NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

RUI JORGE RODRIGUES CORREIA

Bachelor in Computer Science and Engineering

NEXTBLOCKS 2

ENHANCING BLOCK PROGRAMMING WITH INTERACTIVE INPUTS AND CUSTOMIZABLE LEARNING TOOLS

MASTER IN COMPUTER SCIENCE AND ENGINEERING

NOVA University Lisbon
September, 2025

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

DEPARTMENT OF
COMPUTER SCIENCE

NEXTBLOCKS 2

ENHANCING BLOCK PROGRAMMING WITH INTERACTIVE INPUTS AND CUSTOMIZABLE LEARNING TOOLS

RUI JORGE RODRIGUES CORREIA

Bachelor in Computer Science and Engineering

Adviser: Fernanda Barbosa
Assistant Professor, NOVA University Lisbon

Co-adviser: Carmen Morgado
Assistant Professor, NOVA University Lisbon

MASTER IN COMPUTER SCIENCE AND ENGINEERING

NOVA University Lisbon
September, 2025

NextBlocks 2
ENHANCING BLOCK PROGRAMMING WITH INTERACTIVE INPUTS AND CUSTOMIZABLE LEARNING TOOLS

This document was created with the (pdf/Xe/Lua)LaTeX processor and the NOVAthesis template (v7.1.29) [1].

Acknowledgements

First and foremost, I extend my sincerest thanks to my supervisors, Professor Fernanda Barbosa and Professor Carmen Morgado. I am grateful for their guidance, immense patience and invaluable insights throughout this journey. Their expertise and critical feedback were crucial in shaping the direction of this work and challenging me to achieve my best. Their encouragement during the many months was a constant source of motivation.

I would like to acknowledge the NOVA University of Lisbon, specifically through the School of Science and Technology, and the NOVA LINCS research laboratory for providing the academic environment and resources necessary for this research. My gratitude also extends to all the Professors from the Department of Computer Science whose knowledge and dedication during these last 5 years laid a strong foundation for this project.

A special word of thanks goes to my colleagues. The stimulating discussions, shared challenges and spirit of collaboration made this journey far more enriching. Whether it was brainstorming solutions to complex problems or simply offering a supportive word, their camaraderie was invaluable.

Finally, I am immensely thankful to the teachers and students who generously participated in the user testing of NextBlocks. Their time, willingness to engage and constructive feedback were crucial for the evaluation and refinement of this work. This thesis would be incomplete without their essential contributions.

”

“Tell me and I forget.
Teach me and I remember.
Involve me and I learn.”

— Benjamin Franklin
(Writer, scientist, philosopher)

Abstract

Computer science education plays a key role in shaping the future of individuals and societies in an increasingly digital world. As the demand for digital skills continues to grow, investing in computer science education is essential for building an equitable, innovative and technologically advanced society. Programming is a powerful tool for creativity, problem-solving and innovation, shaping the way people interact with technology. It enables individuals to break down complex problems in order to design efficient solutions.

Block-based programming has become a cornerstone of modern computer science education, offering an accessible and engaging introduction to coding for learners of all ages. By using visual interfaces and drag-and-drop blocks, these platforms lower barriers to entry, enabling beginners to grasp fundamental programming concepts without the cognitive overload of syntax errors or complex text-based coding environments. As the demand for skilled programmers continues to grow, block programming remains an indispensable part of cultivating the next generation of computational thinkers and innovators. Platforms like Scratch and Code.org have further amplified the role of block programming, making coding education more engaging and accessible than ever before. However, these platforms lack some collaboration and feedback tools, are not implemented in a commonly used platform and do not allow for full customizability.

This thesis presents the development of NextBlocks 2, a new version of the NextBlocks plugin that aims to fill these gaps by providing a block programming environment which lets teachers guide students along a path they find appropriate. This is achieved through highly customizable blocks and exercises, integrated feedback and collaboration systems and being implemented into Moodle, the most commonly used learning platform in the world. This development consisted of the introduction of innovative features such as the ability to disallow specific blocks per exercise, an interactive input system and different languages for text code translation. A key outcome of this work was the submission and acceptance of NextBlocks into the official Moodle Plugin Directory, making it publicly available for educators and institutions worldwide.

Keywords: Block Programming, Moodle, NextBlocks, Blockly, Visual Programming

Resumo

A educação em ciência da computação desempenha um papel fundamental na formação de indivíduos e sociedades num mundo cada vez mais digital. À medida que a procura por competências digitais continua a crescer, investir na educação em ciência da computação é essencial para construir uma sociedade equitativa, inovadora e tecnologicamente avançada. A programação é uma ferramenta poderosa para a criatividade, resolução de problemas e inovação, moldando a forma como as pessoas interagem com a tecnologia e permitindo que indivíduos decomponham problemas complexos para projetar soluções eficientes.

A programação baseada em blocos tornou-se um pilar da educação moderna em ciência da computação, oferecendo uma introdução acessível e envolvente à programação para qualquer aluno. Através da utilização de interfaces visuais, estas plataformas reduzem as barreiras de entrada, permitindo que iniciantes compreendam conceitos fundamentais da programação sem o peso cognitivo dos erros de sintaxe. Plataformas como o Scratch e o Code.org reforçaram ainda mais o papel da programação por blocos, tornando a educação em programação mais envolvente e acessível do que nunca. No entanto, estas plataformas carecem de algumas ferramentas de colaboração e *feedback*, não estão implementadas numa plataforma amplamente utilizada e não permitem total personalização.

Esta dissertação apresenta o desenvolvimento do NextBlocks 2, uma nova versão do plugin NextBlocks que visa colmatar estas lacunas, fornecendo um ambiente de programação por blocos que permite aos professores orientar os alunos por um percurso que considerem adequado. Tal é conseguido através de exercícios altamente personalizáveis, sistemas de *feedback* e colaboração integrados e estar implementado no Moodle, a plataforma de aprendizagem mais utilizada no mundo. Este desenvolvimento consistiu na introdução de funcionalidades inovadoras, tais como a possibilidade de restringir blocos específicos por exercício e um sistema de entrada interativo. Um resultado fundamental deste trabalho foi a aceitação do NextBlocks no Diretório Oficial de Plugins do Moodle, tornando-o publicamente disponível para educadores em todo o mundo.

Palavras-chave: Programação por Blocos, Moodle, NextBlocks, Blockly, Programação Visual

Contents

List of Figures

1 Introduction

In an increasingly digital world, where technology shapes nearly every aspect of our lives, having an easy access to the digital world is more important than ever. Having the knowledge and skills to think computationally not only offers many diverse job opportunities but also helps the development of critical and logical thinking, which is crucial to perform in many everyday tasks.

1.1 Motivation and Context

Programming is seen as one of the best ways to develop Computational Thinking (CT) [2], which is a crucial skill for everyday tasks. According to [2], "The educational benefits of being able to think computationally transfer to any domain by enhancing and reinforcing intellectual skills". CT helps individuals break problems down into smaller parts to then come up with step-by-step solutions which is useful not only in computer science but also in many other fields, such as engineering, finance or biology since all of these may involve computational approaches.

Learning programming has become an essential part of modern education, as staying up to date with technological advancements is crucial in today's fast-evolving world. As technology continues to shape industries and everyday life, equipping students with programming skills prepares them to be successful in a tech-driven future [3]. However, teaching programming effectively remains a challenge. Many different methods have been tried over the years with varying results, but many of these ended up facing complaints of being too difficult and counter-intuitive to use without the aid of professional tutors [4]. With this in mind the field on Visual Programming emerged, which allows for students to see what they are creating in a visual and interactive way instead of just lines of code. Visual Programming can take many shapes but the method that is focused on this thesis is Block Programming [5].

Block Programming, used in websites such as Scratch [1] or Code.org[2], allows for the

[1] Scratch, https://scratch.mit.edu/
[2] Code.org, https://code.org/

creation of diverse and intuitive coding challenges with different values of difficulty using visual blocks that can easily be understood by inexperienced users. There are previous studies[6][7] that suggest that Block Programming can be used to help children develop computational thinking in primary school environments. Although Block Programming appears to be an effective approach to teaching programming, there is a lack of environments integrated in Learning Management Systems (LMS) that work as a learning tool for the world of programming. Furthermore, the currently available environments do not provide teachers with the flexibility to create custom exercises, limiting their ability to adapt content to the specific needs and skill levels of their students.

NextBlocks is a programming environment that was developed in a previous NOVA SST[8] thesis[9] where students can create programs using visual and interactive blocks.

NextBlocks was developed as an activity plugin integrated in Moodle [10], an LMS used worldwide for different educational purposes, making it more accessible for students and teachers. Blocks can be combined in order to create functional programs which are visually appealing and easy to understand. To develop a program, many different blocks are available to be used which do different functionalities, included in topics like logic, math, text, lists, variables and functions. NextBlocks also allows for automated testing using predefined input files and for teachers to create custom activities for their students, including customized inputs and their respective expected outputs, custom blocks, chat and reaction systems to get feedback on the exercise and comments to ask questions and send feedback. According to the evaluation performed in [9], both students and educators are satisfied with NextBlocks and showed interest in using the platform.

This thesis focuses on developing a new version of NextBlocks, introducing innovative features such as an interactive input system and a setting for teachers to limit how many instances of each block type are available to use in an exercise, and culminating in its successful submission to Moodle's plugin directory[11]. This update aims to improve the usability of the plugin for both students and teachers and improve its worldwide access, positioning NextBlocks as a practical and effective choice for educators seeking an ideal environment to teach programming.

1.2 Objective

The goal is to enhance the development of NextBlocks by implementing features designed to improve the user experience for both teachers and students, and finally submit the plugin to the directory for publication.
Major features:

- **Revamp of the input/output system**: This update implements an interactive terminal that lets the user input any values asked by the system during execution time, allowing for interactive programs. Previously the plugin's IO system required the user to input any value into a small square on one of the blocks or in pre-defined

tests before the program is run, making it so that there was no way to interact with the program while it is running.

- **Different languages for code samples**: The objective is to add other languages to the code samples that translate the block program into programming code, such as Python, in addition to the existing JavaScript.

- **Limit the number of blocks used**: The intention is to create a new menu when creating an exercise where the teachers can limit how many of each type of block a student can use in a specific exercise since teachers might want students to not use a specific block, or to only use a limited amount of that block.

Minor features:

- Rework the chat system to remove the requirement of an external running server.

- Improve the comments system by allowing users to add answers to previously created comments in blocks.

- Move the automatic submission evaluation to an external API.

1.3 Methodological Approach

This thesis follows a structured development and evaluation methodology, organized into three main phases:

1. **Proposal and Design**: Based on an analysis of existing block programming environments and the limitations identified in the original NextBlocks plugin, a set of improved features was proposed. These include an interactive input/output system, block usage limits and translation to Python.

2. **Implementation and Integration**: The proposed features were developed using an iterative development process. The implementation adhered to Moodle's plugin development standards, including the integration of Privacy, Backup/Restore and Language APIs to ensure compatibility and readiness for public distribution.

3. **Evaluation and Validation**: The platform was assessed through structured user testing sessions with teachers and students, employing both quantitative measures (System Usability Scale, Likert-scale surveys) and qualitative feedback. The evaluation aimed to validate usability, pedagogical usefulness and technical robustness. The work was further validated through presentation and discussion at the EDULEARN25 international education conference, and culminated in the plugin's submission and acceptance into the official Moodle Plugin Directory.

This methodology is guided by Design Science Research (DSR) principles, in which the NextBlocks 2 plugin is the design artifact created to address a recognized educational problem. The process includes problem identification, artifact development based on existing knowledge and rigorous evaluation through user testing and public presentation, therefore contributing a practical solution to the field of educational technology.

This methodology ensured a user-centered and iterative development process, where feedback from target users guided refinements and validated the practical relevance of the implemented features.

1.4 Document Organization

This document is organized under the following structure:

- Chapter 2 - Background Knowledge: Explains specific platforms and concepts needed to understand the proposal, including the Moodle platform, Block Programming and the NextBlocks environment.
- Chapter 3 - Related work: Analyses previously created environments that are similar to NextBlocks in functionality and objective, indicating their similarities and differences.
- Chapter 4 - Proposal: Presents the new and updated features to be developed during this thesis
- Chapter 5 - Implementation: Details how the implementation of the new and updated features was made.
- Chapter 6 - Evaluation: Explains the evaluation process with students and educators and its results.
- Chapter 7 - Conclusions and Future Work: Summarizes the results of the project and suggests future changes to the environment.

2

Background

This chapter will present the background knowledge required to understand and comprehend the proposal. It will show relevant concepts related to Moodle, Block Programming and the NextBlocks environment.

2.1 Moodle

Moodle (Modular Object-Oriented Dynamic Learning Environment)[10] is a robust, open-source learning management system (LMS) designed to provide educators, administrators, and learners with a single secure and integrated platform to create personalized learning environments.

2.1.1 Course Management

Moodle facilitates the creation, organization and delivery of courses for teachers. Courses can be structured into sections or topics, enabling a clear and logical progression through the material. Moodle offers various tools and configurations to help educators create courses tailored to specific subjects or objectives:

- **Course formats**[1]**:** Moodle supports multiple course formats such as weekly format (each week a new activity becomes available), topic format (the course's activities are grouped by topic), single activity and social format (centered around a discussion forum for collaborative or informal learning).
- **Hierarchical Organization**[2]**:** Courses can be broken into sections or modules, which may contain resources (documents, videos, links) and activities (assignments, quizzes, forums).
- **Templates and Duplication**[3]**:** Educators can reuse existing course templates or duplicate course content, saving time when creating similar courses.

[1] Moodle course formats, https://docs.moodle.org/405/en/Course_formats/
[2] Moodle course sections, https://docs.moodle.org/20/en/Course_sections
[3] Moodle templates, https://docs.moodle.org/405/en/MoodleDocs:Templates

2.1.2 User roles

Moodle allows control over what users can do within a course through the use of roles[4]:

- **Teacher Roles:** Full control over course creation, resource management and grading.
- **Non-Editing Teachers:** Can teach and grade but cannot modify course content.
- **Students:** Access to participate in activities and view resources.
- **Custom Roles:** Institutions can create tailored roles with specific permissions, such as "Tutor" or "Observer".

2.1.3 Activities and resources

Moodle provides a wide range of activities and resources designed to facilitate teaching and learning. These tools are highly customizable, enabling educators to create dynamic and engaging online courses tailored to the needs of their students. Below is an overview of the key activities and resources offered by Moodle:

Moodle Activities[5] are interactive elements that encourage student participation, collaboration, and assessment. Some of the most commonly used activities include:

- **Assignments** allow teachers to collect work from students, review it, and provide feedback. They support various submission formats, including text, files, or links, and can integrate with plagiarism detection tools.
- **Quizzes**: The quiz module enables educators to design and administer quizzes with multiple question types, such as multiple-choice, true/false, short answer and matching. Quizzes can be timed, graded automatically, and configured for multiple attempts.
- **Forums** facilitate online discussions, promoting interaction and collaboration among students and teachers. They can be used for Q&A, peer reviews, or topic debates.
- **Chat** The chat activity allows real-time communication, making it ideal for virtual office hours, group discussions, or live Q&A sessions.
- **Workshops** are peer-assessment activities where students evaluate and provide feedback on each other's work according to pre-defined criteria.
- **Lessons** offer a structured way to present content interactively, guiding students through a series of pages with questions and branching based on their responses.

[4]Moodle roles, https://docs.moodle.org/405/en/Roles_and_permissions
[5]Moodle activities, https://docs.moodle.org/405/en/Activities

- **Glossary** enables the creation of a collaborative or instructor-managed dictionary of terms, enhancing knowledge sharing and study support.
- **Wikis** allow students to collaboratively create and edit content within a course, fostering teamwork and shared learning experiences.

Moodle Resources[6] are non-interactive elements used to deliver information to students. Some of the key resources include:

- **Files**: Teachers can upload documents, presentations, spreadsheets, or other files for students to download and review.
- **Pages** allow instructors to create custom web pages directly within Moodle, embedding text, multimedia, and other elements.
- **URLs** This resource links to external websites, videos, or other online content, providing additional learning materials and references.
- **Books**: The book resource organizes content into a multi-page format, making it easy to present extensive material in a structured manner.
- **Labels** are used to insert text, images, or multimedia directly onto the course homepage, helping to organize content or provide quick instructions.

Moodle's activities and resources can be combined and tailored to suit the unique requirements of each course. Additionally, many of these tools integrate with third-party services and plugins, further expanding their functionality. By leveraging these tools effectively, educators can create an engaging and interactive learning environment that meets diverse educational objectives.

These activities and resources reflect Moodle's commitment to providing flexible and accessible solutions for online learning. Their versatility supports the delivery of content, assessment of learning and facilitation of collaboration, making Moodle a powerful platform for modern education.

Moodle provides built-in tools for tracking learner progress, engagement and performance. Professors can check the progress of all students in a single page, like shown in Figure 2.1. Students can also check their own report as shown in Figure 2.2.

2.1.4 Plugins

Moodle's plugin system[7] allows developers and educators to customize and enhance the platform to meet diverse educational needs. Plugins in Moodle are modular components that extend the core functionality of the system by adding new features, modifying existing ones, or integrating third-party services.

[6] Moodle resources, https://docs.moodle.org/405/en/Resources
[7] Moodle plugins, https://docs.moodle.org/405/en/Installing_plugins

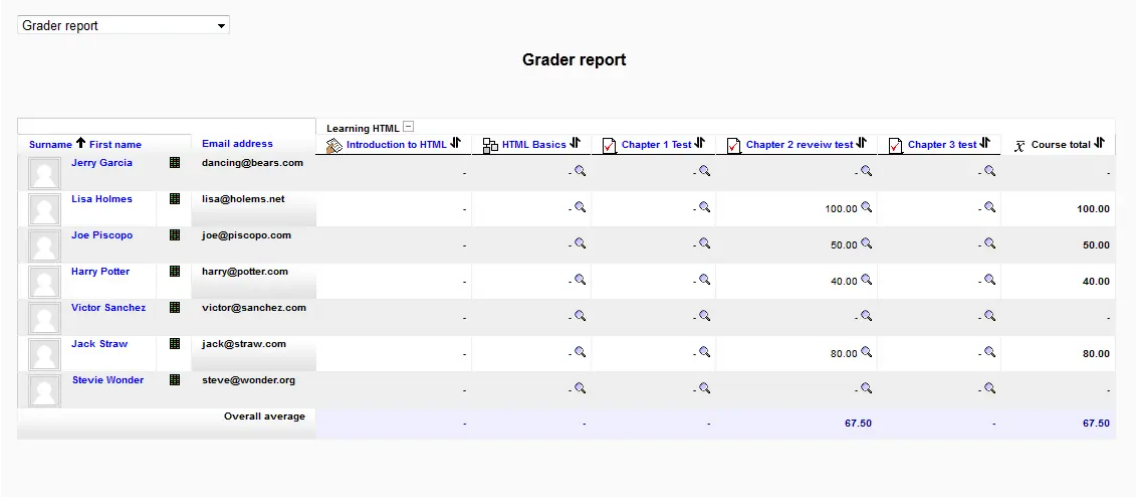


Figure 2.1: Moodle Global Report [12]

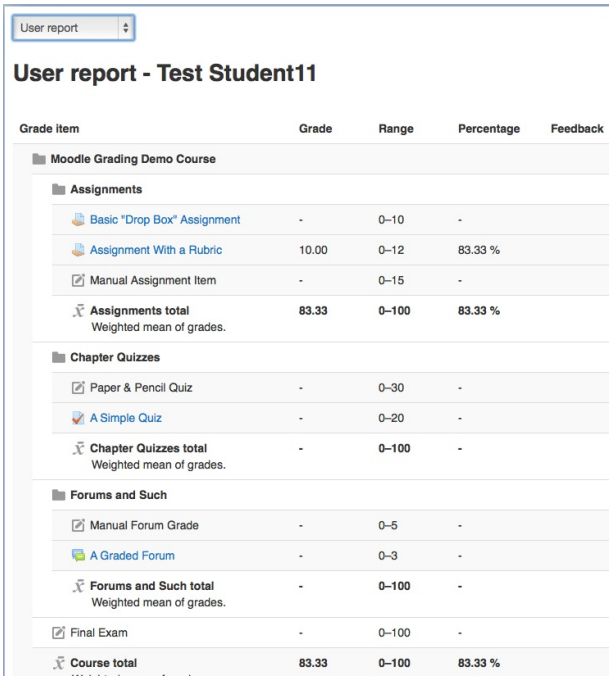


Figure 2.2: Moodle Personal Report [13]

Moodle supports a wide variety of plugin types, each designed for a specific purpose within the platform. Some of the most common types include:

- **Activity Modules** add new types of activities that can be used in courses, such as quizzes, forums, or wikis. Custom activity plugins allow developers to create tailored learning interactions for specific use cases. NextBlocks[9] is implemented as an Activity Module.
- **Blocks** are small, modular elements that appear on the side of course pages or dashboards. They provide supplementary information or tools, such as calendars, recent activity logs, or quick access links. Advanced Notifications[14] is a plugin

implemented as a Blocks module where teachers can send notifications to students in their courses.

- **Themes** control the visual appearance of the Moodle interface, allowing institutions to customize the design, layout, and branding to align with their identity. Academi[15] is one of these plugins.

- **Quiz**: Plugins in this category add new question types to Moodle's quiz module, such as drag-and-drop matching, essay questions, or specialized formats for mathematical equations. CodeRunner[16] is a coding environment which is implemented as a Quiz module.

- **Reports**: Reporting plugins generate detailed analytics and insights about user activity, course progress, or system performance, helping educators and administrators make data-driven decisions. Course Size[17] is a Reports plugin which shows administrators how much disk space a course is taking.

- **Repositories and Portfolios**: Repository plugins enable integration with external file storage services, such as Google Drive or Dropbox, for easier file management. Portfolio plugins allow users to export work or course content to external portfolio systems. Repository GitHub[18] is a plugin implemented as a Repository module which allows Moodle integration with GitHub repositories.

The Moodle plugin system is built using PHP and adheres to Moodle's modular architecture and coding guidelines. Developers can create plugins using Moodle's APIs and development tools, which ensure compatibility with the core platform. Moodle provides comprehensive documentation and a supportive developer community to assist in plugin creation and troubleshooting.

Plugins can be developed to address specific institutional needs, such as integrating external systems, adding unique activity types or automating administrative tasks. Once developed, plugins can be tested locally and shared with the broader Moodle community via the Moodle Plugin Directory[11].

The Moodle Plugin Directory[11] is a centralized repository where developers can upload and share their plugins with the global Moodle community. Educators and administrators can browse and download plugins to enhance their Moodle installations. Each plugin in the directory is reviewed for quality and compatibility before being made publicly available, ensuring reliability and security.

2.2 Block Programming

Block Programming [5] is a Visual Programming approach where users create programs by stacking and connecting blocks that represent instructions.
Visual Programming [19] is a coding style that allows for users to create functionalities

using graphical elements present in an interface rather than writing the code manually. Programming visually makes it possible to directly manipulate programming elements, making it feel like the user is building a puzzle. To create a functioning program, the user needs to link the different pieces in such a way that the expected output is produced.
While this method can have some downsides, such as sometimes limiting the complexity of the graphical elements, it has a lot of advantages, such as accessibility to new programmers, since it's easier to interact with graphical elements than with text, and visualization that helps understanding the program flow and how different components interact. There are previous studies [19] that suggest that visual programming can be a helpful tool to teach computational thinking and programming principles to young children in a helpful and enjoyable way.
In Block Programming these graphical elements are blocks. Blocks can have different functionalities and are usually color coded based on their functionality (for example, logic blocks could be green while math blocks would be orange). Blocks are grouped in categories such as:

- **Control blocks**: Conditional statements and loops.
- **Event blocks**: Buttons and timers.
- **Data Blocks**: Variables and lists.
- **Operators**: Math operators and logic operators.
- **Custom blocks**: Users can define specific reusable functions.

