Majalah Ilmiah UNIKOM Vol.22 No. 1

BIDANG
ILMU KOMPUTER & TIK

APRIL 2024, 3 - 8
p-ISSN : 1411-9374, e-ISSN : 2527-7030

PENGARUH *BUILDER DESIGN PATTERN* PADA PERFORMA APLIKASI DENGAN MENGGUNAKAN BAHASA PEMROGRAMAN GOLANG

NABILA KHAIRUNISA[1], RADEN MOHAMAD RISHWAN[2], RAIHAN AKBAR SAPUTRO[3], CARUDIN[4]
Universitas Singaperbangsa Karawang[1,2,3,4]
2010631170101@student.unsika.ac.id

ABSTRACT

In the industrial era 4.0, technological developments are increasingly progressing, including the field of software engineering. Therefore, a solution is needed to overcome problems during software development, especially application performance. The solution uses design patterns, one of them is the builder design pattern. The Builder Design Pattern is a design pattern in the world of programming that is used to simplify the process of creating complex objects by breaking and breaking code into simpler parts. The method used in this study is an experimental approach, in which researchers look for the influence between variables by using the benchmark evaluation method. This research was conducted by creating two program codes, one using the Builder Design Pattern and the other without using the Builder Design Pattern. Then, the two codes are tested using several performance metrics such as execution time and memory usage. The results of this research show that the use of the Builder Design Pattern can significantly affect performance in some cases. Therefore, the use of the Builder Design Pattern must be considered properly depending on the needs of the application being built to produce an application with maximum performance.

Key Words: *Design Pattern, Builder Design Pattern, Benchmark, Execution Time*

ABSTRAK

Pada era industri 4.0, perkembangan teknologi semakin mengalami kemajuan termasuk bidang rekayasa perangkat lunak. Oleh karena itu, diperlukan solusi untuk mengatasi masalah saat develop perangkat lunak khususnya performa aplikasi. Solusi tersebut dengan menggunakan design pattern, salah satunya builder design pattern. Builder design pattern merupakan salah satu design pattern dalam dunia pemrograman yang digunakan untuk mempermudah proses pembuatan objek kompleks dengan cara memecah dan kode menjadi beberapa bagian yang lebih sederhana. Metode yang digunakan dalam penelitian adalah pendekatan eksperimental, dimana peneliti mencari pengaruh antar variabel dengan menggunakan metode evaluasi benchmark. Penelitian ini dilakukan dengan membuat dua kode program, dimana satu menggunakan Builder design Pattern dan yang lainnya tanpa menggunakan builder design pattern. Kemudian, kedua kode tersebut diuji dengan menggunakan beberapa metrik performa seperti waktu eksekusi dan penggunaan memori. Hasil penelitian menunjukkan bahwa penggunaan builder design pattern dapat mempengaruhi performa yang cukup signifikan dalam beberapa kasus. Oleh karena itu, penggunaan builder design pattern harus dipertimbangkan dengan baik bergantung pada kebutuhan aplikasi yang dibangun untuk menghasilkan aplikasi dengan performa yang lebih maksimal.

Kata Kunci: *Design Pattern, Builder Design Pattern, Benchmark, Execution Time*

PENDAHULUAN

Pada era industri 4.0, perkembangan teknologi semakin mengalami kemajuan termasuk bidang rekayasa perangkat lunak. Perkembangan dan pengembangan aplikasi menjadi hal yang sangat penting. Dengan perkembangan teknologi, penggunaan aplikasi menjadi luas dan beragam [1]. Maka dari itu, salah satu faktor krusial dalam pengembangan aplikasi adalah kinerja yang baik, seperti kecepatan dalam eksekusi, efisiensi dalam penggunaan memori, dan penggunaan CPU yang optimal. Dalam hal ini, pemrograman *design pattern* dapat menjadi solusi untuk optimalisasi performa aplikasi [2].

Namun ada hal lain yang perlu diperhatikan dalam pengembangan aplikasi, seperti aplikasi yang mudah dibaca atau *human readable*, kecepatan dalam membangun aplikasi, dan bagaimana kode yang kita buat dapat lebih mudah dikembangkan baik itu untuk kita sendiri maupun orang lain jika dalam pengembangannya bersama tim [3].

