Jurnal Jawara Sistem Informasi Vol 2.1 (2024)
e-ISSN : 3025-874X

Perancangan Bahasa Pemrograman Edukatif Sederhana "Menara Code" untuk Pembelajaran Pemrograman Dasar

Muhammad Romadhona Kusuma
[1)]MENARA Community (Komunitas Digital, Membangun Ekosistem Nasional, Aplikasi & Riset Anak Bangsa)
E-Mail : m.romadhona.kusuma@gmail.com

ABSTRAK

Dalam era digital yang terus berkembang, kemampuan memahami pemrograman dasar menjadi keterampilan penting bagi berbagai kalangan, terutama pemula. Artikel ini mengusulkan perancangan Menara Code, sebuah bahasa pemrograman edukatif sederhana yang dirancang untuk memperkenalkan konsep-konsep dasar pemrograman secara lebih mudah dan interaktif. Menara Code menggunakan sintaks yang intuitif dan dapat dijalankan langsung melalui antarmuka berbasis web, sehingga meminimalkan hambatan teknis dalam proses pembelajaran. Bahasa ini dirancang dengan pendekatan bertahap yang menggabungkan teori automata, tata bahasa bebas konteks (context-free grammar/CFG), evaluasi ekspresi, serta teknologi pembelajaran mesin untuk memberikan umpan balik otomatis terhadap kesalahan logika dan penjelasan konsep. Selain itu, pengembangan Menara Code juga mengacu pada teori scaffolding untuk menyesuaikan beban kognitif secara progresif, serta memperkenalkan Teori Menara, sebuah pendekatan konseptual yang dikembangkan oleh penulis untuk menggambarkan pembelajaran sebagai proses bertingkat, dimulai dari pondasi konsep dasar hingga pemahaman lanjutan.Dengan kombinasi pendekatan tersebut, Menara Code diharapkan dapat menjadi media pembelajaran yang efektif, adaptif, dan progresif bagi pemula dalam dunia pemrograman.

Kata Kunci – Bahasa Pemrograman Edukatif, Pendidikan Komputer, Sintaks Sederhana, Teori Menara, Scaffolding, Umpan Balik Berbasis AI

ABSTRACT

In the rapidly evolving digital era, the ability to understand basic programming has become an essential skill for various groups, especially beginners. This article proposes the design of Menara Code, a simple educational programming language developed to introduce fundamental programming concepts in a more accessible and interactive way. Menara Code employs intuitive syntax and can be executed directly through a web-based interface, minimizing technical barriers in the learning process. The language is designed with a step-by-step approach, integrating automata theory, context-free grammar (CFG), expression evaluation, and machine learning technology to provide automated feedback on logical errors and concept explanations. Additionally, the development of Menara Code is guided by the scaffolding theory to progressively adjust cognitive load, and introduces the Tower Theory—a conceptual framework developed by the author to represent learning as a layered process, starting from foundational concepts toward advanced understanding. Through this combined approach, Menara Code is expected to serve as an effective, adaptive, and progressive learning medium for programming beginners.

Keywords – Educational Programming Language, Computer Education, Simple Syntax, Tower Theory, Scaffolding, AI-Based Feedback

1. PENDAHULUAN

Pemrograman komputer telah menjadi keterampilan dasar yang sangat diperlukan dalam berbagai bidang, terutama dengan pesatnya kemajuan teknologi informasi. Namun, banyak pemula merasa kesulitan untuk memulai belajar pemrograman karena kompleksitas sintaks dan konsep dalam bahasa pemrograman yang ada. Bahasa seperti Python telah berhasil mengurangi hambatan ini melalui desain sintaks yang sederhana dan mudah dipahami.