Each block can interact with other blocks through the use of connectors (Figure 2.3). Each block can have one or more connectors which are designed so that they can only be connected to blocks that would make the program syntactically correct, avoiding unexpected syntax errors.

Figure 2.3: Block Connectors [20]

An example of a block programming program is shown in Figure 2.4. This program will go through every number between 0 and 9 and, if the number is odd print it, else if the number is even print its square.

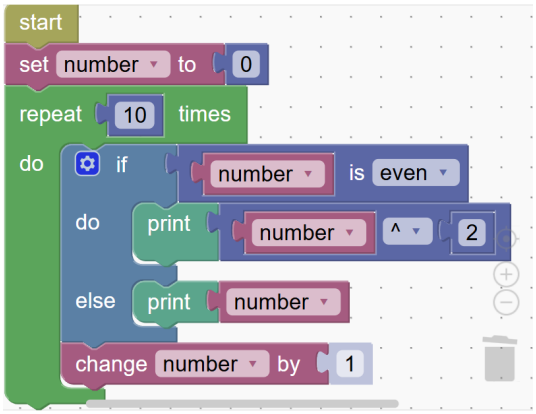
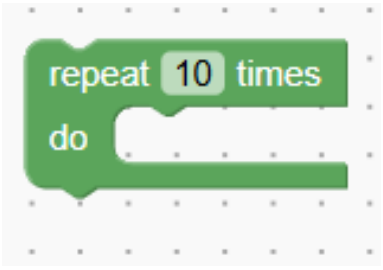


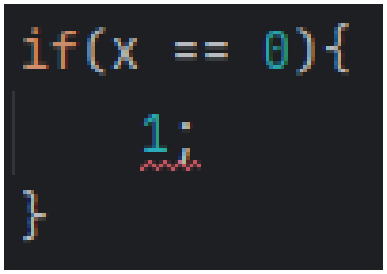
Figure 2.4: Block programming[5] program example

(a) Java (15 items)

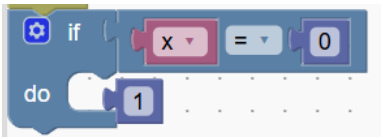


(b) Block Programming[5] (2 items)

Figure 2.5: For loop in text language and block programming



(a) Java



(b) Block Programming

Figure 2.6: Syntax Error vs Error Prevention

Block programming is often used for educational purposes [5] as it simplifies coding concepts and helps beginners understand programming logic in a more visual and intuitive way, without having to worry about syntax. As an example, a "for" loop that takes 15 items in Java can only take 2 in Block Programming, as shown in Figure 2.5

In Figure 2.6 it is visible that Java allows code writing with syntax errors, only showing the errors afterwards. In contrast, block programming does not allow the connection of blocks that would create syntax errors.

However, block programming often struggles with scalability since graphical environments can very easily get cluttered and usually does not allow for complex programming functionalities, such as multithreading or memory management.

In most block programming environments, the blocks are organized in a toolbox (Figure 2.7), which is an interface with every available block the user can use. To create the program the users simply need to drag the blocks from the toolbox into the workspace and connect them. The environment will then translate the graphical program into code for the users to be able to run it and see its output.

Validation in block programming is the process of ensuring that a program constructed with blocks meet predefined requirements and behave as intended. Given that block programming environments are often used in educational settings to teach beginners, validation plays a crucial role in providing clear and constructive feedback. Common methods of validation in block programming include:

- **Output Validation:** Compare the program output with a pre-defined expected output. This method verifies if the program is working as intended and it is typically done with multiple test cases. This method is used in coding platforms such as Code.org[7] and Tynker[21].
- **Syntax Validation:** Some environments, such as Scratch[6], design blocks so that they can only connect with other blocks in ways that make syntactic sense.
- **Runtime Validation:** Environments like Code.org[7] highlight what part of the code is being executed in real-time, allowing users to pinpoint where runtime issues might occur, such as infinite loops or impossible mathematical operations.
- **Path Finding Validation:** In some maze related tasks, programming platforms evaluate programs by checking whether the program fulfills the proposed task, which might include reaching the end point from a starting position, finding the smallest path or reaching multiple goal positions. Platforms that use this method include Code.org[7] and Blockly Games[22].

These environments typically use Blockly[23], an API developed by Google made for developers who want to create their own block programming environments based on Google's Blockly Library, a block programming web-based environment. This API includes functionalities such as custom block creation, code generation and workspace management.

2.3 NextBlocks

NextBlocks [9] is a block programming environment (Figure 2.8) integrated as a Moodle[10] activity, although it was not yet available in the official Moodle plugin directory[11]. It was designed to make programming more visual and intuitive and offers several innovative

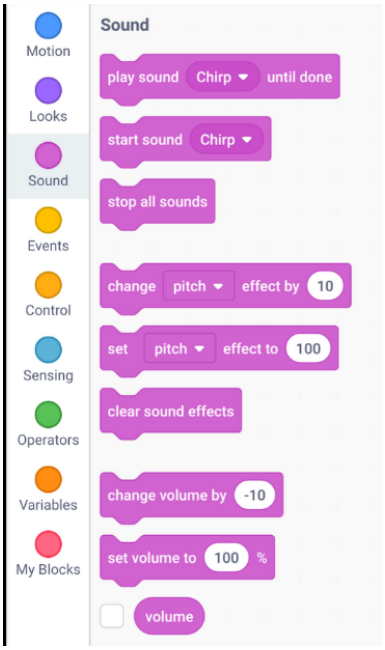


Figure 2.7: Block Programming Toolbox [24]

advantages that make it a powerful tool for programming education and distinguish it from other block programming environments.

One of its key strengths is customizability. Teachers can design custom exercises and custom blocks adapted to their course based on what the students need.

NextBlocks is implemented using the Blockly [8] API [23], which is a flexible library that developers can use to integrate block programming into applications.

Another significant advantage of NextBlocks is feedback and collaboration, since students and teachers can interact using the chat, reaction and comments systems, providing an interactive way for students to help each other and to give feedback to the teacher about the exercise. During the learning process, interaction between teachers and students is fundamental. Without this possibility students might not know what to do, or how to do it. The teacher's guidance is crucial for a positive learning experience. Most of the block programming environments used commonly lack this aspect, which makes learning way more difficult than it needs to be. In other fields of learning this interaction is very common. There are multiple studies [25] [26] that suggest that the student-teacher interaction is beneficial and sometimes even necessary for students' success.

NextBlocks is an activity plugin integrated in Moodle, a widely spread Learning Management System (LMS), making it more accessible and taking advantage of Moodle's features. Being integrated as a Moodle activity also facilitates evaluation, since it can automatically generate feedback and grade reports for each student based on whether they succeeded in each activity, allowing for faster grading for students and time saving for teachers.

[8] Blockly, https://developers.google.com/blockly

The teacher interface in NextBlocks allows educators to create exercises with tools for customization and evaluation. Key features include:

- **Exercise Creation:** Teachers can use default Moodle exercise customization tools such as changing name and exercise description, adding automated tests, selecting evaluation methods and defining time limits for the realization of the exercise. NextBlocks also allows teachers to create custom blocks during exercise creation which can then be used by any student attempting that exercise.
- **Automated Tests:** Teachers can upload test files defining input-output cases for automated grading. These files include test cases separated by specific delimiters, which are used to compare expected and actual program outputs.
- **Custom Blocks:** Teachers can design custom blocks using Blockly's Block Factory, adding flexibility to the exercise.

The student functionalities are implemented into a block programming environment integrated in Moodle and consist of six main areas and three additional buttons:

- **Workspace:** The central area where programs are built by connecting blocks to a pre-placed start block. Every available block is shown on a side bar and can be dragged to the main work area to be used in the program.
- **Reaction Bar:** Provides clickable feedback options for students to share their experience with an exercise.
- **Output Terminal:** Presents buttons to run the program manually or through automated tests configured by the teacher and displays program execution results.
- **Text Code:** Converts block-based programs to JavaScript code for execution, enabling students to view the equivalent text code of their block designs.
- **Chat:** Facilitates communication between students and professors for feedback and questions. Messages are stored in a database for persistent availability.

Additional functionalities include the ability to add comments to blocks for discussions or clarifications between students and teachers and to manage variable values directly within the workspace.

2.3.1 Teacher Interface

Teachers play a crucial role in guiding students through the learning process and NextBlocks provides a robust interface tailored to their needs. This interface is designed to simplify the creation, management and evaluation of programming exercises, ensuring that teachers can focus on delivering quality education.

This subsection details the core aspects of the teacher interface, including exercise creation, test configuration, custom block creation and evaluation methods.

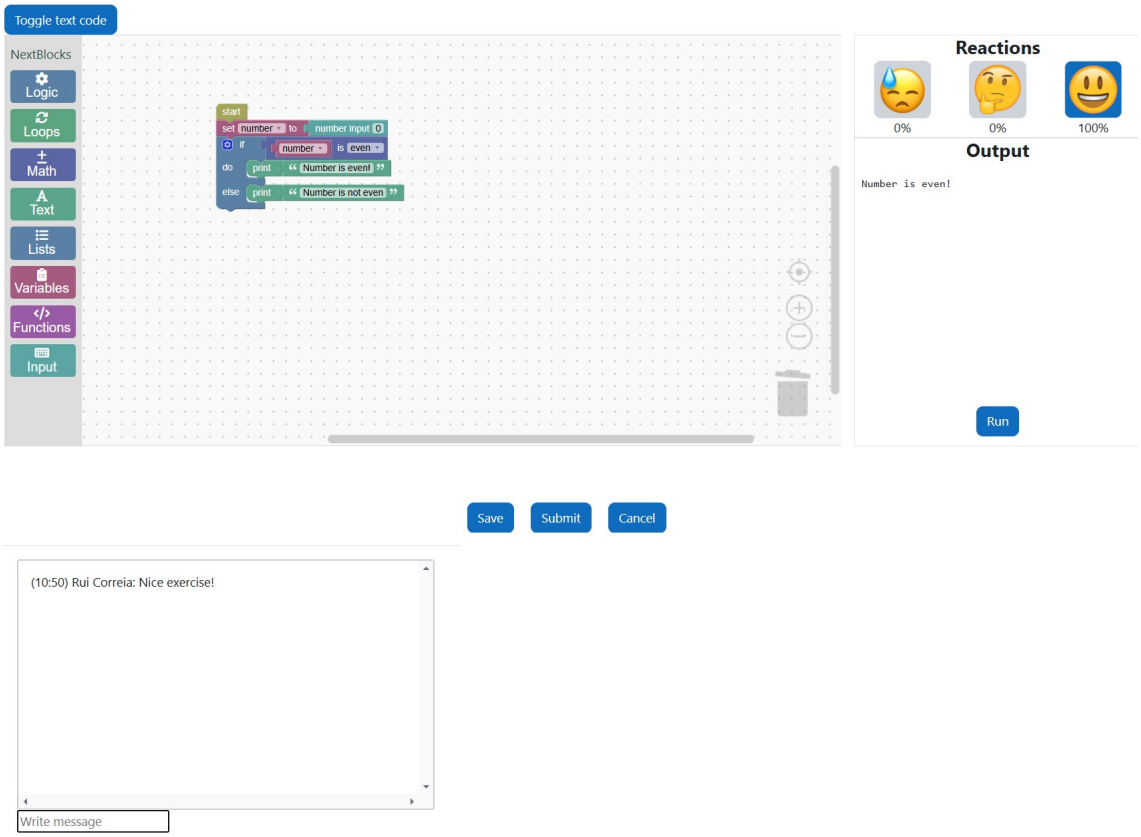


Figure 2.8: NextBlocks student program development interface with toolbox, workspace, reactions, output terminal and chat

2.3.1.1 Exercise creation

When a teacher starts creating a NextBlocks exercise, an interface will be presented similar to the one in Figure 2.9. Besides the standard Moodle customization such as exercise name and description, the teacher is able to add automated tests through text files, custom blocks for students to use and define which evaluation methods will be used. Teachers can look at students' submitted programs through the Grades page and manually grade them, as shown in Figure 2.10. If automated tests are being used, each program will already include the automatic grade, which can be manually changed by the teacher.

2.3.1.2 Tests

Teachers can create tests in order to automatically evaluate student's submissions. These tests can be added to the exercise by adding a text file using the interface shown in figure 2.11.

The files must be written in a specific language, with one example shown in Appendix A. Its format starts with any number of test cases separated by a vertical bar (|). Each test case consists of any number of input variables separated by an underscore (_), where each input variable consists of the variable name, followed by its type and, on the next

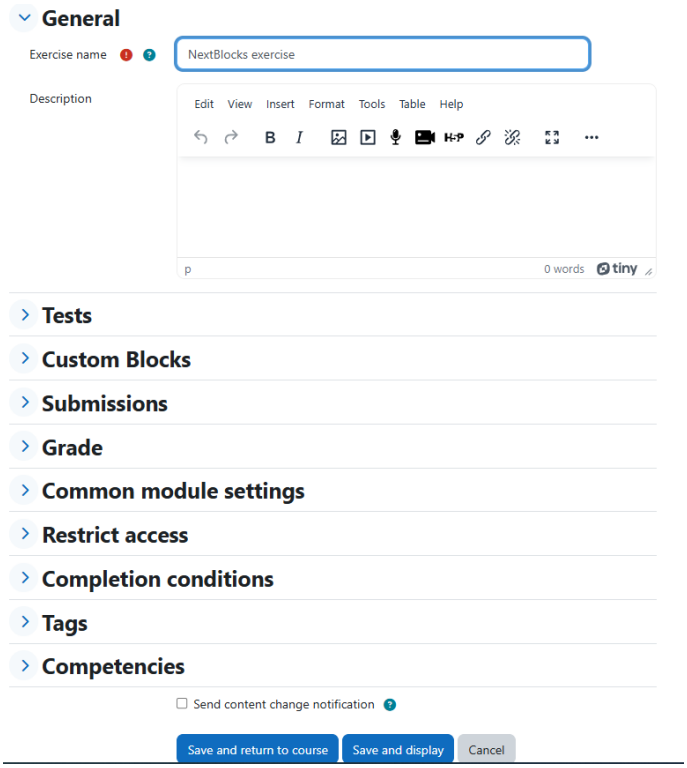


Figure 2.9: NextBlocks Exercise Creation

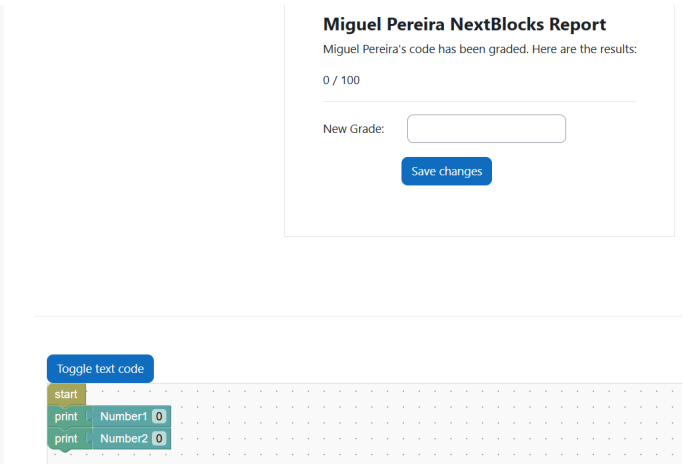


Figure 2.10: Teacher interface when seeing student submission

line, the value of the variable. The end of each test case is represented by a hyphen (-) followed by the expected output.

Each test consists of one of more inputs (each input will have the value of each input variable) and one output. Each input variable present in the test file will show up as a block directly in the workspace, as shown in Figure 2.12. The program will use each input in the student's program and compare the output with the test's expected output and, if both outputs matched, the test is considered successful.

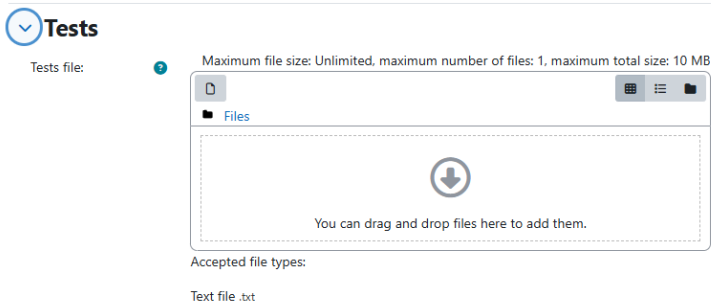


Figure 2.11: NextBlocks file input box for automated tests text file

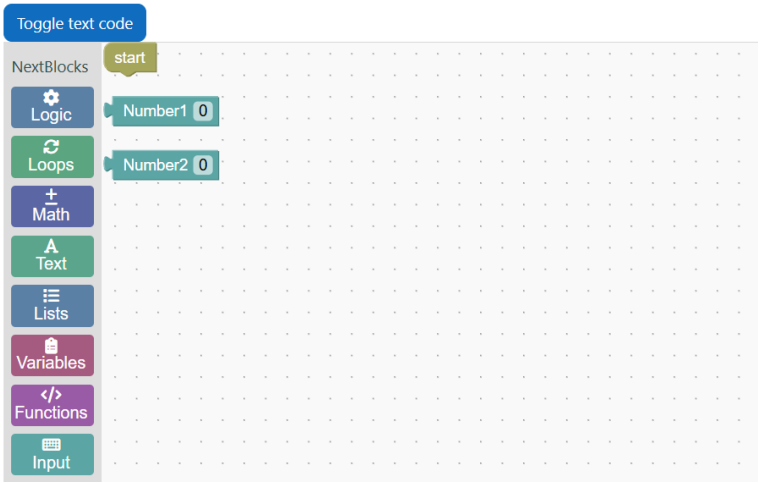


Figure 2.12: NextBlocks input blocks from automated tests' variables

2.3.1.3 Blocks

NextBlocks includes the default blocks provided by Blockly, which are part of categories such as logic, loops, math and text. Besides these, NextBlocks already has pre-defined custom blocks, such as the start block to indicate the start of a program, a print block that prints a value to the terminal and a category related to variables which have blocks users can use to create and manage variables that allow for memory usage during execution time. An example of a program using these blocks is shown in Figure 2.13.

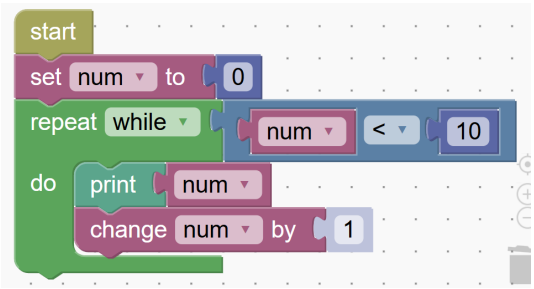


Figure 2.13: NextBlocks program using start, print and variable blocks

Teachers can add any custom blocks they find appropriate by entering their definition and generator, as shown in Figure 2.14. Both of these can be created interactively using

the blockly block factory[27], whose link is shown to the teacher on this page. Based on the program given by the user, the factory automatically generates the definition and generator for the specific block which can then be copied into NextBlocks' interface.

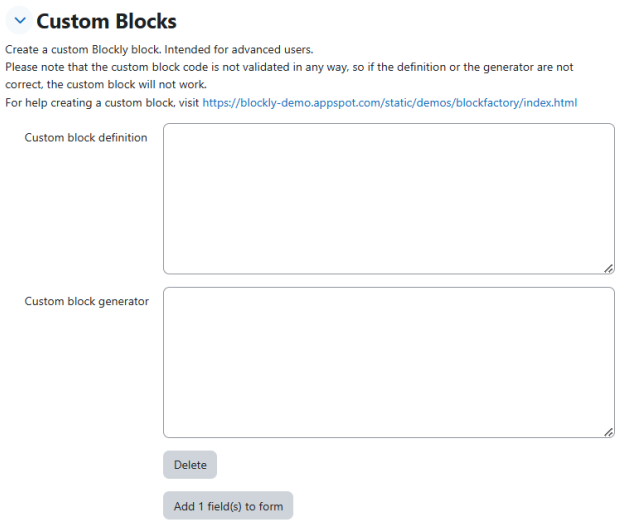


Figure 2.14: NextBlocks custom blocks interface

2.3.1.4 Evaluation method

Since NextBlocks is integrated as a Moodle activity, teachers can use different automated classification methods according to what they find appropriate for each exercise, as shown in Figure 2.15. They can choose how many submissions each student can do and choose different types of grading, with types such as point or scale, as well as defining the maximum grade and the grade needed to pass.

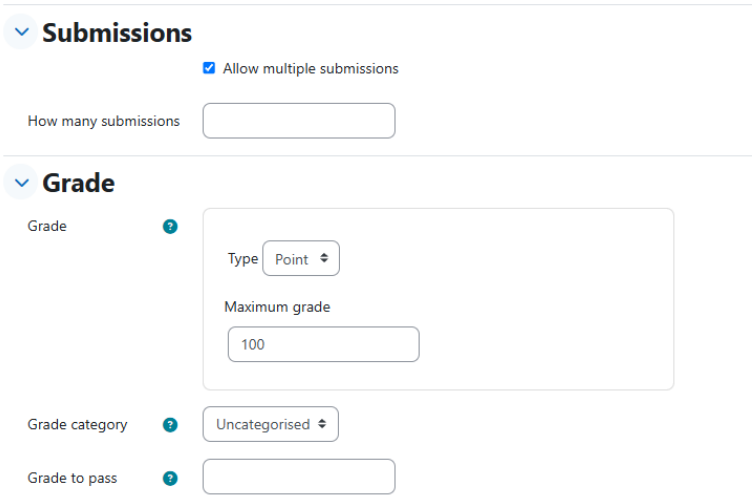


Figure 2.15: Moodle Evaluation method

2.3.2 Student Interface

The student development environment is shown in Figure 2.8. This interface is composed of 6 main areas: Text code, toolbox, workspace, reaction bar, output terminal and chat. It also includes 3 extra buttons: Save, to save the current workspace in a database for later use in this exercise, submit, to submit the current program for evaluation and cancel to exit the exercise.

2.3.2.1 Toolbox

On the left side of the interface there is a toolbox where the user can access every block available that they need to create a working program. To avoid cluttering each block is assigned 1 of 9 categories (such as logic, loops or math). and only these categories are shown in the interface, not every block. To access the blocks from a specific category the user just needs to click the category name, as shown in Figure 2.16 . Each block can be dragged into the Workspace an unlimited number of times to be used in the program.

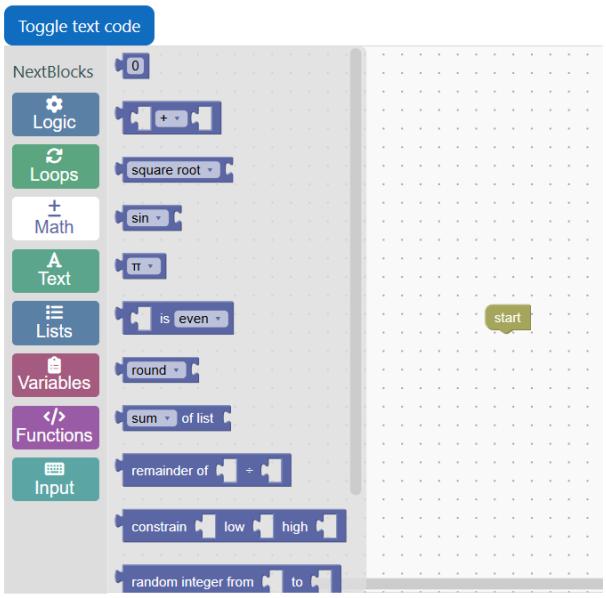


Figure 2.16: NextBlocks toolbox category

2.3.2.2 Workspace

The workspace is in the center of the interface and its where the user creates the program. They can drag the blocks from the toolbox and organize them in the workspace, below the pre-placed start block. Variable values can be changed directly by editing a number or string block, like shown in Figure 2.17. Every block connected to the start block will be executed, from top to bottom, when the "Run" button is pressed on the Output Terminal. Any block outside of the block sequence that includes the start block is ignored.

