Education and Information Technologies (2022) 27:1585–1607
https://doi.org/10.1007/s10639-021-10663-4

Introduction of two new programming tools in Bengali and measurement of their reception among high-school students in Purba Bardhaman, India with the prototypic inclusion of a vector-biology module

Bishnu Goswami[1] · **Sarmila Pal**[2]

Received: 22 December 2020 / Accepted: 11 July 2021 / Published online: 29 July 2021

Abstract

We introduce two new software tools, Bongojontro and Bongojontro Baksobandi, aimed at reducing the barriers to programming for native speakers of Bengali, the fifth most spoken language in the world. The highlights of these software include programming in the native language of Bengali, simpler construction of programs which is friendly for beginners, and the possibility of creating and using modules which can be used to incorporate a level of abstraction that can be helpful for users of different technical skills and roles. We introduced the software to students of two semirural schools in Purba Bardhamman, West Bengal, India. The participants were a section of class XI students of age group 16–17 from both the schools. 40 students provided the full data in the succeeding survey, with 2 providing incomplete data. Among those who participated in the survey, it was found that the reception was overwhelmingly positive, with mean score greater than 6(out of 7) in 12 out of 15 survey questions. The scores were especially high on the usage of their native language on the software and its easy workflow. However, the mean "ease of learning" score was a bit low (4.45/7) compared to the other high ratings. The prototypic vector-biology module, which was a part of Bongojontro Baksobandi, also received very favorable reviews. Further work along these lines using the software and its modules seems to be a promising avenue for useful research and inclusive development in education and information technologies.

Keywords Vector biology · Programming · Beginner · Education · Native language · Bengali

✉ Bishnu Goswami
bishnu.cert@gmail.com

Extended author information available on the last page of the article

1 Introduction

Computer literacy and usage of technological tools have been greatly emphasized in the recent years in most countries, including India, as a part of education as well as a notable skill for the job seekers in the economy. In the context of education, programming is one of the computing skills that is in high demand and will be in demand for the near future. However, there are significant barriers to learning programming among school-age students in various developing or newly industrialized countries, India being an example of the latter. One of these barriers is the barrier of language (Acharya Himanshu, 2019), as most programming tools are in English. Secondly, programming in many of the popular languages are complex and require considerable training to use them.

There has been research initiatives to incorporate notions of basic programming in schools. This has been tried in primary schools, such as by directly measuring the perception by the children (Chiazzese et al., 2017) or through conducting workshops for teachers in primary schools (Monjelat, 2017), among others. Measuring the effect of using beginner programming tools on student engagement, reflective thinking and problem solving skills have also been tried recently (Yildiz Durak, 2020). Perception of the beginner friendly programming tool Scratch was studied among secondary school children in South Africa, with the results indicating older students finding it useful only for introductory study (Marimuthu & Govender, 2018).Correlating the effects of programming on reasoning and mathematical skills for older students, of high school age, was another direction of research (Psycharis & Kallia, 2017). In the socio-cultural context, research on structured approach to teaching programming through newer perspectives, for example, have been researched with positive results (Sentance et al., 2019). All of these points towards the usefulness of newer tools and methods to teach programming in the school level in more efficient and effective ways. In our focus on the barriers of language, a study spanning 86 countries with 840 responses points to the fact that non-native English speakers barriers with reading instructional materials, technical communication, reading and writing code, and simultaneously learning English and programming(Guo, 2018).

Here in this paper, we introduce two new programming tools which can introduce the learners to code in their native language of Bengali. Bengali is an Indo-Aryan language which is spoken by 228 million native speakers as their first language and by another 37 million as their second language, making the language the fifth most spoken language in the world (CIA, 2018). The language is the official and national language of Bangladesh and is the official state language in two states of India, namely West Bengal and Tripura.

The first of our newly introduced programming tools is named Bongojontro, which can introduce the learners to code in their native language of Bengali. The second tool another application called Bongojontro Baksobandi, which uses a modular approach and can be used by coders as well as users who'll use the coded software. The two tools are interoperable. In the second application, a module based on vector-biology is made, and it is used as a prototype to show

how it can be used in the context of vector biology. In the said module, geospatial information system, in the form of publicly available open-source maps, is also illustrated, as well as the possibility of linking to a popular email service is illustrated.

In the Covid-19 pandemic, the importance of using computer-based learning tools has skyrocketed, with many schools and colleges resorting to online-only modes of education for the time being. However, many developing and newly industrialized countries are far from being accustomed to this new mode of learning. For example, in the case of Massive Open Online Courses (MOOCs), only about 10% of the users finish them (Alraimi et al.), which shows that there is a lot to be done to make the users literate and motivated enough to complete them. For students of the lower socioeconomic classes and rural area in India, literacy in basic computing is still in its infancy, and measures of inclusive development using technology are being undertaken on large scale to rectify the gap. Only about 20.66% of rural students and 69.70% of urban students used computers for various academic purposes in selected districts of the relatively developed state of Andhra Pradesh (Sampath Kumar & Shiva Kumara, 2018), for example. Information and communication technologies are also being used in parallel to address multiple literacies including health literacy, financial literacy and eSafety for low-literate learners (Nedungadi et al., 2018). One of the drawbacks of such initiatives is the language barrier as most forms of computer literacy depend on a working knowledge of English, especially when instructions to use various programming languages or GUI tasks are involved. The language barrier is sometimes still there even if the student was taught in an English medium school, such as in a study done on a suburban school (Bhattacharya, 2013).

In the context of using computing tools in public health, where a prototype is shown in this paper which was developed using Bongojontro Baksobandi, there has been a lot of development in the last five years. The prototypical example vector-biology module in this paper has drawn inspiration from the much larger multi-platform dengue-entomological surveillance system used in Malaysia (Ibrahim & Quan, 2017). Household surveys, such as the done in rural Kenya (Ngugi et al., 2020), can be incorporated with information technology tools in countries like India which has cheap internet rates. In developed countries, the combination of these two subjects has been observed in a study where an application was developed to combat Zika virus transmission using web and mobile technologies (Neira-Tovar et al., 2018). Previously an android application with the focus on vector biology and associated public health context was developed and corresponding survey work was done around it in the neighboring country of Bangladesh (Goswami et al., 2020a, b) and India (Goswami et al., 2020a, b), from where the module in Bongojontro Baksobandi was inspired from.

The latest version of Bongojontro and Bongojontro Baksobandi is available at the website "https://bongojontro.itch.io/main" (without quotes).

