrix of the selected SMOTE–KNN pipeline.

The results demonstrate the effect of class imbalance on the original KNN model. Although its overall accuracy was relatively high, its performance for Low and Medium profiles was substantially weaker. Oversampling improved minority-class recognition, supporting the use of resampling methods when unequal class frequencies influence neighborhood-based classification. SMOTE produced the

p-ISSN 2407-4489
e-ISSN 2528-7303

highest macro F1-score by generating additional minority-class points within the training portions of the cross-validation procedure (Chawla et al., 2002).

The selected distance-weighted configuration also allowed closer neighboring profiles to contribute more strongly than more distant profiles. This is consistent with the similarity-based nature of KNN, in which predictions are influenced by the local arrangement of labeled observations in the feature space (Cunningham & Delany, 2021).Nevertheless, the model performance represents agreement with the operational labels defined in this study. The categories were derived from weighted-score and minimum-indicator rules rather than from an independent expert diagnosis. Therefore, the model should be interpreted as a classifier that reproduces the operational categories from multidimensional score profiles, not as an independent determination of students' cognitive ability.

The confusion matrix also shows that all five misclassifications occurred between adjacent categories. No Low profile was classified as High, and no High profile was classified as Low. The single Low error shifted to Medium, while the remaining errors occurred between Medium and High. This pattern indicates that the observed errors remained between adjacent operational categories rather than across the two extreme categories. The Medium category was the most ambiguous class, with the lowest precision (78.57%) and F1-score (81.48%). This can be explained partly by its position between the two boundary categories: Medium profiles may share local feature-space characteristics with both Low and High profiles, while the operational rubric can assign different labels when minimum-indicator constraints are triggered. The limited number of Low (7) and Medium (13) empirical records further increases sensitivity to individual cases. Therefore, the high overall performance should not obscure the remaining uncertainty near category boundaries.

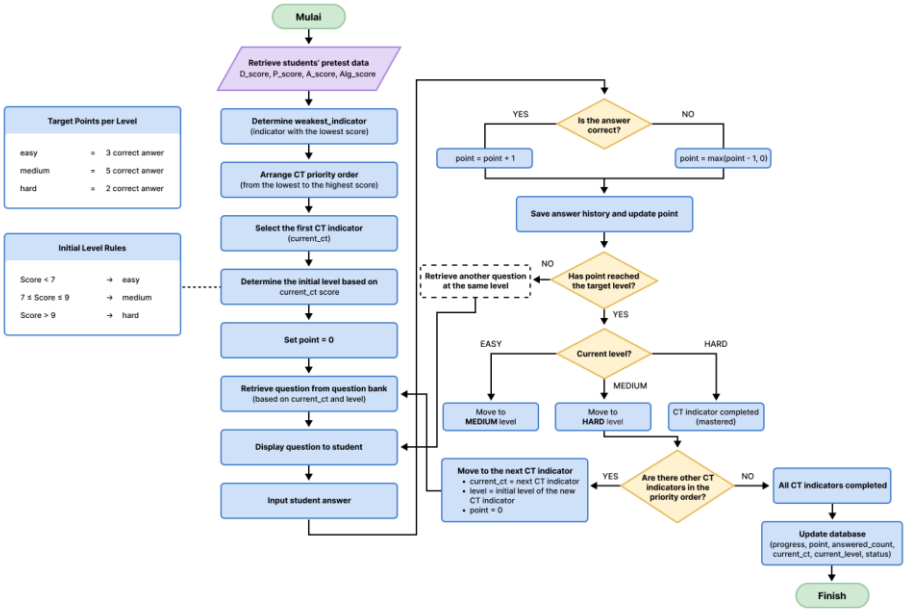


Fig 4. Rule-based adaptive-treatment mechanism implemented in the developed system.

Adaptive Treatment Implementation

The developed system separated the roles of KNN classification and rule-based adaptation. The KNN output represented the student's overall computational thinking category and was used to adjust the level of detail of adaptive hints. Meanwhile, the order and progression of treatment were controlled by explicit rules. The rule-based mechanism first ranked decomposition, pattern recognition, abstraction, and algorithmic thinking from the lowest to the highest pretest score. When two or more indicators had

p-ISSN 2407-4489
e-ISSN 2528-7303

the same minimum value, the predetermined order of decomposition, pattern recognition, abstraction, and algorithmic thinking was applied. This deterministic rule prevented inconsistent treatment assignment for tied scores.