Menanggapi kebutuhan tersebut, artikel ini memperkenalkan Menara Code, sebuah bahasa pemrograman edukatif yang dirancang untuk menyediakan lingkungan belajar yang ramah bagi pemula. Bahasa ini mengusung sintaks yang intuitif, struktur kode yang jelas, serta memungkinkan eksekusi kode secara langsung melalui antarmuka berbasis web. Artikel ini membahas proses perancangan bahasa tersebut, teori-teori pendukungnya, alur implementasi teknis, serta fitur pembelajaran mesin yang ditanamkan untuk memberikan umpan balik cerdas dan mendukung proses belajar yang efektif.

Nama Menara Code dipilih untuk mencerminkan filosofi pembelajaran yang sederhana, terstruktur, dan progresif. Istilah “Menara” digunakan sebagai simbol pendekatan bertahap dalam pembelajaran, di mana proses belajar dipandang sebagai pembangunan struktur

bertingkat—dimulai dari pondasi pemahaman dasar hingga mencapai konsep-konsep lanjutan. Pemilihan nama ini sejalan dengan pendekatan scaffolding dan konseptualisasi Teori Menara, yakni gagasan bahwa setiap lapisan pengetahuan harus dibangun secara berurutan dan saling menopang.

Menara Code ditujukan sebagai media pembelajaran pemrograman tingkat awal (entry-level language) bagi pemula, pelajar, dan komunitas non-IT. Bahasa ini tidak dimaksudkan untuk bersaing dengan bahasa pemrograman lain, melainkan sebagai jembatan awal untuk memahami konsep dasar secara bertahap melalui pendekatan yang sederhana, interaktif, dan adaptif.

Penelitian ini bertujuan untuk merancang dan mengimplementasikan bahasa pemrograman edukatif sederhana Menara Code yang dapat membantu pemula memahami konsep dasar pemrograman secara bertahap, interaktif, dan mandiri.

2. TINJAUAN PUSAKA

Dalam merancang sebuah bahasa pemrograman baru, terutama yang ditujukan untuk pemula, sangat penting untuk memahami fondasi teoritis yang melandasi sistem bahasa pemrograman modern. Bagian ini membahas berbagai konsep utama yang menjadi dasar perancangan Menara Code, mulai dari elemen dasar bahasa pemrograman hingga pendekatan pembelajaran mesin dalam pendidikan

A. Konsep Dasar Bahasa Pemrograman

Bahasa pemrograman adalah alat yang digunakan untuk menyampaikan instruksi kepada komputer. Desain bahasa pemrograman terdiri dari tiga elemen utama:

- Sintaks: Aturan yang menentukan struktur instruksi dalam kode.
- Semantik: Makna dari instruksi yang diberikan.
- Pragmatis: Bagaimana bahasa tersebut digunakan dalam konteks tertentu.

Bahasa seperti Python mengutamakan keterbacaan, menjadikannya sangat populer di kalangan pemula. Filosofi "readability counts" dari Python menjadi inspirasi utama dalam desain Menara Code untuk menciptakan sintaks yang jelas dan mudah dimengerti.

B. Teori Automata

Dalam perancangan bahasa pemrograman, teori automata sangat penting untuk menganalisis dan memproses input kode. Dua elemen utama dalam teori ini adalah:

- Lexical Analysis (DFA): Proses pemecahan kode menjadi token-token yang lebih kecil seperti variabel, operator, dll.
- Parsing (CFL): Mengidentifikasi struktur kode berdasarkan aturan tata bahasa tertentu.

Menara Code mengimplementasikan kedua proses ini untuk menganalisis dan memproses input pengguna, memecahnya menjadi token dan memeriksa apakah sintaksnya benar.

C. Hierarchy of Programming Languages (Chomsky)

Bahasa pemrograman diklasifikasikan dalam hierarki Chomsky berdasarkan kompleksitas aturan tata bahasa. Menara Code dirancang sebagai bahasa pemrograman tingkat tinggi dengan menggunakan aturan context-free grammar (CFG) yang sederhana. Ini memungkinkan bahasa ini digunakan oleh pemula tanpa memerlukan pemahaman yang dalam tentang teori bahasa formal.