2 Bongojontro implementation

Bongojontro is developed to act as a software application to create other applications using the Bengali language. The alphabet has been kept in English so as to support the commonly available hardware keyboards and to prevent confusion among spellings, as there are a lot of different spellings for the same word in Bengali script, which is somewhat rectified by the limitations of phonetics in the standard US-English keyboard. Moreover, the application is developed to make software development much simpler than many competing standards, with inbuilt support of various GUI elements which can be utilized in the code the user writes in Bongojontro.

2.1 The GUI and programming environment

The main screen of the application consists of the logo and has an editor underneath. When the user inputs in the keyboard, the role of the editor becomes apparent, and the letters are shown on the screen. Keeping up with the trend of modern applications, the interface is free from clutter, with only a tiny logo in the top-left corner and a plain violet background. By pressing the F1 key, as customary in various other programs, basic help is provided, in both the languages of English and Bengali (Fig. 1).

The major functions of the IDE are explained with the keyboard shortcuts attached to them. Besides the usual usage of backspace and related keys, these four keys perform special functions:

- F1 (Function key 1): Displays the help
- F5 (Function key 2):Saving the current code in a text file
- F9 (Function key 9):Executing the current code being displayed

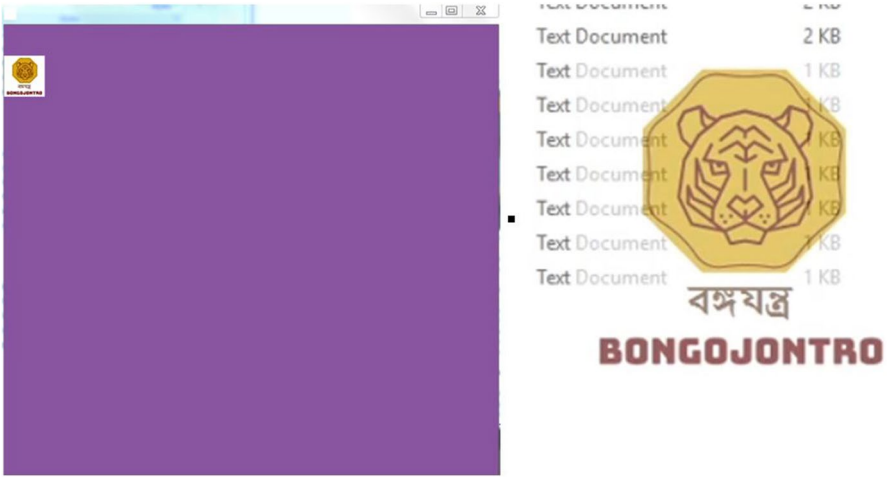

Fig. 1 The starting screen of Bongojontro and the translucent logo being shown at startup

- F12 (Function key 12): Executing the code which was copied previously, such as from a text-editor like Notepad++.

The GUI supports the basics like text reflow and large size of text which is suited for beginner programmers and computer users in general (Fig. 2).

When it comes to the complete programming environment, Bongojontro is expected to run on all Windows 7 (including older 32-bit versions) Operating System (OS) versions and above, such as the current builds of Windows 10, older Windows 10 versions, Windows 8.1 and 8. It has been tested to work in many of such environments. It is tiny in size, with the current version taking up only 2.57 MB of disk space. The executable file is also portable, in the sense it can be copied to other devices without the requirement of installation.

The code editor in GUI is very basic and is suited only for very simple programs. Usage of programming text-editors, such as the lightweight and open-source Notepad++is recommended in the included help file. In the future versions of Bongojontro and Bongojontro Baksobandi, we plan to include a file to be used with "User Defined Languages", a feature of Notepad++. Using the included file, the said text editor will support better formatting and highlighting of Bongojontro and Bongojontro Baksobandi programs, making it even easier for programmers to understand and edit the code.

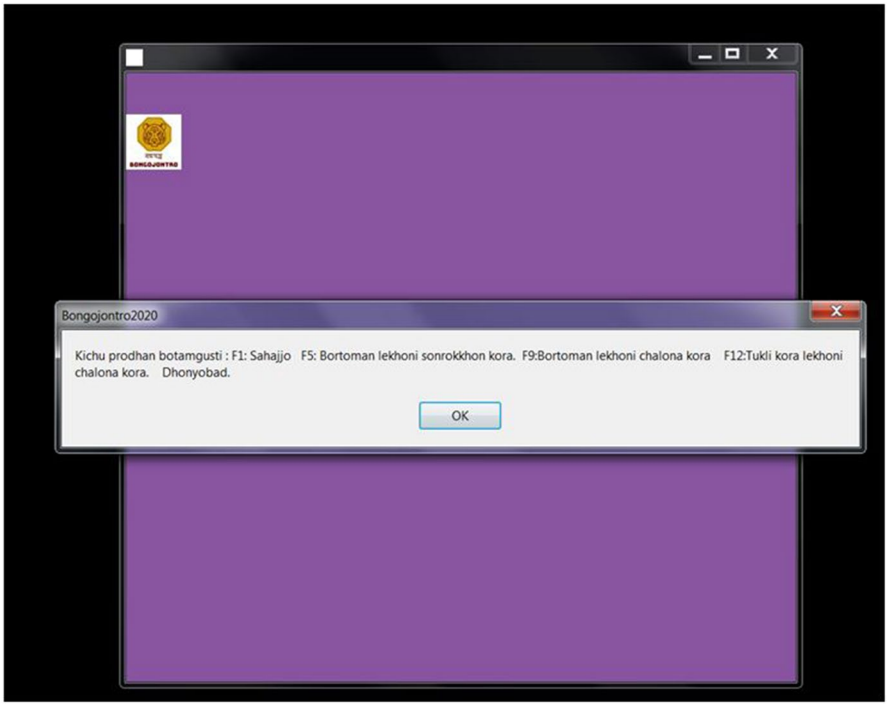

Fig. 2 The minimal help dialog in the GUI

2.2 Common programming operators

Bongojontro supports the common programming operators such as those shown in the table below:

+,-,*,/	Addition, subtraction, multiplication, division
=	Assignment operator
==	Compares two variables, returns true if a variable is equal to the other variable
!=	Compares two variables, returns true if a variable is NOT equal to the other variable
>	Compares two variables, returns true if the value on the left is greater than the value on the right
<	Compares two variables, returns true if the value on the left is smaller than the value on the right
&&	Boolean AND
||	Boolean OR
"	Quote for (eg:message) strings

2.3 The Programming commands

Bongojontro supports the following commands, and more can be added through a modular approach through Bongojontro Baksobandi. All these commands start with a "`" (without quotes), to make the beginner user aware that a specific command is being written. A few among the list of commands which is available in the latest version of Bongojontro is shown below, along with a short description and example. These texts were also provided as a help file within Bongojontro and Bongojontro Baksobandi, and the tone is similar to an user guide of the programming syntax. The included sample programs also utilize many of the commands.