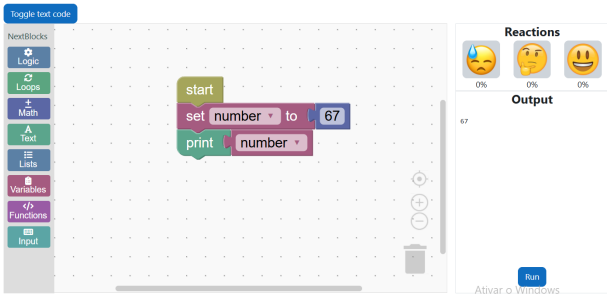


Figure 2.17: NextBlocks program editing a variable's value

2.3.2.3 Reaction bar

In the upper right corner of the interface there is a reaction bar with 3 possible reactions. Each reaction is clickable and serves as feedback for the professor and other students about the exercise. Below each reaction there is a percentage indicating the distribution of the reactions across all students for that specific exercise. This functionality is important for feedback, as users can indicate how they feel about that specific exercise.

2.3.2.4 Output terminal

This terminal consists of a text box, a "Run" button and possibly a "Tests" button, if automated tests are being used. When the "Run" button is pressed, the program currently in the Workspace is executed and the output is presented in the box.
If the teacher added a test file during exercise creation, there will be a "Tests" button next to the "Run" button. Clicking this button will show the available tests and run them, showing whether each of them passed or failed, as shown in Figure 2.18. Clicking on one of the tests will show the details of that test, shown in Figure 2.19

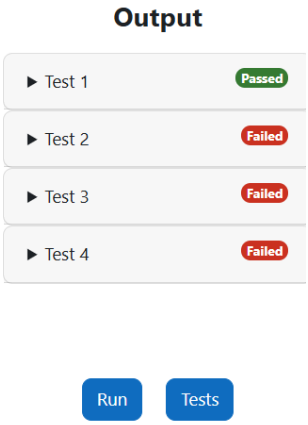


Figure 2.18: NextBlocks tests feedback

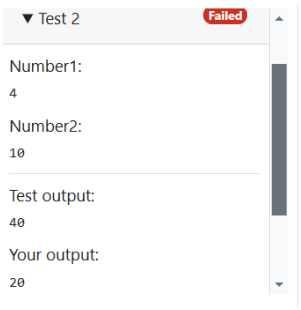


Figure 2.19: NextBlocks test details

2.3.2.5 Text code

On the top left of the workspace there is a "Toggle text code" button. Clicking this button automatically converts the current program into JavaScript code, including a print function that prints a value to the Output terminal. While students can view this code to better understand text-based programming languages, it remains non-editable. The generated code will execute when the user presses the "Run" button in the Output terminal. The block program in Figure 2.20 is converted to the equivalent text code in Figure 2.21.

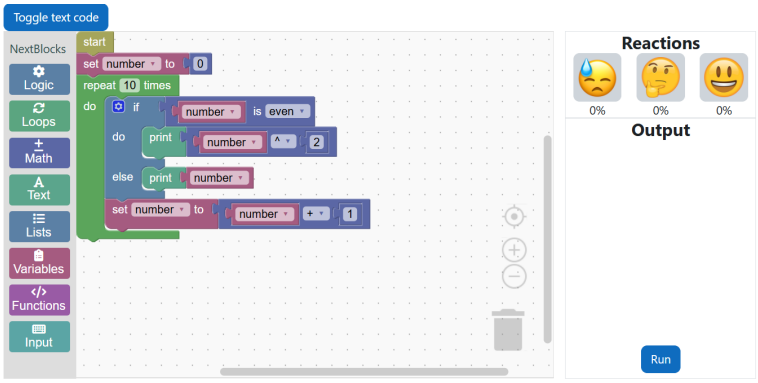


Figure 2.20: NextBlocks example program in blocks

Figure 2.21: NextBlocks example program in text

2.3.2.6 Chat

At the bottom of the student interface there is a chat, unique to each exercise. There users (students and professors) can chat about the exercise, including feedback and question answering. They also see every message that was sent in that specific exercise since its creation, even if the user was not online, since the messages are saved in a database using a custom web service function. NextBlocks' chat works as an external service, which means that the exercise organizer needs to have the chat service running separately for it to show up in the interface for each user. If the service is not running the plugin still works without showing the chat box.

2.3.2.7 Comments

By right clicking on a block and selecting "Add comment" from the options available, a user can add a comment to that specific block. This can be used in many ways, including students questioning teachers about functionalities or best practices and teachers questioning students about intent behind using that specific block. If a block has a comment attached to it, it will show up as a button behind the label of the block with a question mark. Clicking this button generates a text bubble below the block with a comment, as shown in Figure 2.22. Users can edit this bubble by clicking on it.

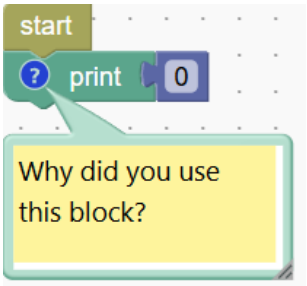


Figure 2.22: NextBlocks comment example

2.3.3 Interaction

There are 3 types of user interaction present in NextBlocks:

- **Chat:** Students can interact with other students and teachers in chat to ask for help, advice, give and receive feedback and ask questions. This tool can be very helpful if a student does not know how to do a specific exercise and wants some guidance.

- **Comments:** Students can leave comments on specific blocks in their program which teachers can see and answer after submission. Teachers can also create comments to ask questions or give feedback about specific blocks of the program.

- **Reactions:** Students can give feedback on an exercise through reactions. The percentage of reactions of each type is shown to every student and teacher in the exercise interface.

2.4 Summary

Block programming plays a crucial role in making coding accessible and engaging for learners of all levels. By removing the barriers of complex syntax, users can focus on problem solving and creativity, fostering a deeper understanding of core programming concepts.

NextBlocks[9] facilitates a way for students to learn programming in an interactive, intuitive and entertaining way, while getting feedback from teachers and collaborating with other students. This plugin introduces innovative tools in the field of block programming, such as being integrated in a worldwide distributed LMS and allowing for full customization from teachers.
Although NextBlocks is an improvement in the world of Block Programming environments, some adjustments are needed when compared with similar environments. During the user testing process of the current NextBlocks version, some negative aspects were identified, such as:

- Of the users who studied the student interface, 75% answered that the comment system was not easy and/or intuitive to use. They indicated that it was not clear that they had to right-click on a block to add a comment and that they did not know how to edit a comment after creating it.

- Some students indicated that they spent too much time looking for the blocks they wanted, due to the large number of blocks in the toolbox.

- Some students tried to duplicate the input blocks that get the values from the test files if they wanted to use those values more than once, instead of storing them in a variable. This was not expected during the development of the plugin and ended up breaking the program's execution.

These issues highlight areas where NextBlocks needs refinement to improve usability and user satisfaction. Addressing these problems and adding innovative features, such as an interactive input system and the ability to limit the usage of specific blocks, are essential steps for NextBlocks to be a truly effective and competitive tool in the domain of block programming environments.

3

Related Work

This chapter focuses on showing other work similar to the NextBlocks plugin, explaining its similarities and differences. The studied environments were chosen based on popularity and similarity with NextBlocks. This analysis can be helpful to identify gaps in the block programming world and in the NextBlocks plugin, providing insights for improvements in NextBlocks.

3.1 Kodable

Kodable [28] is a coding application directed to young children that creates simulated game scenarios where the children need to give basic directional instructions to a character in order to get them from a starting point to their destination, as shown in Figure 3.1. These activities motivate kids to learn how to think computationally from a very young age. According to [28], their research suggests that digital apps such as Kodable can be used to teach young children foundational coding skills while entertaining them, making it a more enjoyable experience.

3.2 Scratch

Scratch[6] is a web environment designed by MIT which encourages users to create interactive game animations using block programming, shown in Figure 3.2. While this tool has an option to create custom blocks, the functionality of these blocks need to be the same as a combination of already existing blocks. This restriction limits the possibilities since it's not possible to create new functionalities, only to compact a set of instructions into a single block. According to[6], Scratch can be used to teach Computational Thinking habits to primary school children.

3.3 Code.org

Code.org[7] is a website that enables students to learn coding while being supervised by their teachers, that can assign them specific courses and monitor their progress, with

Figure 3.1: Kodable Application [29]

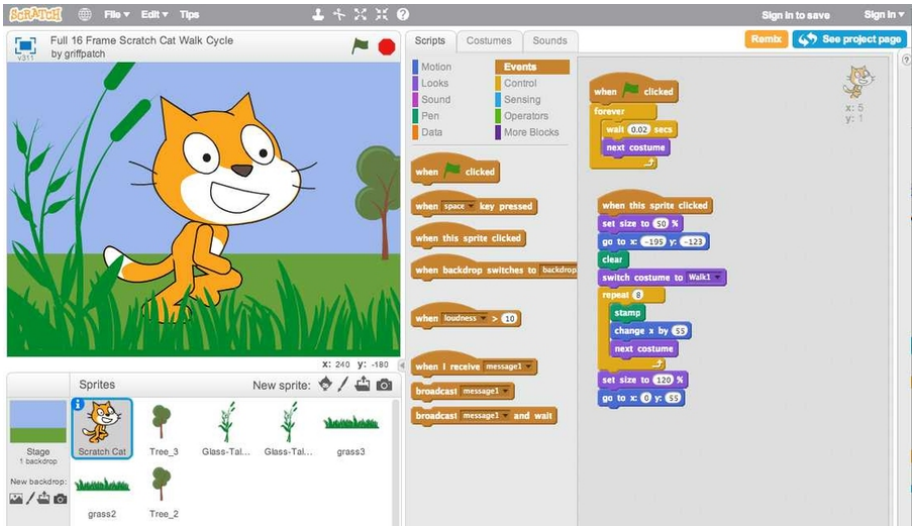


Figure 3.2: Scratch [30]

an interface similar to Figure 3.3. Teachers can create classrooms, assign courses to classrooms from a catalog shown in Figure 3.4 and choose which chapters are shown to students, allowing for some customizability. While this can be very helpful in learning environments, Code.org does not allow teachers to create custom courses (only to use the ones already available on the website) or to create custom blocks.

3.4 MakeCode Arcade

MakeCode Arcade [32] is a tool developed by Microsoft that allows for the development of retro arcade games using blocks, shown in Figure 3.5. This tool shows the program in a simulated game console. While this mechanism may be interesting for game development

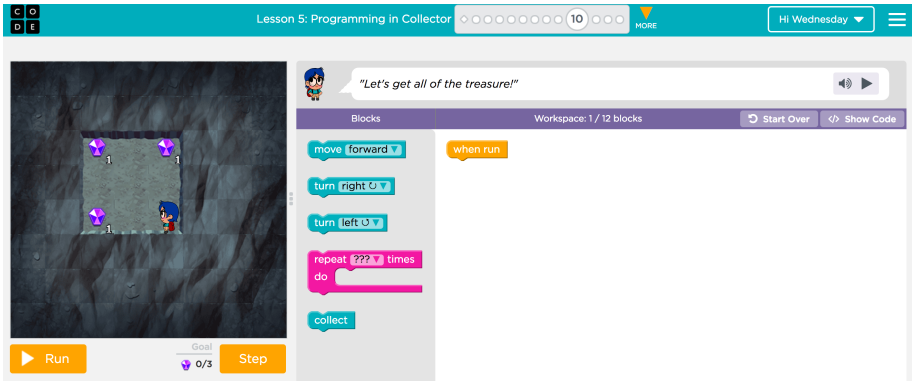


Figure 3.3: Code.org [31]

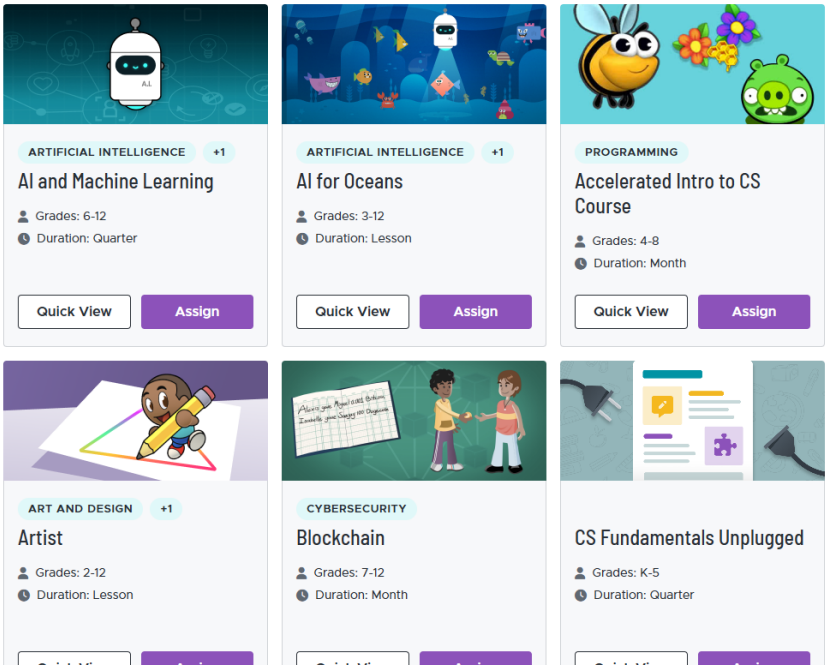


Figure 3.4: Code.org [31] Course Catalog

aspirants, it doesn't allow for the creation of customized blocks making it less suitable for learning environments. It also does not include some necessary basic blocks such as math operators or string modifiers, since it focuses uniquely on arcade game development.

3.5 BlockPy

BlockPy[34] is a web-based Python environment developed by the University of Delaware which allows for learners to create programs using blocks or python code, using the interface shown in Figure 3.6. If they use blocks, a sample of the python code generated will be provided to the user. This environment does not allow for educators to create custom exercises that automatically evaluate students' outputs (teachers would have to manually evaluate students' programs) neither does it have a functionality to create custom

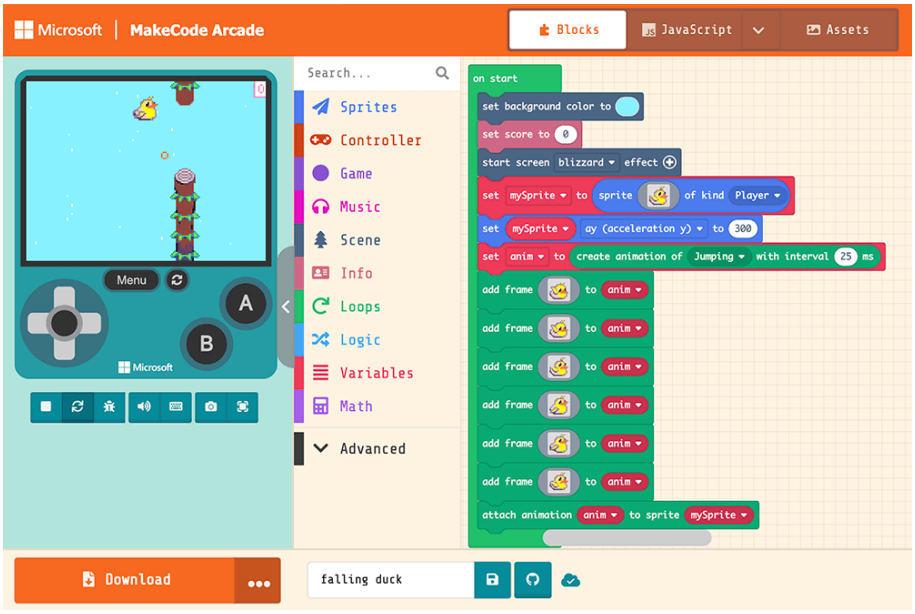


Figure 3.5: MakeCode Arcade tool [33]

blocks by the user.

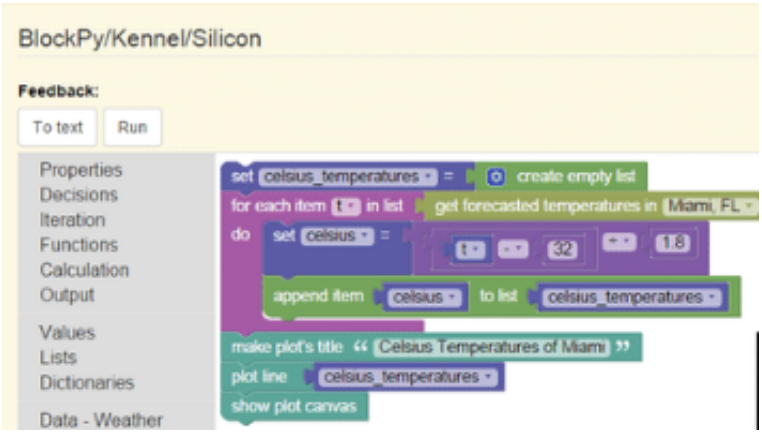


Figure 3.6: BlockPy environment [35]

3.6 Comparative analysis

This section includes a comparative analysis of various block programming environments, focusing on their unique features, strengths and limitations. Every studied environment uses Blockly[23] as a baseline for the blocks. The analysis is divided into three key areas: the learning environment and exercise creation, programming capabilities and feedback and collaboration. The objective of this structured examination is to identify the gaps in existing solutions and to highlight what could be done to address them.

3.6.1 Learning environment and exercise creation

Out of the studied block programming environments, NextBlocks is the only one that is integrated into a widely known LMS, which means it is possible to combine it with other resources and activities that are also integrated in Moodle. It is also the only one that allows teachers to create fully customized exercises.
Code.org[7] allows teachers to choose what exercises they want students to see, but the choice is limited to the courses already existing on the website. The other studied environments do not have any exercise organization functionalities that teachers can use for customization.

3.6.2 Programming

Currently none of the studied block programming environments supports fully interactive input similar to the one in text programming languages, with the use of a terminal. Scratch[6] and MakeCode Arcade[32] allow users to directly interact with actors by moving and rotating them, but no other part of the program can be interacted with during execution time.
Scratch[6] and NextBlocks[9] are the only studied environments that allow creation of custom blocks that simulate functions. However, for Scratch, these are local and cannot be made available to other people. NextBlocks blocks can be made available by the teacher for all students in a specific exercise, and it already includes some blocks added on top of Blockly[23] default blocks, such as the start block. The only 2 environments of the studied ones that automatically translate block programs into text code are Blockpy[34], translating to Python, and NextBlocks[9], translating to JavaScript.

3.6.3 Feedback and collaboration

While some of the other studied environments offer global and/or classroom chats, NextBlocks[9] is the only one that offers separate feedback and collaboration for each exercise. Although feedback can be given through reactions, the chat is crucial for collaboration. NextBlocks also has the comments system, where teachers and students can leave messages in specific blocks, which can be used for questions or feedback.

3.7 Summary

It is clear that currently available block programming environments lack some key functionalities that would make learning programming not only more efficient, but also more entertaining, rewarding and customizable, such as an interactive input system and translation to different text languages. During this thesis, NextBlocks will be improved with these key functionalities in order to make learning programming better for everyone involved.

4

Proposal

When comparing popular block programming environments, it becomes evident that there is a lack of LMS integration, presenting challenges particularly for inexperienced users, who are often the primary audience for these platforms. This combined with other innovative features, such as Custom Exercise Creation and Feedback/Collaboration, makes NextBlocks a groundbreaking platform which has potential to improve the process of learning for both teachers and students.

This dissertation focused on the development of NextBlocks 2, an enhanced version of the original Moodle plugin designed to address pedagogical limitations and improve user experience for both educators and learners. The implementation centered on introducing key features that include an interactive terminal for real-time program input/output, a reworked testing system and its migration to an external server for evaluation reliability, a redesigned threaded comment system for collaborative feedback, an improved chat system, configurable block usage limits for guided learning, Python translation capability alongside existing JavaScript output and a restructured student interface optimizing workspace usability.

4.1 System Architecture

NextBlocks 2 is implemented as a Moodle activity plugin integrated into Moodle's modular architecture while extending its functionality with a complete block programming environment. The system's architecture follows a client-server model, with distinct components handling user interaction, program execution and data persistence. Figure 4.1 illustrates the high-level architecture of NextBlocks 2.

The architecture comprises four main layers:

1. **User Interface Layer (Client-Side)**: Built using the Blockly library in JavaScript, this layer provides the visual block programming environment within the user's web browser. It includes the toolbox, workspace, chat interface, comments system and the terminal. This layer is responsible for rendering the blocks, capturing user interactions and sending program data to the server for evaluation.

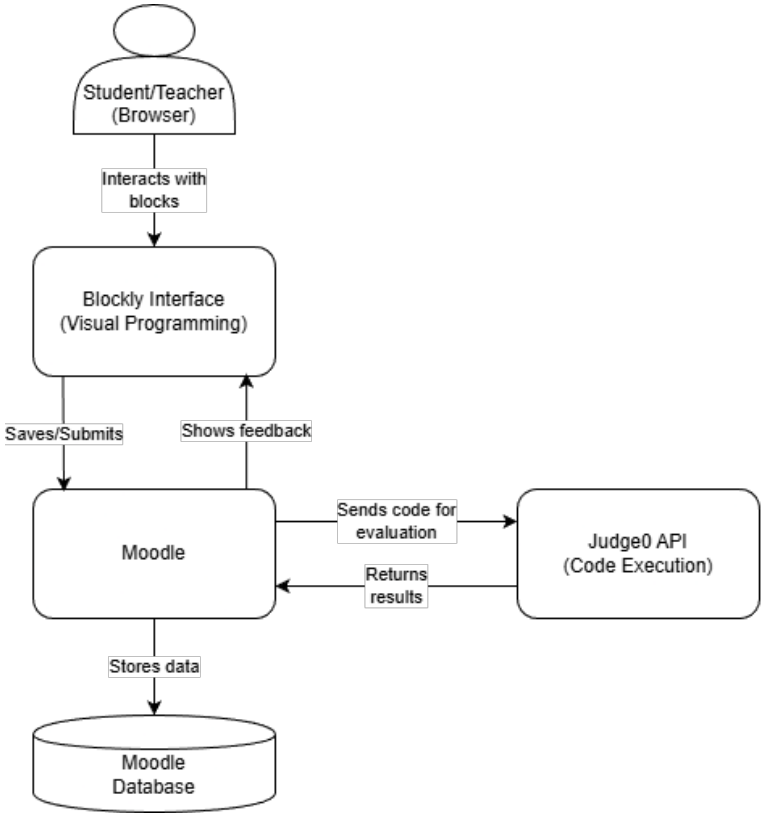


Figure 4.1: NextBlocks 2 system architecture

2. **Moodle Integration Layer (Server-Side)**: Implemented in PHP, this layer handles the core Moodle plugin logic. It manages:
   - **Activity Lifecycle**: Exercise creation, configuration and gradebook integration via Moodle's standard activity APIs.
   - **Data Management**: All plugin-specific data (submissions, comments, chat messages, custom blocks, block limits) is stored in dedicated database tables within Moodle's schema.
   - **Compliance and Portability**: Implements Moodle's Privacy API (GDPR compliance), Backup and Restore API and Language String API (AMOS) to meet directory submission requirements and ensure data portability.
3. **Execution and Evaluation Layer**: To ensure security and reliable grading, program execution is delegated to an external API. When a student submits a program, the plugin sends the block code (converted to JavaScript) and test cases to the Judge0 code execution engine via the Sulu API gateway. The external service runs the code in a sandboxed environment, compares outputs and returns the results, which Moodle then records in the gradebook.
4. **Data Persistence Layer**: Utilizes Moodle's native PostgreSQL database. All user data, exercise settings, submissions and collaborative content (comments, chat) are

stored in relational tables, ensuring data integrity and leveraging Moodle's existing user and course management.