The starting level was determined from the score of the active indicator. Scores below 7 initiated treatment at the easy level, scores from 7 to 9 initiated treatment at the medium level, and scores above 9 initiated treatment at the hard level. A correct answer increased the mastery point by one, whereas an incorrect answer reduced the point by one without allowing a negative value. The mastery targets were three points for easy, five points for medium, and two points for hard. As shown in Figure 4, the system repeatedly selected questions according to the active computational thinking indicator and current difficulty level. When the mastery target was reached, students progressed to the next level. Completion of the hard level marked the active indicator as mastered, after which the system moved to the next indicator in the priority order. The treatment ended after all four indicators satisfied the mastery requirements.The treatment interface displayed the learning path, active indicator, difficulty level, current mastery points, and target points. After an incorrect answer, the system presented an adaptive hint on the subsequent question. The amount of information in the hint was adjusted according to the KNN ability category. The system also provided an option to skip treatment, but the student had to confirm that permission had been received from the teacher. This confirmation reduced the possibility of accidental skipping and stored the treatment status as skipped.

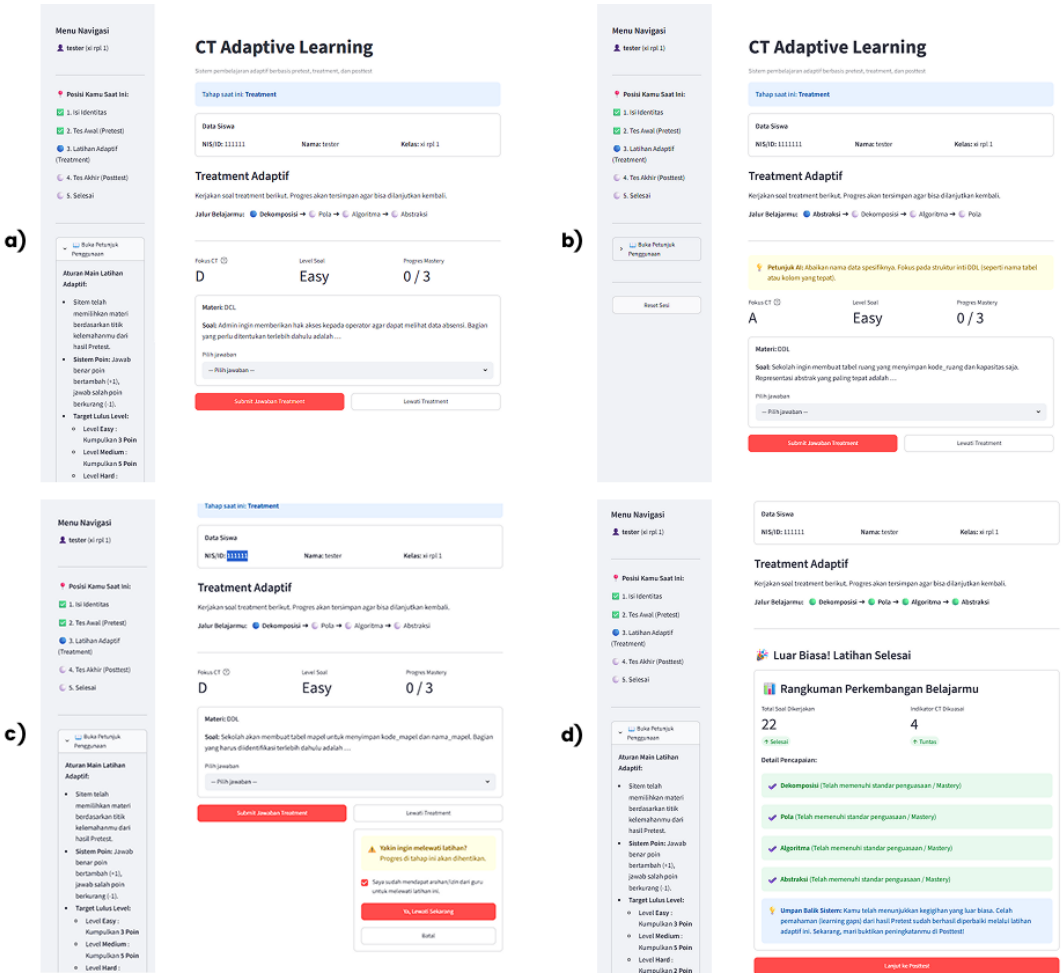


Fig 5. Adaptive-treatment interfaces: (a) indicator priority, active level, and mastery progression; (b) adaptive hint following an incorrect response; (c) teacher-authorized treatment-skip confirmation; and (d) treatment-completion summary.

p-ISSN 2407-4489
e-ISSN 2528-7303

The treatment-completion interface summarized the number of questions answered and the computational thinking indicators that had reached mastery. This implementation reflects the principle that adaptive systems should use learner information to modify feedback and learning progression rather than present an identical path to every learner (Brusilovsky, 2001). The division between classification and rule-based control also improved traceability because each change in treatment level or indicator could be linked to an explicit rule.