Barta

In the context of programming, *barta (*Bengali for 'a message') can be regarded as one of the simplest commands. Using *barta*, the string inside this function will show up in the computer display like a pop-up letter.

The command *barta* will show up multiple times in a random program the programmer writes. Therefore, the correct usage of this command should be well-noted by the programmer.

Example

This command should be used in Latin alphabet in the current version of the program. Here, the pop-up message "Nomoskar Prithibi"("hello world") will be the result while running the code.

```
`barta("Nomoskar Prithibi");
```

Jodi, Notuba and Njdi

In programming, *jodi* ('if in English) implies a condition. A program can be written and run without the use of conditions. An example of such a program is the

display of a simple message. However, for slightly more complex programs, we almost always see the application of some conditions.

Notuba implies the alternative to be followed if the conditions in jodi are not met. Its English equivalent is "else".

Njdi, implies another alternative, separate from the first *jodi*, and also separate from notuba, the latter being the default if none of the *jodi* and (possible multiple) *notuba*(s) is true. Its English equivalent is "else if".

Using *jodi*, a part of the program can be kept separate from the other parts. For example, let us consider the case when the day of the week is asked to the program operator. If the day is between Monday and Friday, then a pop-up box should display "Working day". If the day is a Saturday or a Sunday, then a pop-up box should display "Holiday", using the command *notuba*. Here, these two pop-up messages are separate from each other. The usage of *jodi* and *notuba* is required for the successful results in this case. If Saturday is to be considered as a half day, *njdi* can be used to keep that part separate.

Example(s)

This command must be used in Latin alphabet in this version. The Latin command is *jodi*. With this command, the commands *notuba* and *njdi* is used in this example.

Here, a day of the week is inputted and it is determined if the day is a holiday, a half day, or a working day.

```
____
kau = 1
baar = `nau_lekha("Chuti nirnoi ","Barti likhun","sombar");
`jodi(baar = = "sombar"||baar = = "mongolbar"
||baar = = "budhbar"||baar = = "brihospotibar"
||baar = = "sukrobar")
{
`barta("Din ti kajer din.");
}
`njdi(baar = = "shonibar")
{
`barta("Din ti ordhodibosh chutir din");
}
`notuba
{
`barta("Din ti chutir din");
}
____
```

Choluk

In the current context of programming, *choluk* is an important command. various scenarios in programming calls for the repetition of a command-set multiple times. In those cases, *choluk* can be very useful as a command. Its English language programming equivalent is "while", as in a "while-loop".

As an example, using *choluk*, a set of commands 100 lines long can be expressed only in two lines.

choluk must be used with a condition. This condition ensures that commands under the ambit of *choluk* continues to run as long as the condition is met. Care must be taken to avoid conditions that can result in an infinite loop.

Example(s)

The command *choluk* is being used here to display the numbers 1 to 10 in 10 consecutive pop-up messages. In programming, it is not required to keep single line statements of a loop under the second-brackets (braces). But as we use two statements under the loop, the second-brackets are necessary.

Currently, the command should be written in the Latin alphabet in the form of *choluk*. In this example, the programmer must also know the usage of the command *barta*.

```
____
kau = 1
`choluk(kau < 11)
{
`barta(kau);
kau = kau + 1;
}
____
```

As this loop starts from 1, it must be run until the number 11(where it returns false and the two statements within the loop are ignored). Alternatively, the examples below will also yield the same results. The difference between the operators ' < ' and ' < = ' must be kept in mind for these examples.

```
____
kau = 0
`choluk(kau < 10).
{
`barta(kau);
kau = kau + 1;
}
____
```

or,

```
____
kau = 1.
`choluk(kau < = 10).
{
`barta(kau);
kau = kau + 1;
}
____
```

Nau lekha

Sometimes while programming, we may require the user to input a string of letters. This can be a name, an address, or some code numbers for a specific use.

Currently, this command should be written in the Latin form, as *nau_lekha.* A default string can be set for this input. If the programmer doesn't want to include a default string, a double quotes ("") can take its place.

It should be kept in mind that this command should not be used to directly input numbers. The returned string should be manipulated if the string is to be manipulated like numbers.

Example(s)

As an example, we can write:

```
naam=`nau_lekha("Nam
nothiboddho","Apnar nam likhun","Ajay");
```

Here, a pop-up message will be shown to the user. It will contain the string "Apnar nam likhun" under the heading "Nam nothiboddho". In the input field, the name Ajay will be shown. If the user doesn't change this name, the default value of Ajay will be considered for the variable naam.

If the user deleted the default string, the variable will be containing an empty string(""), as in the example below:

```
naam=`nau_lekha("Nam
nothiboddho","Apnar nam likhun","");
```

Sonkha

It is often a requirement in a program to get a string from the user and convert it into a number. In those cases, this command is useful.

This command should be entered using the Latin alphabet in the current version of the software. This command should be applied to a string. This string can be declared in the body of the text itself. Alternatively, the user can be asked to input a string containing only numbers.

For the second scenario, the user can be asked to input a string using the command nau_lekha.

Numbers can be added, subtracted and other arithmetical operations can be performed on them. As these are not available for strings, it is important to convert the strings to numbers first.

Example(s)

We can use:

```
khoroch_lekha=`nau_lekha("Khoroch","Ei masher
khorcha koto taka holo?","5000");
lekha=`sonkha(khoroch_lekha).
```

In the above example, a pop-up input box will be displayed and in the Latin alphabet, the string " Ei masher khorcha koto taka holo?" will be displayed. The default value of 5000 (in string) will also be shown.

It must be well understood that the return from this command will be a string, and not a number. In the last command, the variable lekha will convert the string to a number and save it in that format using the sonkha command. For this code to work correctly, no non-number string should be entered in the pop-up input box. Some trimmings from the string might be necessary from the string if the need arises.

Lekha

Contrary to the last command, it is sometimes required to take a number and convert it into a string. In those cases, this command is useful.

Currently this command must be inputted using the Latin alphabet. This command should be performed over a number.

Numbers can be used for arithmetic manipulations. But one of the drawbacks of numbers is that they cannot be written one after the other in a message without some explicit separation. The examples will make it clear.

