Received 14 April 2023, accepted 19 May 2023, date of publication 31 May 2023, date of current version 19 July 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3281761

RESEARCH ARTICLE

IrisMath: A Blind-Friendly Web-Based Computer Algebra System

ANA M. ZAMBRANO[1], DANILO I. PILACUÁN[1], MATEO N. SALVADOR[1], FELIPE GRIJALVA[2], (Senior Member, IEEE), NATHALY OROZCO GARZÓN[3], (Senior Member, IEEE), AND HENRY CARVAJAL MORA[3], (Senior Member, IEEE)

[1]Departamento de Telecomunicaciones y Redes de Información, Escuela Politecnica Nacional, Quito 170525, Ecuador
[2]Colegio de Ciencias e Ingenierías "El Politécnico," Universidad San Francisco de Quito (USFQ), Quito 170901, Ecuador
[3]Faculty of Engineering and Applied Sciences (FICA), Telecommunications Engineering, Universidad de Las Américas (UDLA), Quito 170503, Ecuador

Corresponding author: Henry Carvajal Mora (henry.carvajal@udla.edu.ec)

This work was supported by Escuela Politécnica Nacional through the Project Sistema de Cálculo Numérico-Algebraico Virtual Interactivo Para Estudiantes de Ingeniería con Discapacidad Visual under Grant PTT-21-02.

This work involved human subjects or animals in its research. Approval of all ethical and experimental procedures and protocols was granted by Technical Cooperation Agreement between Research Project PTT-21-02 from Escuela Politecnica Nacional and the Asociación de Invidentes Milton Vedado from Quito, Ecuador.

ABSTRACT Visually impaired individuals face challenges in pursuing higher education, particularly in technical courses related to engineering. The lack of specialized tools and resources that allow for proper development in academic activities is a significant barrier to their admission, retention, and graduation from higher education institutions. By considering this, this paper presents the development of a blind-friendly Computer Algebra System (CAS) called IrisMath. This system enables visually impaired people to perform mathematical operations commonly used in engineering. IrisMath is a web application inspired by Jupyter Notebooks and developed using a Layered Architecture to provide modularity. It offers a variety of output formats, including LaTex, CMathML, JSON, and audio formats. Our CAS has undergone an extensive assessment of its functional, non-functional, and usability requirements, demonstrating its potential as a tool for engineering students with visual disabilities.

INDEX TERMS Blind people, Python, synthesized voice, sonification, computer algebra system.

I. INTRODUCTION

According to the World Health Organization (WHO), there are around 252.6 million people with some type of visual impairment around the globe, of which 36 million are classified as totally blind [1]. Among these people, there is a high degree of school dropout, so the percentage is estimated at around 40% in countries like the United States and is exceeded in Latin America [2].

In the particular case of Ecuador, where this research work is carried out, there is a total of 11.6% of the population with some degree of visual disability. In particular, 40% of these people have a disability greater than 75%, of which only 2,906 people are studying basic education and approximately 1,100 people are studying at university [3]. The latter group faces greater challenges due to the lack of specialized tools and resources that allow their correct development in academic activities, especially in technical courses related to engineering. This undoubtedly limits their likelihood of admission, retention, and graduation from higher education institutions.

Among the most widely used software tools in exact sciences and engineering are Computer Algebra Systems (CAS), such as Matlab, Wolfram Mathematica, and Maxima CAS [4]. Unfortunately, these tools lack features that make them accessible and usable by people with visual disabilities. Although there are software tools that can read what is displayed on a computer screen, they are not suitable for

The associate editor coordinating the review of this manuscript and approving it for publication was Fabrizio Messina.

environments that involve mathematical expressions. The main issue is that these programs are designed to read a text and not equations or mathematical expressions. As a result, users may listen to text in a linear way, which can generate ambiguities when trying to understand the details of a mathematical expression. Therefore, there is a need to develop tools that enable visually impaired people to comprehend mathematical expressions in detail, taking into account all of the subtleties that may be involved in these expressions.

This work proposes a new platform called IrisMath that acts as a web prototype, connecting to the Maxima CAS tool via an accessible and intuitive interface to perform numerical-algebraic calculations. It also features speech synthesis adapted to the nonlinear semantics characteristic of the resulting mathematical expressions. The aim of IrisMath is to provide academic support to blind engineering students and overcome potential limitations they may face during their training.

The rest of this work is organized as follows. Section II presents the state of the art and highlights the current problems associated with the lack of research on inclusive software. In addition, the main contributions of our proposal are presented. Section III defines the architecture and the development process for IrisMath. In Section IV, the results obtained are analyzed, and discussions around these results are presented. Finally, Section V concludes the work and presents options for future research.

II. STATE OF THE ART AND CONTRIBUTIONS

This section describes previous works related to the proposal presented. Specifically, these works propose accessibility tools for people with visual disabilities. In addition, the key contributions of our proposal are summarized.

Pearson Accessible Equation Editor (AEE) presents a system for creating mathematical expressions through a web application that is based on external screen readers such as NVDA or JAWS [5]. This system allows the input and output of information related to mathematical notations through a Braille system. For the inputs, AEE uses an external updateable display. Furthermore, the outputs are converted to MathML language, which is an XML-based markup language intended to represent mathematical notations in Braille format [6].

L-MATH [7] is a system that enables the editing and inspection of mathematical formulas. Writing and reading of mathematical expressions are achieved through the BlindMath and TalkingMath modules, respectively. With BlindMath, the visually impaired student can enter mathematical formulas using a computer keyboard that can then be converted to LaTeX code. On the other hand, TalkingMath uses an original adaptive algorithm to read formulas according to human reading habits.

Another popular platform is the LAMBDA system [8]. This is a system of access to computational mathematics, which is based on a linear mathematical notation that allows access to mathematical expressions through the Braille code and synthetic voice. This system has a Braille version with 256 unique characters (LAMBDA code) based on the Braille representation of 8 points, which includes new symbols that allow for the representation of mathematics in a linear way. These symbols can be represented visually and in Braille. Among the distinctive features of LAMBDA is its ability to solve basic mathematical operations (for example, addition, factorial, and trigonometric operations).

DOSVOX [9] is an autonomous system developed by the Federal University of Rio de Janeiro and is currently widely used in Brazil. Within the DOSVOX system, there are two tools that allow the execution of mathematical operations: MATVOX and FINANVOX. The financial calculator FINANVOX [10], allows to perform financial and statistical calculations. This software tool allows operations such as compound interest, amortization, depreciation, mean, and standard deviation, among others. This is achieved by emulating and expanding on the functions of the popular Hewlett-Packard HP-12C financial calculator. On the other hand, DOSVOX encompasses more than 80 open-source tools that can be accessed through spoken menus, allowing visually impaired users to perform various activities such as sending emails, playing music, and creating documents and spreadsheets.