Functional Testing Results

The web application was evaluated through 24 black-box testing scenarios covering navigation, student identity, pretest, diagnostic results, adaptive treatment, posttest, and final-result functions. The summary of the testing results is presented in Table 5.

All scenarios produced outputs consistent with their expected results. The navigation stepper changed according to the active stage, instructions adapted to the current page, and the reset function returned the system to the identity stage. The identity and assessment forms prevented submission when required information or responses were incomplete.

Table 5. Summary of Black-Box Testing Results

Module	Scenarios tested	Passed	Failed	Success rate (%)
Navigation sidebar	3	3	0	100
Student identity	3	3	0	100
Pretest	3	3	0	100
Diagnostic results	2	2	0	100
Adaptive treatment	8	8	0	100
Posttest	3	3	0	100
Final result and restart	2	2	0	100
Total	**24**	**24**	**0**	**100**

All scenarios produced outputs consistent with their expected results. The navigation stepper changed according to the active stage, instructions adapted to the current page, and the reset function returned the system to the identity stage. The identity and assessment forms prevented submission when required information or responses were incomplete.

The diagnostic-result tests confirmed that the four indicator scores, total score, KNN category, and treatment focus were displayed after pretest completion. The adaptive-treatment tests verified answer validation, mastery-point increases and decreases, adaptive-hint presentation, level progression, treatment skipping, and treatment completion.

Posttest testing confirmed that the system displayed a warning when treatment had been skipped, saved complete responses to the database, and prevented duplicate final submissions. The final page displayed the pretest score, posttest score, gain value, treatment status, and learning-session summary. These findings indicate that the implemented functions operated according to the predefined functional requirements.

p-ISSN 2407-4489
e-ISSN 2528-7303

The 100% result should be interpreted as functional conformity within the tested scenarios and environment. It does not demonstrate educational effectiveness, usability satisfaction, or reliability under every possible network and user-load condition.

Classroom Implementation and Study Limitations

The system was implemented in XI RPL 1 from May 4 to May 19, 2026, through Streamlit Cloud. Students accessed the application using Google Chrome, primarily through smartphones. Of the 37 recorded student sessions, 34 completed the full workflow, while three remained at the pretest stage. Several submissions were interrupted when the school Wi-Fi connection was unstable, but the process was completed successfully after the affected students used an independent mobile-data connection. This indicates that stable connectivity remains important during response submission and database storage.

The classification model was developed using 64 empirical profiles from one vocational-school context, including only seven Low-category records. Moreover, the target classes were based on an operational scoring rubric rather than an independent expert diagnosis. The reported results therefore represent the model's ability to reproduce the predefined categories. Future evaluation should involve larger and more diverse datasets and improved recovery mechanisms for interrupted submissions.

CONCLUSIONS

This study developed a web-based computational thinking assessment system that integrates indicator-level scoring, K-nearest neighbor classification, adaptive hints, and transparent rule-based treatment. Among the evaluated pipelines, SMOTE–KNN with distance weighting and $k = 9$ achieved the highest macro F1-score of 89.75%, with 92.19% accuracy and 88.59% balanced accuracy. All 24 black-box testing scenarios produced the expected outputs.

The practical contribution of the system is its ability to transform an aggregate assessment result into an indicator-level diagnostic profile and a structured treatment pathway. For teachers, this provides information about which CT component requires earlier support rather than relying solely on a total score. For students, the system provides differentiated hints and treatment progression according to their assessment profile while keeping progression decisions traceable through explicit rules. These findings demonstrate the functional feasibility of integrating data-driven classification and rule-based adaptation in one assessment workflow; they do not establish that the system improves learning outcomes. The model was developed from a limited single-school dataset and operationally defined target categories. Future studies should therefore validate the approach using larger and more diverse datasets, independent category validation, and experimental designs specifically intended to examine educational effectiveness.

ACKNOWLEDGMENT

The authors would like to thank the Computer Science Education Study Program, Universitas Pendidikan Indonesia, and the vocational high school in Bandung for supporting this research. The authors also express their appreciation to the teachers, students, instrument validators, and all parties who contributed to the development, implementation, and evaluation of the system.

REFERENCES

Agbo, F. J., Oyelere, S. S., Suhonen, J., & Adewumi, S. (2019). A systematic review of computational thinking approach for programming education in higher education institutions. In *Proceedings of the 19th Koli Calling International Conference on Computing Education Research* (pp. 1–10). Association for Computing Machinery. DOI: 10.1145/3364510.3364521.