Example(s)

As an example, if we do not use the command *lekha*, as in this code:

```
maine = 150;
bonus = 100;
`barta("Maine ebong bonus por por likhle
:" + maine + bonus);
```

We will not get the desired result.

In addition to the available commands previously described, the latest version of Bongojontro supports some additional commands which are currently at beta stages. These were previously available only in Bongojontro Baksobandi. These include the two commands:

Chitro

Chitro is the Bengali translation of "picture". Using this command, a picture file can be directly displayed to the user.

Example(s)

As an example, we can use command *chitro*, as in this code:

```
`chitro("data/im1.jpg");
```

It is mandatory that a folder named "data" is in the directory the executable is located. Inside that folder, a JPEG format file with the name "im1.jpg" must exist. If these two conditions are not met, the code will not run correctly.

Jal

Jal is the Bengali translation of "net". Using this command, an interactive browser window can be directly displayed to the user.

Example(s)

As an example, we can use command *jal*, as in this code:

```
`jal("www.wikipedia.org");
```

This will open the free online encyclopedia Wikipedia to the user. The user can then interact with the webpage.

A total of eight sample programs were provided with Bongojontro which could be copied to the clipboard and be executed after running an instance of Bongojontro. The usage of these sample programs were put forward to the later survey participants in a video clip along with narration of the usage of the said programs.

2.4 The modular approach

This approach is illustrated in a second executable of Bongojontro which is termed as Bongojontro Baksobandi (Bengali for the executable "in a box"). Bongojontro supports an innovative modular approach to programming which is expected to be particularly suitable for novice users. In this approach, code is divided into at least three files, *maap.txt, khata.txt* and *surur.txt.* In Bengali translation, the following three files mean "dimensions", "exercise copy" and "that which is in the start" respectively. The first file dictates the dimensions of the window and its title text, in just three lines of code, which are the width, height and the title text respectively. The second file dictates the elements that is to be shown in the window in the form of graphics. The third file dictates the code that is to be run in start-up. More files are supported in later beta versions but the focus for now is on these three files which are suited for beginner programmers and users.

This modular approach introduces an additional layer to the software paradigm, which differentiates programmers and users who will run the program but not write any code. This makes Bongojontro a platform to build software atop of.

An use of the modular approach is given in a model application. The application is related to insect vector-biology in the context of public health. In the application, the user is informed in Bengali language that they will be shown the picture of three genera of mosquitoes, which are of *Aedes*, *Culex* and *Anopheles,* and they have to input which species are found in their region which has the menace of mosquito-borne diseases. Then the application shows three diagrams of the three genera. Using a graphical box, the user then can enter the genera of mosquitoes that are menacing their own area. A second box is prompted, where the user can enter the diseases that are being caused by the mosquitoes in their area (an example disease, such as Dengue, is prefilled). After taking the input, a third box is prompted, which asks the user to input what personal protection methods they do use to curb the spread of mosquito bites and subsequent disease-risk. The example of *mosari* (Mosquito net) is prefilled in this box. After this, another GUI box is shown which asks the user to enter how satisfied they are with the existing eradication drives against mosquitoes, which is used to get a picture of the public health quality in the region. Finally, the coordinates of the region and a zoom factor are sought, which for the demonstration had to be inputted manually. After getting all the inputs, they are saved in a text file and the map of the region with some labeling is shown. For the demonstration, the open-source and free OpenStreetMap were used. An optional opening of online email website, GMail in the example, followed, where the data so far collected can be pasted and sent in an email. All of the dialogs inside the application was in the local language of Bengali (Figs. 3, 4, and 5).

2.5 The code of the previous module

We will now showcase the program code for the example of the modular approach. Below is a screenshot of the code in Notepad++. Notably, the code can be run