MATVOX [11] is an interpreter of computer algorithms that helps to write and compile pseudocode from a text editor called EDIVOX, which also allows working with mathematical notation such as complex numbers and matrices.

The CASVI system [12] is the best predecessor of our proposal. It provides an alternative tool to help people with visual disabilities who are studying engineering and exact sciences. The system serves as a bridge between the students and existing computer algebra system (CAS) tools, allowing them to write, edit, evaluate, and solve mathematical expressions. It is important to note that CASVI is not described in technical scientific papers; its analysis has been limited to studying the interaction with the users.

As a summary, Table 1 details the main contributions of previous works and the ones of our proposal. Based on the literature review, it is evident that the development and implementation of accessible tools present a great challenge since it has become a very important need for the inclusion of students with visual disabilities. This is the main motivation for our work and that is why IrisMath arises, as a pedagogical aid, being a web platform that improves the management of tasks (assignments and grades) between a sighted teacher and a blind student.

This project, carried out by several work teams from different universities, undoubtedly improves the current state of inclusion in engineering. More specifically, our main contributions are summarized as follows:

- In contrast to systems like CASVI [12], a platform structured under a Layered Architecture to provide modularity is developed. Overall, our structured layered architecture with modularity provides a solid foundation for software development by enhancing maintainability, scalability, flexibility, testability, integration, and collaboration. It helps build robust and adaptable software that can evolve over time while minimizing the impact of changes and promoting efficient development practices.

- A significant improvement in voice synthesis by jumping from linear to non-linear translation is performed. In comparison with CASVI [12] that only offers a linear synthesis of complex mathematical expressions, incorporating both linear and non-linear equation reading capabilities enables access to complex equations, supports advanced mathematics education, enhances comprehension, and prepares users for higher-level mathematics. It empowers visually impaired individuals to navigate and engage with mathematical content across a wide range of disciplines and educational levels.
- A greater variety of output formats are provided, such as LaTeX, CMathML, JSON, and audio formats. By providing a greater variety of output formats, software aimed at visually impaired individuals can ensure inclusivity, accommodate individual preferences and accessibility needs, and enhance compatibility with complementary assistive technologies. It promotes a more personalized and empowering user experience, empowering visually impaired users to access and interact with information in ways that suit them best.
- A web platform is developed in order not to present limitations to the user, being able to use any hardware and software as long as a web browser is available. By leveraging the advantages of web platforms, visually impaired users can benefit from increased accessibility, flexibility, and choice. The hardware and software agnosticism, combined with the ease of access and seamless updates, contribute to a more inclusive and empowering user experience for individuals with visual impairments. This contrasts with previous systems such as LAMBDA, CASVI, or MATVOX.

Details about the development process are presented in the following section.

III. ARCHITECTURE AND DEVELOPMENT PROCESS

In this section, the components and language of the system are presented as well as the structure and operation of the prototype. Besides, the voice synthesis operation is also described.

As shown in Fig. 1, the system contemplates two types of actors: Students (users with visual disabilities) and Teachers (users with full capacity to use a graphical interface) with their own activities. More details related to the development of the platform are described below.

A. DEFINITION OF COMPONENTS AND LANGUAGES

This subsection details the architecture, which has been developed taking into account the following considerations:

- Users with visual disabilities do not have the possibility of using a classic graphical interface since they cannot position themselves on the screen.

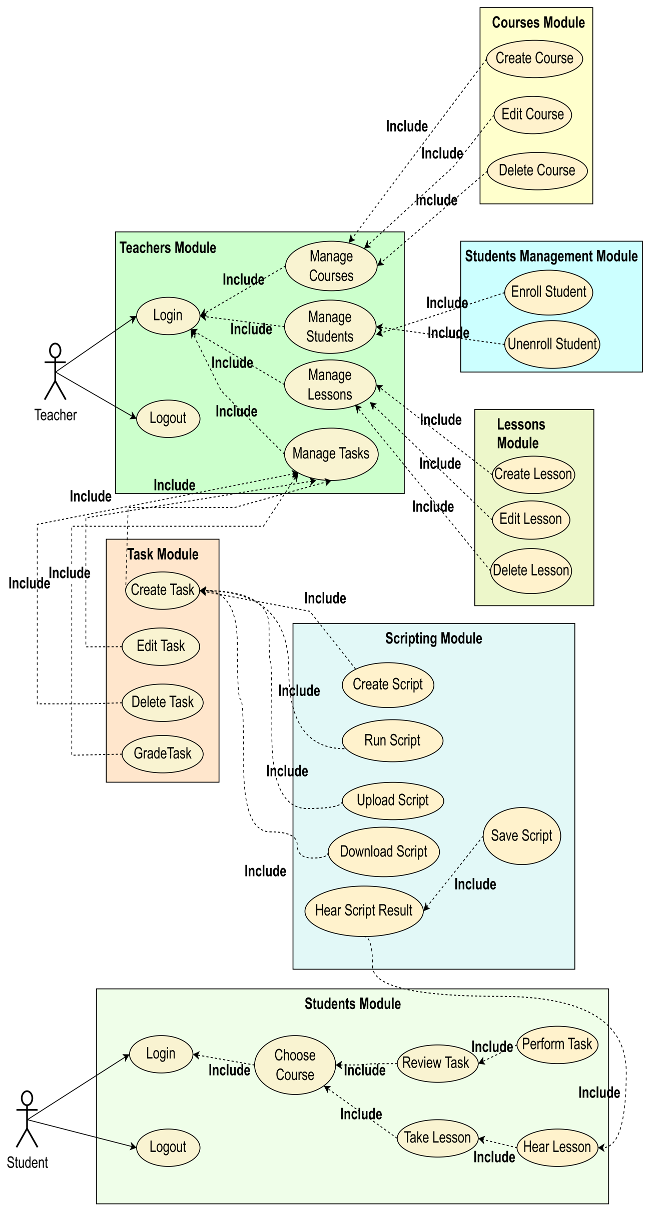


FIGURE 1. Actors and their activities on IrisMath (Use Case Diagram).

- Users with visual disabilities must use the interface solely through a keyboard as the input device to the system.
- The users must be provided with auditory feedback, which contains as much information as possible for the correct understanding of the actions they are performing while taking care of the user's cognitive load [13].
- To ensure unrestricted scalability, the system should be limited to working with CPU, display, keyboard, and mouse (for sighted users).

For the development of this project, a layered architecture has been used, which allows for a modular and easily scalable system. The layers have been developed using Python and JavaScript programming languages, with the MAXIMA

TABLE 1. Main features of current Math CAS for visually impaired individuals.

