Implementation and Utilisation of the Didactic AlgoPoint Application to Facilitate Teacher-Student Group Collaboration

Marcin Stefanowicz and Anna Sasak-Okoń[a]
Maria Curie-Skłodowska University, Pl. M. Curie-Skłodowskiej 5, 20-031 Lublin, Poland

Keywords: Teacher-Student Collaboration, Flowcharts, AlgoPoint, Authoring Didactic Tool, Visual Programming, Block-Based Programming.

Abstract: The paper presents the implementation and evaluation of a teacher-student collaborative module for the AlgoPoint educational application. The module helps teachers manage students, create groups, and develop assessment tools based on local networks. Various approaches to teaching programming and algorithms are explored, emphasizing computational thinking and teacher-student collaboration. The core components of the module are presented, including database structure, user and group management, and test sections. In addition, the results of a classroom study are analyzed, confirming the positive reception of the application by the school community.

1 INTRODUCTION

Before becoming a skilled programmer, it is essential to understand the core principles of computational thinking (CT). This foundation is typically built during early education, particularly in primary and secondary schools (Henderson et al., 2007; Yadav et al., 2011). Problem-solving, often seen as a key element of programming education, can be fostered through the use of effective tools and teaching methods, which are essential to develop strong CT skills (Romeo et al., 2017). Visual tools can offer significant support in the learning process. In recent years, educational games that teach programming through play have gained popularity, with options available to both younger and older students (Rugelj and Lapina, 2019). Another aid in learning programming is the use of flowcharts. These visual diagrams provide a clear representation of the program elements and their relationships, helping students grasp the logic behind algorithms without getting overwhelmed by syntax or semantics. Flowcharts guide learners in organizing algorithms logically, ensuring that they can be effectively translated into functional programs (Weintrop, 2021; Kraleva et al., 2019).

Student-teacher cooperation and group work play a crucial role in effective programming education. When students work together, especially under the guidance of a teacher, they benefit from shared knowledge and different perspectives, which improves understanding of complex topics and encourages a deeper engagement with programming tasks. In a classroom environment, cooperative learning has shown a positive impact on academic performance, as students who work in groups tend to outperform those who learn individually. Platforms that allow collaborative coding and peer review can enhance the learning experience by providing real-time feedback and enabling group members to tackle challenges together (Öztürk, 2023; Tsay-Vogel and Brady, 2010).

AlgoPoint (Stefanowicz and Sasak-Okoń, 2023) is an application designed to assist high school students who find it difficult to grasp the fundamentals of programming and algorithms. It serves as a valuable resource for self-directed learning of introductory programming or as an instructional aid for computer science teachers. The application boasts a range of features, including an intuitive Flowchart Editor and tools for constructing algorithms using customizable blocks and connectors. These algorithms can be thoroughly analyzed and tested through a built-in console with the option to convert them to source code.

This paper presents the implementation of a new collaborative teacher-student module for AlgoPoint. It allows teachers to manage individual students' work, group them for collaborative tasks, and create tests based on flowchart projects within a local network. This tool aims to enhance both individualized

[a] https://orcid.org/0000-0002-4593-120X

Stefanowicz, M. and Sasak-Okoń, A.
Implementation and Utilisation of the Didactic AlgoPoint Application to Facilitate Teacher-Student Group Collaboration.
DOI: 10.5220/0013365200003932

In *Proceedings of the 17th International Conference on Computer Supported Education (CSEDU 2025) - Volume 1*, pages 538-546
ISBN: 978-989-758-746-7; ISSN: 2184-5026

instruction and group learning, offering teachers more control over their students' progress and assessment in algorithmic problem-solving.

The paper text that follows contains 5 Sections. Section 2 discusses related work. Section 3 briefly presents basic functionalities implemented in AlgoPoint. Section 4 describes the newly implemented teacher-student group collaboration module. Section 5 details the findings from research conducted in a real-world high school setting, demonstrating the practical utility of the application. Section 6 provides the final conclusions and reflections on the study.

2 RELATED WORK

Although numerous textual and visual programming languages are used in computer science education today, none of them facilitate teacher-supervised group collaboration combined with block-based code building. Starting from simplifications of existing programming languages such as MiniScript (Strout, 2021), Coral (Edgcomb et al., 2019), or SIMPLE (Rababaah, 2020) through games that support learning programming, such as Tuk Tuk (Koracharkornradt, 2017) or AstroCode (Bione et al., 2017).