This layered, integrated architecture allows NextBlocks 2 to combine the visual simplicity of block programming with the administrative power of a world-leading LMS, while offloading resource-intensive code execution to a secure, specialized external service.

4.2 Revamp of the input/output system

The original NextBlocks environment required all program inputs to be predefined in test files prior to execution, preventing any runtime interaction. To simulate the dynamic input behavior of text-based programming languages (e.g., Python's `input()` function [36]), a dedicated input block that enables direct communication between the user and the environment during program execution was implemented. This block is available in the newly introduced toolbox category Input/Output, as shown in Figure **??**, which also contains the "print" block developed in the previous version. The input block features 3 main components:

- A prompt string parameter displayed to users through the terminal when the block is executed. This string can be introduced by connecting another block that returns a string to the right side connector of the input block. If this block is not present during execution, the "Input" block treats the prompt string as an empty string.
- Execution pausing functionality that halts program flow until the user submits an answer to the prompt.
- An output connection that returns user responses as strings to any block connected to the left side connector.

The program exemplified in Figure 4.2 has the same functionality as the following Python program:

```
print(input("Car price: "))
```

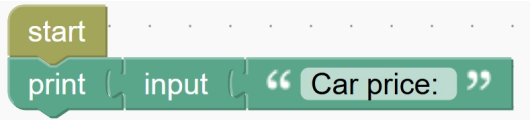


Figure 4.2: NextBlocks 2 program exemplifying use of input block

When that program is executed, the terminal shows the prompt and waits for the user's answer, as shown in Figure 4.3

The values entered by the user in response to the input block's prompt are always treated as strings. Since users might need to input numbers, a second block was created

Figure 4.3: NextBlocks 2 terminal during execution of input block

called "text to number" which is available within the Math category. This block takes as input a string and returns the number corresponding to that string. The program exemplified in Figure 4.4 has the same functionality as the following Python program:

```
print(float(input("Car price: ")))
```

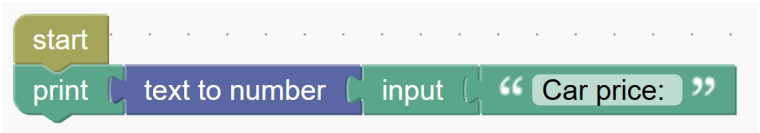


Figure 4.4: NextBlocks 2 program exemplifying use of text to number block

The "text to number" block might generate some different results based on the string that is given:

- If the string only contains digits and an optional point for decimal numbers, it is converted to a number with those characters ("10.24" is converted to the number 10.24 [10 units with 24 hundredths])
- If the string contains digits followed by other characters, it converts the digits to a number until it reaches the first character that is not a number or the first point ("12.3abc4" is converted to the number 12.3 [12 units and 3 tenths]).
- If the string starts with a character that is not a digit then an error is thrown and the execution stops. The program shown in Figure 4.5, when executed, shows the error in Figure 4.6.

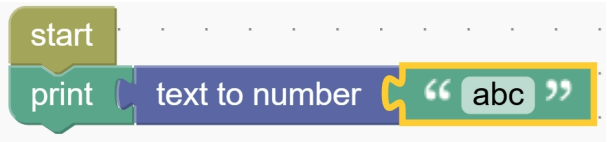


Figure 4.5: NextBlocks 2 program that tries converting non-numeric text into a number

Figure 4.6: NextBlocks 2 error when trying converting non-numeric text into a number

The "input" and "text to number" blocks allow students to input values to the program during execution time, introducing true interactive capabilities that fundamentally transform NextBlocks into a dynamic, effective and entertaining programming environment. Some programs that were not possible previously can now be done. An example is the program that is shown in Figure 4.7, which reads numbers until the number 0, and then prints how many of those are even numbers.

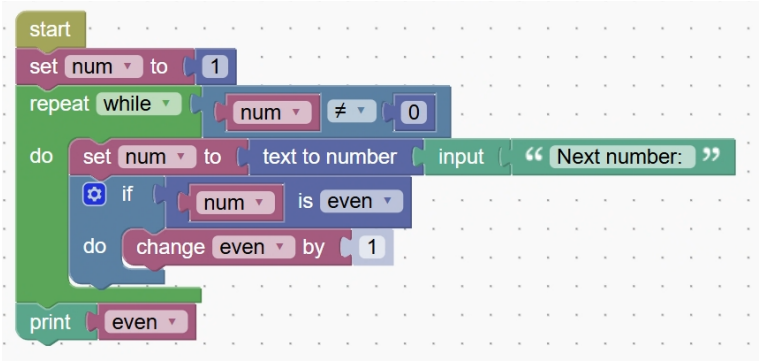


Figure 4.7: NextBlocks 2 program that reads numbers until the number 0 and prints how many were even

4.3 Reworked testing system

To comply with standard test file syntax used in other environments that handle submissions of programs, such as Mooshak[37], the test cases were split. Instead of being all cluttered in the same file, now each test case requires 2 files, called inputX.txt and outputX.txt, where X is any integer that will serve as the test case identifier. The identifers do not need to have any specific order and they do not need to be sequential (for example, it is possible to have a test 4 without a test 3). However, these identifiers will be used to decide the order of which the tests will be shown to the students and executed (smaller identifiers are shown and executed first). The input files should contain the input values for that test case, one per line. The output files must contain the expected output for that test case. Both the input and output files can be empty if there is no input or no expected output, respectively, but they need to exist in the system. If there is a test case with an input file but no output file, or vice versa, that test case is ignored. An example of an input and output files are shown in Figure 4.8, which refer to the program that reads numbers until it reads the number 0, and then prints how many of those numbers were even.

Instead of automatically creating blocks for the inputs, the tests now make the values available when the user asks for an input using the "Input" block. When the student runs the tests on their program, the system maps test case input values sequentially to input blocks in execution order. For example, the first time an input block is executed, the environment will automatically assign it the first input value from the current test being

```
2
3
9
10
90
0
```

(a) Input file

```
4
```

(b) Output file

Figure 4.8: NextBlocks 2 input and output files example

run. Each block can have more than 1 value assigned if the block is executed multiple times in the same program execution. The test validation follows these rules:

- **Correct number of inputs**: Each value is assigned to the block in the same position in the sequence (first value → first block).
- **Insufficient input blocks**: Extra values are not used.
- **Excess input blocks**: Execution aborts with explicit error. While running the program shown in Figure 4.9 (Reading and printing 4 input values) using a test file that only provides 3 input values the output shows the error shown in Figure 4.10.

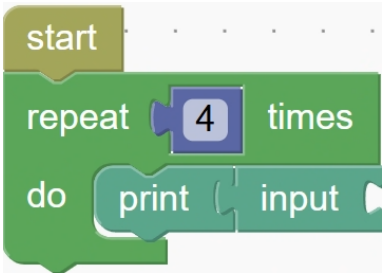


Figure 4.9: NextBlocks 2 program that reads and prints 4 input values

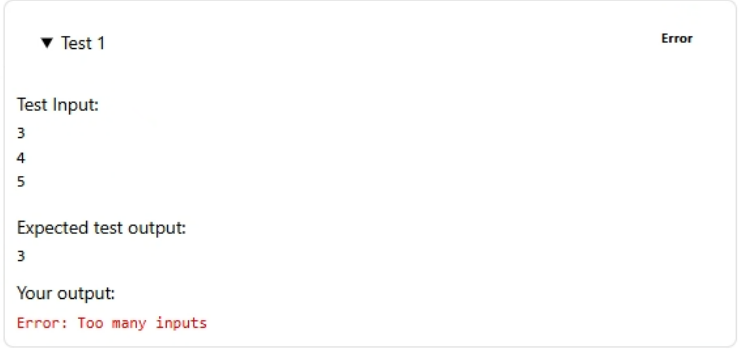


Figure 4.10: NextBlocks 2 tests error when input blocks are in excess

Even though students can add prompts to input blocks, like the "Car price: " shown in Figure 4.11, these prompts are ignored by the environment when running automated tests.

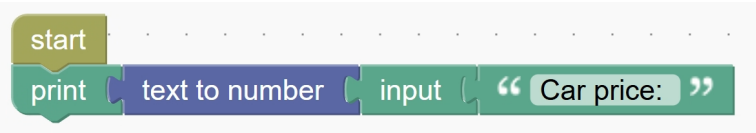


Figure 4.11: NextBlocks 2 program exemplifying use of prompts in input block

This redesign fundamentally transforms NextBlocks' automatic testing system in a way that simplifies test management and mirrors commonly used programming environments, such as Mooshak which is commonly used for programming exercises and also has a testing system that requires test files to be separate by test case.

4.4 Program submission

When a user submits their program for evaluation, the tests are run in order to automatically grade the submission. Each test input values are run through the submitted program and the output is compared with the expected output present in the test files. The automatic grade attributed to the submission is the percentage of test cases that had the correct output, converted to either points or a scale system depending on the exercise's settings.

Running this directly in the user's browser, which happened in the previous NextBlocks version, would be unsafe since it would be very vulnerable to manipulation of the results by anyone with some knowledge in web development [38]. With this in mind, the evaluation of the programs was changed from the user's browser to an external API.

4.5 Block to Python translation

In the previous NextBlocks version, the block programs created by the students would be automatically converted into JavaScript, showing it to the user and being used to execute the program. While this is enough for program execution, it might not be enough for a complete learning process, since different users might want to understand how other text programming languages work.

NextBlocks 2 implemented the automatic translation into Python, one of the most widely used languages for learning purposes [39]. By clicking the "Toggle text code" button on the top of the interface, users are now presented with 2 options: JavaScript (selected by default) and Python. Users can switch between both languages using the buttons shown in Figure 4.12. The selected language is highlighted with a green background and its code translation from the user's program is shown.

While the translation of blocks already available in the Blockly's[23] library is available directly through pre-defined Blockly functions, the same does not happen with custom

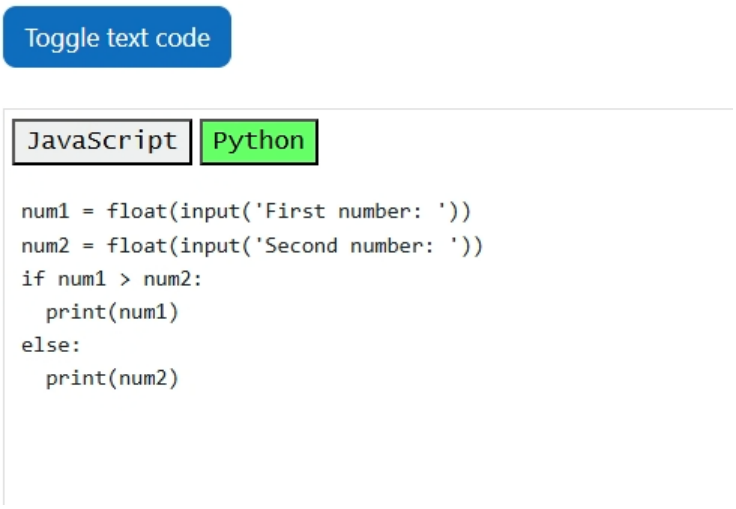


Figure 4.12: NextBlocks 2 block to code translation interface

blocks created by the teachers. This means that, if teachers want their custom blocks to have a translation to Python they need to provide that translation during the creation of the block, in a specific box shown in Figure 4.13. Since Python is not used to execute the program, it is not mandatory for teachers to create the custom block's translation to Python. If the teachers create a custom block without Python translation and any user tries to view that translation, the custom block will be translated to an undefined function with the same name as the block.

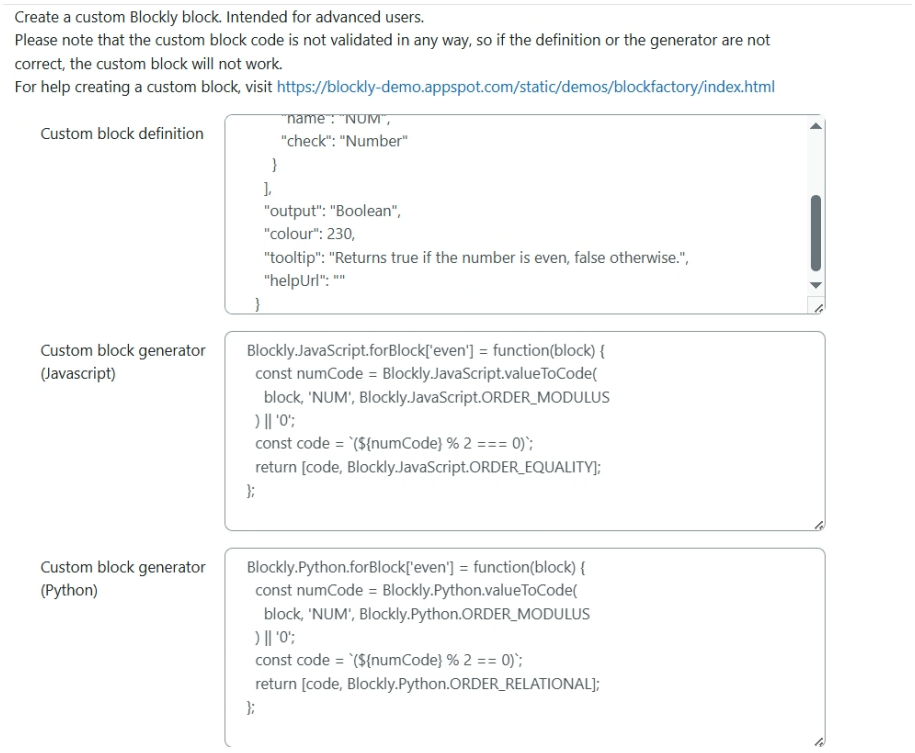


Figure 4.13: NextBlocks 2 custom block Python generator

4.6 Block Limits

Before this new version, students could create the programs in any way they wanted without any restriction as long as it passed the test cases. This often led to bad coding practices which can compromise program comprehension[40].

NextBlocks 2 introduces block limits, a feature that allows teachers to limit how many times each type of block can be used. Some teachers might want to keep all blocks without any restriction to allow for creativity, while others might want to restrict some blocks to lead students to a solution that helps students develop good practices.

While creating an activity, teachers are shown a new tab named block limits, as shown in Figure 4.14. This tab shows every block separated by category, similar to how the students interface's toolbox is organized. Each block has a checkmark called "Infinite" that is enabled by default. Any block that has this checkmark enabled will not be affected by block limits, meaning students can use as many instances of the block as they want. If a teacher turns off the checkmark, a input box appears where the teacher can input a number, representing how many times each student can use that block, in that exercise.

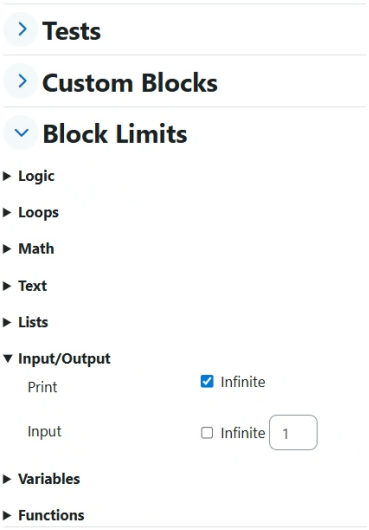


Figure 4.14: NextBlocks 2 block limit menu for teacher during exercise creation

On the student side, when developing the program, if a block has reached the maximum number of instances that can be used, that block will be grayed out in the toolbox and it will not be possible to create more instances of that block. This is shown in Figure 4.15, where the first two blocks can be used, but the others are blocked.

There is also a new button on top of the workspace called "Toggle block limits". When a student clicks this button, the environment opens a new window with every block and its maximum number of instances allowed for that exercise, as shown in Figure 4.16.

Limiting blocks is an optional functionality for teachers who want to guide students toward specific solutions. Teachers who do not want to limit blocks can ignore the feature.

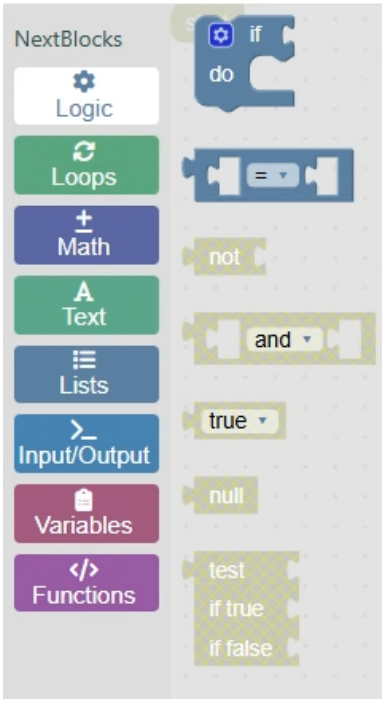


Figure 4.15: NextBlocks 2 toolbox with limited blocks

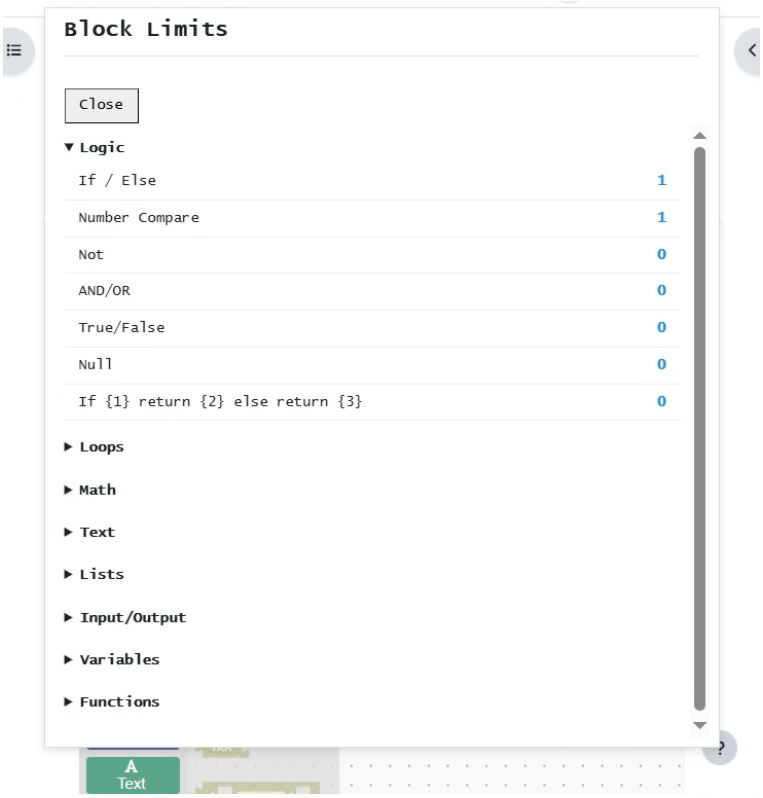
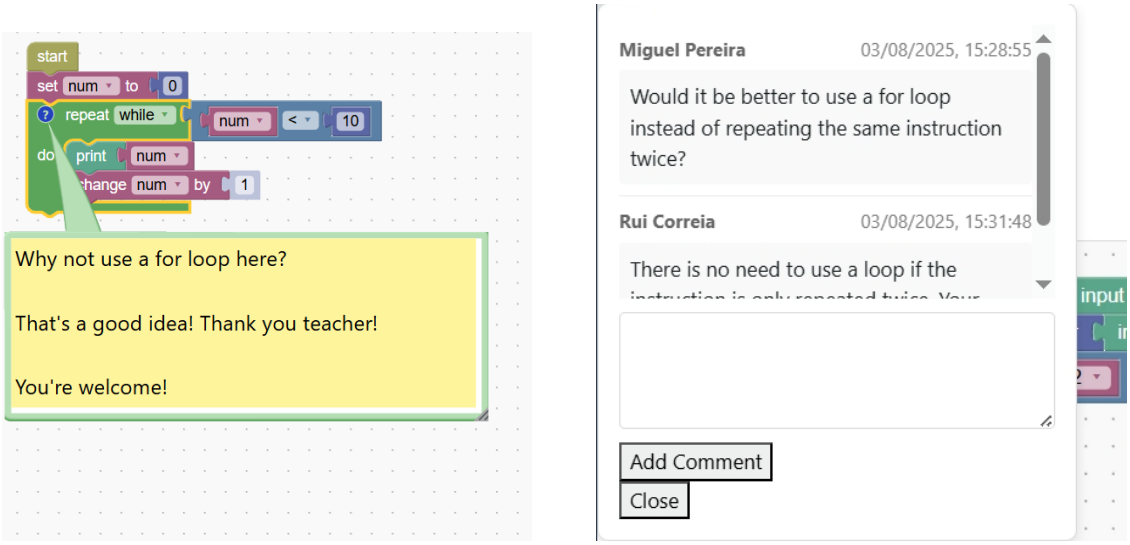


Figure 4.16: NextBlocks 2 block limits window in student interface

4.7 Improved comments system

While adding a single comment to a block is a clear process, if a block started getting too many comments it used to become cluttered since every comment was inside the same box with no clear distinction of when does each new comment start, who created it and when.

To make sure its clear when and who added a comment, each comment is now separated in its own block, with a creator name and a timestamp. If every comment does not fit into the box, the user can scroll down to see the remaining comments. The interface of both versions is shown in Figure 4.17.



(a) NextBlocks 1 (b) NextBlocks 2

Figure 4.17: Comment interaction between a teacher and a student

4.8 Reworked chat system

In the previous NextBlocks version the chat implemented in the plugin only worked if its separate server was running at all times. This was not be viable for some institutions that do not want to keep a service running permanently in their machines and it would not be possible to make this server available through the Moodle Plugin Directory[11]. To fix this issue, the chat system was reworked. Instead of having an external server, there is now a message system integrated with the database that simulates a chat. This change was made to the internal architecture of the plugin, leaving the chat's interface and functionality unchanged for educators and students.

4.9 Restructured Interface

Due to the introduction of new and changes to existing functionalities, the interface of NextBlocks was restructured, as shown in Figure 4.18. Since now the terminal was not only used for output, but also for input, there was a lack of space as it was confined to one of the corners of the screen. The workspace was moved to the bottom of the workspace, making it possible to make use of the full width of the screen for the terminal. Its background color was changed to black to better differentiate from the workspace. The reaction bar was moved to the top making it possible to also widen the workspace, which is the feature that students will spend the most time looking at. The button to toggle the newly introduced block limits window is directly related to the toolbox, so the button was placed above the toolbox, where the "toggle text code" button was in the previous version. The "toggle text code" was moved to the right of the reactions bar.