Design pattern dalam pemrograman dikenal sebagai metode umum yang digunakan dalam mengatasi masalah yang sering dihadapi saat pengembangan aplikasi. *Design pattern* merupakan tipikal solusi untuk masalah desain yang terjadi secara berulang [4]. Salah satu *design pattern* yang biasa digunakan adalah *Builder Design Pattern*. *Builder Design Pattern* merupakan teknik pemrograman yang digunakan untuk memecah pembuatan objek yang kompleks menjadi bagian-bagian yang lebih sederhana. Hal ini dapat menyederhanakan proses pembuatan objek yang kompleks dan meningkatkan fleksibilitas dalam mengubah bagian-bagian dari objek tersebut [5]. *Design pattern* ini memungkinkan pembuatan polydispersity dengan mudah, karena aturan *building* yang memasukkan fungsi sebagai argumen dievaluasi selama tahap *building* [6].

Penelitian ini bertujuan untuk mengetahui bagaimana pengaruh *builder design pattern* terhadap performa aplikasi. Dimana nanti hasilnya dapat membantu *developer* dalam menentukan design pattern yang akan digunakan dan mendapatkan hasil lebih maksimal.

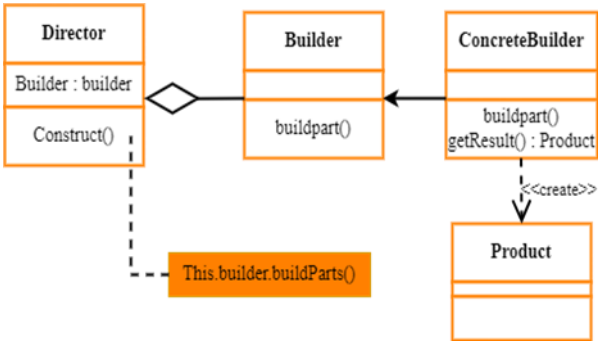
Untuk mencapai tujuan dan memperoleh solusi dari permasalahan sebelumnya, maka perlu adanya tahapan serta metode penelitian yang akan dilakukan. Metode penelitian yang digunakan yaitu pendekatan eksperimental dimana peneliti mencari pengaruh variabel satu terhadap variabel lain.

Penelitian ini dilakukan dengan membuat dua buah kode program, satu menggunakan *Builder Design Pattern* dan yang lainnya tanpa menggunakan *Builder Design Pattern*. Kedua kode tersebut diuji dengan menggunakan beberapa metrik performa seperti kecepatan eksekusi, efisiensi dalam pemakaian memori, dan penggunaan CPU yang optimal. Hasil dari penelitian ini akan memberikan informasi yang berguna bagi pengembang aplikasi dalam memilih teknik pemrograman yang tepat untuk meningkatkan kinerja dan performa aplikasi yang dikembangkan.

TINJAUAN PUSTAKA

Design pattern merupakan solusi tipikal dari masalah yang dapat terjadi pada desain perangkat lunak. *Design pattern* hampir sama seperti *blueprint* yang siap digunakan dan dapat kita sesuaikan untuk menyelesaikan persoalan desain yang berulang dalam koding [7]. Berdasarkan sifatnya terdapat 3 *design pattern*, yaitu *Creational*, pola yang digunakan dalam memecahkan masalah pada mekanisme pembuatan objekt; *Behavioral*, pola yang digunakan dalam memecahkan masalah dengan melakukan komunikasi sesama objek; serta *Structural*, pola yang digunakan dalam memecahkan masalah yang bersinggungan dengan komposisi kelas dan objek [8]. Setiap *design pattern* terdiri atas nama, pemaparan permasalahan yang sesuai, pemaparan solusi yang ditawarkan, serta konsekuensi penggunaannya.

Builder design pattern dapat digunakan untuk mempermudah proses pembuatan objek yang bersifat kompleks dengan menyediakan antarmuka yang lebih sederhana dan dapat mengisi secara individual nilai melalui pemanggilan metode [9]. *Design pattern* ini memiliki dua bagian utama yaitu builder untuk membuat objek dan *director* untuk mengarahkan builder dalam membuat objek dengan cara yang benar. Memisahkan proses pembuatan objek dari kelas utama yang dapat meningkatkan fleksibilitas dan dapat memudahkan pemeliharaan kode merupakan dasar dari *builder design pattern* [8]. Dengan menggunakan *builder design pattern*, dapat memudahkan proses pembuatan objek yang kompleks dan dapat meningkatkan performa aplikasi secara efisien.