D. Evaluasi Ekspresi

Bahasa pemrograman seperti Menara Code dirancang untuk mengevaluasi ekspresi matematika dengan cara yang sederhana. Ketika variabel diberikan nilai, ekspresi seperti z = x + y dihitung secara otomatis setelah menggantikan variabel dengan nilai yang ada di memori.

E. Teori Interaktif dan Komputasi Real-Time

Untuk memberikan umpan balik yang cepat kepada pengguna, Menara Code mengimplementasikan model komputasi real-time menggunakan event-driven programming. Ketika pengguna menekan tombol "Run Code", kode dieksekusi langsung, dan hasil atau pesan kesalahan ditampilkan segera di layar.

F. Teori Desain Berbasis Pengguna (HCI)

Desain Menara Code mengutamakan kemudahan penggunaan, terutama untuk pemula yang baru belajar pemrograman. Dengan meminimalkan sintaks yang kompleks, pengguna dapat dengan mudah menulis dan menjalankan kode tanpa harus menghadapi kesulitan dalam memahami konsep-konsep pemrograman tingkat lanjut.

G. Pembelajaran Mesin dalam Pembelajaran Pemrograman

Pengembangan bahasa pemrograman ini juga mengintegrasikan konsep Pembelajaran Mesin (Machine Learning) untuk memberikan umpan balik yang lebih cerdas. Feedback langsung untuk kesalahan logika dan penjelasan tentang konsep menjadi bagian penting dalam membantu pengguna untuk belajar dengan lebih cepat dan efektif. Pembelajaran mesin dapat membantu mendeteksi kesalahan logika dalam kode yang ditulis oleh pengguna dan memberikan umpan balik langsung berdasarkan analisis pola kode yang telah dipelajari sebelumnya.

H. Teori Scaffolding dalam Pembelajaran Pemrograman

Teori scaffolding diperkenalkan oleh Jerome Bruner sebagai pendekatan pembelajaran bertahap yang membantu siswa mengembangkan pemahaman dengan dukungan yang dikurangi secara perlahan seiring waktu. Dalam konteks Menara Code, scaffolding diimplementasikan melalui penyusunan fitur secara bertahap, dimulai dari sintaks dasar seperti penugasan dan ekspresi aritmatika, sebelum masuk ke struktur kontrol atau fungsi. Pendekatan ini

memungkinkan penyesuaian beban kognitif dan mencegah pengguna pemula merasa kewalahan di tahap awal pembelajaran.

I. Pendekatan Membangun Menara: Konstruktivisme Bertahap

Pendekatan ini menganalogikan proses belajar sebagai pembangunan menara: dimulai dari pondasi yang kuat dan dilanjutkan secara bertingkat. Pandangan ini sejalan dengan teori konstruktivisme Piaget dan Vygotsky, yang menekankan bahwa pembelajaran efektif terjadi saat konsep baru dibangun di atas pengetahuan sebelumnya. Dalam Menara Code, konsep ini diterapkan melalui desain kurikulum dan fitur antarmuka yang memungkinkan pengguna membangun pemahaman pemrograman secara spiral dan bertahap, dari dasar hingga konsep yang lebih kompleks.

J. Teori Menara: Pendekatan Bertahap dalam Pembelajaran Pemrograman

Teori Menara merupakan pendekatan konseptual yang dikembangkan oleh penulis sebagai kerangka untuk memahami proses belajar pemrograman secara bertahap dan terstruktur. Dalam teori ini, proses belajar diibaratkan seperti membangun sebuah menara: dimulai dari pondasi yang kuat berupa pemahaman dasar (seperti variabel dan ekspresi), lalu secara bertahap ditambahkan lapisan demi lapisan berupa struktur kontrol, fungsi, hingga konsep lanjutan.