A separate group consists of didactical tools that follow the block-based approach. Here, one of the most well-known tools is Scratch (Meerbaum-Salant et al., 2010). It allows users to develop simple games and interactive programs, specifically tailored for young users to learn through creating their own games and animations. It forms a coding community for children that encourages collaboration, but without direct teacher supervision. Similar sulutions are, for example, Alice, App Inventor or SmartBuilder (Kaya and Yildiz, 2019; Werner et al., 2012).

A block-based code-building approach is also effective in teaching programming to older beginners. Among such applications, Flowgorithm is worth mentioning. The code blocks have distinct shapes for specific functions and are connected by lines with arrows, rather than being stacked like puzzle pieces. The notation for operators in the expressions is mixed, meaning that a single operation may be performed using multiple operators, which could be confusing. However, Flowgorithm offers the advantage of allowing users to preview the source code generated in various programming languages (Cook et al., 2015). Two other applications offering similar features are Raptor and Visual Logic (Kourouma, 2016).

The applications discussed above are primarily designed for self-directed learning, with the teacher's role being more about specifying the problem rather than managing the solution. More advanced teacher features, but without the block-based approach are offered by Tynker. It is an online platform designed to teach programming and coding to children. It offers a variety of tools and resources to help young learners understand the fundamentals of programming through interactive lessons, games, and activities. Teachers can define a variety of programming tasks icluding collaborative projects where students work together to build a larger program or game (Elsawah and Thabet, 2023). Similar functionalities offers CodeCombat - an interactive platform that teaches coding through an engagig game environment. Teachers can set up classes and assign specific levels or lessons to their students and monitor their progress with a teacher dashboard (Choi et al., 2024).

More advanced students focused on text-programming can benefit from applications like CrossCode (Mazumdar et al., 2024) or Blockly (Valsamakis et al., 2020). They offer a collaborative workspace where users can share projects and contribute to the development process together. The collaborative setup is good for educational environments, as teachers can act as the project masters, guiding students through various coding tasks.

Based on our knowledge, AlgoPoint distinguishes itself with several key features, including a high degree of customization for flowchart elements, a detailed Functions and Data Editor, the ability to adjust various algorithm execution parameters, and a modern, user-friendly interface. A new module developed to facilitate teacher-student collaboration significantly enhances the educational value of the application. This module, like most of the analyzed group collaboration systems, includes user management functionality. However, its key distinguishing feature is the ability to create tests based on flowcharts. This can significantly help students understand new programming concepts and serve as a valuable tool for teachers to assess students' existing knowledge.

3 ALGOPOINT APPLICATION

The teacher-student group collaboration module for AlgoPoint (Stefanowicz and Sasak-Okoń, 2023), as well as the original release of the application itself, has been implemented in the free cross-platform Lazarus integrated development environment 2.2.4 based on the Free Pascal 3.2.2 compiler, which provides improved performance and stability (Dietrich et al., 2024). In addition, the Lightweight Networking Library (lNet) package proved to be indispensable. It is a collection of classes and components that facili-

tate the operation of networks mainly using TCP and UDP protocols based on an event mechanism. Moreover, a self-developed version of a graph database has been implemented. The database is saved as a binary file in the application data folder, ensuring the efficiency and compactness of the data storage.

3.1 AlgoPoint User Interface

The main graphical interface of the application consists of four significant areas: a Ribbon-type toolbar having the option to search for actions and blocks (top), a navigation list of function diagrams (left), an embedded console (bottom), a preview of variable values (right), and a block and connection editor in the central part of the window (Fig. 1). The program also has a complex File menu, a window for changing its basic settings, a preview of the generated source code in six programming languages (Free Pascal, C++, C#, Java, JavaScript, Python), and a module for checking the execution of the designed algorithm.

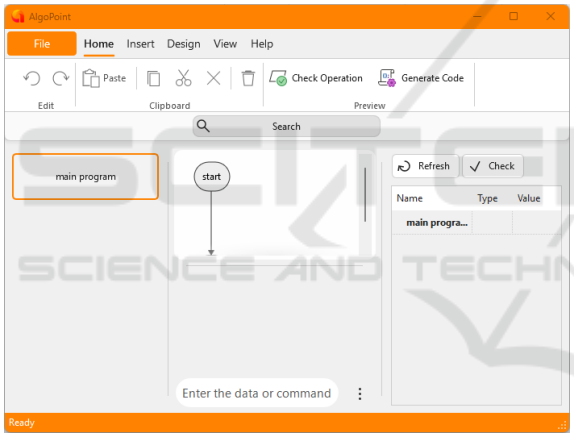

Figure 1: Main application window.

