frontiers | Frontiers in Computer Science

TYPE Original Research
PUBLISHED 28 May 2025
DOI 10.3389/fcomp.2025.1510577

OPEN ACCESS

EDITED BY
Clifford A. Shaffer,
Virginia Tech, United States

REVIEWED BY
Mohammed F. Farghally,
Virginia Tech, United States
Liudmyla Gryzun,
Simon Kuznets Kharkiv National University of Economics, Ukraine
Leigh Little,
SUNY Brockport, United States

*CORRESPONDENCE
Sohail Iqbal Malik
✉ sohail@buc.edu.om

RECEIVED 13 October 2024
ACCEPTED 07 May 2025
PUBLISHED 28 May 2025

CITATION
Malik SI, Mathew R, Tawafak RM, Al-Farsi G and Al-Sideiri A (2025) Investigating the impact of the OOP-SOLVE application on the user's behavior using the technology acceptance model in the programming course.
Front. Comput. Sci. 7:1510577.
doi: 10.3389/fcomp.2025.1510577

Investigating the impact of the OOP-SOLVE application on the user's behavior using the technology acceptance model in the programming course

Sohail Iqbal Malik*, Roy Mathew, Ragad M. Tawafak, Ghaliya Al-Farsi and Abir Al-Sideiri

Department of Information Technology, Al-Buraimi University College, Al-Buraimi, Oman

Introductory programming courses are considered difficult and challenging for students. They have to focus on and develop different skills related to problem-solving and programming domains concurrently. However, most programming courses spend more time teaching programming syntax. Therefore, this study developed and introduced an application, OOP-SOLVE, which focused on algorithmic thinking skills in the object-oriented programming (OOP) domain. The pseudo-code technique is used to create this application. Most of the teaching topics of the OOP course, such as classes, objects, constructors, inheritance, and polymorphism, are covered in this application. Moreover, the application presents each programming question in different sections such as class diagram, main class, test class, execution process, and output. A technology acceptance model (TAM) was used to investigate the acceptance of the OOP-SOLVE application in the OOP course. Moreover, the perceptions of the OOP course lecturers regarding the OOP-SOLVE application were collected by conducting a semi-structured interview. 224 students participated in the survey, and six lecturers participated in the interviews. Results show a positive impact of perceived ease of use, usefulness, and enjoyment on students' attitudes toward their intention to use the application in the course. Lecturers also agreed that the application supported students in the OOP course. Moreover, it promotes students' engagement and enhances collaboration and interaction among students in class activities. In addition to the solution of the given programming statement, the OOP-SOLVE application also presents the execution process of the program along with the output of each programming question. Lecturers also agreed that the application can be a supporting teaching tool in the OOP course.

KEYWORDS

object-oriented programming, algorithmic thinking, TAM, e-learning, computer education

1 Introduction

Programming skills are important for all those who are studying computer science and its related fields (Végh and Czakóová, 2023). On the other hand, different studies (Iqbal et al., 2022; Sohail et al., 2020; Wang et al., 2016; De Raadt, 2008; Ala-Mutka, 2004; Kölling and Rosenberg, 1996) reported that many students found it a hard and challenging task to grasp the precise concepts of the programming domain. There are varieties of programming skills such as algorithmic thinking, problem solving and program design where students need to concentrate

at the same time (Al-Emran et al., 2021). On the other side, most programming courses give more importance to syntax than other skills required in the programming domain (Khan et al., 2021). Consequently, students are familiar with programming syntax and semantics, but they are not able to use these rules while writing a valid program (Winslow, 1996). Moreover, most programming courses promote programming shortcuts (Webster, 1994) (problem statement → codes) in their process of teaching and learning. Therefore, Hromkovic et al. (2016) suggested that computer science students should take algorithmic thinking as a main objective in their studies. Loksa et al. (2016) urged that problem solving skills should be explicitly taught in various courses of programming. Khan et al. (2021) suggested that programming courses should include problem solving strategies in all topics.

This study developed an OOP-SOLVE application and then introduced it in the object-oriented programming (OOP) course which focuses on algorithmic thinking and problem-solving strategies. Almost all of the course topics are covered in the application. Moreover, it is based on pseudo-code technique. Each problem statement is presented by different sections such as class diagram, main class, test class, execution process, and output in the application.

The remaining paper is divided into different sections. Information regarding literature review and research questions of the study are presented in the next section. An introduction of the OOP-SOLVE application and Results are presented in the next two following sections. Finally, research outcomes are presented in the last section.

2 Literature review

Algorithmic thinking is a process to devise steps in a particular sequence to solve a given problem (Katai, 2014). The main goal is to achieve the desired outcomes by formulating the required steps (Hu, 2011). Hromkovic et al. (2016) discussed that algorithmic thinking is one of the most important concepts for computer science students and therefore they should focus and practice it as a main objective in this domain. Kiss and Arki (2017) suggested that strategic focus in education should be algorithmic thinking because those students were handicapped in tertiary education if they do not have a background in it.

Hromkovic et al. (2019) suggested that the learning process in programming should start with algorithmic thinking activities. By this approach, problem solving skills can be developed effectively by students along with programming knowledge at the same time (Malik et al., 2019). Malik and Coldwell-Neilson (2017) introduced an ADRI based approach in programming courses to enhance algorithmic thinking among students. Hromkovic et al. (2016) promoted algorithmic thinking by introducing three examples in the programming course. Kiss and Arki (2017) promoted algorithmic thinking by introducing game-based learning in programming education. Moreover, a puzzle-based game was used among students to promote algorithmic thinking (Chih-Chao and Tzone-I, 2018).

Biju (2013) discussed that Object-Oriented programming (OOP) is a complex and difficult area of study which students often struggle to grasp., particularly when they are transitioning from procedural languages. She suggested focusing on teaching methods such as student-centered learning which promotes problem solving and program writing, examples and practical exercises, pictorial representation, and continuous feedback in the OOP courses. Zainal Abidin and Abdullah Zawawi (2020) introduced Augmented Reality (AR) in teaching OOP concepts. They concluded that AR received positive feedback from users. Moreover, it supports students in understanding the OOP concepts. Ardiana and Loekito (2020) introduced gamification in the OOP course. They discussed that gamification stimulated the activeness and creativity of students in the OOP learning. Moreover, it provided a sense of enjoyment derived from the game. They concluded that the use of gamification increases students' motivation in learning OOP concepts. Végh and Czakóová (2023) discussed that visual programming environments and serious games could be used in OOP learning. They urged that students could visualize abstract concepts of OOP by using visual programming environments. Moreover, the serious games helped students understand the basic concepts of the OOP domain.

Cheah (2020) discussed that students often lack problem-solving abilities which is a big challenge in teaching and learning computer programming. He suggested addressing this issue by developing an effective teaching tool, considering both students' learning methods, and the effectiveness of teaching materials. Mathew et al. (2019) prepared and offered a PROSOLVE game to promote problem solving skills for novices in programming courses. They found that students like the game and it also helps students in enhancing their analytical skills and understanding the programming concepts. Moreover, the lecturers also appreciated the game and considered it as a good alternative method to teach programming concepts to novice programmers. Iqbal and Coldwell-Neilson (2018) offered an ADRI based teaching approach in the programming course. They concluded that the new approach emphasized analytical programming strategies and promoted practice among students.