Setiap lapisan tidak berdiri sendiri, melainkan menopang dan memperkuat keseluruhan struktur kognitif peserta didik. Pendekatan ini mendukung gagasan bahwa pembelajaran efektif terjadi melalui pemetaan konsep secara vertikal dan progresif. Teori Menara mendampingi prinsip scaffolding dengan memberi analogi visual dan kognitif yang kuat, serta relevan dalam desain sistem pembelajaran seperti Menara Code. Teori ini juga selaras dengan filosofi spiral learning dalam konstruktivisme modern.

Berbeda dengan pendekatan sebelumnya, Menara Code tidak hanya mengadopsi sintaks yang disederhanakan, tetapi juga mengintegrasikan umpan balik berbasis AI dan konsep Teori Menara sebagai pendekatan pembelajaran bertahap berbasis konstruktivism.

3. METODE PENELITIAN

Penelitian ini menggunakan pendekatan research and development (R&D) dengan model pengembangan waterfall. Tahapan meliputi analisis kebutuhan pengguna pemula, perancangan grammar dan interpreter, implementasi prototipe berbasis web, serta pengujian fungsional dan usability. Evaluasi dilakukan terhadap beberapa sampel pengguna pemula untuk menilai kejelasan sintaks, kecepatan eksekusi, dan keefektifan feedback logika berbasis pembelajaran mesin.

Evaluasi dilakukan secara deskriptif melalui observasi langsung dan wawancara terhadap 16 pengguna pemula, yang terdiri dari siswa SMA, mahasiswa, santri pesantren, dan peserta umum non-TI. Fokus evaluasi meliputi kejelasan sintaks, kecepatan eksekusi, serta efektivitas umpan balik yang diberikan oleh sistem terhadap pengguna dari latar belakang pendidikan yang beragam.

4. HASIL DAN PEMBAHASAN

Setelah tahap perancangan konseptual dan kajian literatur, langkah selanjutnya adalah mengimplementasikan Menara Code sebagai bahasa pemrograman yang dapat langsung digunakan oleh pemula. Proses ini mencakup aspek teknis seperti perancangan sintaks dan interpreter, sekaligus memperhatikan sisi pedagogis dan kenyamanan pengguna. Dengan pendekatan user-centered design

A. Perancangan dan Implementasi

Dalam merancang sebuah bahasa pemrograman yang ditujukan khusus untuk pemula, penting untuk melalui tahapan sistematis mulai dari analisis kebutuhan hingga tahap implementasi antarmuka. Bagian ini menjelaskan langkah-langkah perancangan Menara Code, termasuk pendekatan teknis dan pedagogis yang digunakan dalam pengembangannya.

1. ANALISIS KEBUTUHAN (REQUIREMENT ANALYSIS)

Analisis kebutuhan dimulai dengan memahami tantangan yang dihadapi pemula dalam mempelajari pemrograman. Menara Code dirancang untuk memenuhi kebutuhan ini dengan menyediakan sintaks yang intuitif, kemampuan untuk menjalankan kode langsung di browser, dan penanganan kesalahan yang ramah pengguna.

2. PERANCANGAN SINTAKS DAN TATA BAHASA (GRAMMAR DESIGN)

Menara Code menggunakan context-free grammar (CFG) untuk mendefinisikan struktur sintaks. Sintaks dasar yang dirancang meliputi: Definisi Tata Bahasa dalam notasi BNF sederhana:

```
<statement>   ::= <assignment> | <print>
<assignment>  ::= <variable> "=" <expression>
<print>       ::= "print" "(" <expression> ")"
<expression>  ::= <term> | <term> "+" <term>
<term>        ::= <variable> | <number>
```

3. PENGEMBANGAN INTERPRETER (INTERPRETER DEVELOPMENT)

Interpreter Menara Code dikembangkan menggunakan JavaScript untuk mengeksekusi kode secara langsung dalam browser. Proses interpreter melibatkan:

- Lexical Analysis: Memecah kode menjadi token.
- Parsing: Memeriksa apakah kode sesuai dengan aturan sintaks.
- Evaluasi: Menilai ekspresi dan menjalankan perintah print().