3.2 The Flowchart Editing Component

The Flowchart Editor centrally located in the main window of AlgoPoint is the core of the application. It enables seamless creation and analysis of designs. The blocks and connections it contains can be formatted using options available on the Design page of the Ribbon (for instance: font name, fill color, border width, inbuilt styles). The design function diagram is based on a start-stop block, extended with other shapes representing specific actions (e.g., data entry, calling a subroutine). Elements can be inserted using commands from the Ribbon. When the one hovers over the connection and clicks the gray plus symbol, the selected block type is added in the designated area and is ready for further editing.

The content of blocks in the editor varies and depends on their type. The user can modify the contents and properties of a block in the Project Management window. For example, in the case of an operations block, one can enter instructions in the source code editor with integrated syntax highlighting. In addition, this editor offers code hinting and auto-complete functions, displaying in a pop-up window a list of available functions, constants and variables for the currently open subroutine flowchart.

3.3 Algorithm Testing

It is worth noting, the application in question offers controls for the testing process (Run/Resume, Pause, and Stop), which allows users to adjust the duration of each step (in seconds) and enable a step-by-step mode. In step-by-step mode, the preview halts after each iteration, facilitating granular analysis. When testing commences, the active block's outline is emphasized through thickening and color inversion for enhanced visibility. To prevent inadvertent changes, most commands are deactivated during testing. The testing process can be manually terminated or automatically stopped when irregularities are detected, such as syntax errors within the operation blocks.

4 TEACHER-STUDENT GROUP COLLABORATION MODULE

In the former version of the application, the user interface directly presented the main program window upon startup. The newly implemented Teacher-Student Group Collaboration module modifies this scheme, initially displaying the Welcome Screen. This window consists of two tabs: Individual Mode and Cooperation. The first allows the user to create a new blank project or open an existing one, previously saved and loaded into the application from disk. The second tab, on the other hand, contains a typical login form. In order for the students to join a test session, they have to check the Connect on Local Network checkbox and enter a valid port and IP address for the teacher's computer.

4.1 Command Palette

The Command Palette is a new helpful solution that significantly improves user interaction with AlgoPoint. It provides the ability to find the necessary information immediately and perform desired actions in a centrally located place, which can translate into time

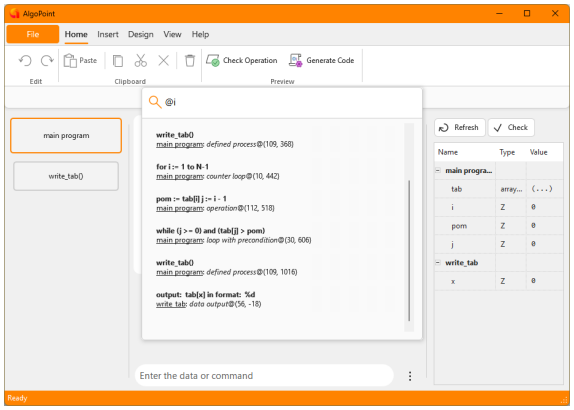

Figure 2: Command palette during the search of blocks by their contents.

savings and increased work efficiency. The Palette is available for any type of user account, both from the keyboard and from the GUI (after clicking the Search field located under the Ribbon). Moreover, the elements displayed in the list of search results are properly formatted, thus ensuring their readability and clarity. It provides a robust search and execution capability, surpassing standard menu options. Its features encompass:

- **Name-Based Search** - users can directly input action names into the command field for comprehensive searches across all menu types.
- **Content-Based Block Search** - by prefixing search terms with '@', users can locate blocks within the project based on their contents, especially convenient for navigating intricate block structures (Fig. 2).
- **Real-Time Expression Evaluation** - expressions preceded by an '=' are instantly evaluated and displayed, providing a convenient alternative to dedicated dialog boxes.
- **Command Listing** - a question mark in the search field yields a list of all executable commands.

Search results can be refined using modifiers (e.g., '@#(ex)+'). The default search employs approximate string matching, prioritizing exact matches before considering substrings. To enforce exact matching, prefix the search term with the exclamation mark. Regular expressions, preceded by '#', can also be used. It ought to be noted that only one modifier can be applied per search.

4.2 Teacher Application View

Once the users have successfully logged into a teacher (or administrator) account, they are redirected to the application's main interface. This window consists of a number of tools that allow both the creation of standard flowchart projects and collaboration at the teacher-student level. Its key element is the new Collaboration special tab on the Ribbon (Fig. 3). It contains a number of appropriately grouped actions, such as managing users, groups, test sections, block constraints in the algorithm flowchart or test creation.