System	Text-to-speech synthesis	Non-auditory feedback	Compatibility with external screen readers	Input formats	Output formats	Architecture type	Compatible platforms	Usability tests
Person accessible Math editor	Simple TTS	-	✓	Text, Braille	Braille	Monolithic	Multiplatform (Web)	-
L-MATH	Simple TTS	✓	✓	LaTeX	Audio formats, LaTeX	Monolithic	Linux, macOS	4 users
LAMBDA	Simple TTS	-	-	LAMBDA, MathML, XHTML, Braile	Monolithic	Microsoft Windows	151 users	
MATVOX	Simple TTS	-	-	Plain text	Plain text, audio formats	Monolithic	Microsoft Windows	6 users
FINANVOX	Simple TTS	-	-	Plain text	Plain text, audio formats	Monolithic	Microsoft Windows	-
Casvi	Simple TTS, lexical and prosodic cues	-	-	Plain text	Plain text	Monolithic	Microsoft Windows	-
IrisMath	Non-linear TTS, auditory aid	-	-	Plain text	LaTeX, CMathML, JSON, Audio formats	Layered	Multiplatform (Web)	10 users

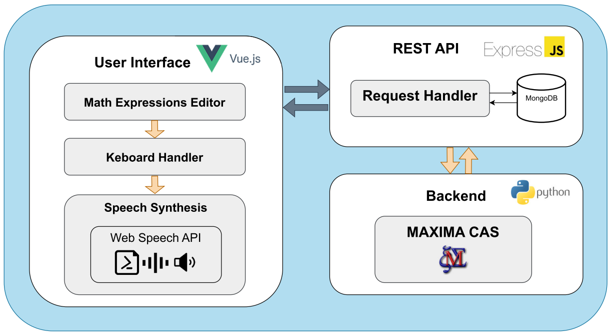


FIGURE 2. Block diagram of our architecture.

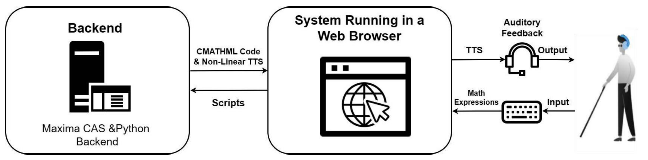

Backend
Maxima CAS &Python Backend
CMATHML Code & Non-Linear TTS
Scripts
System Running in a Web Browser
Auditory Feedback
TTS
Output
Math Expressions
Input

FIGURE 3. Component diagram of IrisMath.

CAS software as the backend for mathematical processing. Maxima CAS was chosen by its efficient symbolic calculations and has been optimized over the years for performance. It is designed to handle complex mathematical computations efficiently, particularly in the domain of symbolic mathematics. In addition, Maxima CAS uses its own high-level programming language, which is designed for symbolic computations. The language is expressive and concise, making it easier to work with complex mathematical expressions, and provides advanced control structures and programming constructs for more advanced symbolic manipulation tasks.

The interface has been developed with the help of the VueJS framework and implemented as a web system, which provides multi-platform capability, allowing the system to be independent of the operating system where it is used. This interface has been developed simulating a web development environment similar to Jupyter Notebooks [14], which has allowed the development of features inspired by it, such as (1) text inputs known as code cells (which allow the writing of mathematical expressions, which will be referred to as scripts from now on), (2) independent execution of cells to offer modularity to the resolution of scripts, and (3) a custom file format based on JSON [15], which stores, with the help of the MongoDB [16], both the inputs of the cells, as well as the outputs processed by the backend that makes use of Maxima CAS.

This undoubtedly delivers the advantage of being exportable and importable for exchange by other users of the system.

The system has also been provided with a Text-to-Speech (TTS) functionality for the mathematical processing results. It is worth noting that this will be much more specialized than the TTS synthesis performed by previous works since they use a simple synthesis that dictates the results linearly. Therefore, IrisMath performs a non-linear synthesis of the results, bringing it closer to the non-linear nature of math expressions that preserves the context and hierarchy of the operations.

Fig. 2 depicts the block diagram of the proposed architecture. Within this architecture, the data exchange formats between the Python backend developed (which uses Maxima CAS), the system interface, and the TTS voice synthesis module are LaTeX and MathML [17] (in its CMathML specification) [18], with LaTeX being the output format provided by Maxima CAS for resulting expressions. However, LaTeX lacks a structure that provides mathematical context to the resulting expressions (therefore ignoring hierarchy and preventing non-linear voice synthesis), which is why the conversion of LaTeX outputs through the SnuggleTex [19] tool to CMathML is necessary. This format allows structuring a mathematical expression in a tree-like form that provides mathematical context to the resulting expressions and preserves their hierarchy.

As mentioned earlier, the mathematical processing required for the resolution of the mathematical expressions entered in the system interface is carried out by the Maxima CAS system, which is an open-source and multi-platform computer algebra system. Fig. 3 shows a component diagram that describes the interaction between the user, the web system, and the MAXIMA CAS system.

The need to implement the CMathML specification to convert the LaTeX format results returned by Maxima CAS for the resolution of mathematical expressions arises from its characteristics. Concretely, MathML has two specifications:

```
<mrow>
    <mi>x</mi>
    <mo>+</mio>
    <mfrac>
        <mi>a</mi>
        <mi>b</mi>
    </mfrac>
</mrow>
```

FIGURE 4. PMathML representation of $x + \frac{a}{b}$.

```
<apply>
    <plus/>
    <ci>x</ci>
    <apply>
        <divide/>
        <ci>a</ci>
        <ci>b</ci>
    </apply>
<apply>
```

FIGURE 5. CMathML representation of $x + \frac{a}{b}$.

PMathML (Presentation MathML) and CMathML (Context MathML). The main objective of PMathML is to describe the structure of a mathematical expression. However, this representation lacks an appropriate semantics for the unequivocal distinction of mathematical operators since these will be encompassed in common tags such as `<mi>`, `<mo>`, and `<mrow>` that cover most operators and variables, which results in expressions that can become ambiguous, making their correct interpretation impossible. In contrast, the main objective of the CMathML specification is to describe the context and semantics of a mathematical expression, adding a much wider range of tags that allow for the identification of an operator or variable independently. Since CMathML contains a specific tag for each mathematical operation, it is more specific than PMathML.

As an example, consider the mathematical expression $x + \frac{a}{b}$, which has been considered for detailing its representation both in the PMathML and CMathML specifications, as shown in Figs. 4 and 5, respectively. As evidenced in Fig. 4, the only tags that make up the structure of PMathML are `<mrow>`, `<mi>`, `<mo>`, and `<mfrac>`. However, the $+$ operator is among a series of tags that are common to all mathematical operations, which would prevent the distinction of its mathematical context. On the other hand, in Fig. 5 it is possible to see that both the division and the addition are correctly identified by a specific tag, namely `<divide>` and `<plus>` respectively, which allows knowing exactly which mathematical operation is being performed when traversing the CMathML tree.