Badriyah, N. L., Thamrin, A. G., & Nurhidayati, A. (2018). Analisis instrumen penilaian hasil belajar mata pelajaran gambar teknik siswa kelas X sekolah menengah kejuruan program keahlian bangunan. *Indonesian Journal of Civil Engineering Education, 4*(2), 93–102. DOI: 10.20961/ijcee.v4i2.27780.

Brusilovsky, P. (2001). Adaptive hypermedia. *User Modeling and User-Adapted Interaction, 11*(1–2), 87–110. DOI: 10.1023/A:1011143116306.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research, 16*, 321–357. DOI: 10.1613/jair.953.

Cunningham, P., & Delany, S. J. (2021). K-nearest neighbour classifiers—A tutorial. *ACM Computing Surveys, 54*(6), Article 128. DOI: 10.1145/3459665.

Delianidi, M., Diamantaras, K., Moras, I., & Sidiropoulos, A. (2025). DK-PRACTICE: An intelligent educational platform for personalized learning content recommendations based on students knowledge state. In *The Barcelona Conference on Education 2024: Official Conference Proceedings* (pp. 391–403). DOI: 10.22492/issn.2435-9467.2024.36.

dos Santos, S. C., & Junior, G. A. S. (2024). Opportunities and challenges of AI to support student assessment in computing education: A systematic literature review. In *Proceedings of the 16th International Conference on Computer Supported Education (CSEDU 2024)* (Vol. 2, pp. 15–26). SciTePress. DOI: 10.5220/0012552500003693.

Forsström, S., & Bond, M. (2024). Computational thinking assessment in primary and secondary education: A meta-synthesis of tools, methods and pedagogical approaches. *International Journal of Computer Science Education in Schools, 6*(4). DOI: 10.21585/ijcses.v6i4.208.

Khatun, S., Jha, A., & Halagappa, P. P. (2023). The effects of problem-based learning model on the computational thinking skills of middle year school students. *International Journal of Current Science, 13*(1), 626–632.

Matayoshi, J., Cosyn, E., & Uzun, H. (2022). Does practice make perfect? Analyzing the relationship between higher mastery and forgetting in an adaptive learning system. In *Proceedings of the 15th International Conference on Educational Data Mining* (pp. 316–324). International Educational Data Mining Society. DOI: 10.5281/zenodo.6853075.

Milala, E. B., Buaton, R., & Fatmaira, Z. (2023). Grouping of outstanding students at Abdi Negara Vocational School using the K-nearest neighbor method. *Journal of Artificial Intelligence and Engineering Applications, 3*(1), 428–432. DOI: 10.59934/jaiea.v3i1.349.

Putra, B., & Sujatmiko, B. (2024). Pengembangan media pembelajaran basis data berbasis web terhadap kompetensi bahasa query terapan di SMK Negeri Mojoagung. *IT-Edu: Jurnal Information Technology and Education, 9*(1), 84–91. DOI: 10.26740/it-edu.v9i1.58647.

Ria, R. R. P. (2023). Asesmen computational thinking di pendidikan matematika: Analisis bibliometrik menggunakan VOSviewer. *EDUPEDIKA: Jurnal Studi Pendidikan dan Pembelajaran, 2*(2), 27–38. DOI: 10.60004/edupedika.v2i2.66.

p-ISSN 2407-4489
e-ISSN 2528-7303

Rosmansyah, Y., Putro, B. L., Putri, A., Utomo, N. B., & Suhardi. (2023). A simple model of smart learning environment. *Interactive Learning Environments, 31*(9), 5831–5852. DOI: 10.1080/10494820.2021.2020295.

Savery, J. R. (2006). Overview of problem-based learning: Definitions and distinctions. *Interdisciplinary Journal of Problem-Based Learning, 1*(1), 9–20. DOI: 10.7771/1541-5015.1002.

Shin, S. S. (2022). Effect of semantic distance on learning structured query language: An empirical study. *Frontiers in Psychology, 13*, 996363. DOI: 10.3389/fpsyg.2022.996363.

Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review, 22*, 142–158. DOI: 10.1016/j.edurev.2017.09.003.

Tan, B., Jin, H.-Y., & Cutumisu, M. (2024). The applications of machine learning in computational thinking assessments: A scoping review. *Computer Science Education, 34*(2), 193–221. DOI: 10.1080/08993408.2023.2245687.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33–35. DOI: 10.1145/1118178.1118215.

Wiyono, S., Wibowo, D. S., Hidayatullah, M. F., & Dairoh, D. (2020). Comparative study of KNN, SVM and decision tree algorithm for student's performance prediction. *International Journal of Computing Science and Applied Mathematics, 6*(2), 50–53. DOI: 10.12962/j24775401.v6i2.4360.