4.2.1 List of Users

The control center for all users who are registered on a given teacher's computer workstation is the User Management window (Fig. 4). This window is divided into two functional areas. The first is the Ribbon located at the top of the window. It contains buttons for performing basic operations on user accounts, i.e. creating, deleting, modifying and importing them from a CSV file. Furthermore, there is an option to force synchronization of information between the application and the database file (by default it always takes place after the window is closed). The remaining part of the window is occupied by a list of users, presenting detailed information about them. This view groups them into columns (account name, first name, last name, role, custom time) and displays them as a vertically scrolling list. To perform a given action on a user, one has to right-click on the row presenting their data.

4.2.2 List of Groups

Another auxiliary window in the teacher-student group collaboration module is the user group manager. The interface of this window consists of three main sections. The top section is the Ribbon, analogous to the user management window. It contains options for adding, deleting and modifying group data, as well as synchronizing with the database. On the left side, below the Ribbon, is a list of group names arranged chronologically by the date of their creation. The rest of the window is occupied by a detailed list of users (with User Name, First Name and Last Name columns). It lists only student accounts, since only they can join an active test conducted on the local network. Multiple selections are allowed for each group, so that many students can belong to multiple groups.

4.2.3 Test Sections

The functionality of flowcharts in the new application version has been extended to include test sections, supplementing standard control and information blocks. Test sections can be placed exclusively on the main branch of the start-stop block, which

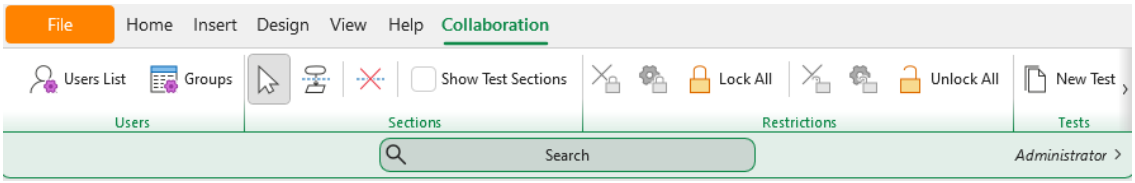

Figure 3: Commands available on the Collaboration tab.

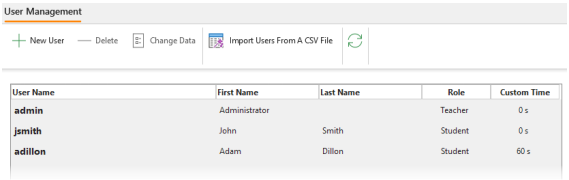

Figure 4: Part of the user management window.

governs the overall algorithm execution and task sequence. Management is facilitated through dedicated actions in the Sections group on the Collaboration ribbon tab. These actions include selection, addition, removal, and visibility toggling of test sections. Initially, test sections are hidden for all user accounts to prevent unauthorized modifications and safeguard flowchart integrity.

Like other block types, test sections can be modified by double-clicking or using the context menu's Edit Content option. This opens a dialog with fields for editing the section's properties. A text field allows for naming the section, which is displayed in the test creator for easy identification. Below, a syntax-highlighted code editor provides the same capabilities as the operation block editor, enabling the implementation of optional initialization instructions for assigned students. These instructions can include variable definitions, default values, and global constant initialization. The table component in the lower part represents and edits the test's positive completion conditions. It consists of three columns: Condition (for entering the condition's source code), Required (to indicate if the condition is necessary for test completion), and Points (to specify the points awarded for fulfilling the condition). Buttons above the table allow for adding and removing rows.

4.2.4 Restrictions of Blocks

The flowchart functionality within the teacher-student collaboration module has been enhanced with access control mechanisms for individual blocks. These controls are accessed via dedicated actions in the Restrictions group on the Collaboration ribbon tab. Teachers can use these controls to block (and unblock):

- **Block Deletion** - when activated, the block cannot be deleted. Users can still modify its content and appearance but cannot remove it from the flowchart. This protects critical flowchart elements from accidental or intentional removal by students.
- **Content and Appearance Changes** - this prevents users from modifying the block's content and appearance. The block becomes locked, and its contents remain unchanged. This provides additional protection and prevents unwanted modifications.
- **All Blocks** - except for the start-stop block and test sections, all blocks in the flowchart are locked. This feature can be used to completely prevent students from making changes to specific parts of the flowchart.

4.2.5 Test Creation Wizard

The test creator constitutes a pivotal component of the collaborative module, facilitating the intuitive and structured design and creation of student assessments by educators. Access to the creator is granted via the New Test button situated within the Collaboration tab of the ribbon. As depicted in Fig. 5, the test wizard interface is partitioned into three primary sections.