As specified earlier, CMathML defines a specific tag for each mathematical operation. Therefore, Table 2 below shows a list of mathematical operations supported in the latest CMathML specification, their respective tags, and their corresponding LATEX operators. These operators are the results of solving mathematical expressions and are returned by Maxima CAS, which are used to be transformed into CMathML using the SnuggleTex tool [20]. In this way, through the use of CMathML, which provides the mathematical context and semantics of mathematical expressions, a much more efficient TTS voice synthesis is achieved compared to previous works.

Consider now the example presented in Fig. 6, where the LaTeX output for the expression arcsin(0) $= 0$ is shown. For this example, Fig. 7 shows the CMathML tree obtained after the execution of the SnuggleTex tool.

As a final stage and in order to provide a way to store and retrieve the results of the processing of mathematical expressions, the MongoDB database manager is used, which is a document-oriented database system. Thus, it allows embedding the results of both LaTeX, CMathML, and TTS in a JSON format that contains all the relevant information about the execution of the system.

TABLE 2. Mathematical operations supported by CMathML.

LaTex Operator	CMathML Tag
+	<plus/>
-	<minus/>
x	<times/>
÷	<divide/>
\vee	<or/>
\wedge	<and/>
\cup	<union/>
\cap	<intersection/>
\setminus	<setdiff/>
\not	<not/>
!	<factorial/>
=	<eq/>
\not=	<neq/>
<	<lt/>
>	<gt/>
\leq	<leq/>
\geq	<geq/>
\equiv	<equivalent/>
\sin	<sin/>
\cos	<cos/>
\tan	<tan/>
\log	<log/>
\exp	<exp/>
\max	<max/>
\min	<min/>

```
\sin^{-1}0 = 0
```

FIGURE 6. LaTeX output from Maxima CAS.

B. STRUCTURE AND OPERATION OF THE PROTOTYPE

Although the target users are visually impaired students, this web interface has been developed with controls such as buttons and text boxes, with special emphasis on the use of MathJAX [21] as a library for displaying the results of mathematical expressions. This is made to facilitate the interaction of Professor-type users. On the other hand, visually impaired users can navigate through the various menus using shortcut keys, which are detailed in Table 3.

```
<math>
    <apply>
        <eq/>
        <apply>
            <arcsin/>
            <cn>0</cn>
        </apply>
        <cn>0</cn>
    </apply>
</math>
```

FIGURE 7. Resulting CMathML tree after conversion by SnuggleTex.

TABLE 3. Available shortcuts for visually impaired users.

Shortcut	Function
Alt+M	Help Menu
Alt+W	Write to current cell
Alt+R	Execute current cell
Alt+L	Listen to current cell
Alt+C	Create new cell
Alt+Q	Next Cell
Alt+Y	Previus Cell

Fig. 8 shows the activity diagram that illustrates the actions to solve a mathematical expression, display the visual presentation, and synthesize the text-to-speech output.

Initially, the system is waiting for input from the user as shown in Fig. 9, which details the initial interface of the system. It consists of a menu bar for storing the current file and managing the processing backend, as well as a toolbar for manipulating and executing mathematical expressions. It also includes a series of sections called cells, which consist of a text editor where mathematical expressions are entered.

Following the process in the activity diagram of Fig. 8, the user enters a mathematical expression to be solved (using a keyboard and with audio assistance at each execution). Fig. 10 presents the process for entering mathematical expressions. The focus of the system should be on the text editor of the cell to be executed, either by clicking on it with the mouse pointer (for sighted users) or through the keyboard shortcut `Alt+W`, which performs the same function and is designed to provide accessibility [22] for visually impaired users. Mathematical expressions to be solved should be entered line by line, with automatic numbering to provide an identifier that allows for feedback to the user.

Once the mathematical expressions have been entered, the user can request their resolution by clicking on the "Execute" button located in the toolbar, or by using the keyboard shortcut `Alt+R`, which allows the interface to request their resolution from the processing backend. After this process is completed, the inputs and outputs are displayed in the results table (one by one), in order to provide better feedback to the users. Fig. 11 shows the results table corresponding to the resolution of the previously entered mathematical expressions.

The table of results presents a series of rows and columns focused on presenting input and output information to the user. In the case of columns, the first column shows the identifier of the result, while the second column shows the result itself. Likewise, the rows show one by one the inputs accompanied by their corresponding outputs. To identify whether a row corresponds to an input or an output, the identifiers `%in` have been used for the inputs, and `%on` for the outputs, with `i` standing for input, `i` for output, and `n` being the identifier that marks the correspondence between them. For example, it can be clearly seen in Fig. 11 that cell `%i1` represents the text: `solve(3*x +4=12)` (this has been entered by the user); while `%o1` represents the output in CMathML format as a response to the equation entered in cell `%i1`.

The aforementioned has been developed through a process that communicates the user interface with the Maxima CAS backend, using a REST API [23] (see Component Diagram in Fig. 3 with the help of the ExpressJS framework [24], which creates a web service that receives processing requests as HTTP methods and is responsible for calling the subsystem for mathematical expression processing, coded in Python. This REST API also stores inputs and results in the MongoDB database manager, which allows the web system to not depend on the internal storage of the computer on which it is running. Fig. 12 presents the Flow Diagram that is followed for processing a script (composed of mathematical operations in a cell `%in`, created in the user interface).

Fig. 12 presents a Flow diagram that represents the proposed algorithm for processing a script containing mathematical expressions. As a first step, a web service is started using the ExpressJS framework, which is responsible for receiving processing requests through the HTTP protocol on port 8000. Subsequently, if a processing request is received, the system attends to it and an instance of the mathematical expression processing backend (coded in Python) is initialized. The backend prepares the input to the system by removing control characters and making it compatible with Maxima's CAS format (i.e., normalization process). Then, the inputs are converted to LaTex format with the help of Maxima CAS, and then to CMathML format, which allows them to be converted into a non-linear language with mathematical context for subsequent synthesis. Next, the inputs are saved in an object called `result`, which has a property called `inputSpeech`. Once the input conversion is completed, the expressions is resolved, and Maxima CAS performs the resolution and formatting of results in LaTex format and then, in CMathML format for conversion to non-linear language with a mathematical context.

The output results are also stored in the `result` object but in the `outputSpeech` property. Once these tasks are completed, a conversion is made from the object in memory (cache) to an object in JSON format, which is returned to the user interface as the processing request result.

