OFFLINE NO-CODE PLATFORM: AN INNOVATIVE SOLUTION FOR DEVELOPING SIMPLE MATHEMATICAL APPLICATIONS WITH APPLICABILITY IN ECONOMICS

ADRIAN BONCEA
Research Engineer, „Constantin Brâncuşi" University of Târgu Jiu, Romania
e-mail: adrian.boncea@e-ucb.ro

ANIELA BĂLĂCESCU,
Prof. PhD., „Constantin Brâncuşi" University of Târgu Jiu, Romania
e-mail: aniela.balacescu@e-ucb.ro

CARINA-ELENA STEGĂROIU
Lecturer PhD, „Constantin Brâncuşi" University of Târgu Jiu, Romania
e-mail: carina.stegaroiu@e-ucb.ro

Abstract
Digitization has been a subject of high interest for many years however this process isn't always equally accessible to all companies due to differences in size, target market and available financial or human resources. The process of digitizing an organization requires using either existing software solutions or developing custom programs tailored to its. Limited resources, especially for small companies, make neither option viable. In this case, the currently proposed solutions are low-code and n-code development platforms. These platforms allow employees with limited or no programming knowledge to create the necessary software provided they have an Internet connection. This paper proposes an alternative no-code solution which would allow creating native platform-specific applications, through a no-code platform designed to solve simple mathematical problems, while also eliminating the requirement for an Internet connection.
Keywords: *no-code development, mathematical programs, simple mathematics, programming, digitization*

1. INTRODUCTION

The continuously accelerating technological evolution is always transforming the life of the modern person, allowing ever-expanding access to information and means for increasing efficiency in all types of human activities. An easily predictable result of the current digital reliance is the increase in demand for digitization, especially in the business space, where maintaining and gaining new clients may be reliant on having an always broadening digital reach and improving workflow efficiency through technological means.

The business digitization process is designed to be desirable from the get-go as the effects of a properly modernized business can increase efficiency and reach exponentially thus boosting profits. The reality however is that the digitization process encounters many challenges, especially in smaller and medium-sized companies. Among the challenges faced in the process of digitization the following can be deemed as the most important:

- Employees' reluctance to change – especially faced in organizations with older employees who lack the skills and understanding necessary to properly make use of digital resources;
- The availability of software which can satisfy the specific requests of the business – many companies have specific workflows which not many programs designed for general use, even in the same field, will have predicted and in the case of niche markets no program may exist at all;

- The cost of hardware and software – the hardware costs can only be reduced by optimizing the acquired digital devices to perfectly fit the requirements of the company however it can be hard to predict future requirements and software costs can be extremely high as there aren't always many options and when there are no options hiring programmers or contracting an external business is required.

Low-code development platforms (LCDP) and no-code development platforms are solutions available to the software availability and cost issues however they aren't perfect, no-code programming limits you to exactly the use cases the platforms were designed to fit with little wiggle room while low-code allows some expandability but requires employees with some programming knowledge, while still not being able to fulfill all cases.

Developing a no-code platform which could satisfy all the needs of any business is possible but would be a huge task as it would require implementing all the functionality of any given programming language in a no-code format while maintaining generated code security. The project proposed would require creating a new Integrated Development Environment (IDE) from scratch, feat impossible to achieve in the time planned for the initial implementation.

A different approach which could help many businesses is focusing on the mathematic parts to create a platform that would allow anyone to easily create a program which could mathematically process data. The reasoning behind targeting the mathematics only is the fact that many types of businesses use some math in their workflows, from statistics operations to processing data returned by data acquisition systems and the employees know these operations the best and would easily be able to convert them into a program were no code required. The mathematical implementation would also allow avoiding the purchase of all-encompassing software suits when one specific operation is required (for example buying a statistics software suite to only use one single operation it offers).

This paper studies the process of developing a no-code platform for creating simple mathematical applications, exposing both the advantages and challenges encountered during this endeavor and the conclusions outline the viability of such a project and the complexity of creating an universal no-code platform.

2. PLANNING AND RESULTS

The digital era has brought many advancements and more people are taking advantage of the available technology to improve their daily lives however we should be aware that not everyone was born into the digital world and some are having a specifically hard time adapting to it, as such any no-code platform developed would have to guarantee an ease of use even for those employees who have never used a computer before but also efficient use for employees who are highly skilled in the daily usage of computer.