Al-Emran et al. (2021) used the TAM model in a programming course to determine the actual use among three E-platforms which are based on the PROBSOL approach. They concluded that usefulness and ease of use of these three E-platforms influenced the behavioral intention of students to use these platforms in their learning process of programming domain. Thongkoo et al. (2020) introduced digital learning tools in programming education courses and used the TAM model to investigate the students' acceptance of these tools in their studies. Results showed that the students appreciated the initiative of using the digital learning tools in the programming domain. Cabada et al. (2018) introduced a learning environment which is based on web 3.0 in a java programming course and used the TAM model to determine the influence of the learning environment on students' behavior. They concluded that the new learning tool supported students in the programming course.

Different tools were prepared and offered in programming education to enhance programming skills and algorithmic thinking. Zhong and Zhan (2024) introduced an intelligent tutoring system in programming learning which promoted motivation and reduced cognitive load. Kazemitabaar et al. (2024) prepared and introduced codeAid tool in a programming class. Calderon et al. (2024) designed and introduced an automated assessment tool for continuous assessment of students' work in programming. Finnie-Ansley et al. (2022) introduced OpenAI released Codex in introductory programming. Alam (2022) revealed that employing robots in teaching computer programming is a promising tool for early childhood STEM education. Iqbal et al. (2022) prepared and offered a

chatbot in programming education. Majid (2014) integrated web 2.0 tools in a programming course. Sohail et al. (2019) promoted algorithmic thinking by introducing the PROSOLVE game in programming courses.

2.1 Research questions

It is clear from the earlier discussion that algorithmic thinking is an important component in learning computer programming. Different tools were developed and introduced in programming education to enhance programming analytical skills. This study prepared and offered an OOP-SOLVE application to promote algorithmic thinking skills of students related to object-oriented programming concepts. Two research questions are proposed to find out the influence of OOP-SOLVE application in the OOP course.

The statements of these two research questions are as follows:

> *RQ1:* What is the feedback of students after introducing the OOP-SOLVE application in Programming education?
>
> *RQ2:* What is the feedback of object-oriented programming lecturers regarding the OOP-SOLVE application?

3 Research methodology and design

For the first research question, a survey was administered to students of the object-oriented programming course after introducing the OOP-SOLVE application. 224 students participated in the survey. Results show that female participants were 116 in the survey and the remaining 108 were male participants. Ethical approval was obtained from the college before collecting the responses to the survey. Moreover, participation in the survey was voluntary and anonymous.

For the second research question, semi-structured interviews were performed with six lecturers of object-oriented programming courses.

The survey consists of twenty-five questions, the statements of which are presented in Table 1 and Figures 1–5. Responses were received using the five-point Likert scale (1 – "strongly disagree" to 5 – "strongly agree").

TABLE 1 Participant demographic information.

Category	Field	Total	Percentage
Participants	IT-BUC students	224	100%
Gender	Female	116	51.8%
	Male	108	48.2%
Major	Information systems	80	35.7%
	Computer science	57	25.5%
	Software engineering	87	38.8%
Degree	Diploma	86	38.4%
	Advanced diploma	3	1.3%
	Bachelor	135	60.3%
Class timings	Morning	107	47.8%
	Evening	117	52.2%
Technology learning interest	Low	37	16.5%
	Medium	73	32.6%
	High	114	50.9%

4 Introduction to OOP-SOLVE application

The OOP-SOLVE is a web-based application that introduces a novel approach to the pseudo-code technique of solving problem statements related to object-oriented programming. The application covers most topics of the OOP course, such as classes, objects, constructors, inheritance, and polymorphism. Questions related to each topic are accessible from the relevant menu bar. Moreover, Figure 6 shows that each problem statement is presented by different sections such as class diagram, main class, test class, execution process, and output.

Figure 7 depicts the interface of the first section of the problem statement in the OOP-SOLVE application. The first section focuses on the problem statement, class diagram, and pseudo-code of the class. The application was developed using visual basic web forms. A programming question is provided at the top of the first section. A class diagram is provided, illustrating the relevant components of the problem statement. The Random Steps text area offers a pseudo-code-based solution to the programming question, as illustrated in Figure 7. The solution steps are randomised each time the user reloads or reopens the program, ensuring that students receive different solutions and concentrate on proposing a solution with every attempt. A user selects and moves the provided solution of the given programming question from the Random Steps text area to the Proposed Steps text area by using the right arrow shown in the Actions area. A given pseudo-code solution is automatically added in the Pseudo Code section for the step moved in the Proposed Steps text area by the user. Four arrows (right, left, up and down) are provided in the Actions area so the user can move the commands right or left between the Radom and Proposed text areas. Moreover, the commands can be moved up or down either in the Random Steps or Proposed Steps text areas using the up or down errors.

This section also shows the errors in the solution in two ways. The right solution is shown in green in the Proposed Steps text area, as shown in Figure 7. The wrong solution or steps are shown in red in the Proposed Steps text area. Moreover, the number of errors in the solution is shown at the bottom of the Proposed Steps text area.

Figure 8 depicts the second section of the OOP-SOLVE application. This section focuses on the test class of the given programming question. The second section is shown only when the right solution for the given programming question is provided in the Proposed Steps text area of the first section. The second section consists of a test class diagram, Random Steps and Proposed Steps text areas, pseudo-code Solution text area, four arrows, No of Errors label, and Input Data box, as shown in Figure 8. The functionality of the second section is similar to the first section except that the Input Data section is included to get variable values from the user related to the given programming question.

Figure 9 depicts the third section of the OOP-SOLVE application which focuses on the execution process of the test class. The section shows how objects, variables are created, and methods are executed as

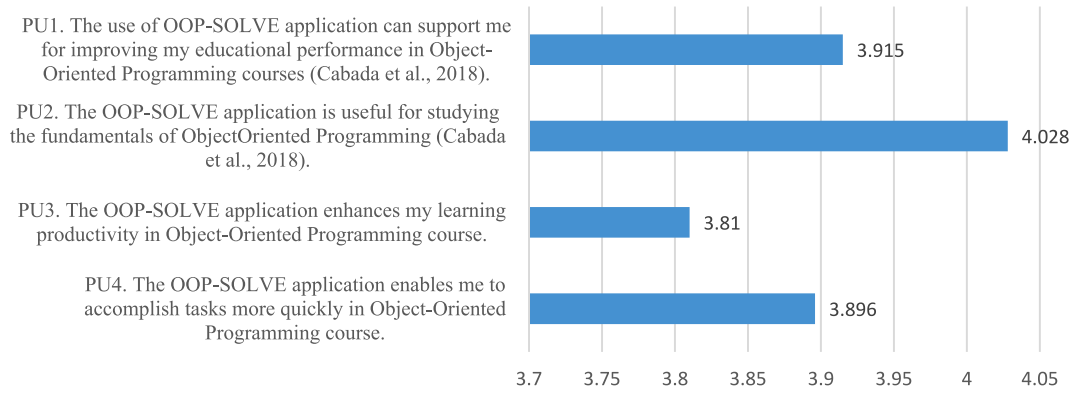

FIGURE 1
Perceived usefulness (PU).

