MANIPAL UNIVERSITY
JAIPUR

Department of Information Technology

School of Information Technology

Manipal University Jaipur

Rajasthan – 303007

Project Report

On

Toy Programming Language

Under the Mentorship of

Ms. Shikha Chaudhary

(Assistant Professor – Senior Scale)

Submitted By

Kartik chitranshi

(219302185)

Nirbhay Kumar

(219302192)

Asmit Aryan

(219302180)

Course Code: IT3230

Submission Date: 22 April, 2024

"PROBLEM STATEMENT"

In the realm of modern education, there's an increasing recognition of the significance of introducing programming concepts to children at an early age. However, existing educational tools often fall short in engaging young learners and effectively imparting fundamental coding skills. To bridge this gap, we envision the creation of a user-friendly, interactive toy programming language tailored specifically for children aged 8-12 years.

This innovative language will boast a simple syntax and an intuitive graphical interface, empowering young learners to craft programs by assembling visual blocks representing various code constructs. By leveraging familiar visual elements and minimizing textual complexity, the language aims to demystify coding and make it accessible to children with diverse learning styles and backgrounds.

The primary goal of this project is to provide an engaging yet educational platform for children to explore programming concepts such as sequencing, loops, conditionals, and variables. Through hands-on experimentation and playful exploration, young users will develop a solid foundation in computational thinking and problem-solving skills. Additionally, the language will incorporate basic debugging functionalities to assist children in identifying and rectifying errors in their programs, fostering resilience and critical thinking.

Ultimately, by fostering a fun and interactive learning environment, this project seeks to inspire the next generation of programmers and equip them with the essential skills needed to thrive in an increasingly technology-driven world. Through collaboration with educators, child psychologists, and programming enthusiasts, we aim to create a toy programming language that not only educates but also entertains, paving the way for a future where coding is as accessible and enjoyable as playing with toys.

"PROJECT OBJECTIVE"

Our project aims to revolutionize the landscape of children's educational tools by developing an innovative toy programming language tailored for young learners aged 8-12 years. Our objective is to create a dynamic and engaging platform that not only introduces children to the fundamentals of programming but also nurtures their creativity and critical thinking skills.

Central to our project is the design of an intuitive and visually stimulating programming environment. Through the use of colorful and interactive blocks, children will be able to construct programs effortlessly, fostering a sense of accomplishment and confidence in their coding abilities.

We seek to empower children to become active creators rather than passive consumers of technology. By providing a diverse range of coding challenges and creative projects, our platform will encourage experimentation and inspire children to unleash their imagination in the world of coding.

Moreover, our project places a strong emphasis on community building and collaboration. Through features such as forums, multiplayer coding games, and virtual coding clubs, we aim to cultivate a vibrant online community where children can connect with like-minded peers, share ideas, and support each other's learning journey.

Ultimately, our project strives to redefine the way children learn and interact with technology, fostering a new generation of confident and creative problem solvers who are well-equipped to thrive in an increasingly digital world.

"INTRODUCTION"

In the realm of modern education, the integration of programming skills has become increasingly vital in preparing the next generation for an ever-evolving digital landscape. Our project embarks on a journey to introduce children between the ages of 8 and 12 to the captivating world of coding through the development of an innovative toy programming language. This language serves as more than just a tool for learning; it's a gateway to unlocking creativity, critical thinking, and problem-solving abilities in young minds.

At the heart of our project lies a commitment to bridging the gap between education and enjoyment. Through a carefully crafted blend of intuitive design and engaging content, our platform provides children with a seamless and immersive introduction to the fundamentals of coding. By transforming complex coding concepts into playful building blocks, we aim to inspire curiosity and empower children to explore the limitless possibilities of programming in a supportive and stimulating environment.

Our project isn't just about teaching children how to code; it's about nurturing a mindset of exploration and innovation from an early age. By encouraging hands-on experimentation and fostering a culture of collaboration, our platform aims to equip children with the skills and confidence they need to thrive in an increasingly digital world. Through guided tutorials, interactive challenges, and a vibrant online community, we aspire to ignite a passion for learning and empower children to become creators, problem solvers, and innovators of tomorrow.