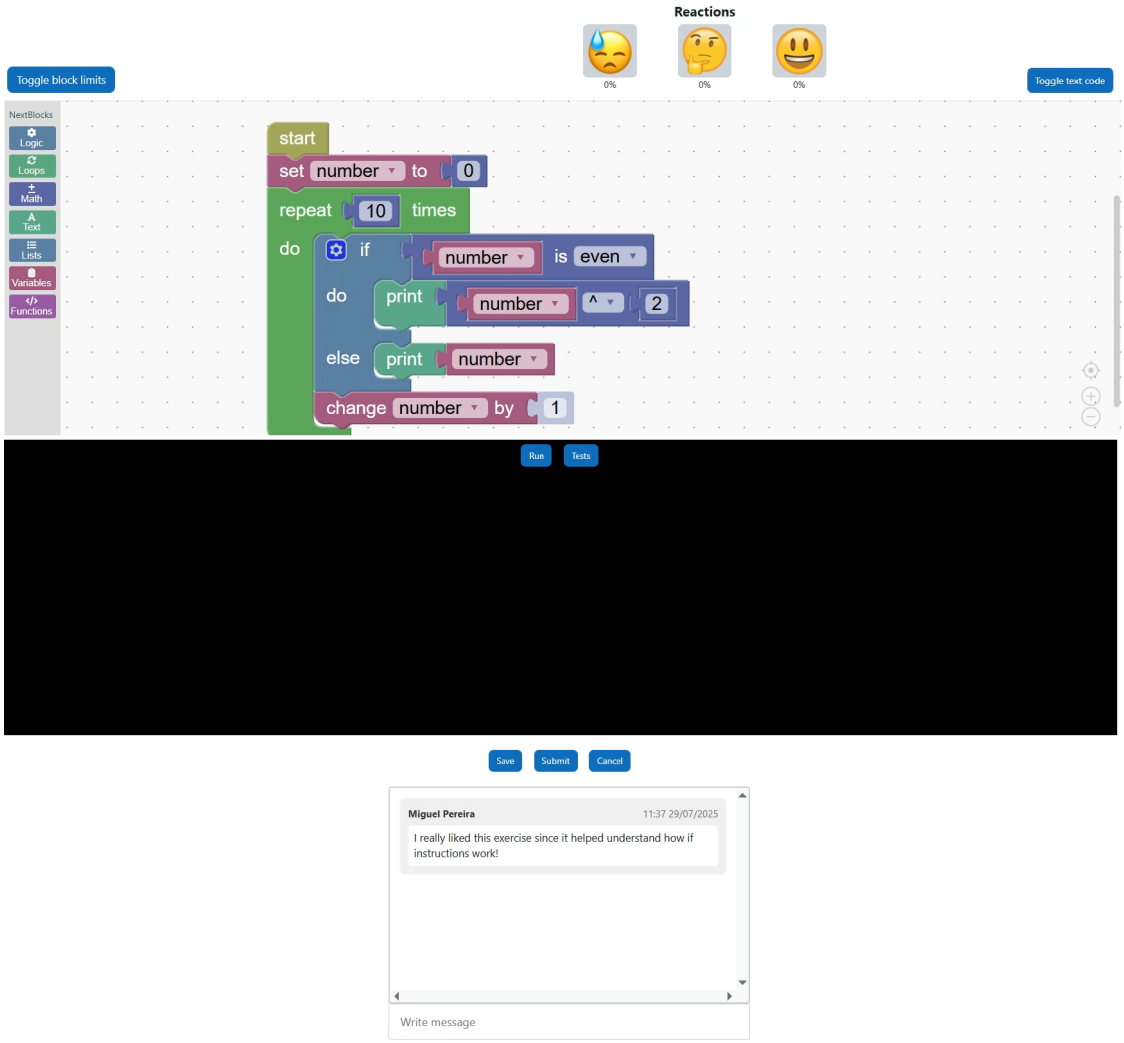


Figure 4.18: NextBlocks 2 restructured interface

4.10 Conclusion

This chapter has detailed the major improvements made in NextBlocks 2, transforming it into a more complete and user-friendly tool for learning programming. The goal was to address the limitations of the previous version and similar platforms by adding and improving features that make the experience better for both teachers and students.

The most significant change was the introduction of an interactive input system. Before, all program inputs had to be set up in advance by the teacher. Now, students can use an "input" block to type answers while their program is running, just like in traditional coding environments. This makes learning more dynamic, interactive and engaging. A new testing system was also created to work with this feature, using a simpler and more standard file format that teachers will find familiar.

In order to help students understand the connection between blocks and text-based code, the automatic translation from block programs into Python was added. This gives learners a clear bridge to the next step in their programming journey while helping teachers during submission evaluation.

A new feature called "Block Limits" gives teachers the possibility to guide students towards the optimal solution. They can now restrict the number of times that a specific type of block can be used in an exercise. This helps guide students away from bad coding practices and encourages them to think more carefully about their solutions.

Improving how people interact within the platform was also a focus in this version. The comment system was redesigned to be more organized, making conversations about specific blocks easier to follow. The chat feature was rebuilt to no longer need a separate server, making it simpler for institutions to manage and allowing for submission of NextBlocks to the Moodle Plugin Directory[11]. Finally, the entire user interface was rearranged to provide more space for developing programs and to make the terminal for input and output larger and easier to use.

These enhancements collectively transform NextBlocks into a more intuitive and pedagogically robust environment that combines the ease of visual programming with the functionality of traditional coding environments, culminating in its successful submission to Moodle's official plugin directory for greater accessibility.

5 Implementation

This chapter details how the new and changed features were implemented in the new version of Moodle's activity plugin NextBlocks, including decisions made, advantages and disadvantages.

Since this development was based on a version that was already previously developed, the general coding tools were kept from that version. Just like before, PHP was used in code related to Moodle's pages, while JavaScript is used for pages that come from Blockly[23].

In the previous NextBlocks version the minimum Moodle version required was 3.11 (build number 2021051700). However, the Build and Restore APIs (Section 5.1.2) were significantly changed during the Moodle 4.0 update. To avoid having to develop separate APIs for different versions, and taking into account that the 3.11 version is no longer supported by Moodle[41], the minimum Moodle version required was changed to 4.0 (build number 2022041900).

NextBlocks uses Blockly[23] version 10 and this was not changed. Even though versions 11 and 12 have been released, these only fix bugs and introduce minor features which are not useful for NextBlocks. The change to any of these versions would imply major changes to NextBlocks implementation without any advantages.

5.1 Moodle structure

Moodle[10] plugins go to different moodle folders based on the plugin type[42]; some examples are blocks plugins that go to the /blocks directory and authentication plugins that go to /auth. Since NextBlocks is an activity plugin, it goes to the /mod directory.

Activity plugins require specific features[43] to be accepted in Moodle's plugin directory[11]. Some of those features are the Privacy API, which manages user data, the Backup and Restore API, which allows teachers to backup and later restore courses with NextBlocks exercises included, and language strings which automatically translate the plugin to other languages.

5.1.1 Privacy API

Just like every other system that can be used in the European Union, NextBlocks has to be up to date with GDPR (General Data Protection Regulation)[44] rules. This can be fulfilled by implementing Moodle's Privacy API[45].

Moodle's Privacy API is a critical framework designed to help Moodle-based learning management systems comply with data protection regulations. It enables plugins to manage personal data transparently, ensuring users can exercise rights such as data access, export, and deletion[45]. Every moodle plugin that is available in Moodle's plugin directory[11] is required to have a working Privacy API.

Any user can request that their data is either exported to them or deleted from the site through the data requests tab, as shown in Figure 5.1. After this request is submitted, site administrators need to allow or deny the request, as seen in Figure 5.2. If the request is allowed, there are 2 scenarios:

- If the request is of export type, the user who created the request can go to the same page and download a zip file with all their data that is saved in that moodle site. That includes all user data and other data that is part of activities in which they are enrolled.
- If it is a deletion, then all data related to that user will be deleted. There are some plugins that, instead of deleting the data, anonymize it. In the case of NextBlocks, all the data from that user is permanently deleted.

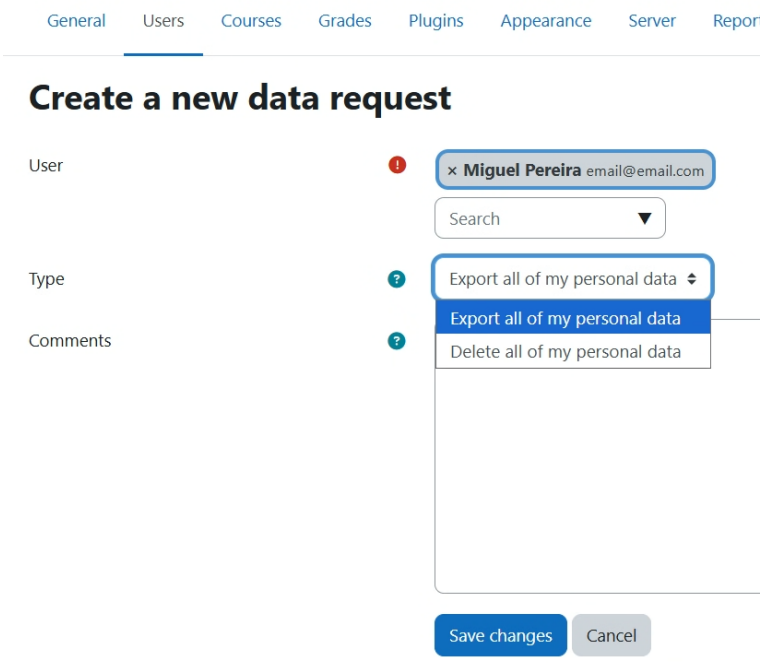


Figure 5.1: Moodle data request creation interface

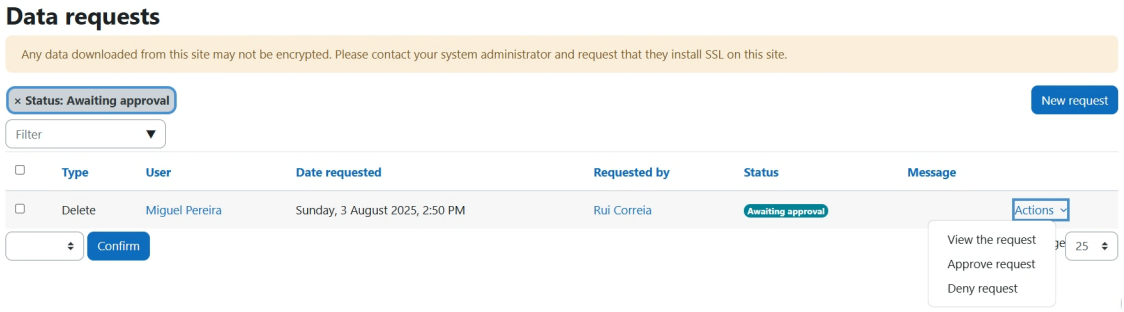


Figure 5.2: Moodle data request list

The activity diagram representing the process is shown in Figure 5.3.

The implementation of the Privacy API is all done in the provider.php file, which is present inside the classes/privacy directory. This file includes functions such as export_user_data and delete_data_for_user which handle the export and delete tasks which are available through this API.

5.1.2 Backup and Restore API

The Backup and Restore API[46][47] provides a standardized framework for saving and recreating Moodle courses, activities and user data. With this API implemented, course administrators can at any point backup an existing course to be restored later with all the same data existing at the time of backup. The restored data from NextBlocks activities include submissions, comments, chat messages, saved programs, test files and all the other information that is part of each exercise.

Through the course settings page, administrators can backup the course into a file with the extension .mbz which contains all the information relating to that course. This file can later be used to restore the course.

The implementation of the Backup and Restore API is done inside the backup/moodle2 directory and it consists of 4 files:

- backup_nextblocks_activity_task.class.php (backup activity task)
- backup_nextblocks_stepslib.php (backup steps library)
- restore_nextblocks_activity_task.class.php (restore activity task)
- restore_nextblocks_stepslib.php. (restore steps library)

Two of these files handle the backup process while the other two handle the restore. Each pair of files is composed of an activity task, which calls each one of steps necessary to backup/restore the course, and a steps library which defines what each step consists of. Since the backup and restore process of NextBlocks consists of duplicating files inside the database tables, modifying only the exercise ids, there is only one step for backup and one step for restore, which are called structure steps. In the backup process it copies all

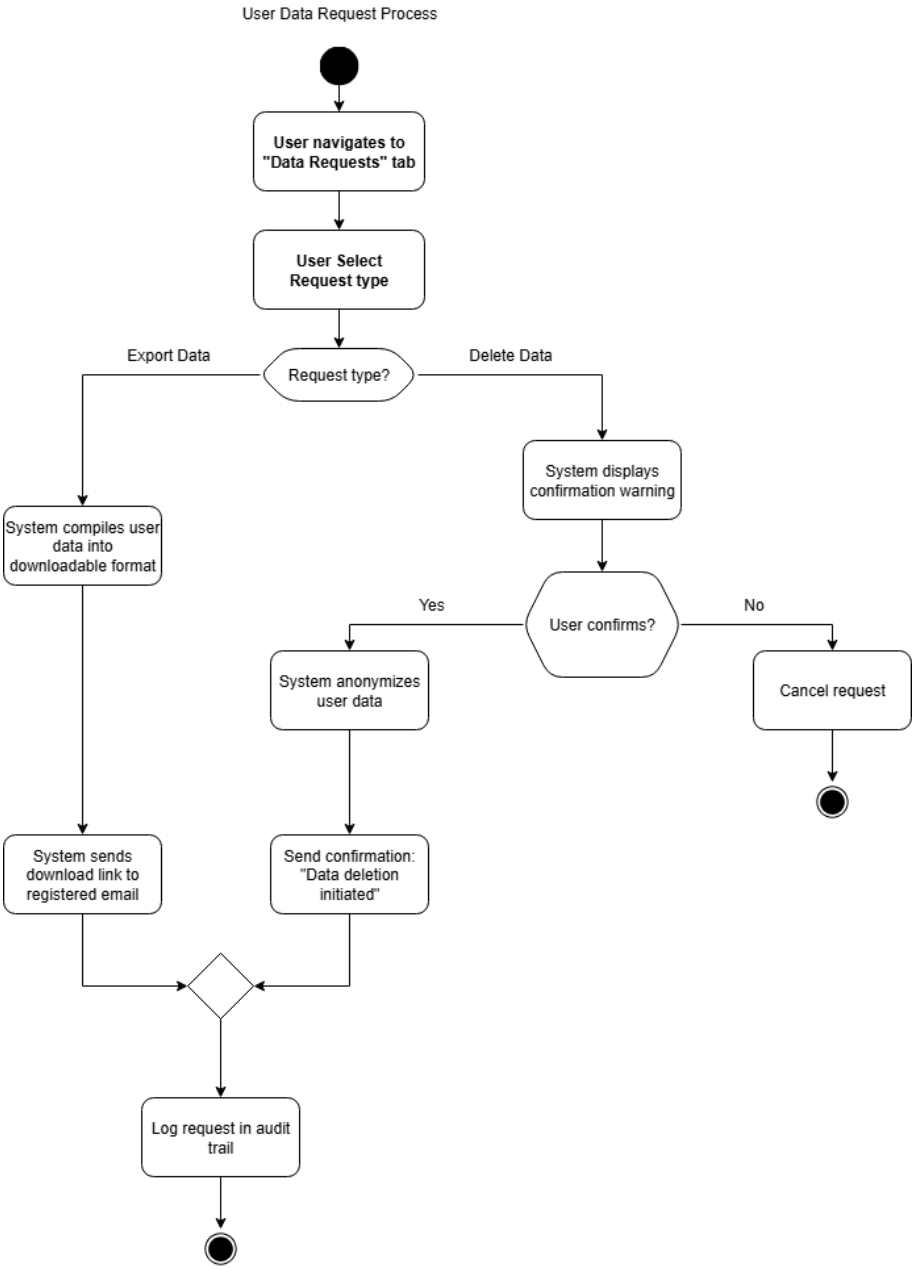


Figure 5.3: Moodle data request activity diagram

database files related to that course into the backup file. During the restore, it gets the database files from the backup file, modifies the needed ids to the new exercise id and puts them back in the database.

5.1.3 Language Strings

Moodle is available to everyone with internet access to use anywhere in the world. However, most moodle plugins are developed in a single language so they are often not accessible to everyone. To combat this issue Moodle developed AMOS (Automated Manipulation of Strings)[48] which automatically translates strings from Moodle plugins into multiple

languages.

For plugin developers, this means that they need to make sure that every string that can be visible to users cannot be inserted directly into the code, but instead in a separate language file which is then submitted into AMOS[48]. Each language string is identified by a string id. As an example, if a developer wanted to add a language string "NextBlocks" with the string id "modulename", this is how they would do it in the language file:

```
$string['modulename'] = 'NextBlocks';
```

In the code, to get strings from this language file the function changes depending on the type of file:

- For PHP files, developers can use the get_string function directly, that only takes the string id as parameter. An example of a call in a PHP file would be $str = get_string('string_id').
- For JavaScript files, they need to specify that the function is inside the 'str' library and what module it belongs to (in NextBlocks the module is always mod_nextblocks), so the call would be str.get_string('string_id', 'mod_nextblocks'). Since it is an asynchronous function it needs to be preceded by an await[49] keyword. An example of a call in JavaScript file would be let str = await str.get_string('string_id', 'mod_nextblocks').

Since the Privacy API is mandatory for plugins to be submitted into Moodle's plugin directory, NextBlocks was changed to make sure every user-facing string is inside the language file which is lang/en/nextblocks.php.

5.2 Database and files

NextBlocks employs a relational database schema within Moodle's existing infrastructure to ensure seamless integration while maintaining data integrity across diverse educational workflows. The schema, visualized in Figure 5.4, comprises eight core tables that manage everything from exercise configurations to real-time collaboration.

The main table is called nextblocks which stores core information about each exercise. This includes the course it belongs to (foreign key referencing Moodle's courses table), the exercise name and timestamps for creation and modification. Additional tables store exercise features like user data (with userid as foreign key from Moodle's users table), which also include data about the user's highest graded submission for that exercise, block limits, custom blocks, comments, and messages. All these tables, except comments, reference the main nextblocks table through nextblocksid to associate data with specific exercises. The comments table does not require this reference because the blockid values, which represent the id of the block which that comment is associated with, are globally unique across all exercises, making exercise identification unnecessary. However, since

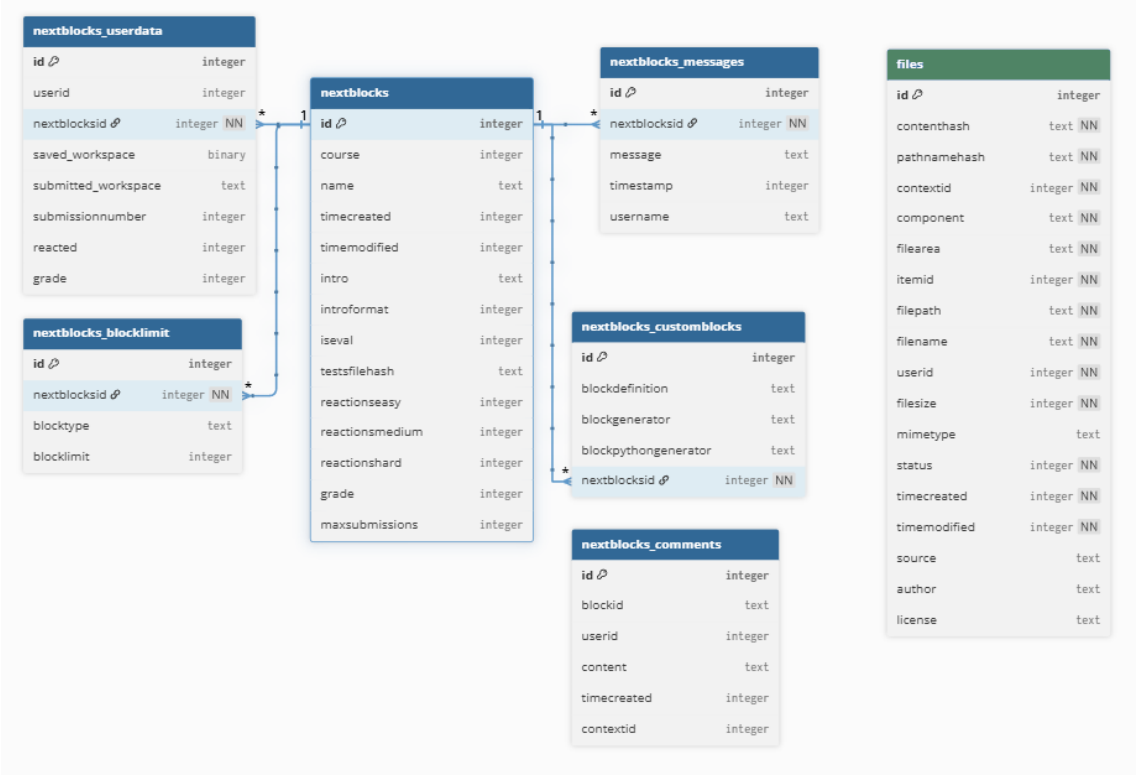


Figure 5.4: NextBlocks 2 database tables

blocks are not directly saved in a database table (they are instead part of the workspace object which is saved in the userdata table), the comments table does not have a reference to any other table.

Beyond the specific tables designed for its unique features, NextBlocks leverages Moodle's core database schema for fundamental activity data. Critical information such as student submissions, grades and general activity completion status are stored within Moodle's native tables, just like any other standard Moodle activity.

5.3 NextBlocks features

This subsection provides a comprehensive technical overview of how NextBlocks' new and improved features were implemented. Each feature has an explanation regarding its architectural foundations and execution workflows. The technical decisions reflect the focus on creating a robust, scalable programming environment that meets diverse educational needs and is well integrated in Moodle.

5.3.1 Implementation of the revamped input/output system

The introduction of the interactive input block explained in section 4.2 required a fundamental shift in NextBlocks' execution model. In traditional synchronous execution used in the previous version, blocking operations, such as waiting for user input, freeze the

browser's main thread, rendering the UI unresponsive. To maintain real-time interactivity while allowing program execution to pause for input, NextBlocks 2 implements an asynchronous, event-driven execution flow leveraging JavaScript's async/await paradigm[49] and Promise-based state management[50]. The execution workflow is shown in a diagram in Figure 5.5

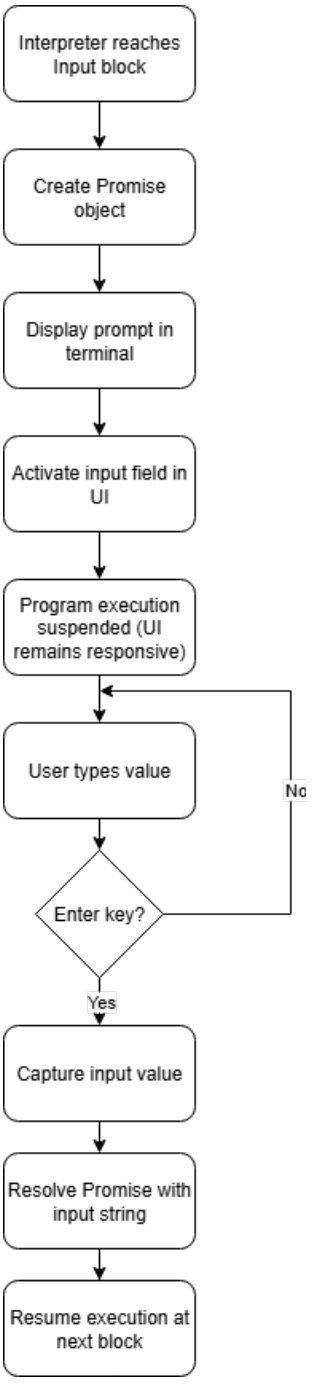


Figure 5.5: NextBlocks 2 input block execution workflow diagram

The implementation of an interactive terminal with an input block is crucial as it allows learners to create responsive applications while maintaining full browser interactivity

during input waits, allowing for the creation of dynamic and interactive programs that were not possible before.

5.3.2 Text to number block

The "text to number" block translates string inputs into numerical values using JavaScript's parseFloat[51] function. Its implementation wraps parseFloat with explicit edge-case handling: if the input string cannot be converted to a number, the parseFloat function returns NaN. NextBlocks' runtime checks for the return of every parseFloat function and, if it is NaN, aborts the execution with the error previously shown in Section 4.2, specifically in Figure 4.6.