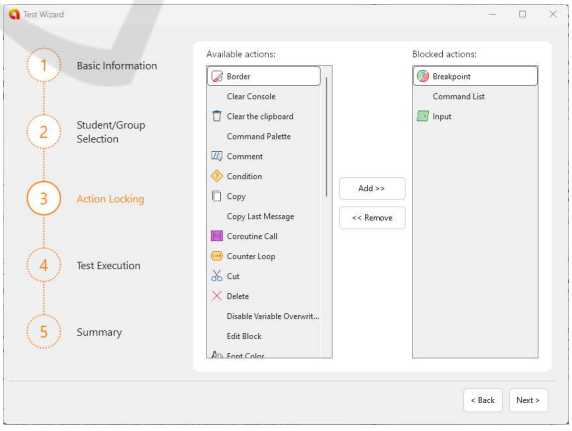

Figure 5: Test Wizard at the stage of blocking selected actions for students.

Positioned on the left-hand side, stepwise progress bar visualizes the stages of the test creation process. Each stage is denoted by a number and a descriptive label, providing indications of the current workflow

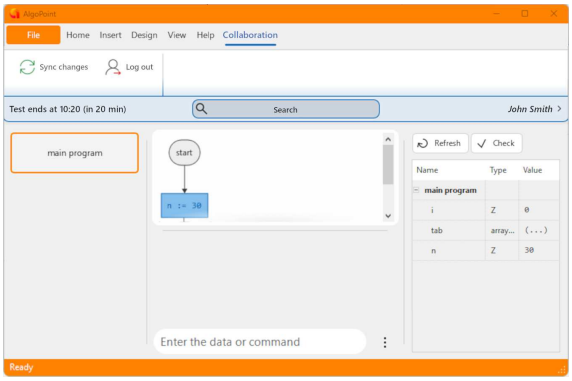

Figure 6: The main window of the application after the student joins the test.

phase. Navigational buttons, Back and Next, are situated in the lower region (right-hand side), allowing for sequential progression through the creation stages. The workspace, occupying the majority of the interface, presents options specific to the current creation phase. This area serves as the platform for data input, decision-making, and configuration of test parameters according to the educator's specifications. Such a covered process for conducting a test can be performed using a simple flowchart as an example. Fig. 7 presents a diagram built for calculating the value of the nth term of Fibonacci sequence. It would need to be further reworked, adding test sections and specifying the appropriate scoring parameters (Fig. 8).

4.3 Student's Application Perspective

As previously outlined, most of the tools within the collaborative module are exclusively accessible to teachers. However, certain changes have been implemented from the student's perspective, particularly during active local network test sessions. The student interface diverges significantly from the standard AlgoPoint view (Fig. 6). Notably, the main File menu, typically located in the upper left corner of the ribbon, is absent. This restriction prevents students from performing unintended actions, such as creating new projects during assessments. Additionally, the Collaboration tab has been modified, featuring a blue color scheme and limited functionalities.

Students are restricted to synchronizing their project status or ending the test by logging out. The quick access toolbar beneath the ribbon displays additional information. A notification on the left side indicates the test's end time for the individual student and the remaining time until completion. During active test sessions, the flowchart is automatically configured according to the test section previously selected by the teacher in the creator.

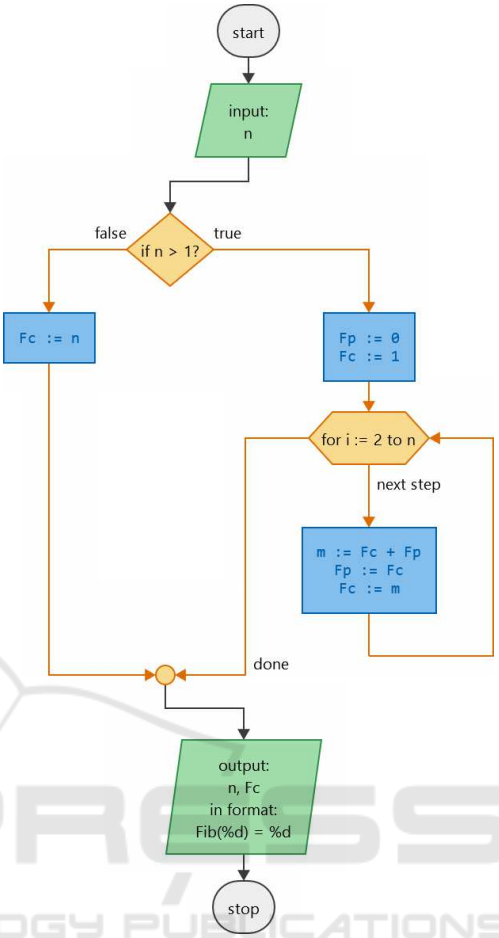

Figure 7: A sample diagram that can be transformed into an interactive test.