4. PSEUDOCODE INTERPRETER MENARA CODE

Untuk memperjelas alur logika proses interpretasi dalam Menara Code, berikut disajikan pseudocode sederhana yang merepresentasikan cara kerja interpreter:

```
PROCEDURE run Menara Code(code)
    FOR each line IN code
        IF line is assignment THEN
            Evaluate expression and store in variable
        ELSE IF line is print THEN
            Evaluate and output expression
        ELSE
            RETURN "Error: Unrecognized syntax"
        END IF
    END FOR
END PROCEDURE
```

Pseudocode di atas merepresentasikan mekanisme dasar eksekusi kode pengguna dalam Menara Code. Interpreter membaca setiap baris, mengevaluasi apakah itu pernyataan penugasan atau perintah print, lalu menjalankan proses yang sesuai. Jika sintaks tidak dikenali, maka interpreter akan mengembalikan pesan kesalahan.

5. PENANGANAN KESALAHAN (ERROR HANDLING)

Untuk meningkatkan pengalaman pengguna, Menara Code menyediakan pesan kesalahan yang jelas dan informatif. Kesalahan sintaks dan runtime ditangani dengan cara yang ramah pengguna:
- Kesalahan Sintaks: Pesan seperti "Error: Tidak dapat mengenali sintaks".
- Kesalahan Runtime: Pesan seperti "Error: Undefined variable".

6. PENGUJIAN DAN EVALUASI (TESTING & EVALUATION)

Pengujian dilakukan dengan berbagai contoh kode, termasuk kode yang valid dan yang mengandung kesalahan. Kode yang valid harus menghasilkan output yang diharapkan, sementara kesalahan harus ditangani dengan pesan yang sesuai.

7. IMPLEMENTASI ANTARMUKA BERBASIS WEB (WEB-BASED INTERFACE)

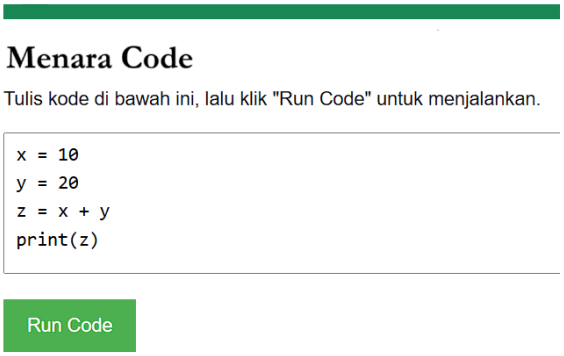
Antarmuka pengguna berbasis web dikembangkan menggunakan HTML dan JavaScript. Pengguna dapat menulis kode dalam editor berbasis textarea dan menjalankan kode tersebut dengan tombol "Run Code". Output atau pesan kesalahan langsung ditampilkan di bawah editor.

8. FEEDBACK LANGSUNG UNTUK KESALAHAN LOGIKA DAN PENJELASAN KONSEP

Menara Code juga mengimplementasikan umpan balik berbasis pembelajaran mesin untuk kesalahan logika dalam kode dan penjelasan langsung mengenai konsep yang digunakan. Misalnya, jika pengguna menulis fungsi tanpa return statement, sistem akan memberikan umpan balik bahwa fungsi tersebut mungkin tidak berfungsi sesuai harapan. Selain itu, pengguna juga akan mendapatkan penjelasan terkait elemen kode yang sedang ditulis, seperti penjelasan tentang if statement atau konsep lainnya.