The results of the processing performed in the system's backend, in both LaTex and CMathML formats and non-linear language with mathematical context, are also stored in a JSON object using MongoDB, which is used to display the results on the screen and also to perform the voice synthesis

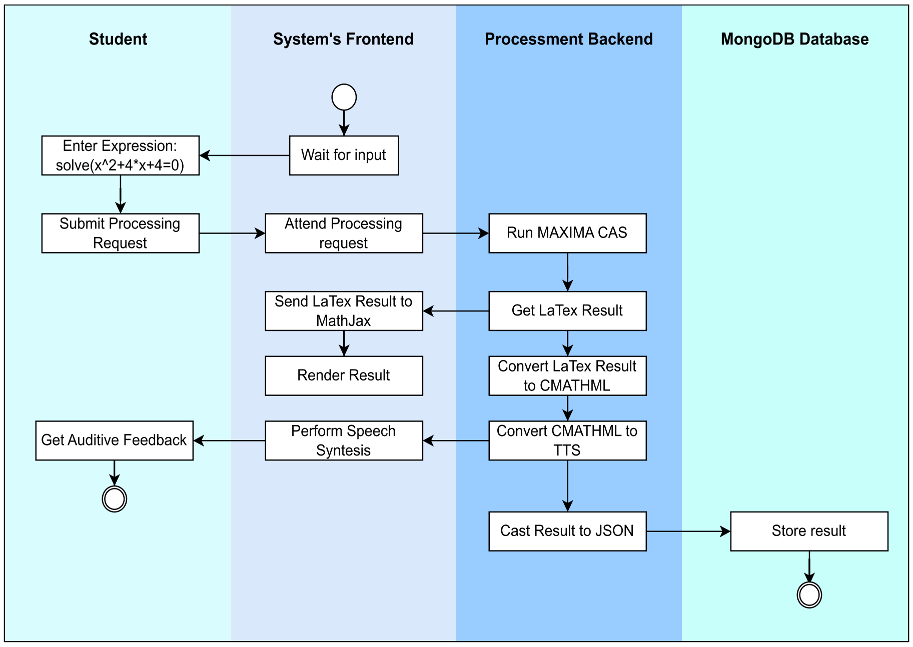


FIGURE 8. Activity diagram. Example to solve a polynomial equation.

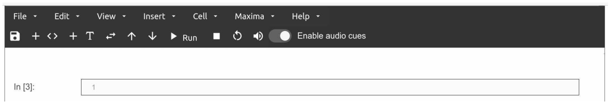
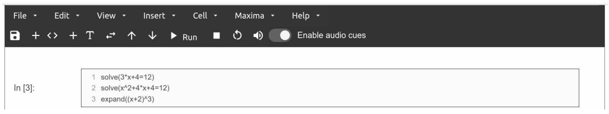
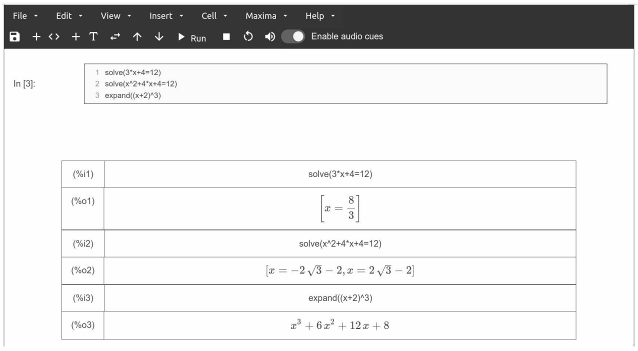


FIGURE 9. Initial interface of the system.



```
solve(3*x+4=12)
solve(x^2+4*x+4=12)
expand((x+2)^3)
```

FIGURE 10. Entry of mathematical expressions.



```
solve(3*x+4=12)
solve(x^2+4*x+4=12)
expand((x+2)^3)
```

(%i1)	solve(3*x+4=12)
(%o1)	$\left[x = \frac{8}{3}\right]$
(%i2)	solve(x^2+4*x+4=12)
(%o2)	$[x = -2\sqrt{3} - 2, x = 2\sqrt{3} - 2]$
(%i3)	expand((x+2)^3)
(%o3)	$x^3 + 6x^2 + 12x + 8$

FIGURE 11. Table of results of the mathematical expressions.

process. Fig.13 shows the structure of the document in which the information for processing mathematical expressions, both inputs, and outputs, is stored.

The format shown in Fig. 13 presents a JSON document with a series of inputs and outputs representing the attributes contained within a cell, which represent the different objects that store the information used by the interface. Each of these is detailed below:

- `input`: Information entered by the user to be processed.
- `output`: Result of processing the mathematical expression in the system's backend, which in turn is subdivided into:
  - `inputScript`: Expressions received in the backend.
  - `outputScript`: Result obtained by Maxima CAS, which is in LaTeX format and is not intelligible by a visually impaired user.
  - `inputSpeech`: Mathematical expressions entered by the user, converted to non-linear language that preserves the context and hierarchy of the operations, used to denote mathematical expressions that are synthesized as speech to provide auditory feedback to the user.
  - `outputSpeech`: Result obtained by processing the input in the backend with Maxima CAS and converted into a non-linear language that preserves the context and hierarchy of the operations, used to denote mathematical expressions to be synthesized to present the result to the user through the system's speaker.
- `run`: Current processing status of the cell.
- `activeCell`: Determines if the cell is currently active to receive keyboard inputs from the user.
- `isTextCell`: Cells are divided into two types. (1) Text cells: which are cells with additional information and will not be processed in the backend, and (2) Script cells: which will be processed by Maxima. isTextCell helps

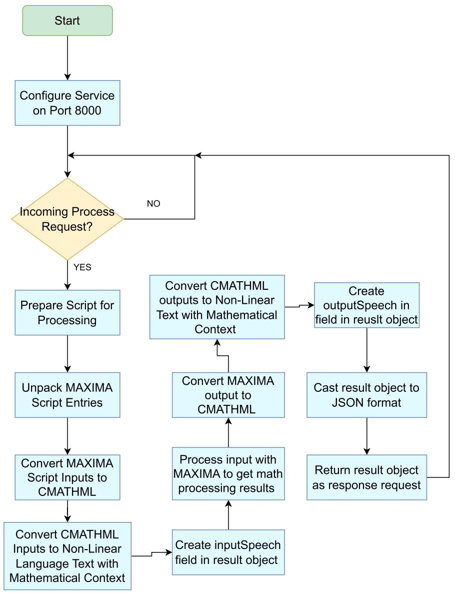
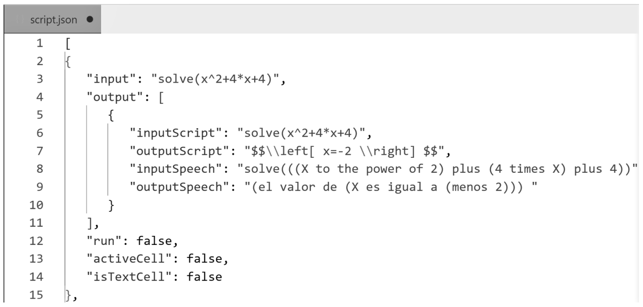

