JITET (Jurnal Informatika dan Teknik Elektro Terapan)
Vol. 14 No. 3, pISSN: 2303-0577 eISSN: 2830-7062 http://dx.doi.org/10.23960/jitet.v14i3.9643

# IMPLEMENTASI POLA DESAIN BUILDER UNTUK FLEKSIBILITAS KUSTOMISASI PADA SISTEM PEMESANAN PRODUK MULTIVARIAN (STUDI KASUS: AYAM GEPREK)

**Muhammad Rafi Catur Wijayanto[1*], Carudin[2]**

[1,2]Fakultas Ilmu Komputer, Universitas Singaperbangsa Karawang; Jl. HS. Ronggo Waluyo, Puseurjaya, Telukjambe Timur, Karawang, Jawa Barat 41361

**Keywords:**
Pola Builder; Sistem Pemesanan Mandiri; Research and Development; PHP; MySQL.

**Corespondent Email:**
mraficatur23@gmail.com

**Abstrak.** Ayam geprek merupakan kuliner populer di lingkungan kampus yang proses pemesanannya seringkali repetitif dan rentan kesalahan. Penelitian ini bertujuan merancang sistem pemesanan mandiri yang fleksibel untuk mengatasi masalah tersebut. Metode penelitian yang digunakan adalah *Research and Development (R&D*) dengan proses pengembangan produk mengadopsi model *Waterfall*. Sistem ini dibangun menggunakan *PHP native* dan basis data *MySQL*, dengan mengimplementasikan Pola Desain *Builder* untuk memfasilitasi kustomisasi produk yang kompleks. Hasilnya adalah sebuah prototipe aplikasi web fungsional yang memungkinkan pelanggan "membangun" pesanan ayam geprek langkah demi langkah sesuai preferensi. Pengujian menunjukkan bahwa *Pola Builder* secara efektif memisahkan logika konstruksi objek dari representasinya, sehingga meningkatkan fleksibilitas, akurasi pesanan, dan skalabilitas sistem. Implementasi ini menunjukkan bahwa *Pola Builder* merupakan solusi yang efisien untuk sistem pemesanan dengan tingkat variasi produk yang tinggi.



**Abstract.** *Ayam geprek is a popular culinary dish in campus environments, yet its ordering process is often repetitive and prone to errors. This research aims to design a flexible self-service ordering system to address this issue. The research method employed is Research and Development (R&D), with the product development process adopting the Waterfall model. The system was built using native PHP and a MySQL database, implementing the Builder Design Pattern to facilitate complex product customization. The result is a functional web application prototype that allows customers to "build" their ayam geprek order step-by-step according to their preferences. Testing revealed that the Builder Pattern effectively separates the object construction logic from its representation, thereby enhancing flexibility, order accuracy, and system scalability. This implementation demonstrates that the Builder Pattern is an efficient solution for ordering systems with a high degree of product variation.*

## 1. PENDAHULUAN

Perkembangan teknologi digital telah mentransformasi berbagai sektor industri, termasuk industri kuliner. Salah satu inovasi yang signifikan adalah sistem pemesanan digital yang menawarkan efisiensi dan kemudahan bagi konsumen maupun produsen [1]. Di lingkungan institusi pendidikan seperti Universitas Singaperbangsa Karawang (UNSIKA), produk kuliner seperti ayam geprek menjadi sangat populer karena harganya yang terjangkau dan rasanya yang dapat disesuaikan

dengan selera [2]. Namun, popularitas ini seringkali tidak diimbangi dengan sistem pemesanan yang efisien. Proses pemesanan yang masih dilakukan secara manual, di mana penjual mencatat pesanan satu per satu, sangat rentan terhadap kesalahan (human error), terutama pada jam sibuk. Kesalahan seperti salah level pedas, jenis nasi, atau topping tambahan sering terjadi dan dapat menurunkan kepuasan pelanggan.

Masalah ini berakar dari kompleksitas objek pesanan itu sendiri. Sebuah pesanan ayam geprek dapat memiliki banyak variasi, seperti jenis nasi (nasi biasa atau nasi uduk), level pedas (level 1 hingga 10), serta berbagai pilihan topping (keju, telur, sosis). Jika dikelola secara manual atau dengan sistem yang tidak fleksibel, kompleksitas ini menjadi tantangan besar [3]. Untuk mengatasi tantangan dalam rekayasa perangkat lunak, penggunaan pola desain (design pattern) telah terbukti menjadi solusi yang efektif untuk masalah-masalah yang berulang [4].