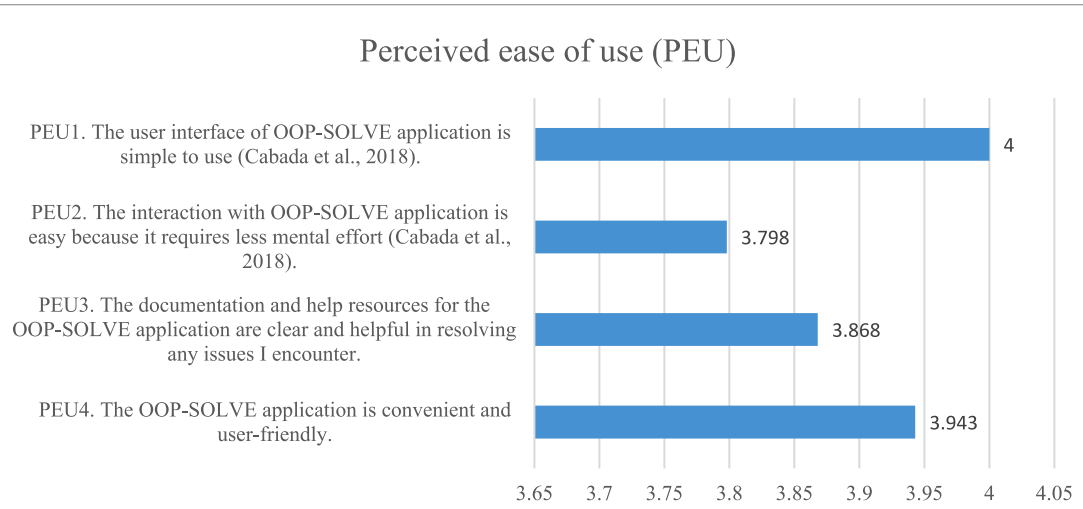

FIGURE 2
Perceived ease of use (PEU).

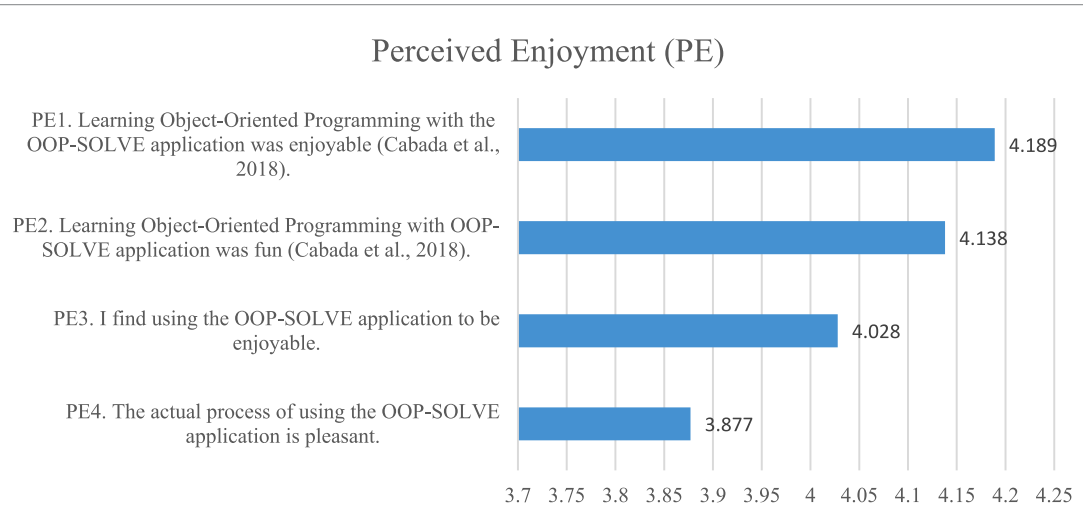

FIGURE 3
Perceived enjoyment (PE).

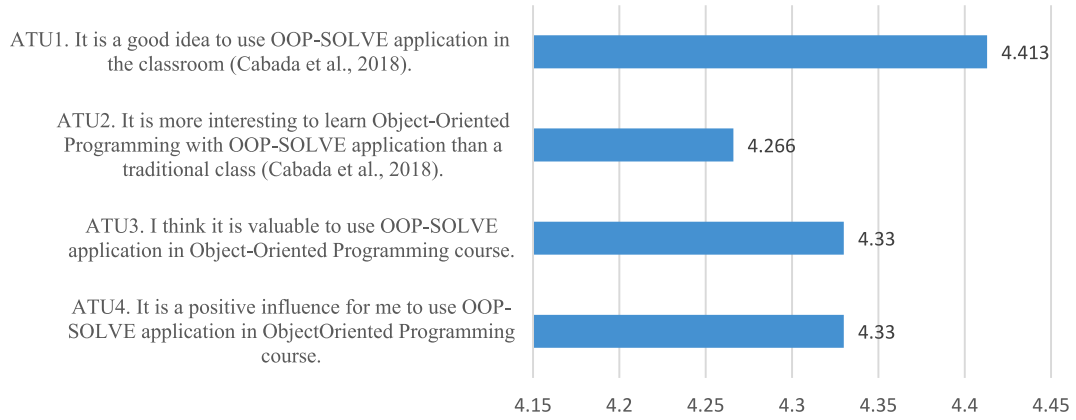

FIGURE 4
Attitude toward using (ATU) OOP-SOLVE application.

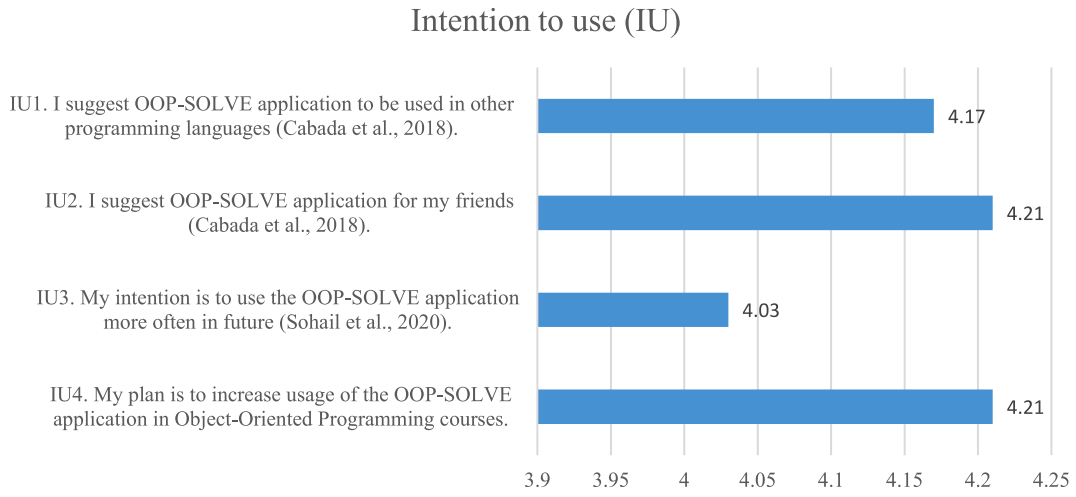

FIGURE 5
Intention to use (IU).

per the instructions of the test class of the given problem statement as shown in Figure 9.

Figure 10 depicts the fourth section of OOP-SOLVE Application which shows the output of the test class of the given problem statement in a tabular form. This section focuses on objects, variables, and methods of the test class.

5 Results

Results of student's survey and semi-structured interviews of Object-oriented programming course lecturers are described in this section. Research question 1 is addressed by probing the responses of student's survey.

> *RQ1:* What is the feedback of students after introducing the OOP-SOLVE application in programming education?

This study used the technology acceptance model (TAM) to investigate the impact of OOP-SOLVE application on the user's behavior. The e-learning acceptance studies used TAM (Davis, 1986) as one of the most accepted theories (Teo, 2009). In this study, the research model consists of five factors such as "perceived usefulness (PU)," "perceived enjoyment (PE)," "perceived ease of use (PEOU)," "attitude toward using (ATU)," and "intention to use (IU)." There are four items in each factor. The questionnaire statements are shown in Figures 1–5. Moreover, the conceptual research model is shown in Figure 11.

The study uses the following seven hypotheses to determine the relationships between five factors of this study:

> *H1:* The perceived ease of use of the OOP-SOLVE positively impacts its perceived usefulness.