5 LIVE MODULE TESTS

5.1 Preparation and Realization of Surveys

To evaluate the functionality and usability of the newly developed teacher-student collaborative module for the AlgoPoint application, a research survey was administered following the implementation phase. We tested the updated software in two phases in Polish schools: June 5, 2024, at the Bolesław Prus High School in Siedlce and June 14, 2024, at the Techni Schools Non-public Programming Technical School in Lublin. Approximately 100 participants (a detailed breakdown is provided in Fig. 9) participated in the study. Initially, teachers were introduced to the software prior to class sessions to familiarize themselves with the new features and create block diagram algorithms for subsequent testing. Teachers designed

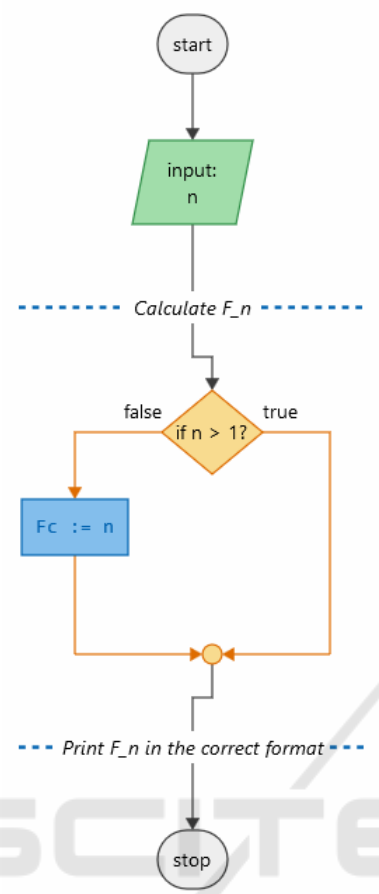

Figure 8: The previous diagram that has been transformed into a test.

block diagrams for tasks such as finding the minimum and maximum values in an array, reversing text, bubble sort, and calculating the greatest common divisor. Subsequently, students were introduced to the application at the beginning of their classes and tasked with implementing specific algorithm segments as defined by their teachers. After a designated period, responses were collected and discussed collectively. Finally, all participants (both teachers and students) completed a survey to gather feedback on the module in question. This comprehensive online survey was administered utilizing Google Forms. It employed a variety of question formats, including scales (1-6), multiple-choice, and open-ended responses, to gather detailed feedback from both teachers and students.

5.2 The Results of Surveys Conducted Among Teachers and Students

Following our 2022 baseline study, a subsequent survey was conducted to evaluate the perceived utility of the revised teacher-student collaborative module. The

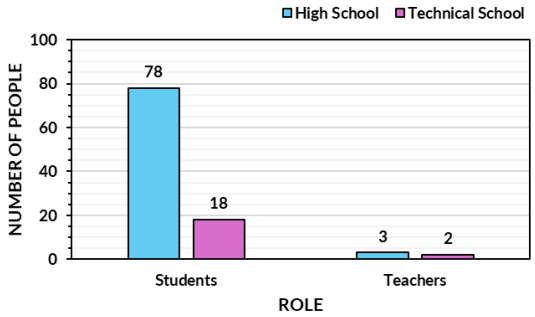

Figure 9: Breakdown of students and teachers by school type.

results of this iterative study revealed a high degree of satisfaction among respondents, with 87% reporting a positive assessment of the module's usefulness. This positive reception underscores the module's potential to enhance collaborative learning experiences. However, it is noteworthy that a significant minority (12%) of participants indicated a lack of sufficient experience to provide a comprehensive evaluation, potentially due to factors such as limited usage or contextual constraints. A negligible portion of respondents (less than 1%) expressed negative sentiments. The results of this study can form the basis for further actions aimed at improving and optimizing the module so that it effectively meets the needs of both teachers and students.

A critical aspect of the collaborative module involved the login process, test-taking experience, and the clarity of the refreshed graphical user interface. These elements were evaluated using a six-point scale. The findings revealed a generally positive perception of these aspects among all participants (Fig. 10). However, a closer analysis of the data revealed a more nuanced picture. While both high school and vocational school students expressed satisfaction, the distribution of responses differed significantly. The vocational school students, demonstrated a higher degree of consensus, with no respondents providing a rating below four. In contrast, high school students exhibited a wider range of opinions.