Gambar 1. Diagram UML *Builder Design Pattern*

Golang merupakan bahasa pemrograman *open source* yang dikembangkan untuk memenuhi kebutuhan *developer* dalam membuat aplikasi yang scalable, efisien, dan mudah dipelajari. Golang memiliki sintaks yang sederhana dan mudah dipahami sehingga mempunyai keunggulan pada sisi kecepatan, keandalan, kesederhanaan, dan skalabilitas. Golang banyak digunakan dalam pengembangan aplikasi *software*, *server*, sistem operasi, dan aplikasi *cloud*. Ada beberapa fitur yang menjadi keunggulan pada bahasa pemrograman golang diantaranya yaitu *goroutines*, *garbage collector*, *cross platform*, *compiled language*, dan *standard library* [10].

METODE PENELITIAN

Metode penelitian yang digunakan yaitu pendekatan eksperimental, dimana peneliti akan mencari bagaimana pengaruh faktor tertentu terhadap faktor lain dengan diawasi secara ketat. Pendekatan tersebut bertujuan untuk melakukan perbandingan suatu akibat tindakan tertentu dengan tindakan yang berbeda. Terdapat beberapa tahapan yaitu mencari studi kasus, menganalisa kasus, penerapan *builder design pattern*, dan pengujian performa.

Gambar 2. Metode Pendekatan Eksperimental

1. Mencari Studi Kasus

Dilakukan pencarian dengan mencari kasus-kasus yang sering terjadi dalam pembuatan aplikasi. Dimana pada kasus tersebut sering digunakan *builder pattern* dalam pengembangannya. Hasil pencariannya yaitu terjadi pada kasus pembuatan objek yang sama namun memiliki *attribute* yang berbeda.

2. Menganalisa Kasus

Menganalisa bagaimana kasus tersebut dapat diimplementasikan ke dalam *builder pattern*. Hal ini tidak dapat digunakan pada semua kasus dikarenakan harus tetap pada tujuan utama dari *design pattern* yaitu mengurangi *boilerplate code* atau pengulangan kode dan meminimalisir kesalahan dalam penulisan kode.

3. Penerapan *Builder Design Pattern*

Pada tahap dilakukan implementasi kode dimana terdapat dua kode, yaitu kode yang menerapkan *builder design pattern* dan yang tidak menerapkan *builder design pattern.* Hal ini dilakukan untuk melakukan perbandingan pengujian performa pada dua kode tersebut.

4. Pengujian Performa

Design pattern yang telah dibuat pada tahap sebelumnya, akan diujikan dengan 3 buah metode. Dimana pada metode pertama akan disimulasikan dengan membuat satu buah objek yang sebanyak-banyaknya, kedua yaitu dengan cara membuat tiga objek yang berbeda sebanyaknya, dan terakhir yaitu sama dengan sebelumnya namun objek yang dibuat dialokasikan pada memori. Lalu akan dilakukan pengujian sebanyak 10 kali untuk memastikan bahwa pengetesan dilakukan dengan cara yang benar dan konsisten.

Selain dengan menggunakan metode pendekatan eksperimental, peneliti juga menggunakan *benchmark* yang merupakan metode atau evaluasi dalam mengukur performa aplikasi maupun sistem dengan memperhatikan berbagai parameter seperti penggunaan memory, waktu eksekusi, kecepatan input/output, dan lainnya. Evaluasi dengan menggunakan *benchmark* biasanya akan mempunyai hasil akhir berupa angka. Semakin tinggi angka hasil *benchmark*, maka dapat diartikan bahwa aplikasi tersebut memiliki performa yang lebih baik. Akan tetapi, ada juga indikator-indikator tertentu, sebab performa sebuah aplikasi memiliki banyak ukuran. *Benchmark* dapat memberikan konsep pembaharuan karena dikenali sebagai metode yang untuk melakukan pembaharuan dengan cara observasi dan sebagai analogi dalam untuk menentukan penerapan terbaik [11].

HASIL DAN PEMBAHASAN

Berdasarkan hasil pengujian dari implementasi *builder design pattern* terhadap performa aplikasi. Peneliti menggunakan beberapa parameter untuk melihat hasil yang lebih maksimal dan membuat beberapa skenario dalam membuat objek.

1. Hasil Pengujian

Ada beberapa parameter dalam pengujian yang dilakukan, yaitu berapa banyak operasi di eksekusi, waktu eksekusi, memori yang digunakan,

dan juga variabel yang dialokasikan ke dalam memori. Untuk mengetahui hal tersebut, peneliti membuat 3 buah skenario. Dimana yang pertama hanya membuat 1 buah objek yang sama, kedua yaitu membuat 3 objek yang berbeda, dan terakhir yaitu membuat 3 objek yang berbeda namun objek tersebut dialokasikan ke dalam memori.