> *H2:* The perceived usefulness of OOP-SOLVE increases the students' attitude toward using the tool in the future.

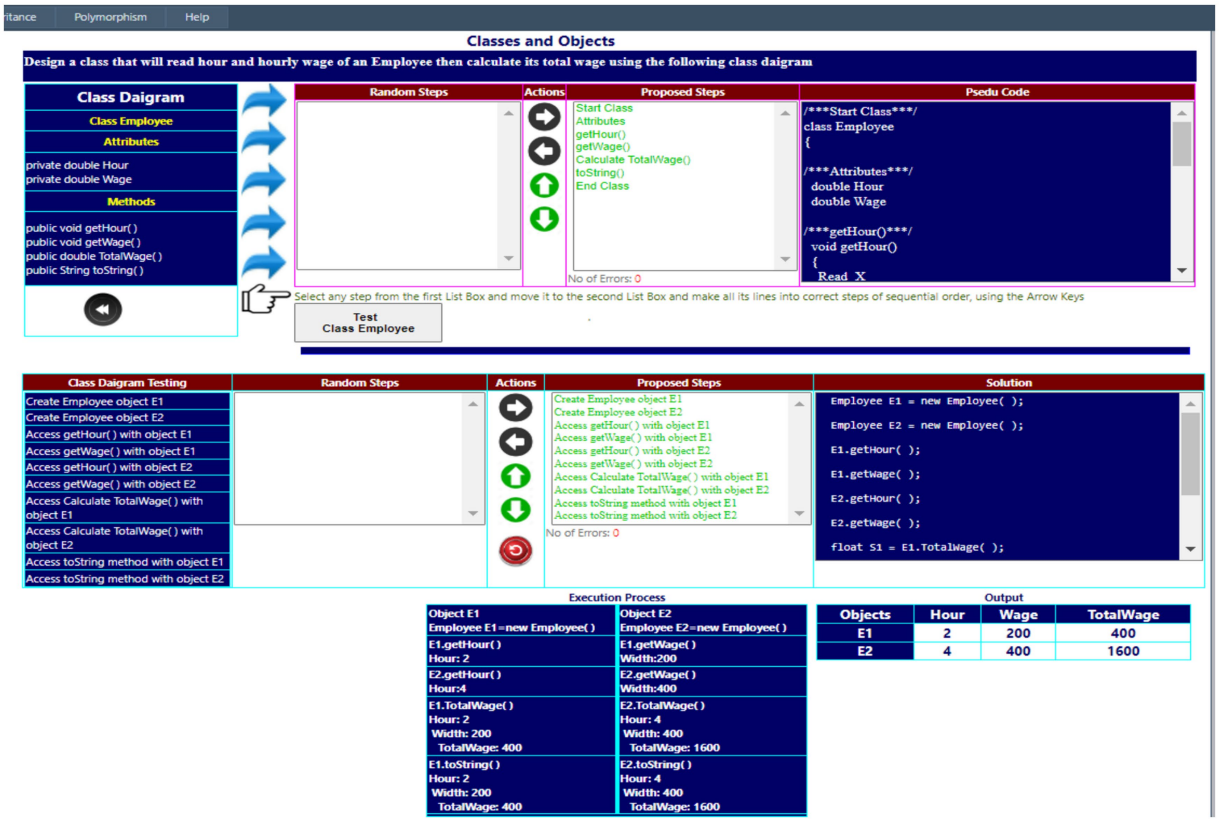

FIGURE 6
Four sections of the problem statement.

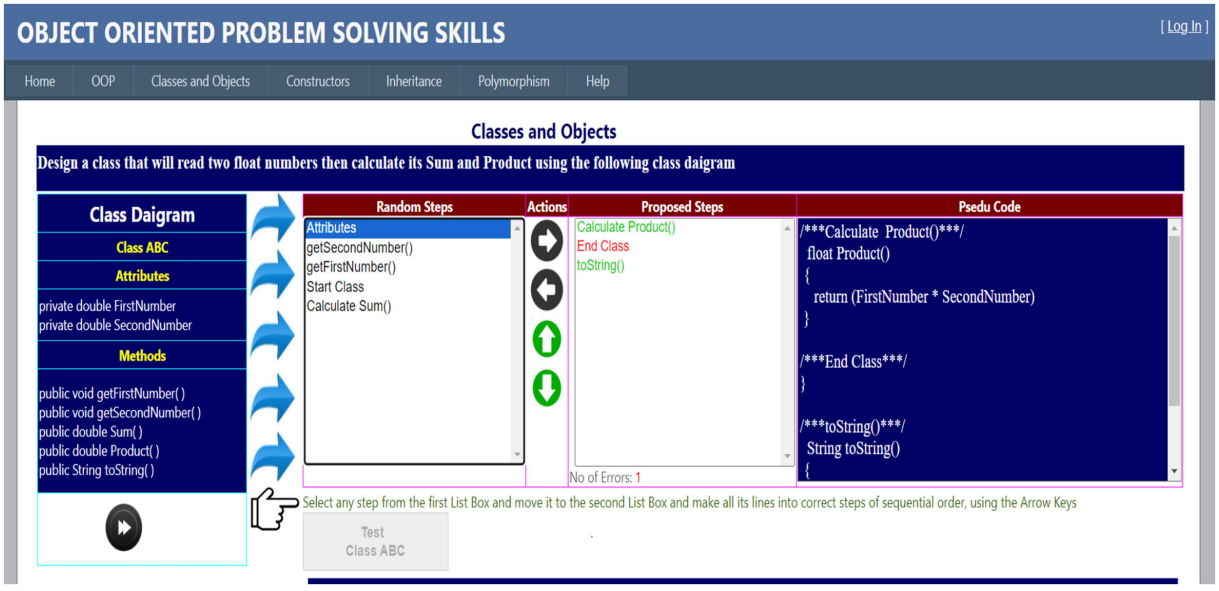

FIGURE 7
First section of the OOP-SOLVE application (class).

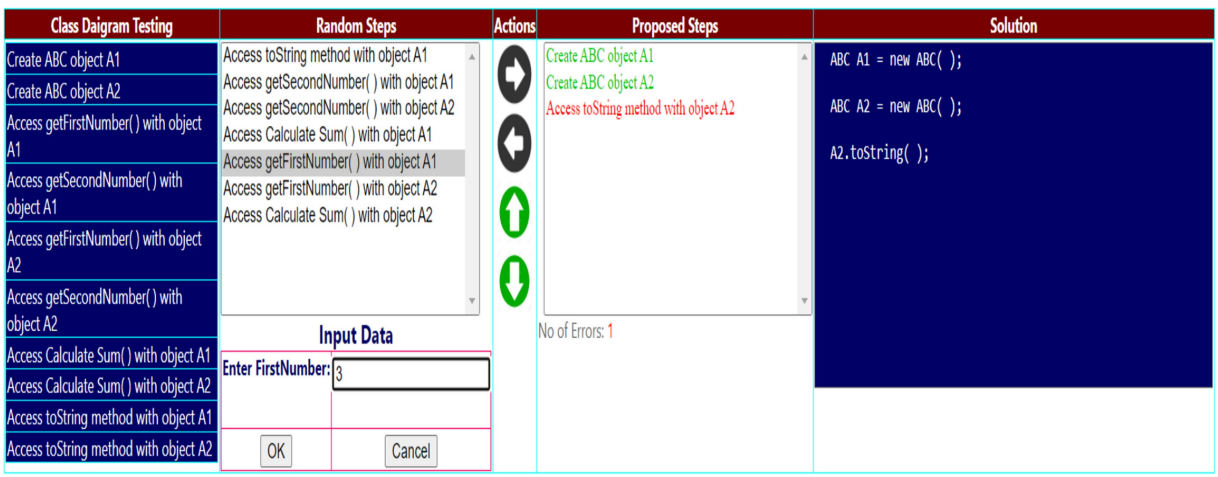

FIGURE 8
Second part of OOP-SOLVE application (test class).