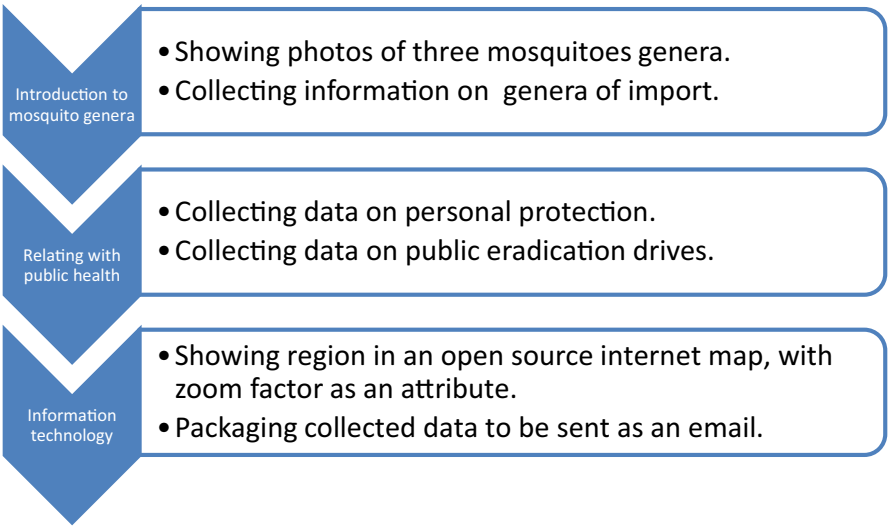

Fig. 3 The workflow of the vector biology module under Bongojontro Baksobandi

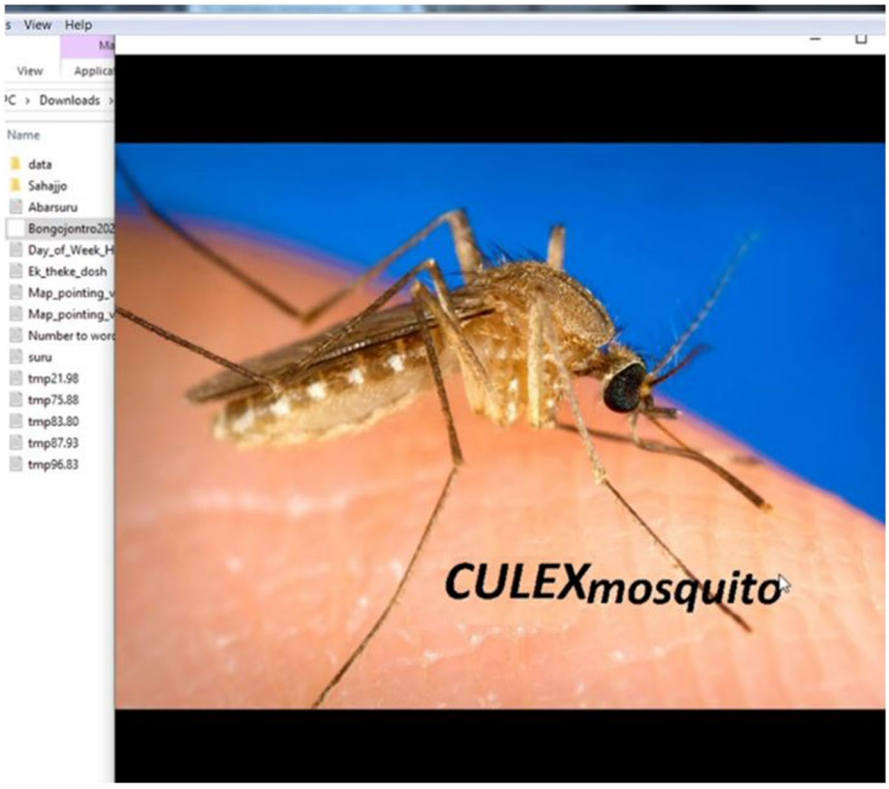

Fig. 4 The vector-biology module in use which utilized Bongojontro Baksobandi

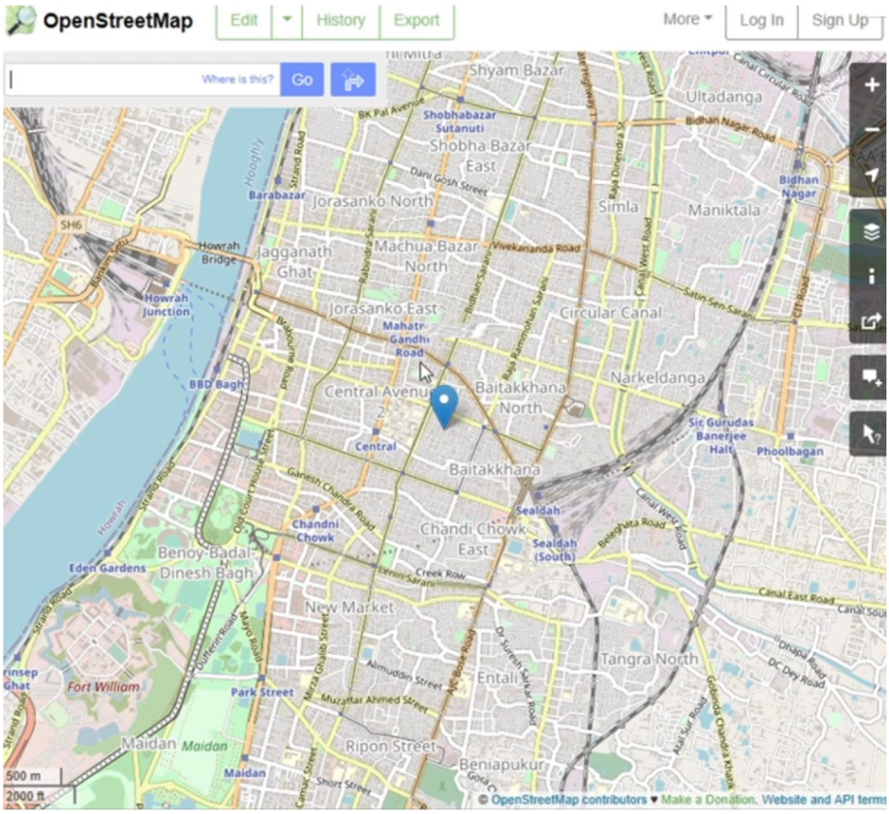

Fig. 5 An example map being shown in the context of vector biology in the same application. The zoom factor was set by the user to a 'distant' view

directly from the current version of Bongojontro. During the time the survey in the next section was conducted, the support of the two new commands were not available in Bongojontro and was available only on Bongojontro Baksobandi. In Bongojontro Baksobandi, an empty *khata.txt* file was included (no graphics were required) along with a *maap.txt* file which had the following three lines of code:

```
800
480
Mosha Protirodh
```

As indicated earlier, these three lines specify the dimensions of the GUI and the title text.

In the *surur.txt,* which specifies the code that is run at starting of the program, we had the following code. It is also shown in a figure, with the code being opened in the text editior Notepad++.

```
`barta(" Tinti moshar projatir chobi dekhano hcche. Erpor Draghima ar Okkhorekha dile map e seta dekhano habe.");
`chitro("data/im1.jpg");
`chitro("data/im2.jpg");
`chitro("data/im3.jpg");
projati=`nau_lekha("Moshar projati","kon projatir mosha mileche?","Aedes ar Anopheles");
rog=`nau_lekha("Elekai rogbalai","Elekai kon roger pradurvab dekha jacche?","Dengue");
bektiguto=`nau_lekha("Bektigoto surokha","Ki ki bektiguto surokha nan moshar birrudhe?","Mosari");
sarkari_obhijan=`nau_lekha("Elekai sorkari nirmul obhijan","Nirmul obhijan niye apni kota sontusto?","Motamuti");
draghima_text=`nau_lekha("Draghima Okkhorekha","Draghima Din","22.5726");
okkhorekha_text=`nau_lekha("Draghima Okkhorekha","Okkhorekha Din","88.3639");
zoom_text=`nau_lekha("Map er zoom","Zoom factor din","14");
`jal("www.openstreetmap.org/?mlat="+draghima_text+"&mlon="+okkhorekha_text+"&zoom="+zoom_text,0);
tothyo="Tothyosomuho: Latitude="+draghima_text+" Longitude="+okkhorekha_text+"Zoom factor="+zoom_text+"Projati="+projati+"Rogbalai="+rog+"Bektigoto surokha:"+bektigoto+" Sarkari obhijane sontusti"+sarkari_obhijan;
```

——

2.6 A description of the vector biology code line-by-line and highlights on usability

As we can see, the code is only 13 lines long, if we ignore the blank lines added for better readability and text-wrapping. The code is accompanied by three image files (im1,im2 and imp3, all in JPEG format) in a specific folder named "data". The details on each step from the vector-biology point of view are already discussed, so we will focus on the programming point of view here. The very first line is to show a message on the whole intent of the program. After the blank lines, lines 4,5 and 6 intend to display three images of interest which also contain the name of the item being shown as a part of the picture. In the next non-blank line 9, the program asks the user to correlate his findings(in the context of a simple study of vector biology: mosquito identification) with the images already displayed. This is done by collecting a string of text, with a default value of "Aedes ar anopheles". The data is saved in a variable. Lines 10–12 collects further data in a similar way, but without the pictures to correlate. They are also stored in separate variables. Lines 13–15 collect numerical data, but they are collected as strings because there is no need for arithmetical operations here. Line 17 opens an URL of maps and plugs in the relevant data collected from two previous variables, in string format. The final line stores

all of the collected data in a single variable which can be used later if the code is extended, in a single block of data (Fig. 6).