Untuk kasus konstruksi objek yang memiliki banyak parameter atau tahapan, Builder Pattern merupakan salah satu pola desain kreasi (creational pattern) yang paling sesuai [5]. Pola ini memungkinkan pembuatan objek kompleks dilakukan langkah demi langkah, memisahkan logika konstruksi objek dari representasi akhirnya. Dengan pendekatan ini, klien dapat membuat berbagai representasi objek yang berbeda dengan menggunakan proses konstruksi yang sama [6]. Hal ini sangat relevan untuk sistem pemesanan ayam geprek, di mana setiap komponen pesanan dapat dibangun secara bertahap tanpa harus membuat konstruktor yang rumit dengan banyak parameter.

Oleh karena itu, penelitian ini mengusulkan pengembangan sistem pemesanan mandiri berbasis web dengan mengimplementasikan Builder Pattern. Sistem ini dirancang untuk memberikan fleksibilitas penuh kepada pelanggan dalam menyesuaikan pesanan mereka, sekaligus meminimalkan risiko kesalahan yang sering terjadi pada sistem manual.

## 2. TINJAUAN PUSTAKA

### 2.1 Metode *Research and Development* (R&D)

Metode *Research and Development* (R&D) adalah suatu proses yang digunakan untuk mengembangkan dan memvalidasi produk baru atau menyempurnakan produk yang sudah ada [7]. Metode ini sangat relevan untuk penelitian yang berorientasi pada hasil berupa produk, baik itu perangkat keras maupun perangkat lunak. Salah satu model R&D yang paling banyak dirujuk adalah model yang dikembangkan oleh Borg dan Gall, yang mencakup sepuluh langkah sistematis: (1) Penelitian dan pengumpulan informasi, (2) Perencanaan, (3) Pengembangan draf produk, (4) Uji coba lapangan awal, (5) Revisi produk utama, (6) Uji coba lapangan utama, (7) Revisi produk operasional, (8) Uji coba lapangan operasional, (9) Revisi produk akhir, dan (10) Diseminasi dan implementasi [8].

Dalam konteks pengembangan perangkat lunak, model ini seringkali diadaptasi dan disederhanakan agar sesuai dengan siklus pengembangan yang lebih cepat [9]. Penerapan R&D dalam pengembangan sistem informasi terbukti efektif untuk memastikan bahwa produk yang dihasilkan tidak hanya fungsional secara teknis tetapi juga valid dan sesuai dengan kebutuhan pengguna di lapangan [10].

### 2.2 *Builder Design Pattern*

*Builder Pattern* adalah pola desain kreasi yang bertujuan untuk "memisahkan konstruksi objek yang kompleks dari representasinya sehingga representasi yang berbeda dapat dibuat dengan proses konstruksi yang sama" [11]. Pola ini sangat berguna ketika proses pembuatan objek memerlukan beberapa langkah atau algoritma yang kompleks. Struktur *Builder Pattern* umumnya terdiri dari empat komponen utama: *Director*, *Builder*, *ConcreteBuilder*, dan *Product* [12].

1. ***Product***: Merupakan objek kompleks yang akan dibuat. Dalam kasus ini, *Product* adalah objek PesananAyamGeprek.
2. ***Builder***: Antarmuka (*interface*) yang mendefinisikan langkah-langkah untuk membangun *Product*. Contohnya, setNasi(), setLevelPedas(), addTopping().
3. ***ConcreteBuilder***: Kelas yang mengimplementasikan antarmuka *Builder* dan bertanggung jawab untuk

membangun serta merakit bagian-bagian dari *Product*.

4. ***Director***: Kelas yang mengonstruksi objek dengan menggunakan antarmuka *Builder*. *Director* tidak terikat pada implementasi *ConcreteBuilder* tertentu, sehingga memberikan fleksibilitas.

Keunggulan utama dari pola ini adalah kemampuannya untuk mengisolasi kode konstruksi yang kompleks dan memungkinkan pembuatan objek secara bertahap. Hal ini meningkatkan modularitas dan keterbacaan kode, terutama pada objek yang memiliki banyak atribut opsional [13]. Beberapa penelitian terbaru menunjukkan bahwa penerapan pola desain seperti *Builder* secara signifikan meningkatkan kualitas, pemeliharaan, dan skalabilitas perangkat lunak [14].