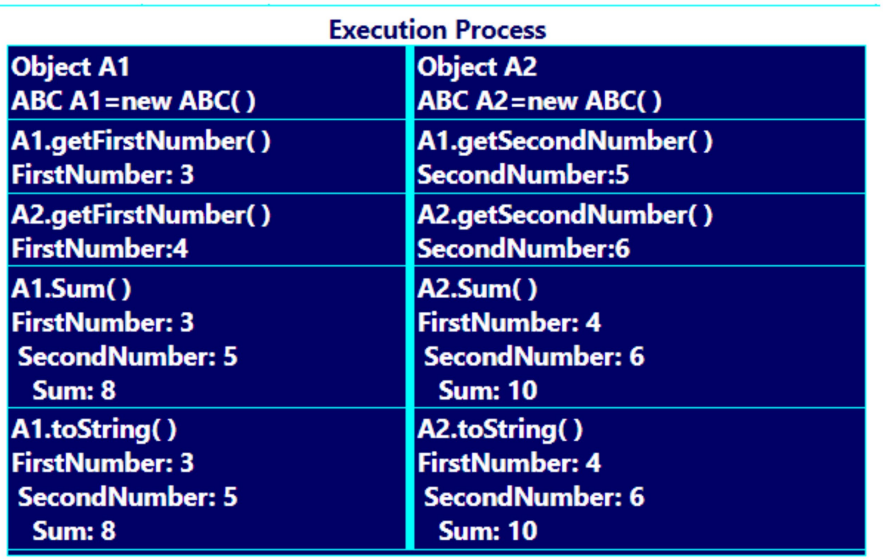

FIGURE 9
Third section of OOP-SOLVE application (execution process).

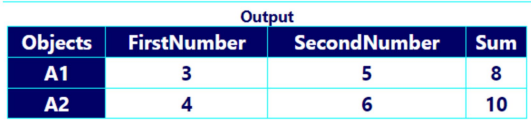

FIGURE 10
Fourth section of OOP-SOLVE application (output).

H3: The perceived ease of use of the OOP-SOLVE enhances the students' attitude to use the tool.

H4: The perceived ease of use of the OOP-SOLVE enhances the students' perceived enjoyment of using the tool.

H5: The perceived enjoyment of the OOP-SOLVE enhances the students' attitude to use the tool.

H6: The student's attitude toward using OOP-SOLVE leads to their intention to use the tool in their programming activities.

H7: The perceived enjoyment of the OOP-SOLVE leads to the intention to use the tool in their programming activities.

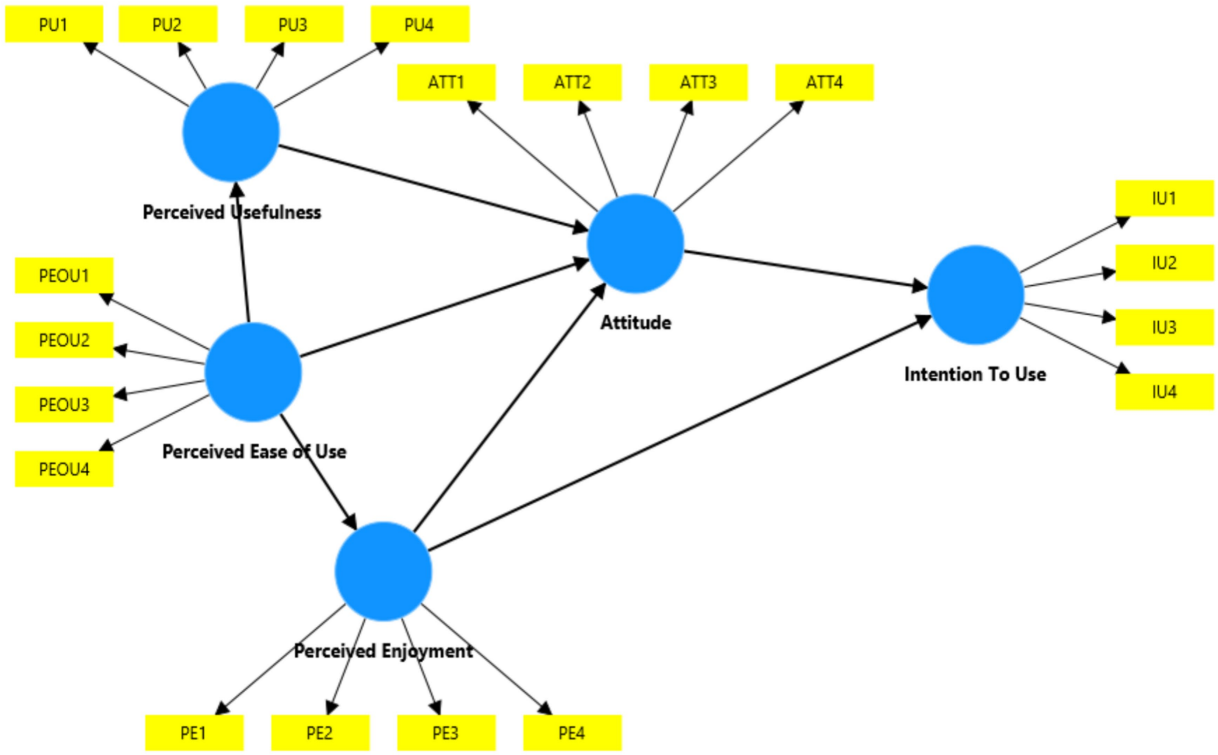

FIGURE 11
Conceptual research model.

5.1 Participants

Participants of this survey were IT students at Al-Buraimi University College, Oman. 224 Students filled in the survey after the OOP-SOLVE application was offered to them in the object-oriented programming course. The demographic information of the participants is presented in Table 1. This section includes six questions relating to gender, major, degree, class timings, and technology learning interest.

5.2 Procedure

The OOP-SOLVE application was offered during the first semester of academic year 2023–24 in the OOP course. Students used the application during class time under the teacher's supervision and then after the class at their own convenience without supervision of the teacher. The responses for the survey were collected during the time frame of 8th to 15th of January 2024.

5.3 Findings

The PLS-SEM program was used to validate the model and test the hypotheses of this study. Figure 1 presents the surveys' questionnaire statements and their mean values regarding the four questions related to perceived usefulness (PU). All mean values are in the range of 3.81 to 4.028.

Figure 2 depicts the mean values of four questions related to perceived ease of use (PEU). The range of mean values is between 3.798 and 4.

Four questions related to perceived enjoyment (PE), along with their mean values, are presented in Figure 3. All mean values are in the range of 3.877 to 4.189.

Figure 4 shows four questions and their mean values related to attitudes toward using (ATU) the OOP application. The range of mean values is between 4.266 and 4.413.

The last section of the survey presents four questions related to the intention to use (IU). All mean values are in the range of 4.03 to 4.21.