The following points about the code are evident, with some of them explained in the tutorial videos referred to in the succeeding section.

- Flow of the program: Each line in the code has a tangible purpose which can be understood by beginner programmer. The lines also illustrate the flow (top-to-bottom) of the operations the code performs. The code is supposed to infuse the minds of the programmer with a concrete step-by-step, easy-to-understand basic programming logic. In other examples, such as the one describing the command "choluk", modification to the usual top-to-bottom at once flow is illustrated. In yet another example, on the use of “Jodi” command, conditional execution of blocks of code is illustrated.
- Data types: The beginner programmer is introduced to various data types, although sometimes in an implicit fashion. Although there are more primitive types, the String type is highlighted in this module, as they are easy to understand and there are fewer chances of error while displaying them using other commands. The conversion of numbers to strings is also supported by using the "Lekha" command, and its use is shown in the example of the command "Choluk".
- Data handling: Concatenation of strings, using arguments in a function and use of default values are some data handling methods in the vector biology module. Opening files in the local file system (images in this case) and opening of websites is made accessible through simple commands.
- Standardization: It is not very uncommon that a beginner programmer learns to work in one computing environment and finds it difficult to migrate to other environments. Keeping this in mind, many parts of the code are similar to other established standard use computer environments. These include the use of brackets inside a function, use of double quotes to denote strings, use of a semicolon to denote end of statements, and use of curly braces in loops.
- Low investment in learning: Apart from the commands in Bengali, there are not any unique operators, special symbols or unique workflow to master. This helps in faster learning as well as easier migration to other established programming languages if the need arises.

Fig. 6 The code of the vector biology module in Notepad + + (free software). Specific syntax highlighting "add-on" for Notepad + + is planned for the next major version

2.7 The love of technology

While the students we worked with in the later methodology section were perhaps not very accustomed to programming on their own, they have worked on multiple projects in and outside their schools which relate to sociological aspects which are common in a less developed country. These include vector-borne disease eradication, environmental awareness and the use of audiovisual learning aids. Bridging the gap between those activities and programming technologies, where they could see that they can provide useful data for research using a program they can easily modify to suit different use cases, was perhaps one major reason they liked using Bongojontro and Bongojontro Baksobandi so much. The module we created was deliberately quite brief, so as to explain their workings in a short amount of time. However, with some more lines and modifications, the module can be utilized to conduct useful research in vector biology, such as by including multiple mosquito species and genera, and various entomological and public health websites where the users can utilize a two-way flow of data. Being more technologically literate would also be helpful to them in the current and near-future world which uses computer technologies in almost every nook and cranny in some way.

3 Survey methodology

3.1 Design

The developed software was tested for its usability, uptake and sustainability as a practical part in the higher educational system, where students can code using Bongojontro. The general public with basic knowledge of computing usage can probably use the software as users but not coders themselves, using the second executable of Bongojontro Baksobandi, due to its modular and layered nature. The aim of this research is to find out how this software is perceived by high-school students of a semi-urban school. To reduce socioeconomic bias, a semi-urban public school was chosen instead of an urban or rural school. The study pertained to the use of Bongojontro in two different schools. Participating students in both the schools were provided with the Bongojontro software and the Bongojontro Baksobandi software. The tutorial videos in the native language of Bengali along with the sample programs were also provided. The surveys were conducted based on the USE questionnaire (Lund, 2001) to get a preliminary picture of the satisfaction with the software application. This questionnaire was chosen as it measures the subjective usability of a product and it is non-proprietary and technology-agnostic. Being a new software product with a fresh paradigm, objective measures to gauge its usability is somewhat unclear at this point, thus the subjective usability was measured first. Additionally, the USE questionnaire has found face validity with relevant and unambiguous descriptions, and its reliability and validity have been reported by previous research (Gao et al., 2018) to be very positive.

3.2 Target participants

The introductory survey was conducted among high school students of the Purba Bardhamman district (under West Bengal, India), with the majority being from Talit Gourshar School and Jagdabad Sashi Bhusan Uccha Vidyalaya. The participants were of a section of class XI students of age group 16–17 from both the schools. Due to the restrictions of the Covid-19 pandemic, in-person visit to the schools was not possible. The regular in-person classes were also closed due to the pandemic, with only emergency purpose activities allowed in the schools. However, many classes were alternatively being conducted in online mode.

We initiated the survey using online video-conferencing and subsequent exchanges were performed in a virtual group on an encrypted messaging app. Meetings were conducted for each of the two schools. The students were briefed about the purpose of the survey by their teachers and we subsequently gave a brief introduction about the new software and the scope of the survey. All the participants in the meeting had some experience with basic computing from their school courses. The participants were shown a series of videos about the two applications and it was supplemented with the software applications being provided for download, along with the sample programs and the vector-biology module. The videos were also sent to them so that they can recheck them if needed.

The students who participated were asked to use the software and the modules for three days, and report back the results in the electronic survey form which will later be handed to them. It was made clear that the electronic forms will be available for only two days, and they should submit their responses within that time. No minimum time of use or other requirements were put forth.

3.3 Data collection and analysis method

After the completion of three days, the survey-result form (electronic) was opened up for input. It was specifically Google forms and was chosen for the supposed reliability of Google's servers. The guidance to filling up the forms, in case of confusion, was attended to by the teachers of the school, in addition to providing clarification by text. The questionnaire was in its original English form, but all the questions were verbally clarified to the students in their native language to prevent confusion. All the students studied English in their primary and secondary schools as a second language. Their proficiency with English, was however not that well, as is common in various public schools of India where English is usually the second language.

After two more days, the survey result form was closed from further input. A total of 42 responses were obtained, from which 2 were discarded because the forms were almost empty, where only the few compulsory entries were filled. Therefore, we had 40 participants who completed the introductory survey for this new software application. In future work, we hope to increase the number of participants and include more institutions, which was a problem back then due to the Covid situation.