To understand the students' experiences during the testing phase, they were asked to identify the most frequently performed actions. The results indicate that high school and vocational school students commonly engaged in editing block contents, managing functions, variables, and constants, and visualizing algorithm execution. These actions, particularly the first and last, are crucial for effective module usage, validating their selection by students. Less frequently used features included accessing the context menu and modifying the block diagram's appearance, which aligns with design expectations. While

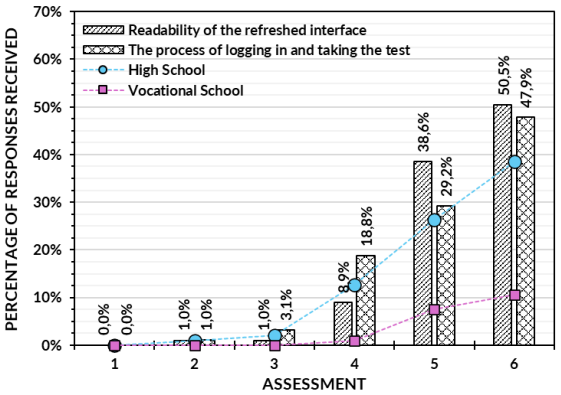

Figure 10: Survey respondents' assessment of the readability of the new interface and the process of of logging in and taking the test (indicating the percentage of each school type).

using the command palette and synchronizing project changes were moderately popular, the former could potentially benefit from increased usage time, and the latter, occurring automatically, was appropriately used with moderation.

Given their broader role in coordinating user accounts and tests, teachers were provided with a more detailed questionnaire to assess their perceptions of the module's functionalities. The results indicated an overwhelmingly positive evaluation of all features, with no ratings below four and a majority of teachers assigned the highest possible score. The command palette was consistently identified as the most valuable feature, highlighting its utility for efficient navigation. Although students demonstrated a lower frequency of use for this feature, teachers, having had more time to explore the application, expressed a significantly higher appreciation for its functionality. Although the overall feedback from the teachers was positive, identifying specific areas for improvement proved challenging. The management of users, groups, and ongoing tests emerged as potential areas where further development could enhance the user experience. Future iterations of the module could incorporate a more specialized dialog box for creating and modifying user data, providing a more detailed and consistent interface. Furthermore, expanding the list of active users during a test to include additional information, such as names, status, and technical indicators, could provide teachers with a more comprehensive overview of test progress.

Complementing the evaluation of specific features, participants were also asked to assess the overall stability of the application (Fig. 11). The findings revealed a high degree of satisfaction, with a substantial majority (80.2%) reporting a positive user experience. A smaller proportion (19.8%) encountered minor issues, primarily related to saving settings. To address this, the relocation of the configuration file could be considered. Notably, nearly half of the participants had prior experience with the application, providing valuable insights into the perceived improvements. These users consistently praised the enhanced performance, intuitive interface, and search functionality of the new version. The feedback also highlighted potential areas for future development, including automatic updates and versions for Linux and macOS.

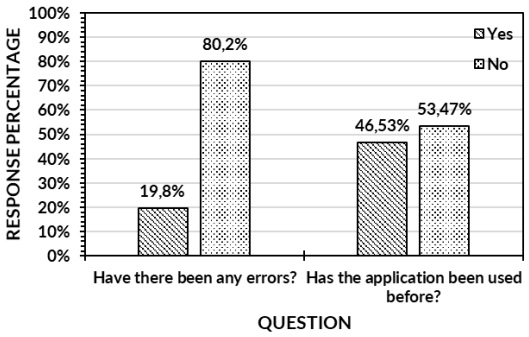

Figure 11: Respondents' answers regarding possible errors of the application and its previous use.

5.3 Afterthoughts on the Outcome of the Questionnaires

Consistent with the findings of a previous study, a significant majority of both students and teachers expressed satisfaction with the new collaborative module. Teachers, in particular, recognized its potential to extend beyond block diagram creation and serve as a comprehensive assessment tool. Features such as the command palette, testing sections, and user management were highly regarded. Although minor issues were identified, a robust update framework is essential for their systematic resolution. Given the module's success, future developments could include a platform for user-generated extensions, fostering a community around the application and potentially increasing its adoption.

6 CONCLUSIONS

The objectives of this study were achieved successfully. The group collaboration module was successfully implemented and presented, and underwent a series of tests in a real-world school environment. Despite occasional minor bugs, the new version of the application performs well. Graph database implementation within the teacher-student collabora-

tion module offers key advantages: high flexibility, enabling facile object and relationship modification without structural disruption, crucial for AlgoPoint's modular expansion; scalability, effectively managing extensive interconnected data, thus accommodating numerous users on a single workstation; and efficient searching, facilitated by the graph structure's accurate representation of complex inter-object relationships. Furthermore, porting the application to platforms such as Linux would enable it to reach a wider audience.