5.4 Reliability analysis

A reliability analysis was performed to determine the internal validity and consistency of each item which is used in each factor. A Cronbach's Alpha test was performed to calculate the validity values of each item used in the research model. The Cronbach's Alpha value of equal to or greater than 0.7 is considered as an acceptable reliability of an instrument. A value of 0.8 or above is considered a good level of reliability. Table 2 depicts that the Cronbach's Alpha values of "Attitude toward using (ATU)," and "Perceived Ease of Use (PEOU)" are at the acceptable level. The reliability values of "Intention to Use (IU)," "Perceived Enjoyment (PE)," and "Perceived Usefulness (PU)" are in the range of good level of reliability.

The results depict that "perceived enjoyment," "intention to use," and "perceived usefulness" factors are good which means that students

enjoyed using the OOP-SOLVE application in the teaching-learning process of object-oriented programming concepts. The remaining two factors (PEOU and ATU) also lie in the acceptable range which means that students perceive that it is easy to use the OOP-SOLVE application.

5.5 Hypothesis testing

The path analysis was performed to verify all hypotheses which show the relationship between pairs of the factors used in this study. Table 3 depicts the path coefficient (β), f-square, and p-value for each hypothesis of this study. The acceptable value of path coefficient (β) is equal to or greater than 0.1, and for p-value it is $p < 0.01$ or $p < 0.001$, and for f-square the acceptable effect size is greater than or equal to 0.02.

The results in Table 3 show that all hypotheses are supported and the relationship between independent and dependent factors are significant. Moreover, all the path coefficient (β), f-square and p-values are in acceptable range.

Results depict that PEOU and PU have a positive impact on the attitude of the students to use the OOP-SOLVE application in learning object-oriented programming concepts. Moreover, students enjoyed using the application in the course which impacted positively on their intention and attitude to use the application.

5.6 Semi-structured interviews with object-oriented programming course lecturers

Results of the second research question are presented in this part of the paper. Research question 2 is explored by organizing semi-structured interviews with lecturers of the object-oriented programming course.

TABLE 2 Cronbach's alpha test for reliability analysis.

Factor	Cronbach's alpha
"Attitude toward using (ATU)"	0.747
"Intention to use (IU)"	0.846
"Perceived ease of use (PEOU)"	0.790
"Perceived enjoyment (PE)"	0.888
"Perceived usefulness (PU)"	0.826

> *RQ2:* What is the feedback of object-oriented programming lecturers regarding the OOP-SOLVE application?

Six lecturers of the OOP course participated in semi-structured interviews. The purpose of the interviews was to explore the lecturers' in-depth feedback regarding the affordances and barriers of the OOP-SOLVE application in the studying process of the OOP courses. The interview was not audio-recorded, and the session notes were taken independently by the moderator and the assistant moderator. They produced a joint session report and shared it with all lecturers for feedback and approval.

The interviews focused on exploring the lecturers' feedback on:

1. Students' experience with OOP-SOLVE application during the course.
2. The effect of OOP-SOLVE application on students' learning.
3. Strengths and weaknesses of the OOP-SOLVE application.
4. Any further suggestions to improve the OOP-SOLVE application for the programming course.

5.6.1 First impression of OOP-SOLVE application

Six lecturers agreed that the first impression of the OOP-SOLVE application in the course was positive. They appreciated the idea of introducing pseudo-code techniques in understanding the OOP concepts. The application supports students to grasp the fundamental building concepts of programming paradigm such as problem analysis, program design without taking care of syntax of computer programming language. One lecturer said, "The OOP-SOLVE application uses a pseudo-code technique, which I believe is the best strategy to start teaching programming."

Lecturers discussed that the application presents each problem statement with class diagram, main class, test class and execution process. This process provides a holistic view of the OOP domain to students.

5.6.2 Students' experience with OOP-SOLVE application

Five lecturers ascertained that the OOP course students felt comfortable with using the application. Moreover, the application provides practice to students for most teaching topics of the OOP domain. Students found that it is easy to use and accessible everywhere, which helps them to get more hands-on experience regarding the OOP concepts covered in the course. One lecturer said, "The OOP-SOLVE application is awesome; students can practice the exercises at home."

TABLE 3 Hypothesis testing results.

Hypothesis	Relationship	β	p value	f-square	Remarks
H1	Perceived ease of use → Perceived usefulness	0.238	0.000	0.213	Supported
H2	Perceived usefulness → Attitude toward using	0.492	0.001	0.354	Supported
H3	Perceived ease of use → Attitude toward using	0.427	0.001	0.403	Supported
H4	Perceived ease of use → Perceived enjoyment	0.531	0.000	0.890	Supported
H5	Perceived enjoyment → Attitude toward using	0.411	0.000	0.613	Supported
H6	Attitude toward using → Intention to use	0.549	0.010*	0.171	Supported
H7	Perceived enjoyment → Intention to use	0.376	0.000	0.529	Supported

$p < 0.01$, * $p < 0.001$, $\beta >= 0.1$.

Six lecturers agreed that students' engagement in the course was promoted by the application. Moreover, the simple design of the OOP-SOLVE application motivates users to practice more programming problems.

5.6.3 Impact of OOP-SOLVE application on students' learning

Six lecturers concluded that the OOP-SOLVE application provided a good impact on students' learning. The application enhances collaboration and interaction among students. The application indicates the mistakes or wrong steps to students which provides them the chance to think about their solution for the given programming question. One lecturer said, "Students learning in the programming domain can be enhanced by providing instantaneous feedback on their code."

Four lecturers agreed that the OOP-SOLVE application also impacted positively on students' attitudes toward completing the exercises. Moreover, it presents the execution process of the given programming question related to the computer memory which provides a clear understanding of the underlying OOP concepts to users.

5.6.4 Strengths of the OOP-SOLVE application

Five lecturers agreed that the application promotes a novel approach by using pseudo-code technique in the teaching-learning process of OOP concepts. The application provides each time new random steps as a solution of the given programming question which promotes critical thinking for solving the given programming question. The application promotes problem solving skills because the given solution is provided with pseudo-code technique and students can only pay attention to the problem analysis and program design. One lecturer said, "Most programming courses offer practice using offline software. The OOP-SOLVE is web-based, which supports students' engagement in the course."

Students can easily access the application through the web. Furthermore, students can practice or finish exercises at their own convenience. Students can consider the OOP-SOLVE as a good supporting tool for practicing the OOP concepts.

5.6.5 Weaknesses of the OOP-SOLVE application

Four lecturers suggested that the option to add more programming questions in the OOP-SOLVE application should be provided. This process helps to increase the question bank and provides more variety of problem statements to students for practice. They also suggested adding tooltips for the pseudo-code terminology used in the OOP-SOLVE application.

5.6.6 Suggestions to improve the OOP-SOLVE application

Five lecturers suggested introducing a login feature for users in the application. This process helps to monitor the progress of the students in the application. They also suggested introducing a system to track the common mistakes of students while solving the problems. This process helps lecturers to focus more on programming concepts where students make more frequent mistakes.

Two lecturers suggested introducing game elements in the application to promote fun and a reward system in the learning process. Moreover, it promotes competition and enhances practice among students.

6 Discussion

The OOP-SOLVE application was developed and introduced in object-oriented programming education to promote algorithmic thinking skills. Students' perceptions regarding the application were collected by administering a survey. Results show that students perceived that it is convenient and simple to use this OOP-SOLVE application. This result is consistent with Malik et al. (2019) who also promoted problem solving skills in programming education by introducing a web-based application. Results show that students perceived that they enjoyed the course after introducing the application in it. This result is consistent with Cabada et al. (2018) who also discovered that students enjoyed a learning environment which is based on web 3.0 in a programming course.