The parseFloat function was chosen over alternatives like parseInt or the Number constructor because it correctly handles both integer and decimal numbers, which is essential for comprehensive math operations, while its behavior of parsing numbers until it encounters an invalid character provides a balance of flexibility and clear error cases for learners.

5.3.3 Implementation of reworked testing system

Instead of taking a single test file, now the system can take any number of test files, as explained in Section 4.3, but it converts them to a single file. This consolidation was done to maintain compatibility with the existing evaluation workflow, which was designed to process a single test file, thereby allowing the new, more flexible test structure to function without requiring a fundamental redesign of the execution engine. To consolidate the files, the environment creates an empty global test file that will include all test cases. Then, it takes each input file 1 by 1 and finds its corresponding output file. If the output file does not exist, it skips to the next input file. It adds a vertical bar (|) to the global file, indicating the start of a new test case. It then adds all the input values, one per line, followed by a dash (-) that separates input from output, and then the output value. Appendix B contains an example of a global test file created by the system from multiple test files introduced by a teacher. All the test related files are saved in the table "files" from Moodle. This table also includes files from other Moodle plugins that are installed. In order to avoid duplicated files with the same name, every nextblocks test file is renamed to contain the nextblocks_ prefix.

To evaluate a student's program, the testing system takes the consolidated test file and organizes all the input values in arrays that work as a buffer, where each test case will have its array of input values. This array is then passed to the program code that will be run. Every time an input block is executed, it will be attributed the first unused value in the array of input values. If there is an input block execution after the array has reached the last value, there is an error shown, as exemplified in section 4.3, Figure 4.10.

5.3.4 Usage of External API for program submission

NextBlocks now uses an external API to evaluate students submissions in order to provide safety and reliability. This is done through Sulu[52], an API monetization platform which combines multiple APIs in a single platform and offers an automated and simplified way to use them. To run the code from the programs and compare them with the expected outputs the chosen API was Judge0[53], an open-source online code execution system, that evaluates the submission and sends it to Moodle's gradebook. Judge0 was selected for its robust, open-source architecture and its proven reliability[54] in handling code execution securely within educational and competitive programming contexts, making it a trusted choice for automated assessment.

The submission of a program to Judge0 is done through the type "POST Submission batch", which allows the same program to be evaluated through multiple test cases sequentially. A POST Submission Batch request requires an array of submissions, each one referring to a specific test case. Each submission requires 4 parameters:

- **Expected Output:** The expected program output in this test case.
- **Language id:** In NextBlocks every program is evaluated using JavaScript, so the language id is always 102 (corresponding to Node.js 22.08.0).
- **Source code:** The program that is being evaluated
- **Input:** The input values of the test case, in a single string with different input values separated by \n.

The response of this POST request is an array of tokens which are used to identify each submission.

After the POST request, the results of the submission are obtained through a GET Submission Batch request. This submission requires an array of tokens and returns an array of submission results, one referring to each token. Each submission result has many fields about the submission, but the only value that NextBlocks uses is the status id, which is 3 if the submission passed all the tests. Other status IDs provide detailed feedback on submission failures, such as 4 for a compilation error, 5 for a runtime error or 6 for a time limit exceeded error. To calculate the grade, the Moodle server calculates the percentage of submissions that got the status id 3. If this submission gets a higher grade than all the previous submissions, it is saved as the submitted_workspace in the nextblocks_userdata table for that student. This submitted_workspace includes all blocks that are part of the workspace, as well as how they are connected. The saved_workspace field, which is often updated when a user clicks the "Save" button, is also updated when the user submits their program, even if it is not the best graded program.

The execution workflow when a student submits their program is shown in a diagram in Figure 5.6

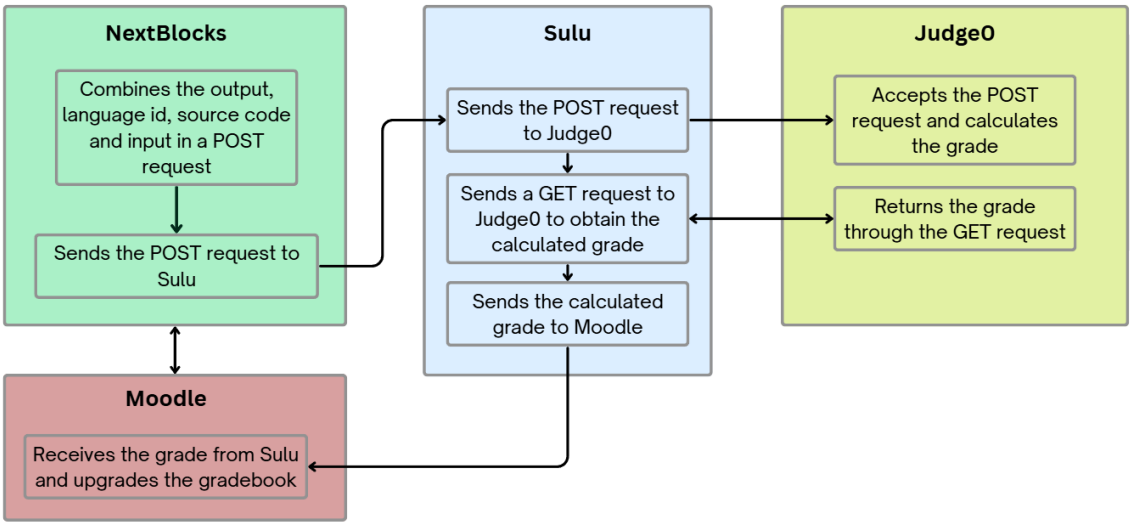


Figure 5.6: NextBlocks 2 external API usage execution workflow diagram

The usage of an external API for evaluation allows teachers to rely on the automatic grading due to additional security, knowing that it is much harder for students to manipulate the results.

5.3.5 Implementation of translation to Python

In the exercise creation interface there is a new box in the "Custom Blocks" tab where teachers can input Python translations to their custom blocks, which is then saved in the nextblocks_customblocks table in the database, in the same entry as the rest of the information about that custom block, such as its id, the definition, the translation to JavaScript and the exercise id.

When creating or evaluating a program, if the student or teacher clicks the button to see the Python translation, a script called codestringPython.js is called which handles the translation in two different ways, depending on where the block comes from: the blocks that are original from Blockly are translated using the Blockly.Python[55] library, while custom blocks are translated through the teacher's translation or, in case this was not made available, the block is translated to an undefined function with the same name as the block.

The code shown in Figure 5.7 is an example of a Python translation for a block called isEven that receives an integer as input and returns a boolean, true if the number is even, false otherwise. This is the code that the teacher would put in the Python translation field when creating the custom block.

The execution workflow for the translation to Python is shown in Figure 5.8

```
Blockly.Python.forBlock['isEven'] = function(block) {
  const numCode = Blockly.Python.valueToCode(
    block, 'NUM', Blockly.Python.ORDER_MODULUS
  ) || '0';
  const code = `(${numCode} % 2 == 0)`;
  return [code, Blockly.Python.ORDER_RELATIONAL];
};
```

Figure 5.7: NextBlocks 2 Python translation of isEven block

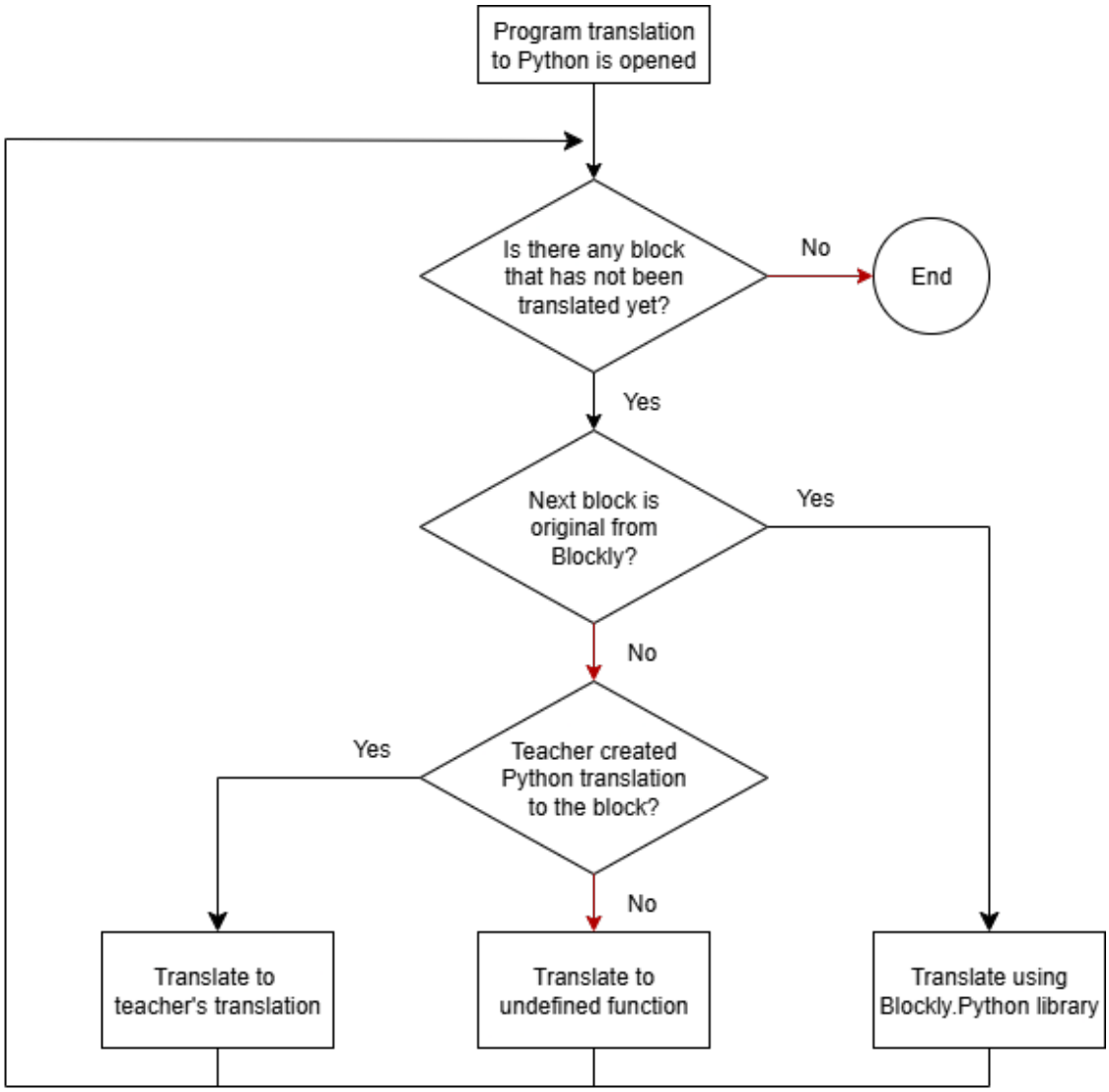


Figure 5.8: NextBlocks 2 Python translation execution workflow diagram

5.3.6 Implementation of block limits

When an exercise is saved, every block limit that is not infinite will be saved in the nextblocks_blocklimits database table. The id is a unique identifier for the entry, nextblocksid is the id of the course that the limit was created on (and a foreign key that references

the id from the nextblocks table). Blocktype is the identifier of the block type that the limit applies to and blocklimit is the maximum number of instances allowed of that block type, in that exercise.

Blockly[23] already has a block limits feature implemented called maxInstances[56]. When the system initializes the Blockly interface, it obtains the list of block limits from the database nextblocks_blocklimits table, ignoring limits from other exercises, and passes it under the name maxInstances to Blockly, which handles the toolbox changes on the student side.

When a student opens the tab to view block limits, a new window is opened that displays all blocks grouped by category, as shown in Figure 5.9. For each block, if there is no limit saved in the nextblocks_blocklimits table, a green infinite symbol is shown next to the block; otherwise the configured numeric limit obtained from the database table is shown in blue.

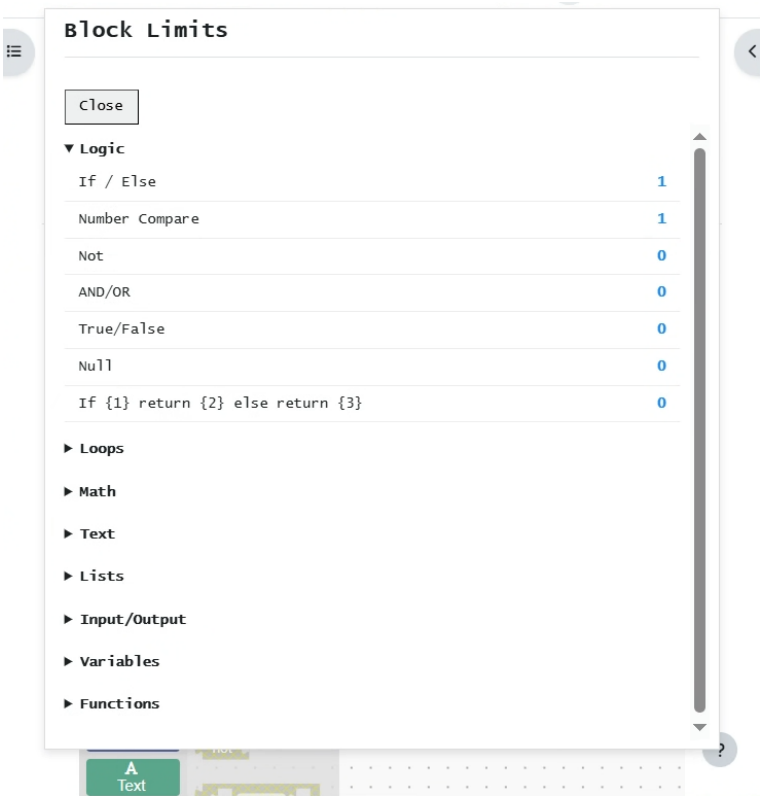


Figure 5.9: NextBlocks 2 block limits window in student interface

5.3.7 Implementation of improved comments system

NextBlocks 2 implements a new comments system and disables Blockly's native comments system. Even though the new system was implemented from scratch, the nextblocks_comments database table that was used in the previous version was kept unchanged since the backend of the system worked as intended.

When a teacher or student opens an exercise, the system goes through every block present in the workspace and searches the nextblocks_comments database table for any comments in that block. Every block with one or more comments gets a triangle warning icon added next to the block's name, as shown in Figure 5.10; clicking that icon opens the comments interface. The comments interface can also be opened from a context menu by right-clicking a block and selecting the "Comments" option.

Opening the comments interface loads all comments for the selected block from the nextblocks_comments database table and displays them. Both teachers and students may add new comments in this interface; new entries are added to the nextblocks_comments table.

Figure 5.10: NextBlocks 2 block comment triangle indicator

5.3.8 Implementation of reworked chat system

As previously explained in section 4.8, there was previously a php server in the server files that worked as a chat server. For chat to show up to teachers and students, the computer running the moodle server also had to be running this php server. Since this might not be viable for some institutions and it wouldn't allow for the plugin to be submitted to Moodle Plugin Directory[11], the chat server was replaced with a message system that aims to simulate a chat.

The nextblocks_messages database table that was used in the previous version was kept unchanged since it had all the necessary parameters.

When a student or teacher opens the interface of an exercise, a timer starts in their page using JavaScript's setInterval function[57]. Every 3 seconds, the runtime checks the nextblocks_messages database table for any new messages and updates the chat box if necessary. If a user sends a message through the chat, this message is shown in the message list and added to the same database table. Every time a user opens the interface of an exercise it automatically loads every previously sent chat message in that exercise from the database and displays them.

5.4 Conclusion

This chapter has outlined the technical implementations that constitute NextBlocks 2. The development successfully enhanced the plugin with the major features proposed, focusing on robust integration within the Moodle ecosystem.

The core achievement was the introduction of an interactive input/output system, which transformed the platform from a static environment to a dynamic one where programs can respond to user input in real-time. This was supported by a reworked testing system that adopted a more standard and manageable file-based approach.

To ensure reliability and security, the automatic grading system was migrated to an external API, preventing potential manipulation of results.

Significant educational features were added, including translation to Python to bridge the gap between blocks and text-based code, and a Block Limits system to allow teachers to guide student solutions. The user experience was refined through an improved comments system, a simplified chat that no longer requires an external server and a restructured interface for better workflow.

After fulfilling Moodle plugin directory[11]'s requirements[43], such as the implementation of the Privacy, Backup and Restore APIs, NextBlocks was submitted for review on the directory on the 18th of May 2025 and it was accepted and published on the 25th of June 2025. It is publicly available, as shown in Figure 5.11, through Moodle's plugins page[1].

The successful submission and acceptance of NextBlocks into the official Moodle Plugin Directory marks the culmination of this development effort. It validates the plugin's quality and makes it accessible to a global audience, fulfilling a primary objective of this thesis and setting the stage for worldwide availability.

[1] NextBlocks in Moodle, https://moodle.org/plugins/mod_nextblocks

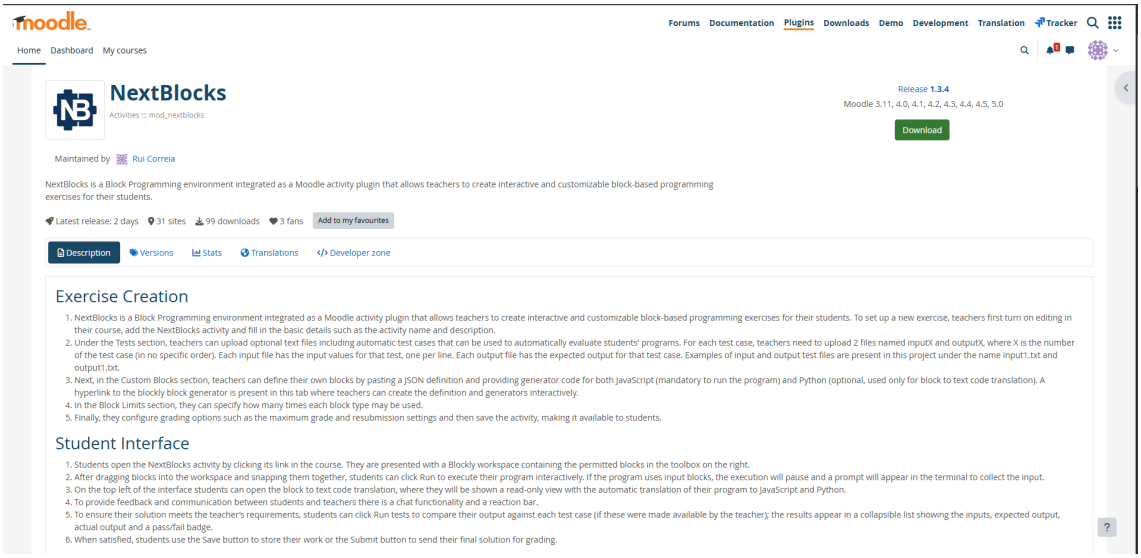


Figure 5.11: NextBlocks Moodle page

6

Evaluation

This chapter presents the evaluation process for NextBlocks 2, designed to assess its functionality and effectiveness. The platform was tested through structured user sessions with its target audiences and presented to the international educational community. The results of these evaluations provide insight into the platform's usability, pedagogical value and identified environment issues and possible solutions.

6.1 User testing

Structured user testing was conducted to evaluate the usability and functionality of NextBlocks 2. The testing focused on the two main user groups: teachers configuring exercises and evaluating submissions, and students solving programming exercises. This section details the methods and outcomes of these tests.

6.1.1 Teacher testing

The teacher-oriented features of NextBlocks 2 were evaluated through structured testing sessions with ten educators possessing diverse teaching backgrounds. The guide shown to the teachers is available in Appendix C. Participants represented a wide spectrum of teaching experience, ranging from 1 to 45 years, with a mean of 23 years and median of 25 years. All participants had previously taught university-level introductory programming courses, with eight having prior experience with block programming environments. Moodle usage frequency varied among participants: four reported using it rarely, two sometimes, three often, and one daily.

6.1.1.1 Quantitative Assessment of Features

Participants evaluated specific features using a 5-point Likert scale where 1 represented "very hard/not useful at all" and 5 represented "very easy/very useful."

Ease of Use Assessments:

The questions asked to teachers relating to ease of use focused on 3 key points: Block

Limits, Comments and NextBlocks overall. The graph representing these results is shown in Figure 6.1. Block Limits was deemed to be the easiest feature to use for teachers, however the comments system and NextBlocks overall were also not considered hard to use, with every teacher rating each feature with a neutral or positive evaluation relating to ease of use.

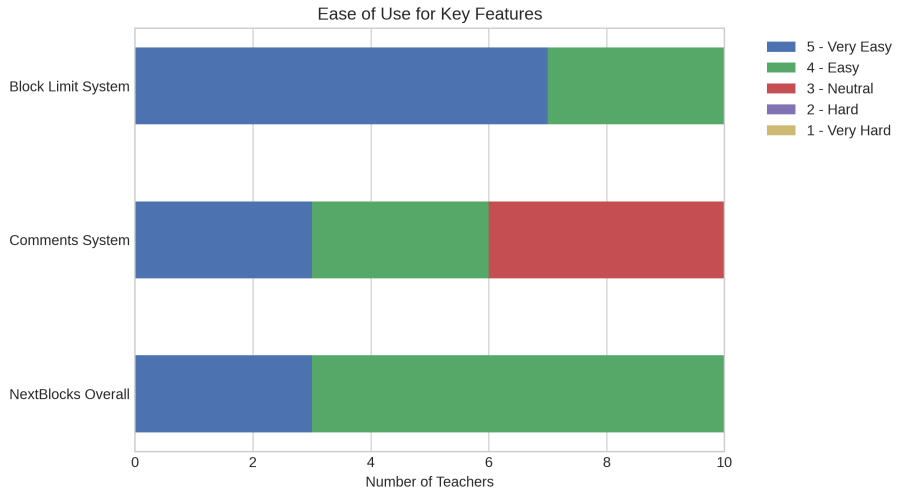


Figure 6.1: Graph representing teachers' user testing answers relating to NextBlocks' ease of use

Usefulness Assessments:
Even though translation to python and the interactive terminal were not asked in the "Ease of use" questions because teachers did not have to interact with them directly, they were still asked about in the usefulness questions. The results of these questions are shown in the graph from Figure 6.2.

The block limit system achieved very good usefulness scores, with all ten participants rating it very useful. The other features achieved worse scores than the block limits, but every teacher suggested that every feature is useful, with a majority of the teachers rating them as very useful.

6.1.1.2 Agreement Statements Assessment

The chart shown in Figure 6.3 shows how the ten teachers rated their agreement with different statements about the teaching ideas behind NextBlocks. These answers provide insights on the plugin's educational value.