2.1. Aspects to consider in designing a no-code platform for mathematics

Many aspects should be considered when choosing how to implement a no-code platform for creating mathematical software. A simple user interface (UI) allows even the least digitally knowledgeable person to easily create the program required for their workflow, however including all the functionality of creating an application into a simple UI can be challenging. Another aspect regarding the way the platform would function is related to how the compilation of the newly developed software would work, packaging a compiler would increase the ease of use but would tie compiler updates to the software package containing both, while the compiler could receive many

independent updates in the meanwhile but requiring the user to manually install it wouldn't give as optimal of a user experience.

The programming language a no-code platform uses for the new software needs to be chosen carefully, the new programs need to be able to run on all required operating systems and also offer satisfying security features to avoid compromising the entire system of a business. C# is the language chosen for the applications produced by the no-code platform as it has a number of easily available security features and allows creating applications for the operating systems:

- Microsoft Windows;
- Google Android;
- Apple iOS;
- Apple iPadOS;
- Apple MacOS;
- Samsung Tizen.

Along the specified operating systems, C# also allows creating web applications if that would be a desired feature in the future development of the application.

Planning the order of implementing features the platform offers is essential in finalizing the project successfully and being focused on mathematical operations it should begin from integer calculations, implementing all required operations and working with variables then expand to non-integer values and operations and then add the possibility to work with files for automation purposes and only as a last step allow creating an user interface for the produced applications, console applications being good enough for the early stages. The implementation plan is:

- Implement basic integer operations (addition, multiplication, subtraction, division, exponentiation, square root);
- Implement working with variables;
- Implement for, switch-case and if-else statements;
- Implement advanced operations (logarithms, etc.);
- Implement working with non-integer values;
- Implement working with files;
- Implement a user interface designer.

Creating a user interface based on this plan is possible by keeping in mind the needed expansions it will go through during the development process.

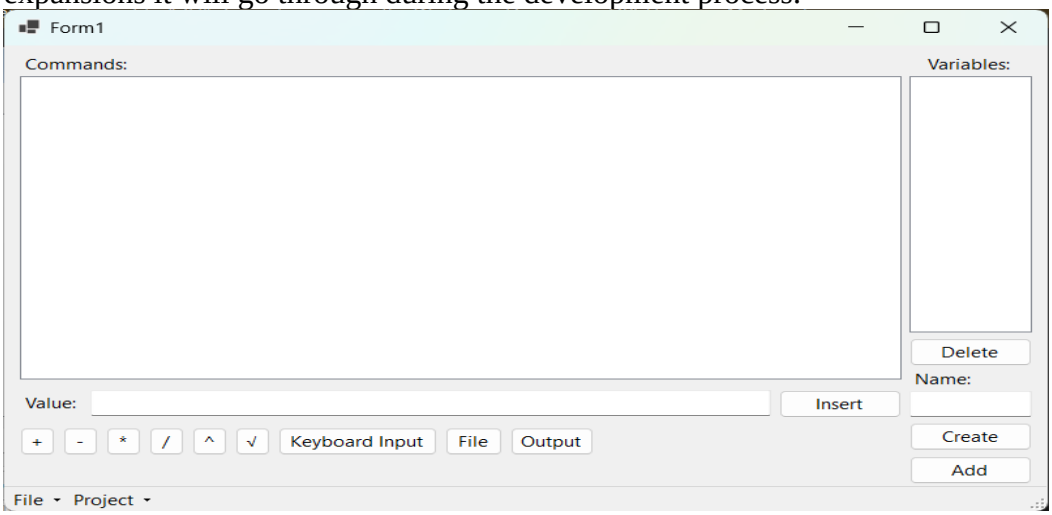

Figure no.1 Initial no-code platform UI

For this purpose, the user interface shown in Figure 1 has been designed as a basic UI for the initial version of the application, giving access to a variable creator, the addition, subtraction,

multiplication, division, exponentiation and square root operations and file operation buttons for future use.

2.2. Aspects regarding the development of a no-code platform for mathematics

In the process of developing a no-code platform many decisions need to be made regarding the project structure and implementation. One of the first decisions to make is the programming language used, as the platform would create C# code it would also make most sense to be coded in C#.

Following the choice of a programming language is the environment used throughout the development process and frameworks to be used. In order to ease the development process the best environment for developing C# code is Microsoft Visual Studio 2022 and the frameworks to be used will be the .NET 8 framework with WinForms for the user interface.

A no-code platform needs to compile code just like the IDE used to develop it and as such it would require installing the .NET SDK on the user PC to provide the dotnet compiler.