Results also showed that students' behavioral intention to use the OOP-SOLVE application in the course was influenced by easy to use and usefulness. This result is consistent with Al-Emran et al. (2021). The OOP-SOLVE application is accessible to students at any place and at their own convenience. They can solve programming questions after the class at their home as well which promotes practice among students. This result is consistent with Winslow (1996) who stated that "the old saw that practice makes perfect has a solid psychological basis" (p. 18). Moreover, Wang et al. (2017) stated that programming training supports in enhancing students' higher order thinking.

The OOP-SOLVE application discourages "programming shortcut" (Webster, 1994) (programming question → codes) and encourages four-step programming approach (programming question → problem solving strategy for main class → problem solving strategy for test class → codes). The four-step programming approach supports users to better understand the programming question and underlying concepts of object-oriented programming before start writing the code. This result is consistent with Sohail et al. (2019) who introduced a game in the programming course to discourage programming shortcuts.

Lecturers in their semi-structured interviews agreed that the OOP-SOLVE is a useful supporting tool in programming education which supports students to practice object-oriented programming (OOP) concepts at their own convenience. Pseudo-code technique is used in the OOP-SOLVE which promotes algorithmic thinking among students. The application provides random steps to plan the solution of the given programming question. This process supports students by taking less cognitive load for devising the solution and focusing more on underlying OOP concepts, program design and structure. This finding is consistent with Shuhidan (2012) who discussed that programming courses involve theoretical concepts and their practice which pushes the cognitive process of many students into overload.

7 Conclusion

This study prepared and introduced the OOP-SOLVE application to promote algorithmic thinking skills among students of object-oriented programming (OOP) course. Pseudo-code technique is used to develop this application. The teaching topics of the OOP domain such as classes, objects, constructor, inheritance, polymorphism, and test classes are covered in the OOP-SOLVE. Questions belonging to the above-mentioned teaching topics are accessible from the relevant menu bar in the

OOP-SOLVE. Each problem statement is presented by different sections such as class diagram, main class, test class, execution process, and output in the application. The technology acceptance model (TAM) was used in this study to explore the acceptance of the OOP-SOLVE application in the course. Semi-structured interviews were performed with the lecturers of the OOP course to collect their in-depth feedback regarding the affordances and barriers of OOP-SOLVE application in the programming domain.

A survey was conducted with the students of OOP class after introducing the OOP-SOLVE application in programming education. The PLS-SEM program was used for validating the model and testing the hypotheses of this study. Results show that all hypotheses are supported in this study. Moreover, the perceived usefulness, ease of use, and enjoyment have a positive influence on the attitude of students toward their intention to use the OOP-SOLVE application.

Lecturers agreed that the OOP-SOLVE application supported students' learning in the programming domain. The pseudo-code technique is used in developing the application which provides students an opportunity to focus more on program analysis and design. The application promotes algorithmic thinking skills, enhances interaction and collaboration among students. Students can access the application anytime and anywhere which impacts positively on students' attitudes toward completing the exercises. The application also presents the execution process and output of each problem statement which provides students a better understanding of the underlying concepts of the OOP domain. The application design is simple which motivates students to practice programming problems related to the OOP domain. The application is a good supporting tool for students to practice OOP concepts.

7.1 Implications for future research

The OOP-SOLVE application is based on the pseudo-code technique. We plan to introduce the flow chart technique, which provides students with a visual presentation of the programming questions. The OOP-SOLVE application provides information about the number of errors in the users' solution. We plan to add specific information about the errors to the users. Tooltips will be added for the pseudo-code terms used in the application. The OOP-SOLVE application will be offered in other computer subjects such as data structure, web programming, or other STEM or business subjects.

Data availability statement

The raw data supporting the conclusions of this article will be made available by the authors, without undue reservation.

Ethics statement

The studies involving humans were approved by Research Ethics Committee/Buraimi University College. The studies were conducted in accordance with the local legislation and institutional requirements. The ethics committee/institutional review board waived the requirement of written informed consent for participation from the participants or the participants' legal guardians/next of kin because students' participation was voluntary and anonymous.

Author contributions

SM: Conceptualization, Funding acquisition, Writing – original draft, Writing – review & editing. RM: Conceptualization, Software, Writing – original draft, Writing – review & editing. RT: Investigation, Methodology, Writing – original draft, Writing – review & editing. GA-F: Conceptualization, Data curation, Formal analysis, Writing – original draft, Writing – review & editing. AA-S: Software, Validation, Visualization, Writing – original draft, Writing – review & editing.

Funding

The author(s) declare that financial support was received for the research and/or publication of this article. This work was supported by MOHERI, Oman. The grant number is (BFP/RGP/ICT/23/454).

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Generative AI statement

The authors declare that no Gen AI was used in the creation of this manuscript.

Publisher's note

References

Alam, A., (2022). Educational robotics and computer programming in early childhood education: a conceptual framework for assessing elementary school students' computational thinking for designing powerful educational scenarios. In 2022 International Conference on Smart Technologies and Systems for Next Generation Computing (New York, USA: ICSTSN) (pp. 1–7). IEEE.

Ala-Mutka, K., (2004). Problems in learning and teaching programming – a literature study for developing visualizations in the codewitz-minerva project, Codewitz needs analysis. Available online at: http://www.cs.tut.fi/%7Eedge/literature_study.pdf (Accessed December 20, 2013).

Al-Emran, M., Malik, S. I., Arpaci, I., and Mathew, R. (2021). "Comparison of e-learning, m-learning, and game-based learning applications for introductory programming courses: an empirical evaluation using the TAM" in Recent advances in technology acceptance models and theories. eds. M. Al-Emran and K. Shaalan. Studies in systems, decision and control, Springer, Cham. 335, 293–309. doi: 10.1007/978-3-030-64987-6_17

Ardiana, D. P., and Loekito, L. H. (2020). "Gamification design to improve student motivation on learning object-oriented programming" in Journal of physics: Conference series. 2nd International Conference on Vocational Education and Technology (IConVET) (Bali, Indonesia: IOP Publishing) 1516:012041. doi: 10.1088/1742-6596/1516/1/012041

Biju, S. M. (2013). "Difficulties in understanding object-oriented programming concepts" in Innovations and advances in computer, information, systems sciences, and engineering. eds. K. Elleithy and T. Sobh (New York: Springer) 152, 319–326. doi: 10.1007/978-1-4614-3535-8_27

Cabada, R. Z., Estrada, M. L. B., Hernández, F. G., Bustillos, R. O., and Reyes-García, C. A. (2018). An affective and web 3.0-based learning environment for a programming language. *Telematics Inform.* 35, 611–628. doi: 10.1016/j.tele.2017.03.005

Calderon, K., Serrano, N., Blanco, C., and Gutierrez, I. (2024). Automated and continuous assessment implementation in a programming course. *Comput. Appl. Eng. Educ.* 32:e22681. doi: 10.1002/cae.22681

Cheah, C. S. (2020). Factors contributing to the difficulties in teaching and learning of computer programming: a literature review. *Contemp. Educ. Technol.* 12:ep272. doi: 10.30935/cedtech/8247

Chih-Chao, H., and Tzone-I, W. (2018). Applying game mechanics and student-generated questions to an online puzzle-based game learning system to promote algorithmic thinking skills. *Comput. Educ.* 121, 73–88. doi: 10.1016/j.compedu.2018.02.002

Davis, F. D. (1986). Technology acceptance model for empirically testing new end-user information systems: theory and results. MA, USA: Massachusetts Institute of Technology.

De Raadt, M., (2008). Teaching programming strategies explicitly to novice programmers', PhD thesis, University of Southern Queensland, Australia. Available at: https://research.usq.edu.au/item/9yy76/teaching-programming-strategies-explicitly-to-novice-programmers (Accessed June 17, 2024).