There was a very strong agreement on several key teaching concepts. All teachers strongly agreed that using block limits is a good way to force students to explore different ways to solve a problem. A large majority also agreed on other core ideas. Nine teachers strongly agreed that the comment system improves communication between teachers and students. Similarly, eight teachers strongly agreed that being able to see the Python translation helps students move from block programming to text programming and six

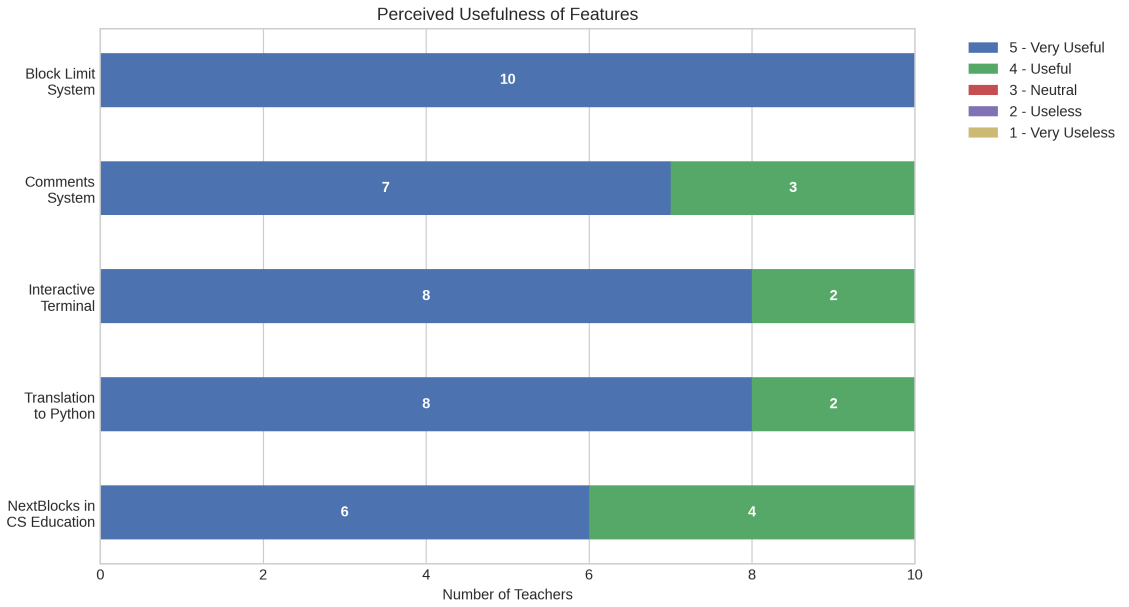


Figure 6.2: Graph representing teachers' user testing answers relating to NextBlocks' usefulness

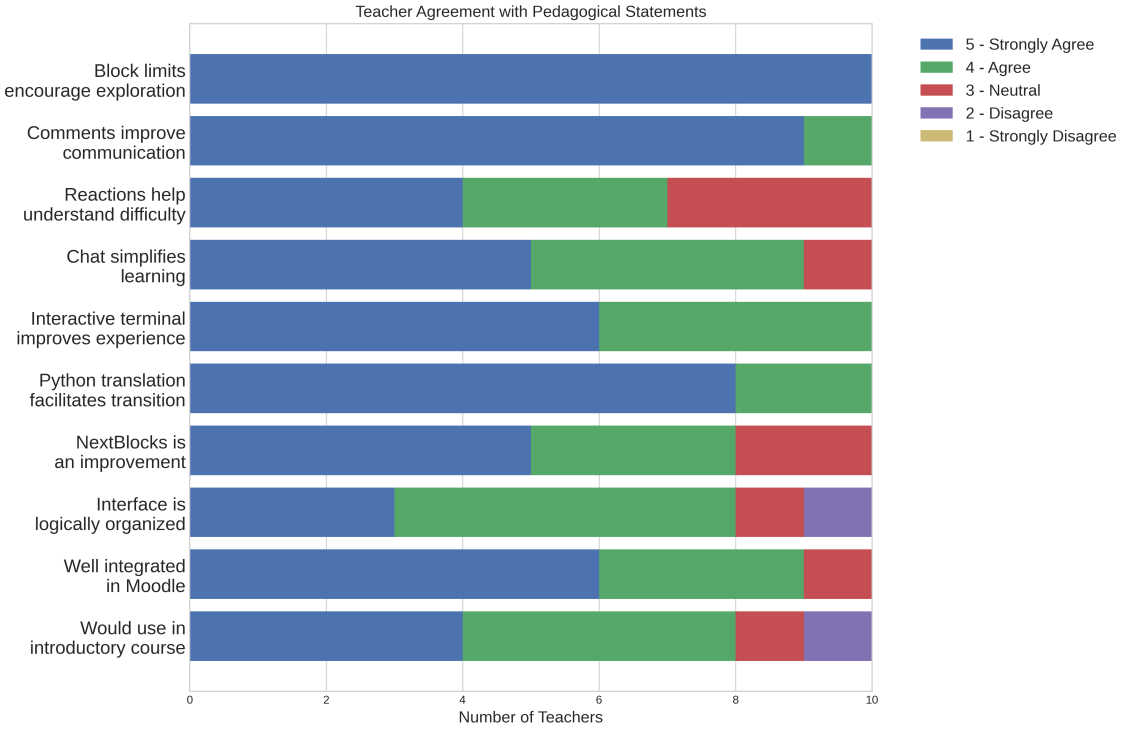


Figure 6.3: Graph representing teachers' user testing agreement answers

strongly agreed that the interactive terminal improves the learning experience, with the other four agreeing.

Five teachers strongly agreed that the chat simplifies the learning process, with four more agreeing. Opinions on the interface were divided: three teachers strongly agreed it

is logically organized, five agreed, one was neutral, and one disagreed.

The most uncertain opinions were about the reactions system. Only four teachers strongly agreed that the reaction buttons help them understand how difficult an exercise was for students. Three others agreed, and three were neutral, meaning this feature might need to be improved.

Finally, when asked about the future, most teachers were positive. Six teachers strongly agreed that NextBlocks is well integrated into Moodle, and four teachers each strongly agreed or agreed that they would use it in an introductory programming course.

These results confirm that teachers strongly support the main teaching ideas in NextBlocks, especially for guiding student learning and helping them transition to text-based coding. It also shows which areas, like the reactions system, might need more work to be as effective.

6.1.1.3 System Usability Scale

To obtain a standardized and reliable measure of the platform's usability, participating teachers were asked to complete the System Usability Scale (SUS)[58] questionnaire. The SUS is a well-established tool consisting of ten questions that provide a global view of subjective usability assessments.

The results yielded an average SUS score of 93.25 out of 100. According to the accepted benchmarks for SUS scores, this result falls within the "excellent" range[59] and is equivalent to an "A+" grade[60].

This outstanding score strongly reinforces the positive feedback gathered the other questions. It objectively confirms that teachers found NextBlocks 2 to be intuitive, easy to navigate and not cumbersome to use. The high SUS score underscores the success of the interface restructurings and feature implementations previously detailed, demonstrating that the plugin's advanced functionalities are presented in an accessible and user-friendly manner for educators and students.

6.1.1.4 Qualitative Feedback

Open-ended feedback provided specific insights that have already guided platform improvements. Regarding the block limit system, multiple participants noted that the input block was initially miscategorized within the "Text" category rather than a more logical dedicated category. This feedback directly resulted in the creation of a new "Input/Output" category and proper reclassification.

For the comments system, one participant suggested changing the comment icon from a triangle to a more intuitive symbol, while another recommended adding functionality for general submission-level comments alongside the existing block-specific comments.

The reactions system generated suggestions for adding descriptive titles to each reaction icon to clarify their meaning for users. Regarding the chat functionality, one participant suggested adding a title to the chat interface, while another proposed replacing the

permanently visible chat bar with a clickable bubble icon that would expand when needed.

The Python translation feature received particularly positive comments, with multiple participants highlighting its value for both teaching assessment and student learning. Teachers noted that seeing the Python equivalent helps them evaluate submissions more efficiently while helping students understand how block constructs translate to text-based programming.

Several participants offered general observations that NextBlocks would be particularly valuable for middle and high school students with no programming experience, though some expressed reservations about its suitability for university-level computer science courses. This feedback suggests the platform may be most beneficial in introductory and pre-university educational contexts.

The comprehensive feedback collected from teacher testing provided valuable direction for future development priorities. The strong positive ratings for core innovative features like block limits and Python translation, combined with specific constructive feedback, demonstrate both the platform's current effectiveness and its potential for continued refinement toward optimal classroom integration.

6.1.2 Student testing

The student-oriented experience of NextBlocks 2 was evaluated through structured testing sessions with a group of students with different levels of experience. The guide shown to the students is available in Appendix D. Participants represented a range of prior experience, with the number of years of programming classes varying from 1 to 6, with a mean of 3.8 and average of 4.5. Seven of the students reported having prior experience with block programming environments and their frequency of Moodle usage was diverse as shown in Figure 6.4.

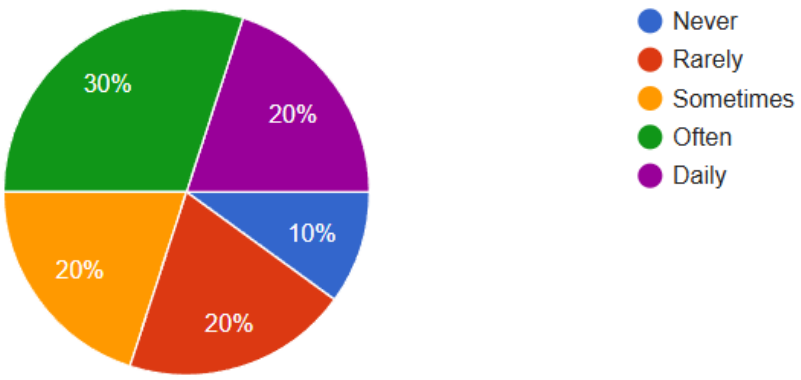


Figure 6.4: Graph representing students' utilization of Moodle

6.1.2.1 Quantitative Assessment of Features

Students evaluated specific features using a 5-point Likert scale where 1 represented "very hard/not useful at all" and 5 represented "very easy/very useful."

Ease of Use Assessments:

As shown in the results from the graph in Figure 6.5, students found NextBlocks highly intuitive, with the overall platform receiving high ease-of-use scores. The interactive terminal was particularly highlighted for its simplicity, closely mimicking the input behavior of text-based languages. The comments system and the block limit indicator were also rated as straightforward to use, indicating that the interface is intuitive.

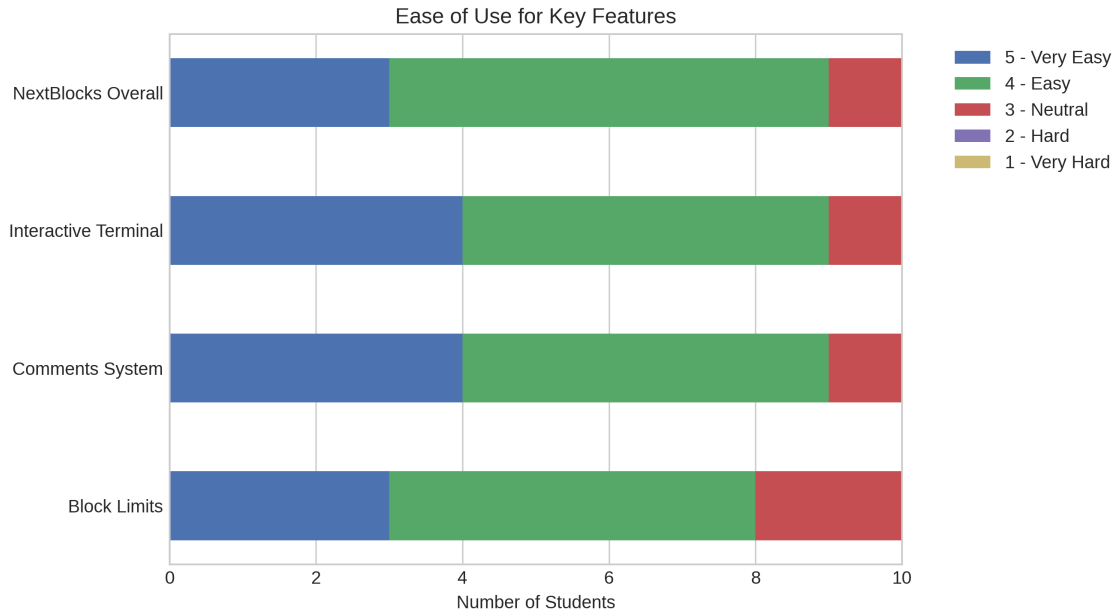


Figure 6.5: Graph representing students' user testing answers relating to NextBlocks' ease of use

Usefulness Assessments:

The perceived usefulness of NextBlocks' features was overwhelmingly positive as shown in Figure 6.6. Students rated the platform as highly useful for computer science education. The interactive terminal was praised for making programming feel more dynamic, and the automatic test execution with detailed feedback was considered crucial for independent learning. Most students agreed that comments are useful for communication between students and teachers and the automatic translation to Python was identified as one of the most useful features, providing a clear bridge to text-based coding.

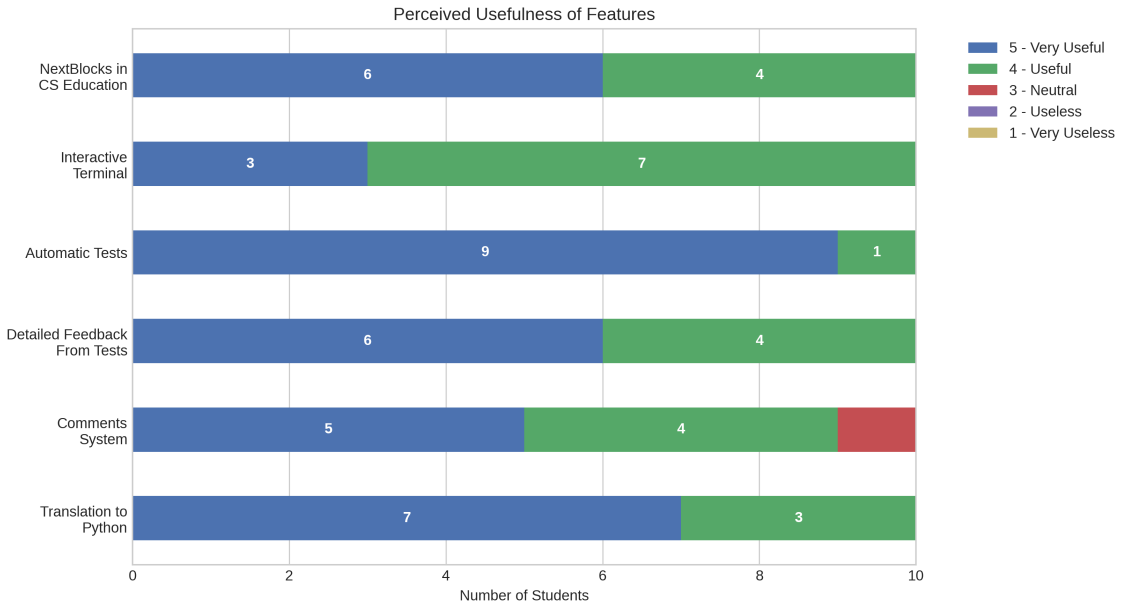


Figure 6.6: Graph representing students' user testing answers relating to NextBlocks' usefulness

6.1.2.2 Learning Perspective Agreement Statements

Students expressed strong agreement with statements about the educational value of NextBlocks, as shown in Figure 6.7. An overwhelming majority agreed that the interactive terminal made programming more dynamic and interesting, while also facilitating debugging. The ability to see the Python translation was almost universally endorsed as a key facilitator for transitioning from blocks to text.

There was strong consensus that NextBlocks is a significant improvement over traditional block programming environments (even though 3 students preferred not to answer this question due to lack of knowledge relating to block programming environments) and that its integration into Moodle is a major advantage. Students agreed that the platform is effective for learning fundamental programming concepts like loops, variables, and logic, and nearly all participants affirmed that it is an excellent tool for introducing programming to complete beginners.

Opinions on the interface were positive, with most students finding it logically organized. The chat and comment systems were also viewed favorably, with students agreeing that these features simplify the learning process and improve communication with teachers.

6.1.2.3 System Usability Scale

To complement the teacher assessment, students also completed the System Usability Scale (SUS) questionnaire. The results yielded an average SUS score of 89.75 out of 100 which, according to standard benchmarks, also falls within the "excellent" range[59] and

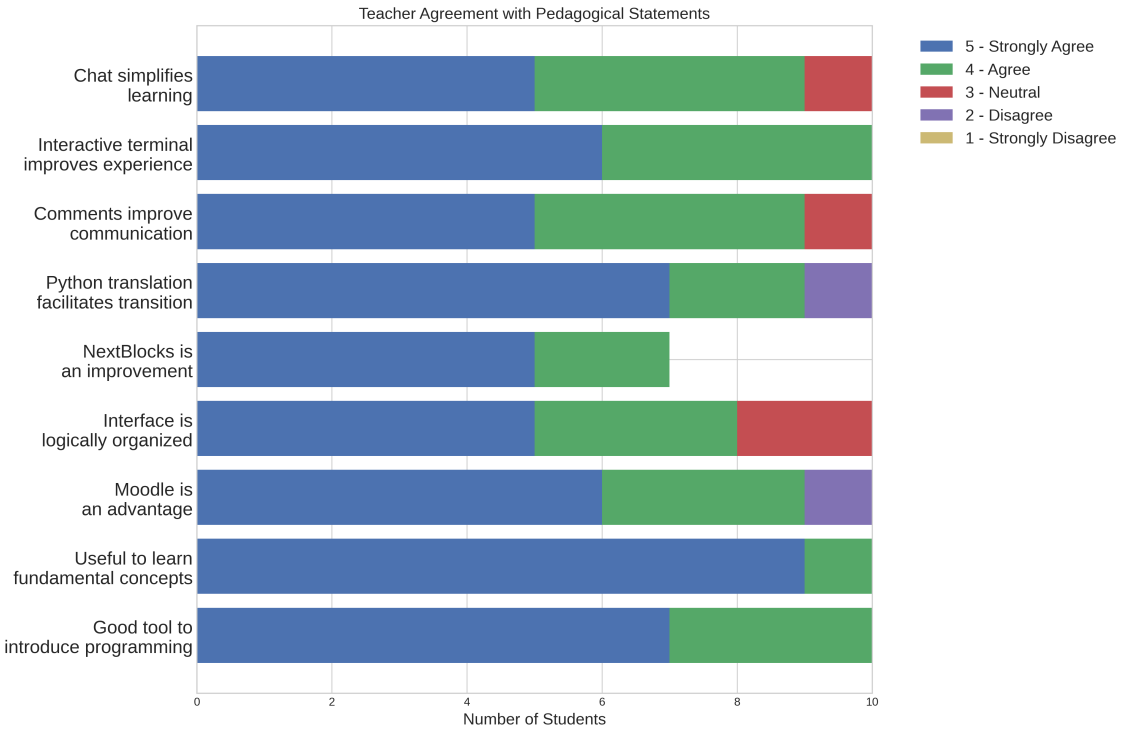


Figure 6.7: Graph representing students' user testing agreement answers

corresponds to an "A+" grade[60].

This outstanding score from the student perspective objectively confirms that NextBlocks 2 is not only powerful from a pedagogical standpoint but also intuitive and satisfying for the learner.

6.1.2.4 Qualitative Feedback

Open-ended feedback from students provided valuable insights into their experience, highlighting both strengths and areas for future refinement.

The feedback was overwhelmingly positive regarding the platform's core functionality. Many students spontaneously mentioned the engagement factor of the interactive terminal, noting that it made their programs feel more dynamic. The automatic translation to Python was frequently cited as a key feature for understanding and learning the programming concepts. The block limits feature, while sometimes described as challenging, was acknowledged by several students as a useful constraint that pushed them to think smarter.

However, the feedback also revealed specific usability challenges. A number of students reported difficulty discovering how to add comments, with the requirement to right-click on a block not being an intuitive action for all users. This suggests a need for a more discoverable UI element for initiating comments.

Additionally, several students suggested that information about block limits would be more accessible if displayed directly next to each block in the toolbox, rather than in a

separate window. While this is a valuable suggestion, its implementation is currently constrained by the underlying Blockly[23] library, which does not natively support modifying the toolbox interface to that extent.

The comprehensive feedback from students confirms that NextBlocks 2 successfully achieves its goal of being an engaging and effective tool for novice programmers. The constructive criticism provides clear, actionable directions for enhancing usability in future iterations, while the positive reception underscores a strong product-market fit for its intended educational context.

6.1.3 Comparative Analysis

The results of the user testing show a clear improvement in usability and user satisfaction from NextBlocks 1 to NextBlocks 2, both for teachers and for students.

In the original version, teachers reported an average System Usability Scale (SUS) score of 78.3, which already corresponds to a "good" level of usability. However, in NextBlocks 2 this value increased significantly to 93.25, placing the system in the "excellent" range and very close to the maximum score. This improvement indicates a much stronger perception of ease of use, clarity and overall quality by educators in the newer version.

A similar trend is observed in student evaluations. In NextBlocks 1, students obtained an average SUS score of 77.5, which also corresponds to a good usability level. In NextBlocks 2, the student SUS score rose to 89.75, representing a substantial increase and placing the platform in the "excellent" usability category. This suggests that students found the second version noticeably easier to understand, more intuitive and more pleasant to use.

This progression is particularly relevant because SUS improvements of this magnitude usually reflect a strong refinement of interaction design and user experience. The consistency of the improvement across both user groups strengthens the conclusion that NextBlocks 2 represents a significant evolution in usability when compared to the original version.

Furthermore, the results show a reduction in usability variability. In NextBlocks 1, responses were more dispersed, indicating mixed perceptions among participants. In NextBlocks 2, scores are more uniformly high, suggesting a more stable and predictable user experience across different profiles of users.

In summary, the user testing results demonstrate that NextBlocks 2 achieves a higher level of usability than NextBlocks 1, with both teachers and students rating the platform in the top usability tier. This confirms that the second version not only maintains the positive reception of the original system but significantly strengthens it in terms of perceived quality and user satisfaction.

6.2 Conference presentation

On July 1, 2025, NextBlocks 2 was presented at the EDULEARN25[61] international education conference in Palma de Mallorca, Spain. This gathering of 800 educators and technology specialists from different education levels and backgrounds provided an ideal platform to showcase the plugin's pedagogical innovations. During the session, attendees explored NextBlocks' unique features through live demonstrations and use-case scenarios.

Educators were given access to the plugin where they could experiment with the environment in any way they found appropriate, such as creating customized programming exercises with block limits and testing the interactive terminal. Valuable qualitative feedback emerged from these sessions: some high school teachers praised the intuitive UI for young learners, while university instructors highlighted the Python translation feature as a critical bridge to text-based coding. All attendees received direct access to the Moodle plugin directory link and many showed interest in using the plugin in the future.

This conference served not only as validation of NextBlocks' educational value but also helped spread it to institutions worldwide. The reviewed article about NextBlocks was published in the EDULEARN25 Proceedings[62].

6.3 Conclusions

This chapter presented a comprehensive evaluation of NextBlocks 2, focusing on its usability, pedagogical value and overall effectiveness through structured user testing with both educators and students, as well as its presentation at an international educational conference.

The user testing revealed overwhelmingly positive feedback from both target groups. Teachers rated the platform highly in terms of ease of use and usefulness, particularly highlighting the Block Limits feature as a powerful tool for guiding student learning. The System Usability Scale (SUS) scores, 93.25 for teachers and 89.75 for students, confirm that NextBlocks 2 is not only functionally robust but also exceptionally user-friendly. Students appreciated the interactive terminal and the Python translation feature, which together create a more dynamic and transparent learning experience.

Qualitative feedback provided actionable insights for future improvements, such as enhancing the reactions system, improving comment visibility and refining the chat interface. The conference presentation at EDULEARN25[61] further validated the platform's relevance and potential for global adoption, with educators expressing strong interest in integrating NextBlocks into their teaching practices.

In summary, the evaluation demonstrates that NextBlocks 2 successfully meets its design goals of being an interactive, customizable and pedagogically effective block programming environment. The positive reception from both users and the academic community underscores its readiness for real-world educational deployment and its potential to significantly enhance programming education.

7

Conclusions and Future Work

This dissertation documented the design, implementation and evaluation of NextBlocks 2, a significant evolution of a block programming environment deeply integrated into the Moodle[10] learning management system. The project was driven by the goal of enhancing the original plugin with innovative features that address pedagogical gaps in existing tools, focusing on interactivity, customizability and a structured educational workflow. This chapter provides a conclusive summary of the work, reflects on the achievement of the initial objectives, proposes directions for future development and offers final considerations on the project's contribution and impact.