The C# language provides any required mathematical operations as part of the Maths class and expanding upon it with external libraries is an option to be considered for future development as it could provide more advanced mathematical functions or even statistics functions aa a publicly available component.

2.3. Implementation results

The process of creating the app according to the established plan had a high degree of complexity and proved to be extremely time consuming, the project being still in a very basic form of functioning after a month of work, not even able to call the dotnet compiler itself, instead just generating the code files which need to be manually put through the compiler using the command line interface (CLI).

The biggest issue is balancing safe code with an easy-to-understand user interface and complex functionality, although all chosen components are very well documented the attempt at implementing the entire mathematical part of a programming language in a simple drag-and-drop/click and enter interface while requiring communication with external tools increases the complexity of the project significantly.

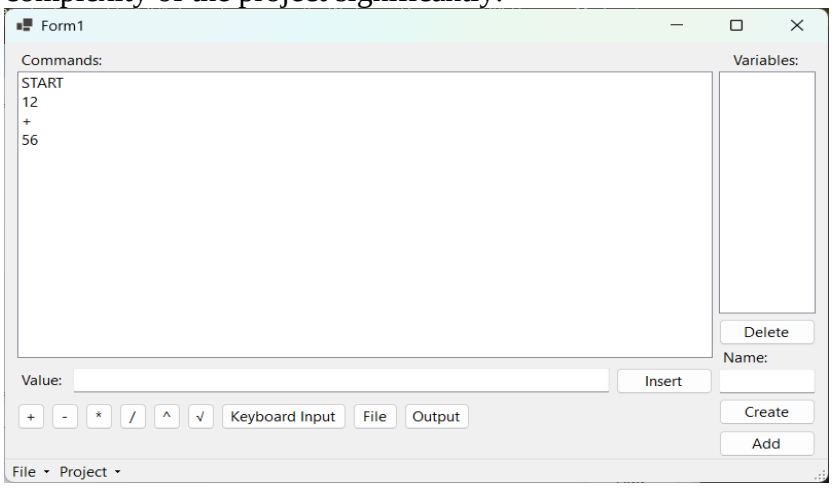

Figure no.2 Simple operation in the UI

Figure 2 shows the application running and a simple operation implemented inside the UI, calculating the sum between the numbers 12 and 56, the files submenu allows exporting the generated code as a C# compiler file (.cs format).

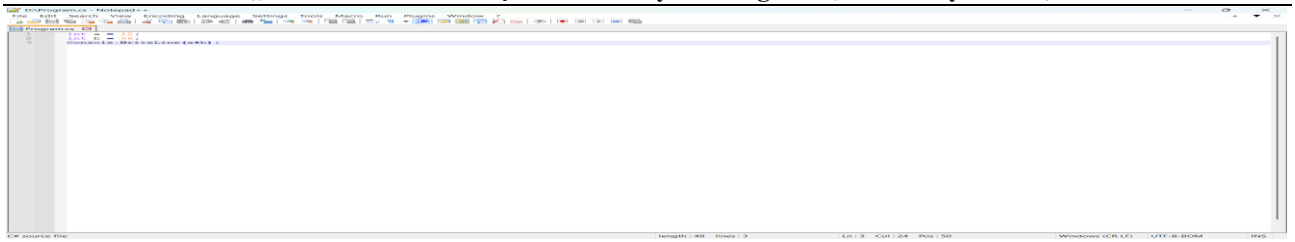

Figure no.3 Generated code

Figure 3 shows the generated code opened in the Notepad++ text editor, showing it internally used placeholder variables for the calculation in order to make debugging easier, option which shall be removed later on in the development cycle to ensure no clashing between user variables and program variables exists.

Figure no.4 Generated program running

Figure 4 shows the generated program running in the console after having been compiled using the dotnet compiler, as it can be seen it gave directly the answer to the question without writing the values of the other variables, 68 being indeed the sum of 12 and 56.

Further development of the application will begin by adding the possibility to create variables then by adding conditional expressions and introducing all the operations available in the Math class. After adding the ability to work with non-integer values modifying the core of the current UI will be necessary to allow both using other types in operations and specifying the type of each variable, a help item will be available explaining in simple terms the differences between the different numeric types (int, double, float, etc.) and many error prevention/exception catching functions will be implemented in order to avoid dealing with compilation errors due to user error.