REFERENCES

Bione, J., Miceli, P., Sanz, C. V., and Artola, V. (2017). Astrocode: a serious game for the development of computational thinking skills. In *9th Int. Conf. on Education and New Learning Technologies (Barcelona)*.

Choi, W. C., Choi, W. C., and Choi, I. C. (2024). Investigating the effect of the serious game codecombat on cognitive load in python programming education. In *2024 IEEE World Engineering Education Conference (EDUNINE)*, pages 1–6.

Cook, D. D. et al. (2015). Flowgorithm: Principles for teaching introductory programming using flowcharts. In *Proc. American Society of Engineering Education Pacific Southwest Conf.(ASEE/PSW)*, pages 158–167.

Dietrich, J. W., Siegmar, N., Hojjati, J. R., Gardt, O., and Boehm, B. O. (2024). Cyberunits bricks: An implementation study of a class library for simulating nonlinear biological feedback loops. *ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal*, 13:e31762–e31762.

Edgcomb, A. D., Vahid, F., and Lysecky, R. (2019). Coral: An ultra-simple language for learning to program. In *2019 ASEE Annual Conference & Exposition*.

Elsawah, W. and Thabet, R. A. (2023). The effectiveness of tynker platform in helping early ages students to acquire the coding skills necessary for 21st century. In *International Conference on Information Systems and Intelligent Applications*, pages 381–397, Cham. Springer International Publishing.

Henderson, P. B., Cortina, T. J., Hazzan, O., and Wing, J. M. (2007). Computational thinking. In *Proceedings of the 38th SIGCSE technical symposium on Computer science education*, pages 195–196.

Kaya, K. Y. and Yildiz, İ. (2019). Comparing three free to use visual programming environments for novice programmers. *Kastamonu Eğitim Dergisi*, 27(6):2701–2712.

Koracharkornradt, C. (2017). Tuk tuk: A block-based programming game. In *Proceedings of the 2017 Conf. on Interaction Design and Children*, pages 725–728.

Kourouma, M. K. (2016). Capabilities and features of raptor, visual logic, and flowgorithm for program logic and design.

Kraleva, R., Kralev, V., and Kostadinova, D. (2019). A methodology for the analysis of block-based programming languages appropriate for children. *Journal of Computing Science and Engineering*, 13(1):1–10.

Mazumdar, S., Das, S., Naskar, S., Chowdhury, S., Haldar, D., Bhattacharjee, A., and Das, A. (2024). Design and development of real-time code editor for collaborative programming. *IARJSET*, 11:340–349.

Meerbaum-Salant, O., Armoni, M., and Ben-Ari, M. (2010). Learning computer science concepts with scratch. In *Proceedings of the Sixth ACM int. workshop on Computing Education research*, pages 69–76.

Rababaah, A. R. (2020). A new simple programming language for education. In *2020 15th International Conference on Computer Science & Education (ICCSE)*, pages 145–149. IEEE.

Romeo, M., Lepage, A., and Lille, B. (2017). Computational thinking development through creative programming in higher education. *Int. Journal of Educational Technology in Higher Education*, 14(1).

Rugelj, J. and Lapina, M. (2019). Game design based learning of programming. *Proc. Int. SLET, CEUR Workshop*.

Stefanowicz, M. and Sasak-Okoń, A. (2023). Algopoint as an original didactic tool for introductory programming using flowcharts. In *Proceedings of the 15th International Conference on Computer Supported Education - Volume 1: CSEDU*, pages 162–170. INSTICC, SciTePress.

Strout, J. (2021). Miniscript: A new language for computer programming education. In *2021 6th Int. STEM Education Conference (iSTEM-Ed)*, pages 1–4. IEEE.

Tsay-Vogel, M. and Brady, M. (2010). A case study of cooperative learning and communication pedagogy: Does working in teams make a difference? *Journal of the Scholarship of Teaching and Learning*, 10:78–89.

Valsamakis, Y., Savidis, A., Agapakis, E., and Katsarakis, A. (2020). Collaborative visual programming workspace for blockly. In *2020 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 1–6.

Weintrop, D. (2021). The role of block-based programming in computer science education. *Understanding computing education*, 1:71–78.

Werner, L., Campe, S., and Denner, J. (2012). Children learning computer science concepts via alice game-programming. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education*, pages 427–432.

Yadav, A., Zhou, N., Mayfield, C., Hambrusch, S., and Korb, J. T. (2011). Introducing computational thinking in educational courses. In *Proceedings of the 42nd ACM technical symposium on Computer science education*, pages 465–470.

Öztürk, B. (2023). The effect of cooperative learning models on learning outcomes: A second-order meta-analysis. *Educational Policy Analysis and Strategic Research*, 18:273–296.