TOY PROGRAMMING LANGUAGE AND ITS IMPACTS: -

The impact of a toy programming language on young learners is multifaceted and far-reaching. Firstly, it provides children with a tangible introduction to the world of coding, demystifying complex concepts and making them accessible even to those with no prior experience. By engaging with colorful blocks and interactive interfaces, children develop a solid foundation in computational thinking, logical reasoning, and problem-solving skills, which are invaluable in navigating an increasingly technology-driven society.

Moreover, a toy programming language fosters creativity and innovation by empowering children to bring

their ideas to life through code. By providing a platform for experimentation and exploration, it cultivates a mindset of curiosity and ingenuity, encouraging children to think outside the box and develop innovative solutions to real-world problems.

Beyond individual skill development, the widespread adoption of a toy programming language has broader implications for education and society as a whole. By integrating coding into the curriculum at an early age, schools can better prepare students for future careers in STEM fields and equip them with the digital literacy skills needed to thrive in the 21st century workforce. Additionally, a generation of proficient coders has the potential to drive technological innovation, fuel economic growth, and address global challenges through collaborative problem solving and innovation.

In essence, the impact of a toy programming language extends far beyond the realm of coding; it lays the groundwork for a future where creativity, critical thinking, and technological proficiency are not only valued but also cultivated from a young age, empowering children to become active participants in shaping the world around them.

TYPES OF TOY PROGRAMMING : -

Toy programming languages come in various forms, each tailored to cater to the unique needs and preferences of young learners. One type of toy programming language utilizes visual programming interfaces, where children can drag and drop colorful blocks representing code constructs to create programs. These visual languages, such as Scratch and Blockly, offer a hands-on approach to learning programming concepts and are particularly well-suited for younger children or those who are new to coding.

Another type of toy programming language focuses on physical computing and tangible interaction. These languages, like Arduino and Raspberry Pi, enable children to connect hardware components such as sensors, LEDs, and motors to their code, allowing them to create interactive projects and tangible prototypes. By bridging the gap between the digital and physical worlds, these languages offer a unique opportunity for children to explore the principles of electronics and engineering in addition to coding.

Additionally, some toy programming languages are designed to simulate real-world environments or games, providing children with a context-rich learning experience. For example, platforms like Code.org's Minecraft and Tynker's game-based coding courses allow children to program characters and elements within popular games, fostering engagement and motivation while learning to code.

Ultimately, the diversity of toy programming languages reflects the varied interests, learning styles, and developmental stages of young learners, providing them with a range of options to explore and experiment with coding in a way that best suits their individual preferences and abilities.

CHARACTERISTICS OF TOY PROGRAMMING LANGUAGE :-

Toy programming languages share common characteristics that distinguish them from production-grade languages:

Simplicity: Toy languages are intentionally simple, featuring minimal syntax and a narrow scope of features to facilitate comprehension.

Focused Purpose: These languages often focus on specific educational goals, such as teaching basic programming concepts, demonstrating language features, or simulating domain-specific scenarios

Limited Expressiveness: While sufficient for educational purposes, toy languages may lack the expressiveness and versatility of mainstream languages used in professional software development.

Ease of Implementation: Toy languages are designed to be straightforward to implement, allowing learners to gain hands-on experience in language construction.

"TIMELINE OF PROJECT"

Week 1-2

- Project planning and research - Define the objectives and scope of the toy language. Identify key features and syntax inspired by JavaScript

Week 3-4

- Lexer Implementation - Develop a lexer to tokenize input code into meaningful tokens. Implement token definitions for identifiers, keywords, operators, and literals

Week 5

- Syntax analysis - Define grammar rules for the toy language using parsing techniques. Implement parsing logic to construct an AST from tokens. Test the parser with

Week 6-7

- Execution - Implement interpreter functions for statements, expressions, and control flow. Handle variable declarations, assignments, and basic operations. Test the interpreter with simple programs to verify execution.

- Language Enhancements and Refinements - Add support for functions, conditionals, loops, and error handling. Refactor and optimize codebase for clarity and performance. Conduct comprehensive testing to ensure robustness.

Week 9

- Debugging - Conduct final testing and bug fixing. Polish documentation, project report, and presentation materials. Rehearse presentation and demo to ensure clarity

"TECHNOLOGY & METHODS USED"

In developing this project, a combination of modern technologies and methodologies was employed to ensure efficiency, functionality, and user-friendliness. The following technologies were utilized:

HTML (Hypertext Markup Language): The foundation of the project, HTML was used to structure the content and layout of the web pages, providing a semantic and organized structure for the toy programming language platform.