Dalam melakukan test, kami melakukannya menggunakan laptop dengan spesifikasi yang dapat dilihat pada tabel dibawah. Hal ini menjadi catatan yang cukup penting dikarenakan ada kemungkinan perbedaan jika menggunakan spesifikasi yang berbeda.

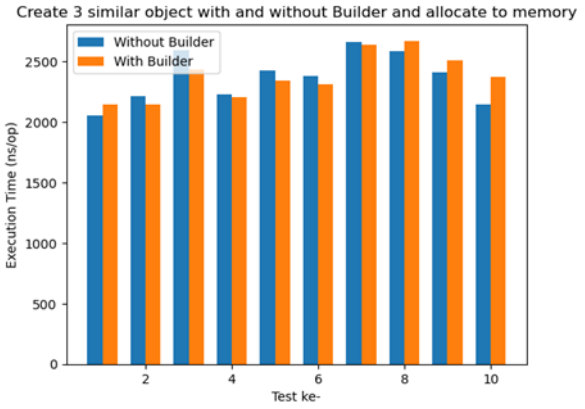
Tabel 1. Spesifikasi Komputer

OS	EndeavourOS Linux x86_64
Host	Swift SF314-41 V1.10
Kernel	5.15.94-1-lts
CPU	AMD Ryzen 5 3500U with Radeon Vega Mobile Gfx (8) @ 2.100GHz
Memory	12GB
Runtime	go version go1.20.1 linux/amd64

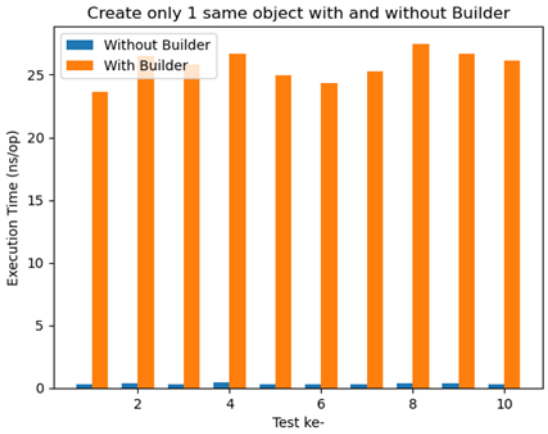
2. *Execution Time*

Execution Time didefinisikan sebagai waktu yang dihabiskan oleh sistem untuk mengeksekusi tugas menggunakan cara lain dan dapat mengatakan waktu selama program berjalan. Pada saat proses eksekusi *library runtime* digunakan. Ada beberapa operasi dasar yang terjadi pada saat eksekusi termasuk membaca instruksi program dalam menjalankan tugas atau menyelesaikan tindakan [12]. Berikut merupakan grafik hasil pengujian pengaruh *builder design pattern* terhadap performa aplikasi yang dapat dilihat pada gambar 3, 4, dan 5.

Gambar 3. Grafik Pengujian Menggunakan 3 Objek dan Mengalokasikan ke Memori

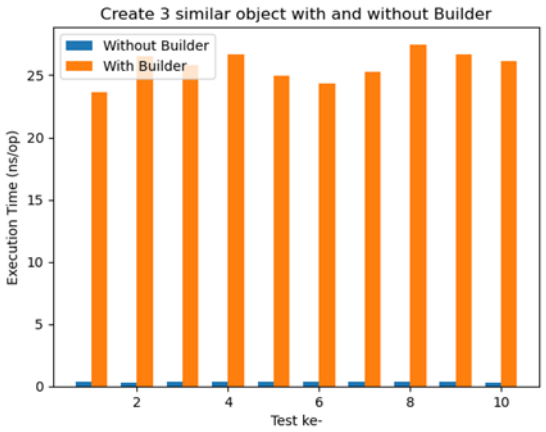


Gambar 3 menunjukkan pengujian dengan membuat 3 objek serupa dengan dan tanpa menggunakan *builder pattern* dan mengalokasikannya ke memori. Pada pengujian ini memiliki *execution time* dengan perbedaan yang tidak signifikan dan tidak terlalu beasar, artinya pada pengujian ini mempunyai *execution time* yang lambat. *Execution time* tercepat yaitu pada test ke-1 dengan kisaran 2000 ns/op.