Start
Configure Service on Port 8000
Incoming Process Request?
NO
YES
Prepare Script for Processing
Unpack MAXIMA Script Entries
Convert MAXIMA Script Inputs to CMATHML
Convert CMATHML Inputs to Non-Linear Language Text with Mathematical Context
Create inputSpeech field in result object
Process input with MAXIMA to get math processing results
Convert MAXIMA output to CMATHML
Convert CMATHML outputs to Non-Linear Text with Mathematical Context
Create outputSpeech in field in reuslt object
Cast result object to JSON format
Return result object as response request

FIGURE 12. Flow diagram for mathematical expression processing.



```
[
{
   "input": "solve(x^2+4*x+4)",
   "output": [
      {
         "inputScript": "solve(x^2+4*x+4)",
         "outputScript": "$$\\left[ x=-2 \\right] $$",
         "inputSpeech": "solve(((X to the power of 2) plus (4 times X) plus 4))"
         "outputSpeech": "(el valor de (X es igual a (menos 2))) "
      }
   ],
   "run": false,
   "activeCell": false,
   "isTextCell": false
},
```

FIGURE 13. JSON file format that stores processing results in MongoDB.

in identifying them and avoiding sending a text cell for processing.

However, despite all this process, JSON format by itself is not useful for users with visual disabilities. This is why the functionality of speech synthesis of the results has been added, which allows the cells to be dictated one by one, using the identifiers presented above, which provides better feedback to the blind user. This speech synthesis is executed immediately after rendering the results table and can be invoked as many times as necessary through the shortcut `Alt+L`.

C. VOICE SYNTHESIS FOR THE RESULTING MATHEMATICAL EQUATION

After processing the cells in the backend of the system, the platform's interface is capable of taking the obtained results and performing text-to-speech synthesis in order to provide the necessary feedback to users [25].

Text-to-speech synthesis of the results obtained from the backend is carried out using Web Speech API, which is an open API aimed at web browsers that provides accessibility options for users with visual impairments and enables the development of the present system as a web application. Web Speech API implements functions for text-to-speech synthesis, which are used to create a component in VueJS that handles the voice synthesis of the results obtained from the backend. Table 4 presents the methods that make up the speech synthesis which are available in Web Speech API. These have been implemented in the corresponding component, that can be controlled through the shortcuts presented on Table 5.

FIGURE 14. Volunteer users using IrisMath.

IV. EXPERIMENTS AND RESULTS

This section describes the tests developed to evaluate the interaction of users with the IrisMath platform in terms of its usefulness, and its complexity of use. First, we present the experimental setup conditions of our experiments in Section IV. Next, in Section IV-B, we outline the evaluation metrics used to assess our system. Finally, in Section IV-C, we discuss the results.

A. EXPERIMENTAL SETUP

For the execution of the tests, a computer laboratory was set up, in which 5 computers were adapted with the appropriate level of contrast and brightness for use by blind volunteers. This is because, on the visual disability scale, not all blind people are unable to perceive light. In addition, the test group was divided into two scenarios: a simulated scenario consisting of six people without visual impairment, and then, a real scenario where two visually impaired people used the platform, as observed in Fig. 14.

The simulated scenario lasted four weeks, out of which two constituted processes such as environment preparation and user training in the use of the software. It is worth mentioning that in the real testing scenario, this period was doubled due to logistical implications and coordination with the parties involved. Additionally, it is of utmost importance that the visually impaired users undergo prior training in the use of the system.

Regarding the methodologies applied in the execution of the tests, the study was based on current standards and guidelines, which include the Software Engineering Body of Knowledge V3 (SWEBOK) [26], the International

TABLE 4. Methods provided by Web Speech API.

Method	Parameter	Return	Description
SpeechSynthesis. getVoices()	-	Array of SpeechSynthesisVoice objects	Gets the list of voices available in the web browser for synthesis.
SpeechSynthesis. speak()	Utterance	None.	Start speech synthesis from a processing queue.
SpeechSynthesis. pause()	-	None.	Pause speech synthesis without stopping it.
SpeechSynthesis. resume()	-	None.	Restarts speech synthesis where it left off after a pause.
SpeechSynthesis. cancel()	-	None.	Cancels speech synthesis and clears the processing queue.

TABLE 5. Shortcuts available to control speech synthesis.

Command	Function	Description
Ctrl+Alt+S	SpeechSynthesis.speak()	Start speech synthesis at user request.
Ctrl+Alt+P	SpeechSynthesis.pause()	Stop speech synthesis.
Ctrl+Alt+R	SpeechSynthesis.resume()	Restart speech synthesis.
Ctrl+Alt+C	SpeechSynthesis.cancel()	Cancel speech synthesis.

Software Testing Qualifications Boards (ISTQB) Software Testing Certification Body [27], the ISO/IEC/IEEE 29119 standard [28], and the Keystroke-Level Model (KLM) GOMS [29]. At this point, it is worth emphasizing that IrisMath is not required to comply with all the specifications of the aforementioned standards, but they have served as a guide to correctly focus the tests and appropriately dimension resources such as time and (computer and human) resources.

B. EVALUATION METRICS

We evaluated our system using the following metrics:

- **Average duration to perform an action:** We assessed the time required to complete the following actions: accessing the course, accessing the activity, script execution, modifying the script, re-execution of the modified script, and downloading the script. These actions were evaluated by both visually impaired and non-visually impaired users for comparison purposes. Non-visually impaired users served as the baseline.
- **Self-Assessment Manikin (SAM) survey:** This survey measures pleasure, arousal, and dominance using a 9-point scale for various parameters, including equation input, interface interaction, interpretation of audible indications, and result transcription. These indicators directly impact the usability and satisfaction levels of the target users.

C. RESULTS

Table 6 summarizes the number of test cases executed without issues related to functional and non-functional requirements. These results show that 88% of the test cases associated with functional requirements were executed without issues, and 100% of the test cases associated with non-functional requirements were executed without issues. Among the most important functional test cases are equation input, interaction and navigation between interfaces, interpretation of audible indications, and result transcription.

TABLE 6. Summary of test cases associated with the IrisMath platform requirements.

Type of requeriment	Number of requeriments	Associated test cases	Approved test cases
Functional	19	26	23
No Functional	7	18	18
Total	26	44	41

TABLE 7. Average duration, in seconds, of actions performed by visually impaired and not visually impaired users.

Action	Average duration visually impaired users	Average duration non visually impaired users
Accessing the course	8.30	4.15
Accessing the activity	10.50	5.30
Script execution	9.10	6.05
Modifying the script	251.75	213.10
Re-execution of the modified script	8.95	5.55
Downloading the script	7.35	4.85
Total	295.95	239.00

Regarding usability, the completion time in seconds for a task composed of (i) accessing the course and the corresponding activity, (ii) executing the preconfigured script, (iii) modifying parameters in four cells, (iv) re-evaluating the script, and (v) downloading the new script locally, are presented in Table 7. The average activity completion times are compared between a visually impaired and a non-visually impaired user. In the table it is observed that the total activity performed by both groups takes 295.95 seconds (4.9 minutes) and 239 seconds (3.9 minutes), respectively, which does not represent a time in which a person can lose track of the activity or become bored with it. These results are encouraging since the visually impaired user was able to complete all the actions correctly and in a time that is about 1 minute slower than a user without visual disabilities. The longer duration in performing the actions by visually impaired individuals is due to the auditory indications, which are reproduced as guides and sometimes require additional reproductions.

