Bangla Programming Language: A Native Language-Based Approach to Introducing Programming to Early Learners

Md. Anisur Rahman
Head of IT, Programming24 School
Dhaka, Bangladesh
Email: mrahman202047@mscse.uiu.ac.bd, anisur@dcaclab.com
Website: https://programming24.school.blog

Abstract*— Programming is often introduced in English, creating a significant barrier for learners whose primary language is not English. This is especially true for children and early learners, who may already struggle with abstract concepts like logic, loops, and functions. This paper introduces a novel educational initiative—Bangla Programming Language [1], an advanced web-based programming environment that enables learners to write, interpret, and learn code entirely in Bangla. Built using a custom JavaScript-based parser and runtime engine [3], our ecosystem targets young learners, particularly school children, to introduce foundational computational thinking in their mother tongue. In this variant, we detail the complete functional scope of the platform, including its core Bangla syntax module, an interactive canvas drawing engine, voice-activated speaking code modules, and structured logic modules exemplified by a full Snake Game implementation. We outline our design methodology, native runtime literature review, comparative analysis matrix, and future pathways toward full compiler integration.

***Keywords*—** *Bangla Programming Language, Native Syntax, JavaScript Parser, Early Computing Education, Canvas Drawing, Voice-Activated Coding, Game Logic.*

I. INTRODUCTION

As the global digital economy expands exponentially, learning computational logic and programming has evolved from an optional specialized skill to an essential foundational literacy [4]. However, the dominant vocabulary of computing—exclusively based on English keywords—poses an immediate, non-trivial cognitive barrier for early learners from non-English-speaking backgrounds [6]. Children face a dual cognitive overload: they must simultaneously memorize foreign syntax rules and internalize highly abstract logical structures like conditional forks, programmatic loop parameters, and function calls.

To address this linguistic hurdle, our research and development initiative targets the creation of a fully localized, web-accessible development environment that brings computer programming directly into the native linguistic frame of Bangla speakers [1]. The ecosystem translates technical commands into naturally structured Bangla keywords, allows instant direct execution inside the web browser via an optimized runtime client [3], and provides graphical output and spoken feedback systems geared toward engaging young minds.

Unlike theoretical models or static translation scripts, our architecture introduces five primary operational paradigms fully integrated into a uniform user terminal: a standard Bangla Code Runner, a Canvas-based Drawing Environment, an interactive Voice/Speaking module, an execution sandbox supporting complete interactive systems like a classic Snake Game, and a dedicated Bangla-to-C syntax translation bridge for transitioning students toward formal industry languages.

II. LITERATURE REVIEW

The pedagogy of training novice developers in localized configurations has seen diverse theoretical and tool-based iterations in South Asia [5], [7]. Our system addresses structural gaps present in historical and contemporary tools through extensive engineering enhancements.

A. Potaka Editor

The Potaka Editor is a pioneering milestone that champions Bangla programming specifically tailored for children via a hybridized approach utilizing block-based and basic text inputs [2]. Inspired by the visual blocks of MIT's Scratch and Google's Blockly, Potaka successfully decreases syntax errors. However, Potaka is structurally restricted as a visual learning aid rather than an execution engine. It lacks high-fidelity, real-time native programmatic compilation, and expressions involving advanced recursive architectures or sophisticated local variable workflows remain constrained.

B. ChaScript

ChaScript represents an academic scripting language engineered to mirror the conversational patterns of the Bangla language. While it successfully demonstrates a readable syntax flow, it suffers from a critical infrastructure issue: it lacks active browser runtime modules and production-grade execution parsers. Because it remains primarily localized to educational

research documents, it has missed widespread scaling in mainstream schools or public learning repositories.

C. Pankti

Pankti approaches coding from a highly creative and aesthetic angle, treating software script as an expressive or poetic medium. While it presents a uniquely beautiful paradigm for blending human art and computational execution, it intentionally minimizes focus on critical computer science metrics such as multi-dimensional data handling, explicit variable scope isolation, and optimized loops—making it less suitable as a rigid foundation for mainstream engineering instruction.

D. Specialized Tooling

Recent developments have brought tools like Bongojontro and advanced localized generative models like TigerCoder into the ecosystem [9]. While large language models help generate syntax blocks, early learners still require deterministic, low-latency, sandboxed environments that run flawlessly on low-spec hardware without demanding continuous internet connectivity or cloud API usage [8].

III. RESEARCH METHODOLOGY

Our methodology is anchored around answering three critical research questions: (1) Can programming logic be completely contextualized into Bangla without dropping structural or performance standards? (2) Does native semantic processing measurably reduce the initial barrier to entry for young beginners? (3) How can multiple execution frameworks (text, audio, graphics) coexist within a unified browser terminal?

To establish baseline evaluations, the following boundaries were strictly monitored:

- **Inclusion Criteria:** Systems natively implementing non-English parsing; browser-accessible portals requiring zero desktop initialization or external dependencies; targets children or non-technical beginners.
- **Exclusion Criteria:** Frameworks restricted purely to static poetic definitions; systems requiring complex command-line or system-level binary installation; non-functional code repositories lacking automated testing loops.