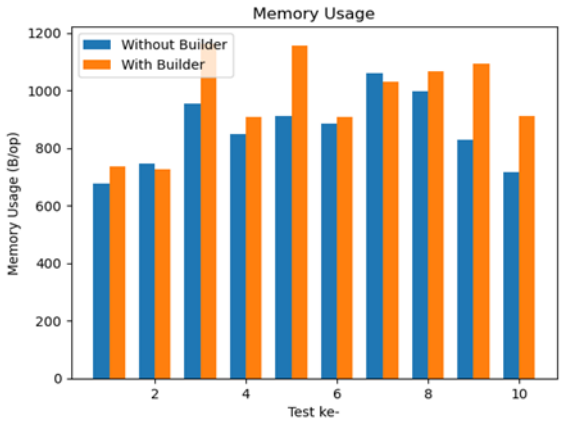
Gambar 4. Grafik Pengujian Menggunakan 1 Objek

Berdasarkan gambar 4, pengujian dengan menggunakan 1 objek menunjukkan bahwa penggunaan *builder design pattern* semakin memperlambat *execution time*. Dapat dilihat pada grafik perbedaannya sangat signifikan.



Gambar 5. Grafik Pengujian Menggunakan 3 Objek

Hasil pada gambar 5, dengan menggunakan 3 objek tanpa pengalokasian ke memori juga menunjukkan bahwa tanpa penggunaan *builder design pattern*, *execution time* yang diperlukan lebih sedikit dan dengan menggunakan builder design pattern justru *execution time* yang diperlukan lebih banyak dan memiliki perbandingan yang sangat signifikan.



Gambar 6. Grafik Penggunaan Memori

Pada gambar 6 pada saat melakukan pengujian, penggunaan memori oleh kedua kode yang dijalankan tidak terdapat perbedaan yang signifikan. Oleh karena itu penggunaan memori tidak mempengaruhi hasil performa pada pengujian.

Berdasarkan gambar 3, 4, 5, dan 6 yang telah dijelaskan sebelumnya, pengujian dilakukan untuk membandingkan objek dengan menggunakan dan tanpa *builder design pattern*. Hasil pengujian menunjukkan bahwa penggunaan *builder design pattern* cenderung memperlambat proses, terutama ketika digunakan untuk membuat satu objek. Dalam pengujian ini, penggunaan *builder design pattern* atau tidak juga tidak berdampak signifikan pada penggunaan memori. Oleh karena itu, pilihan antara menggunakan atau tidak menggunakan *builder design pattern* sebaiknya dipertimbangkan kembali dengan baik, terutama jika performa eksekusi menjadi faktor penentu.

3. Analisis Hasil Pengujian

Berdasarkan hasil pengujian, dapat dilihat bahwa pada *execution time* yang diperlukan tanpa menggunakan builder pattern lebih sedikit dibandingkan dengan menggunakan builder pattern. Hal itu terjadi di semua skenario yang sudah kami buat. Namun, perbedaan cukup signifikan ketika mengalokasikan objek ke dalam memori. Sedangkan pada penggunaan memori memiliki perbedaan sedikit saja.

Berdasarkan hasil tersebut, hal itu dapat menjadi salah satu bukti bahwa builder pattern mempengaruhi performa aplikasi cukup signifikan pada beberapa skenario. Dapat dilihat bahwa *execution time* ketika menggunakan builder pattern cukup signifikan bahkan sampai 25x lebih lambat. Namun, nyatanya skenario yang paling signifikan yaitu variabel yang dibuat akan dialokasikan ke memori dan perbedaan *execution time* tidak terlalu besar.

Pada penggunaan memori, tidak ada perbedaan yang signifikan dari kedua kode tersebut. Namun, perlu diingat bahwa perbedaan tersebut bisa menjadi menjadi salah satu perhitungan ketika memutuskan untuk menggunakan *builder design pattern*.

KESIMPULAN

Hasil dari pengujian menunjukkan bahwa penggunaan *builder design pattern* dapat mempengaruhi performa aplikasi dengan cukup signifikan pada beberapa skenario yang telah di atur. *Execution time* pada aplikasi dengan menggunakan pattern tersebut menjadi lebih lambat, namun perbedaan paling signifikan terjadi saat variabel yang dibuat dialokasikan ke memori dengan perbedaan yang tidak terlalu besar. Dalam penggunaan memori, tidak ada perbedaan yang signifikan terhadap penggunaan *builder design pattern* dan tanpa menggunakan *builder design pattern*. Oleh karena itu, diperlukannya pertimbangan dalam menggunakan *builder pattern* dari segala aspek dan dampak yang mungkin terjadi pada aplikasi. Pengaruh performa ini seharusnya memang terjadi di hampir semua design pattern. Hal ini dikarenakan design pattern dirancang untuk mempermudah proses *development* aplikasi, mengurangi kode yang sama berulang kali, dan mengurangi resiko salah ketik dalam pembuatan aplikasi.