4 Results

4.1 Quantitative results

Using the USE questionnaire, the quantitative aspects of the study were elucidated. In the first table, the usefulness of the Bongojontro and Bongojontro Baksobandi software was elucidated. In the context of general purpose programming by beginners, the responses are shown. This utilized a 1–7 point Likert Scale, where 7 is strongly agreeing with the statement and 1 is strongly disagreeing with the statement. The mean and standard deviation was calculated using the open-source software Libre Office Calc.

Statement	Mean score (Between 1–7)	Standard Deviation(up to 3 decimals)
1. It helps me be more effective	6.05	1.044
2. It helps me be more productive	6.25	0.865
3. It is useful	6.2	0.944
4. It makes the things one would want to accomplish easier to get done	6.05	0.944
5. It saves me time when I use it	6.05	1.044
6. It is easy to use	6.2	1.091
7. It is simple to use	6.25	1.024
8. It is user friendly	6.2	1.044

The next table shows the easiness of the operation of the software. Here, the vector-biology centered module made by Bongojontro Baksobandi was used as an example of application developed by Bongojontro Baksobandi, as well as the regular Bongojontro application was utilized as the subject which was scored in the survey. Participants rated their reviews based on how they liked using the previously mentioned vector-biology module in Bongojontro.

Statement	Mean score (Between 1–7)	Standard Deviation(up to 3 decimals)
9. It requires the fewest steps possible to accomplish what I want to do with it	6.25	0.921
10. I learned to use it quickly	6.15	1.315
11. I easily remember how to use it	5.9	1.687
12. It is easy to learn to use it	4.45	0.768
13. I am satisfied with it	4.7	0.602
14. I would recommend it to a friend	6.5	0.814
15. It is fun to use	6.3	1.095

4.2 Qualitative results

The survey also incorporated qualitative results such as remarks and two-way communication. From this, we can summarize as follows:

- Most people viewed the newly developed application in very positive light.
- Most of the participants praised the initiative, especially about using their native language. They noted that the usage of their native language in commands made some of the example codes much easier to understand.
- Many participants highlighted that the software was "useful" to them while learning about, and doing, programming.
- Some participants commented that the explanation videos need to be shorter and more user-friendly. For the short term, it is hoped that additional videos with a more step-by-step breakdown will be useful. For the long term, short workshops and in-person by-hand demonstrations will probably be useful.
- A few participants noted that as most of the programming tutorials available online were in English, they were difficult to learn from. They loved the tutorial in the videos which highlighted the usage of Bongojontro.
- The participants praised the ease of use of the vector-biology related software created with Bongojontro Baksobandi, commenting that it was easy to use and opined that most laymen can use the software with just a short introduction of what it is and how to use it.

5 Discussion

This study was aimed at introducing this new application of Bongojontro and aid in bottom-up development by empowering the students to have the possibility to begin programming in their native language of Bengali, and assessing its initial response from a small cohort of high-school students. In the shortage of quality teachers and infrastructure in the burgeoning field of basic computing and information technology, using native language in introductory programming has the potential to have far-reaching impacts, including the gradual erosion of the digital divide. As many students of public schools in rural and semirural parts of newly industrialized or developing countries like India are weak in English, the focus on regional language, as we found in this study, is a welcoming change. The similarities with other popular programming languages, such as the usage of braces and functions, also meant that transitioning to more established languages, such as Java or JavaScript, will be easy if the user needs to use it in their higher studies or in professional environments. Moreover, with the rising internet and smartphone usage in India, being acquainted with using software would be useful in yet unforeseen avenues of personal or professional development. The study showed that the reception of Bongojontro and Bongojontro Baksobandi was overwhelmingly positive among the users in the study.

Secondly, the work is also supportive of the "Make in India" initiative. In the initiative, the Government of India encourages manufacturing in India and dedicated investments in manufacturing. While software products do not strictly fall

under manufacturing, many other important processes utilize software on a day to day basis. If more software can be made in the home country, it can be possible to make it more secure, such as when data is stored locally, and more resistant to rogue software which includes backdoors or other nefarious tools. Exposure to software tools that are made in the same country, and sharing the same primary language can also invigorate the youth to be more amenable to the ideas of self-sustainability and lesser dependence on foreign-made software. As Bongojontro has similar syntax to many popular programming languages, transitioning to a more common language (usually in English) will also not be very difficult.

Thirdly, it was shown that using the vector-biology based module of Bongojontro Baksobandi, the study participants found it very easy to collect data about mosquitoes in an endemic area and put in its coordinates to showcase the region in an open source map. This opens up more possibilities in the use of software in the eradication drives which were previously requiring, by the very least, workers with a reasonable grasp of the English language. The code for the vector-biology module was also only around fourteen lines, making it easy to modify and understand for relative beginners. It has to be said that, however, for serious use, some more lines should be requires in the code (error checking, multiple webpages for different purposes etc.).

6 Conclusion

Bongojontro seemed to be an effective tool for beginner programmers in the study, who praised its ease of use, productivity, simplicity and fun of use. The survey responders responded very highly on the parameter of whether they would recommend the application to a friend. Programming in their native language of Bengali made the survey participants very satisfied and many of them lauded the initiative. In the context of Bongojontro being "useful", "easy to use" and "being user-friendly", the mean score of 6.2 out of 7 meant that the software fulfilled its role of being a good fit for budding programmers and program users in Bengali. Helping the young users in "being more effective"(6.05/7) and "being more productive"(6.25/7) were also on the list of high scores for this new software. On the other hand, the score was comparatively on the lower side(while still being high overall) on being "easy to learn to use it"(4.45/7) and "being satisfied with it"(4.7/7). This dichotomy probably points to the fact that while the software is user-friendly, some training still needs to be instilled on the young users and they might have difficulties learning all the examples and included modules on their own. On the question of "satisfaction", some questions still remain open. Were the students not highly satisfied because they are not introduced to more user-editable modules such as those on vector biology, or would they have been more satisfied if the module was on, say, computer-games, which many younger users love? One thing was quite established from the sample, however, and that was this software is largely useful to the students in the survey, with mid-high to high scores in every item of the questionnaire.

The example module developed in Bongojontro Baksobandi provided evidence that software developed using Bongojontro can be used in various applied contexts, and not just in beginner-oriented programming for its own sake. Using the

vector-biology module, an use case can very easily be demonstrated where workers in entomological monitoring or surveys can utilize it to improve their workflow. The small code size of the module also meant that it is easily modifiable by programmers if the need arises.