TABLE I: COMPREHENSIVE FEATURE MATRIX & COMPARISON MAPPING

Feature Matrix	Potaka [2]	ChaScript	Pankti	Our System [1]
Bangla Syntax	Blocks	Text	Poetic	**Deterministic Text**
Execution Platform	Visual	Academic	Aesthetic	**Real-Time JS Engine**
Web Interface	Supported	Unavailable	Limited	**Full Custom IDE**
Graphic Canvas	Basic Shapes	No Support	Creative	**Free-Hand & Vector**
Audio Feedback	No Support	No Support	No Support	**Web Speech Integrated**
Bangla to C Bridge	No	No	No	**Supported (Partial)**
Target Base	Children	Academic	Creative	**Children & Beginners**

IV. SYSTEM IMPLEMENTATION & CORE MODULES

The platform utilizes a modern frontend engine built with clean HTML5, CSS3 configured via Bootstrap components, and custom Google Typography (Hind Siliguri). The foundational parsing logic is processed exclusively on the client side using JavaScript, avoiding performance issues from server roundtrips.

A. Core Bangla Logic Code Module

The parser utilizes dynamic dictionary mapping regular expressions to map native tokens to JavaScript instructions. Keywords like ধরি map directly to `let/var`, যদি maps to `if`, and দেখাও points to standard console outputs or DOM display fields.

Listing 1: Core Logical Syntax Implementation

```
ধরি নাম = ইনপুট ("তোমার নাম কি?");
যদি (নাম == "আনিছুর") {
  দেখাও ("হ্যালো আনিছুর!");
} নাহলে যদি (নাম == "লিখন") {
  দেখাও ("হ্যালো লিখন!");
} নাহলে {
  দেখাও ("ঘোড়ার ডিম! তোমার নাম কি?");
}
```

B. Free-Hand & Programmatic Canvas Drawing Module

To stimulate visual and geometric learning, our ecosystem provides an advanced interactive canvas. This module operates via two parallel streams: direct programmatic shape rendering using localized canvas macros, and an integrated real-time free-hand brush layer that allows early learners to doodle directly over their rendered shapes, blending manual artistry with programmatic execution.

Listing 2: Programmatic Canvas Render Operations

```
আঁকো(আঁকার_বোর্ড)
```

C. Voice-Activated & Speaking Code Module

Leveraging the native HTML5 Web Speech Synthesis API, the platform provides interactive auditory loops. This enables children with reading or visual challenges to interact directly with computing systems, translating

string outputs into distinct, synthesized spoken Bangla vocal streams.

The execution engine interprets string expressions dynamically and targets accessible audio nodes. For instance, declaring a text variable and channeling it through the native vocalization routine prompts the browser engine to stream clear, audible Bangla voice synthesis to the user without external dependencies.

D. Interactive System Logic: Full Snake Game Implementation

To demonstrate the platform's ability to handle game state loops and structured logical patterns, Listing 3 provides a clean representation of the interactive state entry module, removing the prolonged algorithmic components to emphasize terminal activation routing.

Listing 3: Snake Game Coordination Script

```
আঁকো("সাপ_খেলা");
// Arrow বাটন ব্যবহার করে সাপ খেলা খেলো
```

V. LIMITATIONS & FUTURE ROADMAP

While the JavaScript parser works well for client-side applications, it faces performance trade-offs during large-scale execution loops due to its text-substitution architecture. Furthermore, the Bangla-to-C module operates purely as a one-way learning visualizer and is not yet connected to a live, system-level binary compiler.

Our engineering roadmap includes plans to integrate WebAssembly-based backend runtimes directly into the web layout to enable high-speed processing of localized scripts. We also plan to release a dedicated mobile learning application and align our educational modules directly with public school curricula across Bangladesh and West Bengal.

VI. CONCLUSION

The Bangla Programming Language environment bridges the gap between language and logic. By removing English barriers, it enables early learners to build software systems, control graphics canvas architectures, and execute audio scripts in their mother tongue. This democratizes technical training, providing an inclusive platform for the next generation of engineers across South Asia.

REFERENCES

[1] M. A. Rahman, "Bangla Programming Language: A Native Language-Based Approach to Introducing Programming to Early Learner (V4.5)," Programming24 School Research Paper, 2026.

[2] Potaka Editor, "A Bangla programming environment for children," available at: https://potaka-editor.vercel.app.

[3] M. A. Rahman, "Bangla Programming Language Code Library," GitHub Repository, Available at: https://github.com/AnisurRahmanJU/BanglaProLangs.

[4] J. Smith, A. Kumar, and L. Chen, "Localization in Computing Education," *ACM Education Review*, vol. 33, no. 4, pp. 112–129, 2021.

[5] S. Roy and B. Chowdhury, "Teaching Programming in Native Languages: A Case Study on Bengali," *Proceedings of ICALT, IEEE*, pp. 184–188, 2019.

[6] M. Hasan and S. Rahman, "Barriers in Learning Programming in Non-Native Languages," *Journal of Educational Technology*, vol. 25, no. 3, pp. 45–52, 2020.

[7] T. Kundu and R. Alam, "Using Mother Tongue in Programming Education: A Study from West Bengal," *International Journal of Computer Science Education*, vol. 15, no. 2, pp. 75–89, 2017.

[8] A. Karim and L. Nahar, "Comparative Analysis of Programming Tools for School Children in South Asia," *South Asian EdTech Review*, vol. 9, no. 1, pp. 23–34, 2022.

[9] N. Raihan, A. Anastasopoulos, and Q. Zampieri, "TigerCoder: A Novel Suite of LLMs for Code Generation in Bangla," *arXiv preprint*, 2025.