Finally, emotions associated with the use of IrisMath were evaluated considering 3 aspects from the Self-Assessment Manikin (SAM) survey: pleasure, arousal, and dominance, with respect to the parameters equation input, interface interaction, interpretation of audible indications and transcription of the results. The evaluation uses a 9-point scale, whose results are shown in Table 8. From a general perspective,

TABLE 8. Assessment of user perception regarding the key functions of the IrisMath platform. The evaluation uses a 9-point scale.

Parameter	Pleasure	Arousal	Dominance
Equation input	4.58	7.58	5.50
Interface interaction	7.58	6.17	5.50
Interpretation of audible indications	6.67	7.42	7.58
Result transcription	6.83	7.22	7.67
Average	6.42	7.26	6.56
Percentage	71.33%	80.67%	72.89%

IrisMath has achieved a high level of acceptance from users and has demonstrated solid results in most of the evaluated parameters. However, some results have raised concerns, and we have conducted additional queries to users to delve deeper into these issues. Specifically, we would like to address the following points:

(i) The parameter "Equation Input" received a low level of satisfaction, which could be attributed to two factors: firstly, the system presents a delay in the descriptive audio of the keys that were pressed, and secondly, the system does not allow users to navigate through the input, meaning that the elements eliminated when the user presses the "backspace key" are not indicated.

(ii) The low levels of dominance observed in both the "Equation Input" and the "Interface Interaction" parameters have led us to consider providing more time for users to become familiar with all the elements and commands that make up IrisMath. We believe that allowing users more time to become familiar with the system will improve their experience.

V. CONCLUSION

This work presented IrisMath, a web-based Computer Algebra System aimed at visually impaired people that use Maxima as a math engine. Our system is inspired by Jupyter Notebooks, where code cells allow individuals with visual disabilities to execute mathematical operations. IrisMath employs text-to-speech messages to communicate with the user about both the input operations as well as their result.

Our system has been evaluated by both sighted and visually impaired individuals, demonstrating promising results in terms of the time required for users to complete various activities as well as acceptance regarding its use. Specifically, 88% of functional requirements were executed correctly, while 100% of non-functional requirements were met. Although the SAM surveys yielded positive results, we noted that the system has a learning curve that visually impaired users need to overcome. In fact, a steep learning curve is typically expected for CAS software, even for sighted people, due to the intrinsic difficulties of mathematics.

In this study, we have adopted a microservices architecture, which is well-suited to the particular objectives and demands of the project. By decomposing the application into smaller, loosely coupled services, this architecture offers modularity, flexibility, and scalability, enabling independent development, deployment, and scaling of each service. As a result, maintenance becomes more manageable, and development cycles accelerate. Nevertheless, it is worth noting that alternative architectures exist for developing similar applications. Considering these alternatives may be worthwhile for future research or work. In addition, future work may explore the inclusion of more features to IrisMath, such as the ability to plot graphs or allow users to execute source code in programming languages. Another important enhancement would be to include a grading system that allows teachers to evaluate visually impaired users as explored in [30].

REFERENCES

[1] E. B. M. Elsman, M. Al Baaj, G. H. M. B. van Rens, W. Sijbrandi, E. G. C. van den Broek, H. P. A. van der Aa, W. Schakel, M. W. Heymans, R. de Vries, M. P. J. Vervloed, B. Steenbergen, and R. M. A. van Nispen, "Interventions to improve functioning, participation, and quality of life in children with visual impairment: A systematic review," *Surv. Ophthalmology*, vol. 64, no. 4, pp. 512–557, Jul. 2019.

[2] WHO. (2019). *Who Launches First World Report on Vision*. Accessed: Jul. 22, 2022. [Online]. Available: https://www.who.int/es/news/item/08-10-2019-who-launches-first-world-report-on-vision

[3] CONADIS. (2022). *Disability Statistics*. Accessed: Jan. 8, 2023. [Online]. Available: https://www.consejodiscapacidades.gob.ec/estadisticas-de-discapacidad/

[4] A. Díaz, A. G. López, and A. de la Villa Cuenca, "An example of learning based on competences: Use of maxima in linear algebra for engineers," *Int. J. Technol. Math. Educ.*, vol. 18, no. 4, pp. 177–181, 2011.

[5] Pearson Accessibility for Assessments. (2023). *Accessible Equation Editor*. Accessed: Mar. 1, 2013. [Online]. Available: https://accessibility.pearson.com/resources/aee/aee-start.php

[6] MDN. (2022). *MathML*. Accessed: Aug. 31, 2022. [Online]. Available: https://developer.mozilla.org/en-U.S./docs/Web/MathML

[7] R. Neeser. (2022). *L-Math: Linear Algebra for Geometric Applications*. Accessed: Mar. 2, 2022. [Online]. Available: https://l-math.common-lisp.dev/

[8] W. Schweikhardt, C. Bernareggi, N. Jessel, B. Encelle, and M. Gut, "LAMBDA: A European system to access mathematics with Braille and audio synthesis," in *Proc. Int. Conf. Comput. Handicapped Persons*, vol. 4061, 2006, pp. 1223–1230.

[9] J. A. Borges, "Dosvox um novo acesso dos Cegos à cultura e ao trabalho," *Revista Benjamin Constant*, vol. 3, no. 1, Apr. 2017.

[10] P. H. M. Campoverde and L. C. Martini, "Calculadora financiera FINANVOX: Herramienta informática educativa de apoyo para deficientes visuales en Su proceso de formación académica," in *Proc. Brazilian Symp. Comput. Educ.*, vol. 1, 2012, pp. 872–875.

[11] H. da Mota Silveira, "Extensão de recursos e plano de avaliação do matvox: Aplicativo matemático programável de apoio para deficientes visuais," in *Proc. Anais do XXII SBIE (XVII WIE)*, 2011, pp. 592–595.

[12] P. Mejía, L. C. Martini, F. Grijalva, and A. M. Zambrano, "CASVI: Computer algebra system aimed at visually impaired people. Experiments," *IEEE Access*, vol. 9, pp. 157021–157034, 2021.

[13] J. Sweller, "Cognitive load theory and educational technology," *Educ. Technol. Res. Develop.*, vol. 68, no. 1, pp. 1–16, Feb. 2020.