CSS (Cascading Style Sheets): CSS was leveraged to enhance the visual appearance and design of the project, including aspects such as colors, fonts, layout, and responsiveness, ensuring a visually appealing and consistent user experience across different devices and screen sizes.

JavaScript: As the backbone of interactivity, JavaScript played a crucial role in implementing various features and functionalities of the toy programming language platform. It facilitated dynamic content generation, user input validation, and interactive elements, enhancing the overall user engagement and experience.

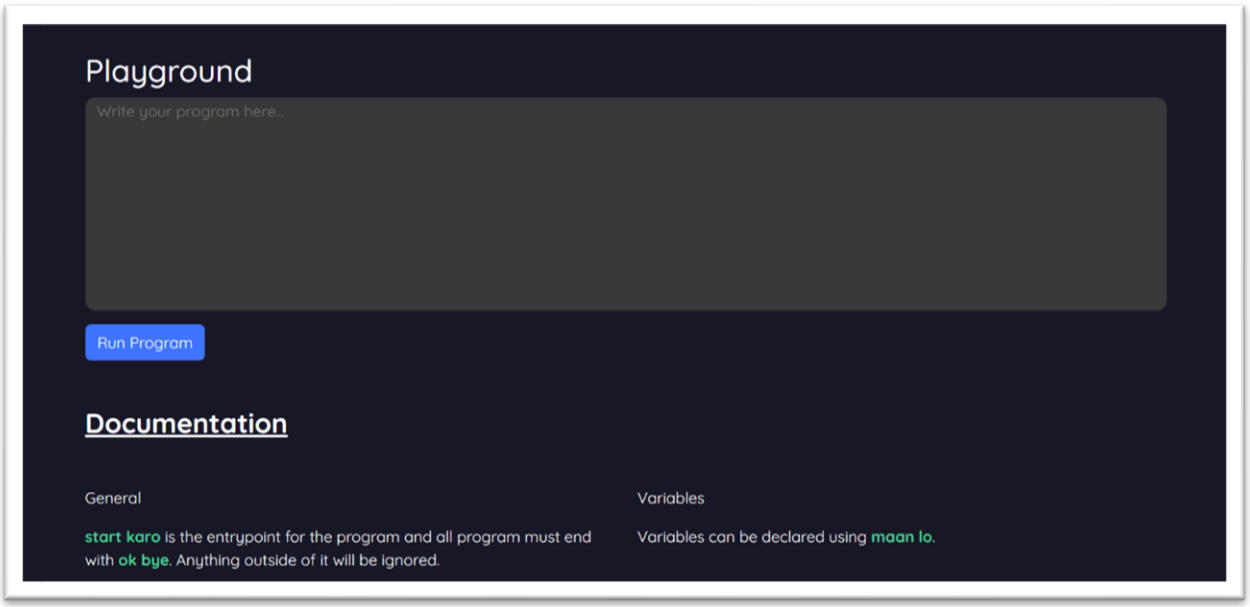
React: React, a JavaScript library for building user interfaces, was utilized to create interactive components and manage the state of the application efficiently. Its component-based architecture allowed for modular and reusable code, simplifying the development process and improving maintainability.