Dengan menggunakan *builder design pattern* terdapat kelebihan yang dapat membantu mengurangi total parameter dalam konstruktor dan oleh karena itu tidak perlu meneruskan null untuk standar opsional ke konstruktor. Sedangkan kekurangan penggunaan *design pattern* ini yaitu pada setiap jenis produk yang berbeda membutuhkan pembuatan *concrete builder* secara terpisah. Dengan kelebihan dan kekurangan tersebut, pengembang selanjutnya diharapkan dapat mencoba menggunakan design pattern lain untuk mencari yang terbaik dan menghasilkan performa yang lebih maksimal.

UCAPAN TERIMA KASIH

Terima kasih kepada semua pihak yang telah memberikan dukungan dan motivasi selama melakukan penelitian ini. Tanpa adanya dukungan dan motivasi tersebut, penulis tidak dapat menyelesaikan penelitian sampai akhir.

DAFTAR PUSTAKA

[1] F. D. Anastasia and I. V. Paputungan, "Implementasi Bloc Pattern pada Pengembangan Front-End Top Up LinkAja untuk Aplikasi Agen46," *Jurnal SNATI*, vol. 2, no. 1, pp. 1–11, 2022.

[2] I. G. B. V. Putraadinatha, D. D. J. Suwawi, and S. Y. Puspitasari, "Pengaruh Design Pattern Terhadap Maintainability Aplikasi Mobile," *e-Proceeding of Engineering*, vol. 8, no. 5, pp. 10954–10961, Oct. 2021.

[3] T. Dürschmid, "Design pattern builder: A concept for refinable reusable design pattern libraries," *Companion Proceedings of the 2016 ACM SIGPLAN International Conference on Systems, Programming, Languages and Applications: Software for Humanity*, pp. 45–46, 2016. doi:10.1145/2984043.2998537.

[4] L. Marchesi, M. Marchesi, G. Destefanis, G. Barabino, and D. Tigano, "Design patterns for gas optimization in Ethereum," *2020 IEEE International Workshop on Blockchain Oriented Software Engineering (IWBOSE)*, pp. 9–15, 2020. doi:10.1109/iwbose50093.2020.9050163.

[5] M. A. Abdukadirova, "The role of builder and building in the development of the country is invaluable," *The American Journal of Interdisciplinary Innovations and Research*, vol. 03, no. 05, pp. 81–84, May 2021. doi:10.37547/tajiir/volume03issue05-14.

[6] M. Girard, A. Ehlen, A. Shakya, T. Bereau, and M. O. de la Cruz, "Hoobas: A highly object-oriented builder for molecular dynamics," *Computational Materials Science*, vol. 167, pp. 25–33, May 2019. doi:10.1016/j.commatsci.2019.05.003.

[7] D. Drozdov *et al.*, "Utilizing software design patterns in product-driven manufacturing system: A case study," *Service Oriented, Holonic and Multi-agent Manufacturing Systems for Industry of the Future*, pp. 301–312, 2020. doi:10.1007/978-3-030-27477-1_23.

[8] "Design patterns," Refactoring.Guru, https://refactoring.guru/design-patterns (accessed Jul. 1, 2023).

[9] J. Pivarski, P. Elmer, and D. Lange, "Awkward arrays in python, C++, and Numba," *EPJ Web of Conferences*, vol. 245, pp. 1–7, 2020. doi:10.1051/epjconf/202024505023.

[10] A. S. Sari and R. Hidayat, "Designing website vaccine booking system using golang programming language and framework react JS," *Journal of Information System, Informatics and Computing*, vol. 6, no. 1, pp. 22–39, Jun. 2022. doi:10.52362/jisicom.v6i1.760.

[11] A. D. Rahma, Y. Rohayati, and I. N. Kusmayanti, "Usulan Perbaikan Media Komunikasi Pemasaran Online Pada Instagram Navara Footwear Menggunakan Metode Benchmarking Dan Analytical Hierarchy Process," *Smart Comp*, vol. 11, no. 2, pp. 239–246, Apr. 2022.

[12] GeeksforGeeks. (2023, May 3). *Measure execution time with high precision in C/C++*. GeeksforGeeks. https://www.geeksforgeeks.org/measure-execution-time-with-high-precision-in-c-c/.