[14] B. M. Randles, I. V. Pasquetto, M. S. Golshan, and C. L. Borgman, "Using the jupyter notebook as a tool for open science: An empirical study," in *Proc. ACM/IEEE Joint Conf. Digit. Libraries (JCDL)*, Jun. 2017, pp. 1–2.

[15] F. Pezoa, J. L. Reutter, F. Suarez, M. Ugarte, and D. Vrgoč, "Foundations of JSON schema," in *Proc. 25th Int. Conf. World Wide Web*, Apr. 2016, pp. 263–273.

[16] C. Győrödi, R. Győrödi, G. Pecherle, and A. Olah, "A comparative study: MongoDB vs. MySQL," in *Proc. 13th Int. Conf. Eng. Modern Electr. Syst. (EMES)*, Jun. 2015, pp. 1–6.

[17] O. Caprotti and D. Carlisle, "OpenMath and MathML: Semantic markup for mathematics," *XRDS, Crossroads, ACM Mag. Students*, vol. 6, no. 2, pp. 11–14, Nov. 1999.

[18] M. Kohlhase and F. Rabe, "Semantics of OpenMath and MathML3," *Math. Comput. Sci.*, vol. 6, no. 3, pp. 235–260, Sep. 2012.

[19] J. F. Sepúlveda and L. Ferres, "Improving accessibility to mathematical formulas: The Wikipedia math accessor," *New Rev. Hypermedia Multimedia*, vol. 18, no. 3, pp. 183–204, Sep. 2012.
[20] D. McKain. (2011). *Snuggletex Generating Content MathML*. Accessed: Mar. 4, 2023. [Online]. Available: https://www2.ph.ed.ac.uk/snuggletex/documentation/generating-content-mathml.htm
[21] D. Cervone, P. Krautzberger, and V. Sorge, "Towards universal rendering in MathJax," in *Proc. 13th Int. Web All Conf.*, Apr. 2016, pp. 1–4.
[22] G. Yang and J. Saniie, "Sight-to-sound human-machine interface for guiding and navigating visually impaired people," *IEEE Access*, vol. 8, pp. 185416–185428, 2020.
[23] M. Collier and R. Shahan, *Microsoft Azure Essentials Fundamentals of Azure*. Redmond, WA, USA: Microsoft Press, 2015.
[24] A. Mardanov, *Express.Js Guide: The comprehensive Book Express.Js*. Victoria, BC, Canada: Lean Publishing, 2013.
[25] P. Mejía, L. C. Martini, F. Grijalva, J. C. Larco, and J. C. Rodríguez, "A survey on mathematical software tools for visually impaired persons: A practical perspective," *IEEE Access*, vol. 9, pp. 66929–66947, 2021.
[26] P. Bourque and R. Fairley, *Guide to the Software EngineeringBody of KnowledgeVersion 3.0 SWEBOK*. Piscataway, NJ, USA: IEEE Press, 2014.
[27] *Software Testing Qualifications Board, Y Hispanic America Software Testing Qualifications Board*, International Software Testing Qualifications Board, Probador Certificado del ISTQB. EEUU: CTFL-ISTQB, Tampa, FL, USA, 2018.
[28] S. W. G. AEN/CTN71/SC7/GT26. (2022). *ISO/IEC/IEEE 29119 Software Testing Standard*. Accessed: Jan. 8, 2023. [Online]. Available: https://in2test.lsi.uniovi.es/gt26/?lang=en
[29] S. K. Card, T. P. Moran, and A. Newell, *The Psychology of Human Computer Interaction*. Mahwah, NJ, USA: CRC Press, 2008.
[30] A. M. Zambrano, S. Corrales, C. Parra, F. Grijalva, and J. A. Zambrano, "A prototype software for grading management aimed at visually impaired teachers," in *Proc. IEEE 6th Ecuador Tech. Chapters Meeting (ETCM)*, Oct. 2022, pp. 1–6.

ANA M. ZAMBRANO received the engineering degree in electronics and information networks from Escuela Politécnica Nacional, Quito, Ecuador, in 2010, the master's degree in communication technologies, systems and networks from the Polytechnic University of Valencia, in 2013, through a scholarship in Ecuador, and the Ph.D. degree in telecommunications, in 2015. She is currently with Escuela Politécnica Nacional. Her research interests include real-time applications, distributed systems, and the Internet of Things together with smart cities.

DANILO I. PILACUÁN was born in Quito, Pichincha, in March 1995. He received the bachelor's degree in computer science. He is currently pursuing the degree with the Faculty of Electronic Engineering and Information Networks, Escuela Politecnica Nacional. His research interests include the development of web applications in languages, such as C#, Java, and Python, and also the development and implementation of the IoT hardware prototypes.

MATEO N. SALVADOR was born in Quito, Ecuador, in 1999. He is currently pursuing the engineering degree in information technology (IT). He studies with Escuela Politécnica Nacional, Quito. In 2022, he received a Scholarship for Academic Excellence from Escuela Polítecnica Nacional. His research interest includes the development and optimization of web applications and network security.

FELIPE GRIJALVA (Senior Member, IEEE) received the B.S. degree in electrical engineering and telecommunications from the Army Polytechnic School, Quito, Ecuador, in 2010, and the M.Sc. and Ph.D. degrees in electrical engineering (major in computing engineering) from the University of Campinas, Campinas, Brazil, in 2014 and 2018, respectively. He is currently a full-time Professor with Universidad San Francisco de Quito (USFQ), Quito. His research interests include spatial audio, machine learning and computer vision applications, and assistive technologies aimed at visually impaired people.

NATHALY OROZCO GARZÓN (Senior Member, IEEE) received the electronic and telecommunications engineering degree from Armed Forces University-ESPE, Ecuador, in 2011, and the M.Sc. and Ph.D. degrees in electrical engineering from the University of Campinas (UNICAMP), Brazil, in 2014 and 2018, respectively. She obtained the HCIA-5G Certification from Huawei, in 2020. She is currently an Assistant Professor with Universidad de Las Américas (UDLA), Quito, Ecuador. Her research interests include digital and wireless communications, computer applications, software simulation, MIMO, cognitive systems, machine learning, and 5G technologies.

HENRY CARVAJAL MORA (Senior Member, IEEE) received the B.Sc. degree (Hons.) in electronics and telecommunications engineering from Armed Forces University-ESPE, Ecuador, in 2009, and the M.Sc. and Ph.D. degrees in electrical engineering from the School of Electrical and Computer Engineering (FEEC), University of Campinas (UNICAMP), Brazil, in 2014 and 2018, respectively. He was the Director of the Technology Transfer Area in the Education, Science and Technology Secretariat (SENESCYT), Ecuador, in 2018. He obtained the HCIA-5G Certification from Huawei, in 2020. He is currently an Assistant Professor with Universidad de Las Américas (UDLA), Ecuador. His research interests include wireless communications, physical-layer security, 5G and B5G technologies, software simulation, computer applications, and machine learning.