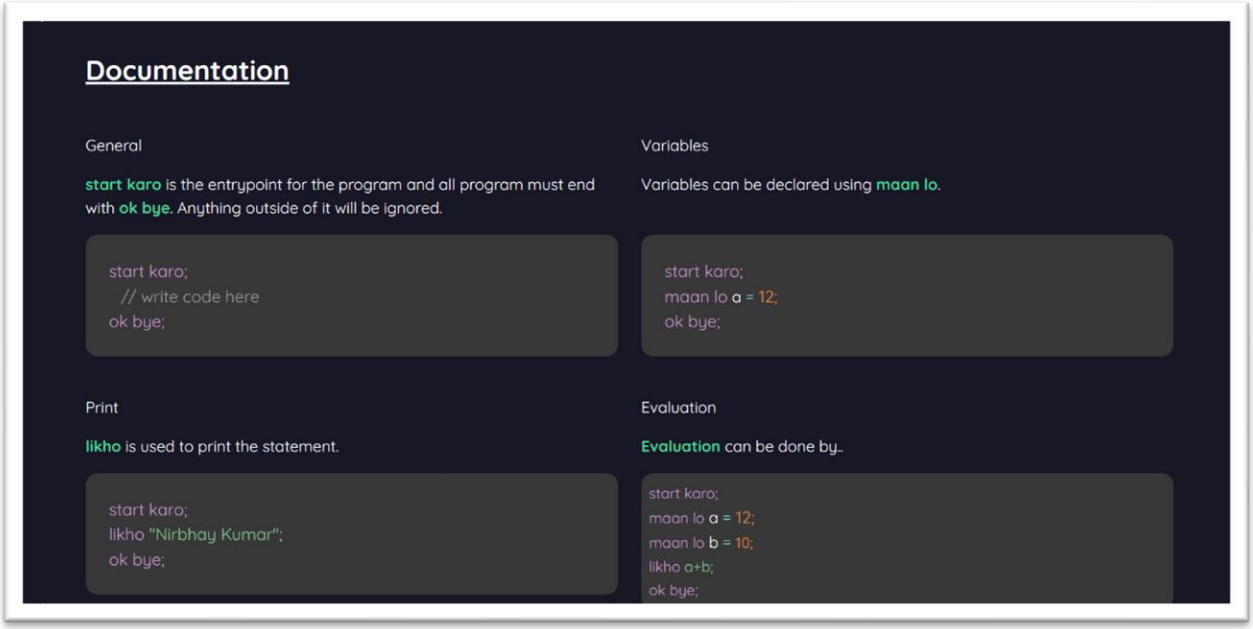
The development process followed agile methodologies, incorporating iterative development cycles and frequent testing to ensure continuous improvement and address any issues or enhancements promptly. Version control systems such as Git were employed for collaborative development and code management, enabling seamless collaboration among team members and facilitating version tracking and rollback when necessary.

Overall, the combination of HTML, CSS, JavaScript, and React, along with agile methodologies, facilitated the creation of a robust, visually appealing, and interactive toy programming language platform that aims to inspire and educate young learners in the fascinating world of coding.

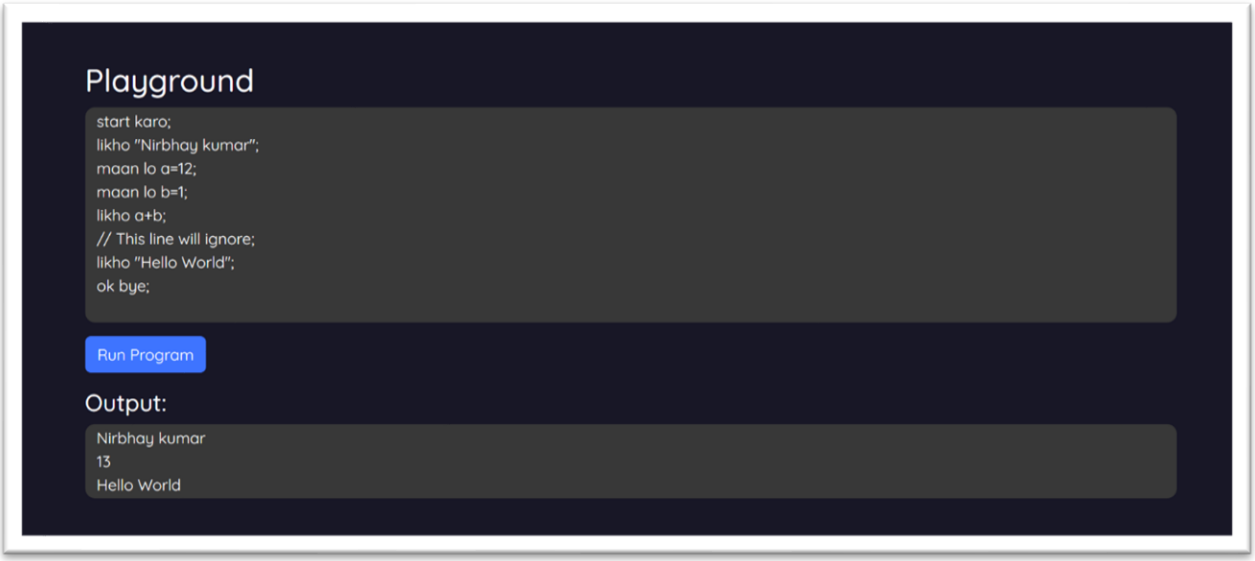
"RESULT"