Finnie-Ansley, J., Denny, P., Becker, B. A., Luxton-Reilly, A., and Prather, J., (2022). The robots are coming: exploring the implications of openai codex on introductory programming. In Proceedings of the 24th Australasian Computing Education Conference. New York, NY, USA: Association for Computing Machinery. 10–19. doi: 10.1145/3511861.3511863

Hromkovic, J., Kohn, T., Komm, D., and Serafini, G. (2016). Examples of algorithmic thinking in programming education. *Olympiads Inform.* 10, 111–124. doi: 10.15388/ioi.2016.08

Hromkovic, J., Kohn, T., Komm, D., and Serafini, G., (2019). Algorithmic thinking from the start, The Education Column, Available online at: http://bulletin.eatcs.org/index.php/beatcs/article/view/478 (Accessed May 10, 2024).

Hu, C., (2011). Computational thinking: what it might mean and what we might do about it. *In* Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education, New York, USA: ACM. 223–227.

Iqbal, S. M., Ashfaque, M. W., Mathew, R., Jabbar, J., Al-Nuaimi, R. S., and Al-sideiri, A. (2022). Fostering the learning process in a programming course with a Chatbot. *Int. J. Online Pedagog. Course Des.* 12, 1–17. doi: 10.4018/IJOPCD.306686

Iqbal, S. M., and Coldwell-Neilson, J. (2018). Gender differences in an introductory programming course: new teaching approach, students' learning outcomes, and perceptions. *Educ. Inf. Technol.* 23, 2453–2475. doi: 10.1007/s10639-018-9725-3

Katai, Z. (2014). The challenge of promoting algorithmic thinking of both sciences and humanities-oriented learners. *J. Comput. Assist. Learn.* 31, 287–299. doi: 10.1111/jcal.12070

Kazemitabaar, M., Ye, R., Wang, X., Henley, A. Z., Denny, P., Craig, M., et al. (2024). "Codeaid: evaluating a classroom deployment of an llm-based programming assistant that balances student and educator needs" in Proceedings of the CHI conference on human factors in computing systems. New York, NY, USA: Association for Computing Machinery. Article 650, 1–20. doi: 10.1145/3613904.3642773

Khan, I., Al-Mamari, A., Al-Abdulsalam, B., Al-Abdulsalam, F., Al-Khansuri, M., Iqbal Malik, S., et al. (2021). "A machine learning classification application to identify inefficient novice programmers" in Advances in visual informatics: 7th international visual informatics conference, IVIC 2021, Kajang, Malaysia, November 23–25, 2021, proceedings 7 (Switzerland: Springer International Publishing), 423–434.

Kiss, G., and Arki, Z. (2017). The influence of game-based programming education on algorithmic thinking. *Procedia. Soc. Behav. Sci.* 237, 613–617. doi: 10.1016/j.sbspro.2017.02.020

Kölling, M., and Rosenberg, J., (1996). BlueJ – a language for teaching object-oriented programming. Proceedings of the twenty-seventh SIGCSE technical symposium on Computer science education. New York, NY, USA: Association for Computing Machinery. 190–194. doi: 10.1145/236452.236537

Loksa, D., Ko, A. J., Jernigan, W., Oleson, A., Mendez, C. J., and Burnett, M. M., (2016). Programming, problem solving, and self-awareness: effects of explicit guidance. Proceedings of the 2016 CHI conference on human factors in computing systems. New York, NY, USA: Association for Computing Machinery. 1449–1461. doi: 10.1145/2858036.2858252

Majid, N. A. A. (2014). Integration of web 2.0 tools in learning a programming course. *Turk. Online J. Educ. Technol.* 13, 88–94. Available at: https://www.tojet.net/articles/v13i4/13410.pdf (Accessed June 17, 2024).

Malik, I. S., and Coldwell-Neilson, J. (2017). Impact of a new teaching and learning approach in an introductory programming course. *J. Educ. Comput. Res.* 55, 789–819. doi: 10.1177/0735633116685852

Malik, I. S., Mathew, R., and Hammood, M. M. (2019). "PROBSOL: a web-based application to develop problem-solving skills in introductory programming" in Smart technologies and innovation for a sustainable future. Advances in science, Technology & Innovation (IEREK interdisciplinary series for sustainable development). eds. A. Al-Masri and K. Curran (Germany: Springer), 295–302.

Mathew, R., Malik, S. I., and Tawafak, R. M. (2019). Teaching problem solving skills using an educational game in a computer programming course. *Inform. Educ.* 18, 359–373. doi: 10.15388/infedu.2019.17

Shuhidan, S. M.. (2012). Probing the minds of novice programmers through guided learning, PhD thesis, retrieved July 2024, RMIT University: Australia.

Sohail, I. M., Al-Emran, M., Mathew, R., Tawafak, R., and AlFarsi, G. (2020). Comparison of E-learning, M-learning and game-based learning in programming education–a gendered analysis. *Int. J. Emerg. Technol. Learn.* 15, 133–146. doi: 10.3991/ijet.v15i15.14503

Sohail, I. M., Mathew, R., Tawafak, R. M., and Khan, I., (2019). Gender difference in perceiving algorithmic thinking in an introductory programming course. In EDULEARN19 Proceedings (Valencia, SPAIN: IATED) 8246–8254.

Teo, T. (2009). Modelling technology acceptance in education: a study of pre-service teachers. *Comput. Educ.* 52, 302–312. doi: 10.1016/j.compedu.2008.08.006

Thongkoo, K., Daungcharone, K., and Thanyaphongphat, J. (2020). "Students' acceptance of digital learning tools in programming education course using technology acceptance model" in 2020 Joint International Conference on Digital Arts, Media and Technology with ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunications Engineering (ECTI DAMT & NCON) (New York, USA: IEEE), 377–380.

Végh, L., and Czakóová, K. (2023). Possibilities of using games in teaching and learning the basic concepts of object-oriented programming, INTED2023 proceedings, 17th International Technology, Education and Development Conference, 5329–5334, IATED, doi: 10.21125/inted.2023.1383

Wang, H. Y., Huang, I., and Hwang, G. J. (2016). Comparison of the effects of project-based computer programming activities between mathematics-gifted students and average students. *J. Comput. Educ.* 3, 33–45. doi: 10.1007/s40692-015-0047-9

Wang, X. M., Hwang, G. J., Liang, Z. Y., and Wang, H. Y., (2017). Enhancing students' computer programming performances, critical thinking awareness and attitudes towards programming: an online peer-assessment attempt. J. Educ. Technol. Soc. 20: 58–68. Available online at: www.jstor.org/stable/26229205 (accessed February 3, 2020).

Webster, M., (1994). Overview of programming and problem solving, Merriam-Webster's Collegiate Dictionary, 10. Merriam-Webster, Springfield, MA. Available at: https://samples.jbpub.com/9781284028645/Programming6e_CH01.pdf (Accessed June 13, 2024).

Winslow, L. E. (1996). Programming pedagogy-a psychological overview. *ACM Sigcse Bull.* 28, 17–22. doi: 10.1145/234867.234872

Zainal Abidin, Z., and Abdullah Zawawi, M. A. (2020). OOP-AR: learn object oriented programming using augmented reality. *Int. J. Multimed. Recent Innov.* 2, 60–75. doi: 10.36079/lamintang.ijmari-0201.83

Zhong, X., and Zhan, Z. (2024). An intelligent tutoring system for programming education based on informative tutoring feedback: system development, algorithm design, and empirical study. *Interact. Technol. Smart Educ.* 22, 3–24. doi: 10.1108/ITSE-09-2023-0182