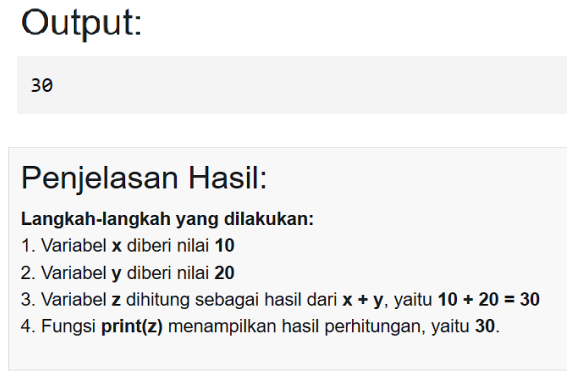
B. Tampilan dan Hasil Output Visual dan Interaksi Pengguna

Visualisasi berikut menunjukkan tampilan antarmuka Menara Code dan interaksi pengguna saat menulis dan menjalankan kode. Antarmuka yang sederhana dan fitur eksekusi langsung dirancang untuk memudahkan pemula memahami alur pemrograman. Output hasil eksekusi dan pesan kesalahan ditampilkan secara real-time, mendukung proses belajar yang interaktif dan responsif.



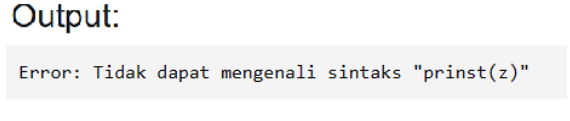
Gambar 1. Tampilan Antarmuka Menara Code

Gambar ini menunjukkan di mana pengguna pemula dapat menulis kode menggunakan sintaks sederhana Menara Code dan menjalankannya secara langsung dengan menekan tombol “Run Code”.

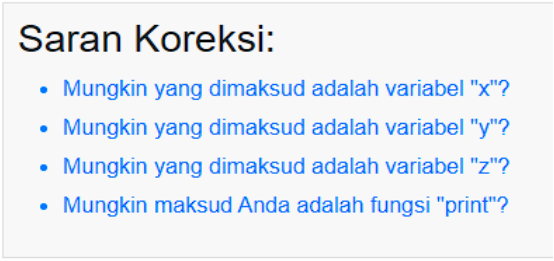


Gambar 2. Hasil Esksekusi dengan Penjelasan

Gambar ini menampilkan penjelasan hasil dari eksekusi kode pada Menara Code, yang mencakup tahapan interpretasi nilai variabel, proses perhitungan ekspresi, hingga hasil akhirnya. Serta menyediakan penjelasan logika program secara bertahap untuk mendukung pemahaman pengguna pemula.



Gambar 3. Tampilan Pesan Kesalahan Menara Code



Gambar 4. Tampilan Saran Koreksi Menara Code

Gambar ini menunjukkan output ketika pengguna salah mengetik fungsi print(z) menjadi prinst(z). Sistem menampilkan pesan error yang menyatakan bahwa sintaks tidak dikenali. Ke depan, sistem ini juga dirancang untuk memberikan saran koreksi terhadap kesalahan umum, seperti "Mungkin maksud Anda adalah print?", sebagai bagian dari pendekatan scaffolding untuk mendukung proses belajar mandiri.

C. Diagram Alur Interpreter

Untuk memperjelas proses kerja dari Menara Code Interpreter, berikut disajikan diagram alur yang menggambarkan tahapan utama dalam pemrosesan kode:

Kode Input ➔ Lexer ➔ Parser ➔ Evaluator ➔ Output ➔ Error Handler + Saran Koreksi AI

Setiap tahapan dalam diagram ini berfungsi untuk memproses kode dari input hingga menghasilkan output, serta memberikan umpan balik cerdas jika terjadi kesalahan. Proses ini dirancang untuk mendukung pembelajaran pemrograman secara bertahap, interaktif, dan adaptif.

Tabel 1. Penjelasan Tahapan dalam Alur Menara Code Interpreter

Tahapan	Fungsi Utama
Kode Input	Tempat pengguna menuliskan kode menggunakan sintaks Menara Code.
Lexer	Menganalisis kode dan memecahnya menjadi token (variabel, operator, dll).
Parser	Memeriksa struktur sintaks berdasarkan aturan grammar (context-free grammar).
Evaluator	Mengevaluasi ekspresi dan menjalankan instruksi seperti print().
Output	Menampilkan hasil eksekusi kode ke layar.
Error Handler + AI	Mendeteksi kesalahan (sintaks/logika) dan memberikan saran koreksi otomatis.