With the dual mode operation, either by writing a sample code or writing/using a module, the flexibility of Bongojontro and its variant Bongojontro Baksobandi is increased. To gauge its applicability in wider scales, however, further development of modules and measuring their reception, both by ordinary non-programmer users and beginner programmers, are to be elucidated.

7 Limitations of this study and future work

Although this study combined both the applications Bongojontro and Bongojontro Baksobandi, sample programs (module) for the latter were only one in number(excluding the example-codes for the commands). This was one limitation of this study which is to be addressed by incorporating more varied modules in number at a later date. Secondly, although the initial survey hinted at great reception by the respondents, the number of participants in the survey can be increased and more nuanced qualitative feedback can be taken. Finally, the software is aimed as a beginner-oriented tool, and how far it can push the envelope and its perception by professional users remains to be seen. Continuous development in this application can give us a clearer picture of the same in the future. The inclusion of more commands since the first iteration of Bongojontro, as we saw in the vector-biology module, is probably a good first step in that direction.

Acknowledgements The authors are thankful to the University Grants Commission of India for providing funds for the research under its UGC-JRF scheme. Special thanks goes to Mr. Joyanta Bhattacharyya, teacher of Jagdabad S.B. Highschool of Panchkula, Purba Bardhamman for providing help in the research by clarifying the doubts of students about the survey process. Thanks also goes to Ms. Puspita Mallick for helping in the survey process.

Funding Bishnu Goswami is funded by the University Grants Commission of India, as a Research Fellow (UGC-JRF).

Data availability Available on reasonable request.

Code availability Bongojontro and Bongojontro Baksobandi will be available to the public at a later date, after the related survey works and research data is published.

Declarations

Conflicts of interests None.

References

Acharya, H. G. (2019). Communicative English Language Teaching and its Challenges for teaching In Indian classrooms. *International Journal of Basic and Applied Research, 9*(7), 623–626.

Bhattacharya, U. (2013). Mediating inequalities: Exploring English-medium instruction in a suburban Indian village school. *Current Issues in Language Planning, 14*(1), 164–184. https://doi.org/10.1080/14664208.2013.791236

Chiazzese, G., Fulantelli, G., Pipitone, V., & Taibi, D. (2017). Promoting computational thinking and creativeness in primary school children. Proceedings of the 5th International Conference on Technological Ecosystems for Enhancing Multiculturality, 1–7. https://doi.org/10.1145/3144826.3145354

Gao, M., Kortum, P., & Oswald, F. (2018). Psychometric Evaluation of the USE (Usefulness, Satisfaction, and Ease of use) Questionnaire for Reliability and Validity. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting, 62*(1), 1414–1418. https://doi.org/10.1177/1541931218621322

Goswami, B., Pal, S., & Pahari, D. P. (2020a). A primary survey for the novel synthesis of methods to combat vector-borne diseases and related long term management. *Asian Journal of Advances in Research* 8–14.

Goswami, B., Pal, S., & Ghosh, K. (2020b). Multi-pronged survey and prototypical information technology application against vector-borne disease in Noakhali, Bangladesh. *Uttar Pradesh Journal Of Zoology*: 7–16.

Guo, P. J. (2018). Non-native english speakers learning computer programming: Barriers, desires, and design opportunities. *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, 1–14. https://doi.org/10.1145/3173574.3173970

Ibrahim, N., & Quan, T. W. (2017). The development of multi-platforms application for dengue-entomological surveillance system. *2017 6th ICT International Student Project Conference (ICT-ISPC)*, 1–4. https://doi.org/10.1109/ICT-ISPC.2017.8075360

Lund, A. M. (2001). Measuring usability with the USE questionnaire. *Usability Interface, 8*(2), 3–6.

Marimuthu, M., & Govender, P. (2018). Perceptions of scratch programming among secondary school students in KwaZulu-Natal, South Africa. *The African Journal of Information and Communication, 21*, 51–80. https://doi.org/10.23962/10539/26112

Monjelat, N. (2017). Programming technologies for social inclusion: An experience in professional development with elementary teachers. *Twelfth Latin American Conference on Learning Technologies (LACLO), 2017*, 1–8. https://doi.org/10.1109/LACLO.2017.8120901

Nedungadi, P. P., Menon, R., Gutjahr, G., Erickson, L., & Raman, R. (2018). Towards an inclusive digital literacy framework for digital India. *Education + Training, 60*(6), 516–528. https://doi.org/10.1108/ET-03-2018-0061

Neira-Tovar,L. A. et al. (2018). Developing an application for the prevention of virus transmitted by the mosquito Zika using web and mobile technologies. Available:Puentes-consortium.org.

Ngugi, H. N., Nyathi, S., Krystosik, A., Ndenga, B., Mbakaya, J. O., Aswani, P., Musunzaji, P. S., Irungu, L. W., Bisanzio, D., Kitron, U., Desiree LaBeaud, A., & Mutuku, F. (2020). Risk factors for Aedes aegypti household pupal persistence in longitudinal entomological household surveys in urban and rural Kenya. *Parasites & Vectors, 13*(1), 499. https://doi.org/10.1186/s13071-020-04378-7

Psycharis, S., & Kallia, M. (2017). The effects of computer programming on high school students' reasoning skills and mathematical self-efficacy and problem solving. *Instructional Science, 45*(5), 583–602. https://doi.org/10.1007/s11251-017-9421-5

Sampath Kumar, B. T., & Shiva Kumara, S. U. (2018). The digital divide in India: Use and non-use of ICT by rural and urban students. *World Journal of Science, Technology and Sustainable Development, 15*(2), 156–168. https://doi.org/10.1108/WJSTSD-07-2017-0021

Sentance, S., Waite, J., & Kallia, M. (2019). Teaching computer programming with PRIMM: A sociocultural perspective. *Computer Science Education, 29*(2–3), 136–176. https://doi.org/10.1080/08993408.2019.1608781

"The World Factbook". www.cia.gov. *Central Intelligence Agency. Retrieved 21 February 2018.*

Yildiz Durak, H. (2020). The effects of using different tools in programming teaching of secondary school students on engagement, computational thinking and reflective thinking skills for problem solving. *Technology, Knowledge and Learning, 25*(1), 179–195. https://doi.org/10.1007/s10758-018-9391-y

Authors and Affiliations

Bishnu Goswami[1] · Sarmila Pal[2]

Sarmila Pal
pal.sarmila@gmail.com

[1] Department of Zoology, The University of Burdwan, Burdwan 713 104, West Bengal, India

[2] Department of Zoology, Hooghly Mohsin College, Chinsurah 712101, West Bengal, India