An important feature that still requires being implemented is the ability to compile directly from within the application, the project submenu does include that option but it is currently non-functional however the plan is to allow selecting the location of the .NET SDK for the required .NET version (version 8 as of the creation of this paper) and then simply pass the compilation commands to the dotnet command line utility. To further enhance the development process project style compatibility with Visual Studio is being considered however that may be hard to maintain long term due to major changes to the project structure happening every few years. The main challenge in implementing any compilation and file operation stands in ensuring application security.

A scenario in which application security could be jeopardized by implementing the compilation functionality is if a malicious actor managed to either replace any of the executables in the .NET SDK folder or intercept the compilation command and append its own specific files to the compilation process, effectively injecting its own code into the application to do whatever it wants it to do later. File operations can be almost as dangerous because, although C# uses a VM (virtual machine) to secure the OS from the programs and has numerous features such as garbage collection and security measures, it is still possible to achieve a stack overflow or underflow, especially when using heavy mathematics, which could then allow a malicious actor to navigate the system memory and inject its own payload at any point it desires, even in the system reserved memory.

Implementing an UI designer is the final step the application will take as a simple mathematics focused no-code platform, many UI frameworks are being considered, among which MAUI (Multi-platform App UI) offers the most interesting features as it allows building a single user interface for any of the supported operating systems (Windows, Android, iOS, iPadOS, MacOS, Tizen) however it does not come without disadvantages, mainly requiring manual setup of certain components necessary to build the application for a specific OS (for example Android Studio for building the Android version of the app or having a MacOS device to compile for iOS and iPadOS).

After all the goals of this project will have been finished it can become the base for a complete no-code C# development environment with hope that in the future it will allow anyone with minimal digital knowledge to build any kind of application or game they may desire using the C# programming language.

3. CONCLUSIONS

Creating a no-code development platform for creating mathematical software could help many companies which are hitting certain roadblocks in their digitization process however this is a long-term project which will face many challenges and have to overcome numerous issues in order to reach its full potential and it could stand at the basis of a universal no-code development environment.

The development process has faced many challenges and there are issues to watch out for, especially in the field of security as using an external compiler leaves few options to confirm the compiler hasn't been hijacked but packaging it with the platform would have been even more disastrous as compilers constantly receive security updates and a month of lagging behind could open the entire software to many attacks. Another security issue to keep in mind is that working with files and using mathematics could pave the way for stack overflow or underflow attacks which could potentially load a malicious payload into reserved system memory.

If properly implemented a platform like this has the potential to bring a great plus value to any small or medium sized business, allowing specific solutions to be made which satisfy the exact workflow the employees are already used with instead of having to adapt to a new workflow. The room to expand into an universal no-code platform which would allow creating any type of software is also an exciting prospect however offering the full capabilities a programmer would have access to without any programming knowledge requires serious planning and testing to not leave newly generated software vulnerable to a whole set of attacks which could compromise the company's entire system and potentially temporarily freeze its activity (if no worse).

REFERENCES

1. El Kamouchi, H., Kissi, M., El Beggar, O. (2023, November). Low-code/No-code Development: A systematic literature review. In 2023 14th International Conference on Intelligent Systems: Theories and Applications (SITA) (pp. 1-8). IEEE.
2. Hurlburt, G. F. (2021). Low-Code, No-Code, What's Under the Hood?. IT professional, 23(6), 4-7.
3. Pătrașcu A., Toader A. F., Bălăcescu A. (2024) An Improved Multi-Objective Hybrid Algorithm for Solving Job Shop Scheduling Problem, In Economic Computation and Economic Cybernetics Studies and Research Vol 58, Issue 3/2024 (pp. 177-192)
4. Rokis, K., Kirikova, M. (2022, September). Challenges of low-code/no-code software development: A literature review. In International Conference on Business Informatics Research (pp. 3-17). Cham: Springer International Publishing.
5. https://learn.microsoft.com/en-us/ef/core/what-is-new/ef-core-8.0/whatsnew

6. https://learn.microsoft.com/en-us/ef/core/what-is-new/ef-core-8.0/plan
7. https://learn.microsoft.com/en-us/dotnet/core/whats-new/dotnet-8
8. https://learn.microsoft.com/en-us/dotnet/csharp/
9. https://learn.microsoft.com/en-us/visualstudio/windows/?view=vs-2022
10. https://learn.microsoft.com/en-us/dotnet/desktop/winforms/overview/?view=netdesktop-8.0
11. https://learn.microsoft.com/en-us/dotnet/api/system.math?view=net-8.0
12. https://learn.microsoft.com/en-us/dotnet/maui/?view=net-maui-8.0