7.1 Conclusions

The development of NextBlocks 2 was guided by a set of clear objectives aimed at transforming the plugin into a more dynamic and pedagogically powerful tool. The analysis of the implemented features and the results from the rigorous evaluation with educators allow for the following conclusions:

The core objectives outlined at the beginning of this thesis were successfully met. The introduction of the interactive input/output system fundamentally altered the nature of programming within NextBlocks. By implementing the dynamic input and text to number blocks, the platform transitioned from a static execution model to a dynamic one, allowing students to create programs that respond to user input in real-time. This achievement successfully bridges a significant gap between block-based and traditional text-based programming, offering a more authentic and engaging coding experience.

Furthermore, the objective of providing multi-language code translation was not only achieved but also identified as one of the most impactful features. The addition of Python translation, alongside the existing JavaScript functionality, was overwhelmingly praised by educators during testing. Teachers highlighted its crucial role helping students understand the connection between visual blocks and textual code, thereby facilitating a smoother transition to professional programming environments, as well as helping teachers evaluate student submissions in a timely manner.

The implementation of the Block Limits feature directly addressed the goal of empowering teachers to guide student learning and prevent the development of poor coding practices. The feedback from teachers was unequivocal, with all participants rating this feature as "very useful". This was further supported by student feedback, which acknowledged that block limits encouraged more thoughtful problem-solving, despite the initial challenge. This strong positive reception confirms the pedagogical value of allowing educators to strategically constrain tool usage to encourage deeper computational thinking and explore diverse problem-solving approaches.

From a technical standpoint, the project significantly enhanced the robustness, security and integration of the plugin. The migration of the automated grading system from the client-side to the external Judge0 API via Sulu dealt with a critical vulnerability, making the evaluation process secure and reliable. Moreover, the full compliance with Moodle's stringent development standards through the implementation of the Privacy, Backup and Restore APIs was a pivotal achievement. This effort culminated in the official acceptance and publication of NextBlocks in the Moodle Plugin Directory, a testament to the project's quality and a milestone that guarantees its accessibility to a global community of educators and students.

Finally, the positive evaluation by educators and students, the successful presentation at the EDULEARN25[61] international conference and the more than 150 downloads[63] of the plugin in the Moodle Plugin Directory[11] serve as strong external validation of the project's relevance and contribution. The high System Usability Scale (SUS)[58] scores from both teachers (93.25) and students (89.75) objectively confirm that NextBlocks 2 is not only pedagogically effective but also highly usable and intuitive. The feedback gathered not only confirmed the utility of the new features but also provided valuable, actionable insights that were incorporated into the final design. The academic recognition further values the significance of this work in the field of computer science education technology.

In summary, NextBlocks 2 has evolved from a promising prototype into a mature, feature-rich and pedagogically valuable platform that effectively addresses the limitations of existing block programming environments. It successfully fulfills its purpose of being a highly customizable, interactive and integrated tool that supports both teachers and students in the learning process.

7.2 Future Work

While NextBlocks 2 represents a substantial advancement in block programming educational tools, the field of educational technology is continuously evolving. Based on the development experience, the feedback received and current trends in computer science education, there are still some improvements that can be made to NextBlocks:

- **Refinement of the reactions system:** As indicated by the teacher evaluations, the reactions system was the feature with the most neutral feedback. A future iteration

could enhance its utility by adding descriptive labels (e.g., "Too Easy,", "Neutral", "Too Hard") and potentially adding more reactions.

- **Expansion of commenting functionality:** Acting on feedback from both teachers and students, future work should address the discoverability of the comment system. Many students reported difficulty finding how to add comments, as the right-click requirement was not intuitive. Additionally, teachers suggested implementing general, submission-level comments alongside the existing block-specific comments.

- **Integration of block limits display in toolbox:** Student feedback indicated that block limit information would be more accessible if shown directly in the toolbox next to each block, rather than requiring a separate window. This would provide immediate visibility of constraints during the creation of the program.

- **Addition of a chat icon:** Some teachers indicated that the chat is hidden at the bottom of the interface, making it harder to find and interact with. A future fix for this issue would be adding a chat bubble that is always shown to a teacher or student viewing the exercise. When this bubble is clicked the chat window would open.

- **Translation to additional languages**: Expanding beyond Python and JavaScript, support for other languages common in computer science education (e.g. C, Java) could widen NextBlocks' applicability.

- **Advanced visualization and debugging tools:** Integrating a step-by-step debugger with visualizations of variable states and program flow would be a significant aid for comprehension. This would help students internalize the execution model and understand the runtime behavior of their programs, making debugging a more intuitive and educational process.

- **Real-Time Program Monitoring:** Implementing a teacher dashboard that displays students' workspaces live as they develop their programs. This would allow educators to monitor progress, identify struggles early and provide live assistance.

- **AI-assisted features**: Integrating generative AI could offer powerful new functionalities. An AI tutor could analyze a student's code to provide real-time, personalized hints. An AI-powered assessment assistant could help teachers by automatically generating initial test cases or providing a first-pass analysis of submitted code for common errors.

7.3 Final Considerations

The development of NextBlocks 2 has been a rewarding journey that includes software engineering and pedagogical design. This project highlights the critical importance of

building educational tools that are not only technically robust but also deeply embedded within the practical ecosystems where teaching and learning occur.

The positive reception from educators at the EDULEARN25[61] international conference and the subsequent publication of the work in the conference proceedings serve as strong endorsements from the academic community. This recognition confirms that NextBlocks 2 addresses a genuine need and makes a meaningful contribution to the field of computer science education.

It is desired that NextBlocks 2 will serve as an effective and inspiring resource for educators worldwide, empowering them to introduce programming concepts in a way that is both engaging and effective. By combining the accessibility of block-based programming with the power of customization, real-time interaction and LMS integration, this project contributes to the vital goal of facilitating computational thinking and programming education.

Bibliography

[1] J. M. Lourenço. *The NOVAthesis LaTeX Template User's Manual*. NOVA University Lisbon. 2021. URL: https://github.com/joaomlourenco/novathesis/raw/main/template.pdf (cit. on p. i).

[2] J. M. Wing. "Computational Thinking: What and Why?" In: *Proceedings of the 24th Australasian Computing Education Conference*. 2010. URL: https://api.semanticscholar.org/CorpusID:63382972 (cit. on p. 1).

[3] S. Abesadze and D. Nozadze. "Make 21st Century Education: The Importance of Teaching Programming in Schools". In: *International Journal of Learning and Teaching* (2020-01), pp. 158–163. DOI: 10.18178/ijlt.6.3.158-163 (cit. on p. 1).

[4] M. Guzdial. "Programming Environments for Novices". In: *Proceedings of the 24th Australasian Computing Education Conference*. 1997. URL: https://api.semanticscholar.org/CorpusID:6339931 (cit. on p. 1).

[5] A. P. J. Perin, D. E. dos S. Silva, and N. M. C. Valentim. "Investigating block programming tools in high school to support Education 4.0: A Systematic Mapping Study". In: *Informatics in Education* 22.3 (2023), pp. 463–498. ISSN: 1648-5831. DOI: 10.15388/infedu.2023.21 (cit. on pp. 1, 9, 11).

[6] J. Fagerlund et al. "Computational thinking in programming with Scratch in primary schools: A systematic review". In: *Computer Applications in Engineering Education* 29 (2020), pp. 12–28. URL: https://api.semanticscholar.org/CorpusID:218938698 (cit. on pp. 2, 12, 24, 28).

[7] U. Kale. "Still a New Kid on the Block? Computational Thinking as Problem-Solving in Code.org". In: *Proceedings of the 2021 AERA Annual Meeting* (2021). URL: https://api.semanticscholar.org/CorpusID:268115664 (cit. on pp. 2, 12, 24, 28).

[8] *NOVA FCT*. 2025. URL: https://www.fct.unl.pt/en (visited on 2024-01-15) (cit. on p. 2).

[9] D. Pereira, F. Barbosa, and C. Morgado. "NextBlocks: An Interactive Block Programming Platform". In: *Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1*. ITiCSE 2024. Milan, Italy: Association for Computing Machinery, 2024, pp. 590–596. ISBN: 9798400706004. DOI: 10.1145/3649217.3653609. URL: https://doi.org/10.1145/3649217.3653609 (cit. on pp. 2, 8, 12, 23, 28).

[10] S. H. Gamage, J. R. Ayres, and M. B. Behrend. "A systematic review on trends in using Moodle for teaching and learning". In: *International Journal of Stem Education* 9 (2022). URL: https://api.semanticscholar.org/CorpusID:246281909 (cit. on pp. 2, 5, 12, 42, 67).

[11] *Moodle plugin directory*. 2025. URL: https://moodle.org/plugins/?q= (visited on 2025-01-15) (cit. on pp. 2, 9, 12, 39, 41–43, 54, 55, 68).

[12] *Changing the Grader Report / Gradebook preferences in Moodle*. 2021. URL: https://www.inmotionhosting.com/support/edu/moodle/grader-report-preferences/ (visited on 2024-12-26) (cit. on p. 8).

[13] *How do I View my Grades in Moodle?* 2024. URL: https://sites.ulethbridge.ca/moodle-answers/2024/02/17/how-do-i-view-my-grades-in-moodle-2/ (visited on 2024-12-26) (cit. on p. 8).

[14] *Moodle Plugins directory: Advanced Notifications*. URL: https://moodle.org/plugins/block_advnotifications (visited on 2025-01-22) (cit. on p. 8).

[15] *Moodle Plugins directory: Academi*. URL: https://moodle.org/plugins/theme_academi (visited on 2025-01-22) (cit. on p. 9).

[16] *Moodle Plugins directory: CodeRunner*. URL: https://moodle.org/plugins/qtype_coderunner (visited on 2025-01-19) (cit. on p. 9).

[17] *Moodle Plugins directory: Course Size*. URL: https://moodle.org/plugins/report_coursesize (visited on 2025-01-22) (cit. on p. 9).

[18] *Moodle Plugins directory: GitHub*. URL: https://moodle.org/plugins/repository_github (visited on 2025-01-22) (cit. on p. 9).

[19] E. Bagley, J. Shumway, and J. Edwards. "Second-grade Students' Use of Visual Programming to Learn Multiplication: Leveraging the Concept of Iteration". In: *Proceedings of the 24th Australasian Computing Education Conference*. ACE '22. Virtual Event, Australia: Association for Computing Machinery, 2022, pp. 76–84. ISBN: 9781450396431. DOI: 10.1145/3511861.3511870. URL: https://doi.org/10.1145/3511861.3511870 (cit. on pp. 9, 10).

[20] *Working with Scratch Blocks*. 2022. URL: https://forum.creaticode.com/topic/73/working-with-scratch-blocks (visited on 2024-11-24) (cit. on p. 10).

[21] *Coding for kids | Tynker*. URL: https://www.tynker.com/ (visited on 2025-01-25) (cit. on p. 12).

[22] *Blockly Games*. URL: https://blockly.games/ (visited on 2025-01-25) (cit. on p. 12).

[23] *Using Blockly's API*. 2024. URL: https://developers.google.com/blockly/guides/configure/advanced/using_blockly_apis (visited on 2024-12-19) (cit. on pp. 12, 13, 27, 28, 35, 42, 53, 65).

[24] *Scratch - Wikipedia*. 2024. URL: https://pt.wikipedia.org/wiki/Scratch (visited on 2024-11-24) (cit. on p. 13).

[25] L. Li and S. Yang. "Exploring the Influence of Teacher-Student Interaction on University Students' Self-Efficacy in the Flipped Classroom". In: *Journal of Education and Learning* (2021). URL: https://api.semanticscholar.org/CorpusID:233367305 (cit. on p. 13).

[26] H. Sun et al. "The Influence of Teacher–Student Interaction on the Effects of Online Learning: Based on a Serial Mediating Model". In: *Frontiers in Psychology* 13 (2022). URL: https://api.semanticscholar.org/CorpusID:247454186 (cit. on p. 13).

[27] *Blockly block factory*. URL: https://blockly-demo.appspot.com/static/demos/blockfactory/index.html (visited on 2025-01-16) (cit. on p. 18).

[28] S. Pila et al. "Learning to code via tablet applications: An evaluation of Daisy the Dinosaur and Kodable as learning tools for young children". In: *Comput. Educ.* 128 (2019), pp. 52–62. URL: https://api.semanticscholar.org/CorpusID:53753172 (cit. on p. 24).

[29] *Kodable – Programming Curriculum for Elementary*. 2024. URL: https://kodable.soft112.com/ (visited on 2024-11-21) (cit. on p. 25).

[30] *Kids coding in the cloud*. 2013. URL: https://news.mit.edu/2013/scratch-two-released-0514 (visited on 2024-11-21) (cit. on p. 25).

[31] *The student experience on Code.org*. 2024. URL: https://support.code.org/hc/en-us/articles/115001347791-The-student-experience-on-Code-org (visited on 2024-11-21) (cit. on p. 26).

[32] P. Voštinár. "MakeCode Arcade: Interesting environment for programming 2D games". In: *2021 IEEE World Conference on Engineering Education (EDUNINE)*. 2021, pp. 1–6. DOI: 10.1109/EDUNINE51952.2021.9429132 (cit. on pp. 25, 28).

[33] *Microsoft MakeCode Arcade Hour of Code 2021 Educator Guide*. 2021. URL: https://arcade.makecode.com/hour-of-code/educators-2021 (visited on 2024-11-21) (cit. on p. 27).

[34] A. C. Bart et al. "Position paper: From interest to usefulness with BlockPy, a block-based, educational environment". In: *2015 IEEE Blocks and Beyond Workshop (Blocks and Beyond)* (2015), pp. 87–89. URL: https://api.semanticscholar.org/CorpusID:40151963 (cit. on pp. 26, 28).

[35] *A complete representation of BlockPy*. 2015. URL: https://www.researchgate.net/figure/A-complete-representation-of-BlockPy-http-thinkcsvtedu-blockpy_fig1_308602816 (visited on 2024-11-21) (cit. on p. 27).

[36] *Python input() Function*. URL: https://www.w3schools.com/python/ref_func_input.asp (visited on 2025-07-26) (cit. on p. 31).

[37] J. P. Leal and F. Silva. "Mooshak: a Web-based multi-site programming contest system". In: *Software: Practice and Experience* 33.6 (2003), pp. 567–581. DOI: https://doi.org/10.1002/spe.522. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/spe.522. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.522 (cit. on p. 33).

[38] M. Musch et al. "ScriptProtect: Mitigating Unsafe Third-Party JavaScript Practices". In: *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*. Asia CCS '19. Auckland, New Zealand: Association for Computing Machinery, 2019, pp. 391–402. ISBN: 9781450367523. DOI: 10.1145/3321705.3329841. URL: https://doi.org/10.1145/3321705.3329841 (cit. on p. 35).

[39] *Octoverse: AI leads Python to top language as the number of global developers surges*. URL: https://github.blog/news-insights/octoverse/octoverse-2024/ (visited on 2025-02-02) (cit. on p. 35).

[40] M. Abbes et al. "An Empirical Study of the Impact of Two Antipatterns, Blob and Spaghetti Code, on Program Comprehension". In: *2011 15th European Conference on Software Maintenance and Reengineering*. 2011, pp. 181–190. DOI: 10.1109/CSMR.2011.24 (cit. on p. 37).

[41] *Moodle 3.11*. URL: https://moodledev.io/general/releases/3.11 (visited on 2025-08-08) (cit. on p. 42).

[42] *Plugin types | Moodle Developer Resources*. URL: https://moodledev.io/docs/4.1/apis/plugintypes (visited on 2025-08-16) (cit. on p. 42).

[43] *Plugin contribution checklist | Moodle Developer Resources*. URL: https://moodledev.io/general/community/plugincontribution/checklist (visited on 2025-08-16) (cit. on pp. 42, 55).

[44] *General Data Protection Regulation - Wikipedia*. URL: https://en.wikipedia.org/wiki/General_Data_Protection_Regulation (visited on 2025-08-02) (cit. on p. 43).

[45] *Privacy API | Moodle Developer Resources*. URL: https://moodledev.io/docs/4.5/apis/subsystems/privacy (visited on 2025-08-02) (cit. on p. 43).

[46] *Backup API | Moodle Developer Resources*. URL: https://moodledev.io/docs/4.5/apis/subsystems/backup (visited on 2025-08-04) (cit. on p. 44).

[47] *Restore API | Moodle Developer Resources*. URL: https://moodledev.io/docs/4.5/apis/subsystems/backup/restore (visited on 2025-08-04) (cit. on p. 44).

[48] *Automated Manipulation of Strings (AMOS) | Moodle Developer Resources*. URL: https://moodledev.io/general/projects/api/amos (visited on 2025-08-04) (cit. on pp. 45, 46).

[49] *async function - JavaScript | MDN*. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/async_function (visited on 2025-08-09) (cit. on pp. 46, 48).

[50] *Promise - JavaScript | MDN*. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise (visited on 2025-08-09) (cit. on p. 48).

[51] *parseFloat() - JavaScript | MDN*. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/parseFloat (visited on 2025-08-09) (cit. on p. 49).

[52] *Sulu - The API monetization platform for the modern AI driven web*. URL: https://doc.sulu.sh/introduction (visited on 2025-07-27) (cit. on p. 50).

[53] H. Z. Došilović and I. Mekterović. "Robust and Scalable Online Code Execution System". In: *2020 43rd International Convention on Information, Communication and Electronic Technology (MIPRO)*. 2020, pp. 1627–1632. DOI: 10.23919/MIPRO48935.2020.9245310 (cit. on p. 50).

[54] H. Z. Došilović and I. Mekterovic. "Robust and Scalable Online Code Execution System". In: 2020-09, pp. 1627–1632. DOI: 10.23919/MIPRO48935.2020.9245310 (cit. on p. 50).

[55] *Code generation | Blockly*. URL: https://developers.google.com/blockly/guides/create-custom-blocks/code-generation/overview (visited on 2025-08-10) (cit. on p. 51).

[56] *Options.maxInstances property | Blockly*. URL: https://developers.google.com/blockly/reference/js/blockly.options_class.maxinstances_property (visited on 2025-08-11) (cit. on p. 53).

[57] *Window: setInterval() method*. URL: https://developer.mozilla.org/en-US/docs/Web/API/Window/setInterval (visited on 2025-08-12) (cit. on p. 55).

[58] R. A. Grier et al. "The System Usability Scale: Beyond Standard Usability Testing". In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 57.1 (2013), pp. 187–191. DOI: 10.1177/1541931213571042. eprint: https://doi.org/10.1177/1541931213571042. URL: https://doi.org/10.1177/1541931213571042 (cit. on pp. 60, 68).

[59] A. Bangor, P. Kortum, and J. Miller. "Determining what individual SUS scores mean: Adding an adjective rating scale". In: *Journal of usability studies* 4.3 (2009), pp. 114–123 (cit. on pp. 60, 63).

[60] J. Sauro. *A practical guide to the system usability scale: Background, benchmarks & best practices*. Measuring Usability LLC, 2011 (cit. on pp. 60, 64).

[61] *EDULEARN25*. URL: https://iated.org/edulearn/ (visited on 2025-08-19) (cit. on pp. 66, 68, 70).

[62] R. Correia, F. Barbosa, and C. Morgado. "NEXTBLOCKS - INTEGRATING A CUSTOMIZABLE BLOCK PROGRAMMING ENVIRONMENT WITH INTERACTIVE INPUTS INTO MOODLE". In: *EDULEARN25 Proceedings*. 17th International Conference on Education and New Learning Technologies. Palma, Spain: IATED, 2025-30 June-2 July, 2025, pp. 5292–5301. ISBN: 978-84-09-74218-9. DOI: 10.21125/edulearn.2025.1325. URL: https://doi.org/10.21125/edulearn.2025.1325 (cit. on p. 66).

[63] *Moodle Plugins directory: NextBlocks*. URL: https://moodle.org/plugins/mod_nextblocks (visited on 2025-09-27) (cit. on p. 68).

A

Old syntax test file example

```
|
_
Number1: number
2
_
Number2: number
10
-
20
|
_
Number1: number
4
_
Number2: number
10
-
40
|
_
Number1: number
40000000
_
Number2: number
1000
-
40000000000
|
```

```
_
Number1: number
0
_
Number2: number
0
-
0
```

B

Global test file example

```
|
2
10
-
20
|
4
10
-
40
|
40000000
1000
-
40000000000
|
0
0
-
0
```

C

NextBlocks User Test Instructions - Teacher Edition

Introduction

Thank you for participating in the second stage of NextBlocks' user testing! This evaluation aims to evaluate the improvements made since the last stage, so functionalities that were not modified might not be questioned about. Nevertheless, you can always leave your opinion about them in the last page of the questionnaire. NextBlocks is a Moodle-based block programming platform that aims to be interactive, customizable and collaborative. The improvements made during the new version include an interactive terminal, functionalities to edit an exercise, block limits, a modified comments system and automatic translation to Python. Your feedback will help improve its functionality.

Purpose

This test is composed of 2 phases. In the first phase you will create a new exercise, while in the second phase you will evaluate a student's submission in a previously created exercise.

Phase 1: Create Even Counting Exercise

Create an exercise called Even Counting where students are expected to create a program that reads 10 integers and print how many of them are even. In the desktop there is a text file with the description of the exercise and a folder with the test files, which can be used to facilitate the creation of the exercise. Since we don't want students to create 10 input blocks to read 10 integers, limit the maximum number of "input" blocks to 1, forcing students to find another solution. Set the maximum number of submissions to 10 and save the exercise. After creating the exercise, you remember that the students might also use 10 'if' blocks instead of using a loop. Edit the exercise and set the maximum number of 'if' blocks to 1. Save the exercise.

Phase 2: Grading & Feedback

Go back to the course and see the grades' page. Find Miguel Pereira's submission to the exercise 'Print the maximum between 2 numbers', which has a 75/100 grade. Review his program and the automatic translation to text code (JavaScript and Python) and see that everything is correct and that he should have gotten the maximum grade. Assume that there is an error with the tests and manually change his grade to 100. Look again at his program and see that he left a comment in one of the blocks. Open the comments interface, read it and answer. Create a new comment in another block, giving feedback to the student about their submission. Finally, check the reactions bar and the chat to see what the students thought about the exercise.

D

NextBlocks User Test Instructions - Student Edition

Introduction

Thank you for participating in the second stage of NextBlocks' user testing! This evaluation aims to evaluate the improvements made since the last stage, so functionalities that were not modified might not be questioned about. Nevertheless, you can always leave your opinion about them in the last page of the questionnaire. NextBlocks is a Moodle-based block programming platform that aims to be interactive, customizable and collaborative. The improvements made during the new version include an interactive terminal, block limits, a modified comments system and automatic translation to Python. Your feedback will help improve its functionality.

Purpose

During this test you will be presented with a previously created exercise and asked to do different tasks related to that exercise in order to try out the different functionalities.

Task Description

1. Open the exercise called ”Print the maximum between 2 numbers”. Read its description to understand what the program you will create is expected to do.
2. Open some of the categories in the toolbox (Left side of the interface) and notice that some of the blocks are grayed and that they cannot be used. Open the Block Limits window to find out what are the defined limits for that exercise.
3. Now that you know how many of each block you can use, start creating your program. At any time you can run the program with manual inputs to test its functionality.
4. Once you have finished creating the program, run the tests to see if your program behaves as expected. You can modify it if necessary.
5. Open the text code translation to see how your program translates to JavaScript and Python.
6. Leave a comment in one of the blocks, a reaction to the exercise and a message in chat.
7. Finally submit your program for evaluation.