LANDING PAGE:-




Documentation
General
start karo is the entrypoint for the program and all program must end with ok bye. Anything outside of it will be ignored.
start karo;
// write code here
ok bye;
Variables
Variables can be declared using maan lo.
start karo;
maan lo a = 12;
ok bye;
Print
likho is used to print the statement.
start karo;
likho "Nirbhay Kumar";
ok bye;
Evaluation
Evaluation can be done by..
start karo;
maan lo a = 12;
maan lo b = 10;
likho a+b;
ok bye;

PLAYGROUND:-


Playground
start karo;
likho "Nirbhay kumar";
maan lo a=12;
maan lo b=1;
likho a+b;
// This line will ignore;
likho "Hello World";
ok bye;
Run Program
Output:
Nirbhay kumar
13
Hello World

"CONCLUSION"

In conclusion, the development of this toy programming language project represents not only a technological achievement but also a significant milestone in the realm of childhood education and digital literacy. Through the strategic integration of HTML, CSS, JavaScript, and React, we have succeeded in creating a platform that transcends mere coding instruction to become a gateway to creativity, critical thinking, and problem-solving skills for young learners.

By providing a playful yet educational environment for children to explore the fundamentals of programming, we aim to inspire a lifelong passion for learning and innovation. The platform's intuitive interface, interactive features, and engaging content empower children to unleash their creativity, experiment with coding concepts, and develop the confidence to tackle challenges in the digital world with ease.

Furthermore, the project's emphasis on community building and collaboration fosters a sense of camaraderie and support among learners, creating a vibrant ecosystem where ideas are shared, friendships are formed, and knowledge is collectively cultivated.

As we reflect on the journey of developing this project, we are filled with pride and excitement for the impact it will have on shaping the future generation of coders, innovators, and problem solvers. With continued dedication and innovation, we look forward to seeing the transformative effects of this project unfold in the lives of young learners worldwide.

"REFERENCES"

1. "Encyclopedia: Definition of Compiler". PCMag.com. Retrieved 2 July 2022.
2. Compilers: Principles, Techniques, and Tools by Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman - Second Edition, 2007
3. Sudarsanam, Ashok; Malik, Sharad; Fujita, Masahiro (2002). "A Retargetable Compilation Methodology for Embedded Digital Signal Processors Using a Machine-Dependent Code Optimization Library". Readings in Hardware/Software Co-Design. Elsevier. pp. 506–515. doi:10.1016/b978-155860702-6/50045-4. ISBN 9781558607026. A compiler is a computer program that translates a program written in a high-level language (HLL), such as C, into an equivalent assembly language program [2].
4. Sun, Chengnian; Le, Vu; Zhang, Qirun; Su, Zhendong (2016). "Toward understanding compiler bugs in GCC and LLVM". Proceedings of the 25th International Symposium on Software Testing and Analysis. Issta 2016. pp. 294–305. doi:10.1145/2931037.2931074. ISBN 9781450343909. S2CID 8339241. {{cite book}}: |journal= ignored (help)
5. Lecture notes. Compilers: Principles, Techniques, and Tools. Jing-Shin Chang. Department of Computer Science & Information Engineering. National Chi-Nan University
6. Naur, P. et al. "Report on ALGOL 60". Communications of the ACM 3 (May 1960), 299–314.
7. Chomsky, Noam; Lightfoot, David W. (2002). Syntactic Structures. Walter de Gruyter. ISBN 978-3-11-017279-9.
8. Gries, David (2012). "Appendix 1: Backus-Naur Form". The Science of Programming. Springer Science & Business Media. p. 304. ISBN 978-1461259831.
9. Hellige, Hans Dieter, ed. (2004) [November 2002]. Written at Bremen, Germany. Geschichten der Informatik - Visionen, Paradigmen, Leitmotive (in German) (1 ed.). Berlin / Heidelberg, Germany: Springer-Verlag. pp. 45, 104, 105. doi:10.1007/978-3-642-18631-8. ISBN 978-3-540-00217-8. ISBN 3-540-00217-0. (xii+514 pages)
10. Iverson, Kenneth E. (1962). A Programming Language. John Wiley & Sons. ISBN 978-0-471430-14-8.