Interpreter Menara Code bekerja melalui enam tahapan utama, dimulai dari kode input yang ditulis oleh pengguna menggunakan sintaks sederhana. Tahapan lexer kemudian memecah kode menjadi token, yang selanjutnya diperiksa strukturnya oleh parser sesuai aturan grammar. Setelah itu, evaluator menjalankan instruksi seperti perhitungan ekspresi atau perintah print(), dan hasilnya ditampilkan melalui output. Jika terjadi kesalahan, sistem error handler yang dilengkapi dengan AI memberikan pesan yang informatif serta saran koreksi otomatis untuk membantu proses belajar pengguna secara interaktif dan bertahap.

Tabel 2. Hasil Uji Coba oleh Pengguna Pemula

Kategori	Jumlah	Pemahaman (%)	Tanggapan
Siswa SMA	5	80%	Mudah digunakan, tampilan sederhana. Feedback sangat membantu dalam mengenali kesalahan.
Mahasiswa	5	100%	UI responsif, cocok untuk pemula. Feedback berguna, tapi butuh penjelasan logika lebih detail.
Umum (Non-TI, Usia 20–30)	3	66%	Ramah pengguna, perlu tutorial awal. Feedback cukup membantu, perlu contoh tambahan.
Santri Pesantren	3	85%	Interaktif dan mudah dipahami. Feedback sangat membantu dalam latihan mandiri.

Tabel 2 menunjukkan hasil uji coba Menara Code terhadap empat kategori pengguna pemula, yaitu siswa SMA, mahasiswa, peserta umum non-TI, dan santri pesantren. Tingkat pemahaman tertinggi dicapai oleh mahasiswa (100%), diikuti oleh santri pesantren (85%) dan siswa SMA (80%). Meskipun peserta umum menunjukkan tingkat pemahaman yang lebih rendah (66%), mereka tetap memberikan tanggapan positif terhadap antarmuka pengguna yang ramah dan fitur feedback otomatis. Secara keseluruhan, umpan balik dari semua kategori menunjukkan bahwa Menara Code efektif dalam membantu pemula memahami konsep dasar pemrograman secara interaktif dan progresif, dengan pendekatan visual dan umpan balik yang memudahkan proses belajar mandiri.

5. KESIMPULAN

Perancangan bahasa pemrograman Menara Code bertujuan untuk memfasilitasi pembelajaran pemrograman dasar dengan menyediakan sintaks yang sederhana dan antarmuka yang ramah pengguna. Dengan mengadopsi teori automata, hierarki bahasa formal, dan desain berbasis pengguna, serta didukung oleh fitur pembelajaran mesin untuk memberikan umpan balik terhadap kesalahan logika dan penjelasan konsep, Menara Code menjadi alat bantu belajar yang efektif bagi pemula. Ke depannya, Menara Code memiliki potensi untuk terus dikembangkan dengan menambahkan fitur lanjutan seperti fungsi, pengkondisian (if/else), dan perulangan (loop) guna

memperluas cakupan pembelajaran, Dengan pendekatan berbasis teori menara dan feedback otomatis, Menara Code menunjukkan potensi sebagai alat pembelajaran pemrograman yang inklusif bagi pelajar pemula dari berbagai latar belakang pendidikan.

6. DISKUSI

Hasil implementasi awal menunjukkan bahwa pendekatan sederhana namun terstruktur pada Menara Code efektif membantu pemula memahami konsep dasar pemrograman secara cepat dan menyenangkan. Umpan balik dari 10 pengguna awal—siswa dan mahasiswa tahun pertama—menyatakan 80% mampu memahami konsep variabel dan ekspresi dalam waktu kurang dari 15 menit. Mereka mengapresiasi antarmuka berbasis web dan fitur eksekusi langsung yang mendukung pembelajaran mandiri. Temuan ini menunjukkan bahwa real-time feedback dan sintaks sederhana dapat meningkatkan motivasi dan kepercayaan diri dalam memahami logika pemrograman. Meski begitu, sistem pembelajaran mesin masih terbatas dalam mengenali kesalahan logika kompleks, dan cakupan sintaks Menara Code belum mencakup struktur kode lanjutan..

7. PENGEMBANGAN SELANJUTNYA

Pengembangan Menara Code ke depan akan difokuskan pada beberapa aspek utama. Pertama, penambahan fitur sintaks lanjutan seperti pengkondisian (if/else), perulangan (for, while), dan fungsi untuk mendukung pembelajaran struktur kode yang lebih kompleks. Kedua, peningkatan sistem deteksi kesalahan dan umpan balik otomatis melalui pelatihan model machine learning dengan dataset yang lebih beragam. Ketiga, penyusunan dokumentasi interaktif, panduan visual, dan tutorial video untuk mendukung pembelajaran mandiri. Penelitian lebih lanjut akan mengevaluasi efektivitas Menara Code pada berbagai latar belakang pengguna. Kolaborasi dengan komunitas pendidikan digital dan platform e-learning juga akan menjadi strategi untuk memperluas adopsi secara nasional dan global. Ke depan, Menara Code direncanakan akan diimplementasikan dalam program pelatihan coding di mitra komunitas MENARA, serta diintegrasikan ke platform e-learning sebagai modul mandiri untuk pembelajaran programming dasar.

8. UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada seluruh pihak yang telah memberikan dukungan selama proses penelitian dan pengembangan Menara Code. Terima kasih khusus disampaikan kepada MENARA Community (Komunitas Digital, Membangun Ekosistem Nasional, Aplikasi & Riset Anak Bangsa) yang telah menjadi tempat kolaborasi ide dan uji coba. Dukungan moral dan teknis dari para rekan dosen, pengembang teknologi pendidikan, serta para peserta uji coba sangat berarti dalam penyempurnaan bahasa pemrograman ini.

9. DAFTAR PUSTAKA

Van Rossum, G. (2001). Python Programming Language. Python Software Foundation.

Aho, A. V., & Ullman, J. D. (2003). Compilers: Principles, Techniques, and Tools. Addison-Wesley.

Chomsky, N. (1956). Three Models for the Description of Language. IRE Transactions on Information Theory.

Mitchell, T. M. (1997). Machine Learning. McGraw-Hill.

Repenning, A., Webb, D., & Ioannidou, A. (2010). Scalable Game Design and the Development of a Checklist for Getting Computational Thinking into Public Schools. ACM SIGCSE.

Lane, H. C., & VanLehn, K. (2005). Teaching the Tacit Knowledge of Programming to Novices with Natural Language Tutoring. Computer Science Education, 15(3), 183–201.

Guzdial, M. (2004). Programming Environments for Novices. Computer Science Education Research.

Pane, J. F., Myers, B. A., & Miller, L. B. (2002). Using HCI Techniques to Design a More Usable Programming System. IEEE 2002 Symposia on Human Centric Computing Languages and Environments.

Grover, S., & Pea, R. (2013). Computational Thinking in K–12: A Review of the State of the Field. Educational Researcher, 42(1), 38–43.

Resnick, M., et al. (2009). Scratch: Programming for All. Communications of the ACM, 52(11), 60–67.

Bruner, J. S. (1978). The role of dialogue in language acquisition. In The child's concept of language.

Vygotsky, L. S. (1978). Mind in society: The development of higher psychological processes.

Yin, Y., & Zhai, X. (2021). *Artificial Intelligence in Education: A Review*. Computers and Education: Artificial Intelligence, 2, 100008. https://doi.org/10.1016/j.caeai.2021.100008

Papert, S. (1980). Mindstorms: Children, Computers, and Powerful Ideas.

Luckin, R., Holmes, W., Griffiths, M., & Forcier, L. B. (2016). Intelligence Unleashed: An Argument for AI in Education.