### *3 Sistem Pemesanan Mandiri Berbasis Web*

Sistem pemesanan mandiri berbasis web merupakan aplikasi yang memungkinkan pelanggan untuk melakukan proses pemilihan dan penentuan pesanan secara independen tanpa memerlukan interaksi langsung dengan staf penjual [15]. Dalam konteks industri kuliner, sistem semacam ini berperan penting dalam meningkatkan efisiensi operasional dan akurasi pencatatan pesanan. Penelitian di bidang teknologi hospitality menunjukkan bahwa adopsi sistem pemesanan digital secara signifikan mengurangi waktu tunggu pelanggan dan memperkecil frekuensi kesalahan pesanan dibandingkan metode pencatatan manual [1].

Dari perspektif rekayasa perangkat lunak, tantangan utama dalam pengembangan sistem pemesanan dengan tingkat kustomisasi tinggi terletak pada pengelolaan kompleksitas objek pesanan. Sebuah pesanan dengan banyak atribut opsional memerlukan arsitektur yang dapat mengakomodasi variasi tanpa menghasilkan kode yang kaku dan sulit dipelihara [3]. Oleh karena itu, pemilihan pola desain yang tepat menjadi kunci keberhasilan implementasi sistem semacam ini. Penggunaan *framework* PHP native dan MySQL sebagai fondasi teknologi dalam penelitian ini merupakan pilihan yang relevan untuk skala usaha mikro hingga menengah, mengingat kedua teknologi tersebut bersifat open-source, memiliki ekosistem yang matang, serta banyak didukung oleh layanan hosting yang terjangkau [9]. Kombinasi antara teknologi yang ramah biaya dan pendekatan arsitektur yang terstruktur melalui pola desain menjadikan sistem yang dikembangkan lebih berpotensi untuk diadopsi secara nyata oleh pelaku usaha kuliner skala kampus.

## 3. METODE PENELITIAN

Penelitian ini menggunakan metode *Research and Development* (R&D) yang diadaptasi dari model Borg dan Gall. Mengingat keterbatasan sumber daya dan waktu, sepuluh langkah dalam model tersebut disederhanakan menjadi lima tahapan utama yang relevan untuk pengembangan sistem informasi, sebagai berikut:

### 3.1 Analisis Potensi dan Masalah

Tahap ini melibatkan identifikasi masalah yang terjadi pada proses pemesanan ayam geprek secara manual. Analisis dilakukan melalui observasi langsung di beberapa kedai ayam geprek di sekitar UNSIKA. Ditemukan bahwa masalah utama adalah tingginya potensi kesalahan pencatatan pesanan akibat banyaknya variasi menu dan kurangnya standarisasi proses.

### 3.2 Pengumpulan Data dan Studi Literatur

Pada tahap ini, dilakukan pengumpulan data mengenai kebutuhan fungsional dan non-fungsional sistem. Selain itu, dilakukan studi literatur mendalam mengenai metode R&D, *Builder Design Pattern*, serta teknologi yang akan digunakan, aitu PHP *native* dan database MySQL. Tinjauan terhadap penelitian serupa juga dilakukan untuk memahami pendekatan yang sudah ada [15].

### 3.3 Desain Produk

Tahap desain mencakup beberapa aktivitas, yaitu:

- **Desain Arsitektur Sistem:** Merancang arsitektur perangkat lunak dengan *Builder Pattern* sebagai komponen inti untuk mengelola pembuatan objek pesanan.
- **Desain Database:** Merancang skema database menggunakan *Entity-Relationship Diagram* (ERD) untuk menyimpan data produk, varian, dan pesanan.
- **Desain Antarmuka Pengguna (UI/UX):** Merancang *mockup* antarmuka yang intuitif dan mudah

digunakan oleh pelanggan untuk melakukan pemesanan mandiri.

### 3.4 Pengembangan dan Validasi Produk

Produk dikembangkan dalam bentuk aplikasi berbasis web menggunakan bahasa pemrograman PHP *native* dan sistem manajemen database MySQL. Setelah prototipe awal selesai, dilakukan validasi melalui pengujian fungsional dengan metode *black-box*. Pengujian ini bertujuan untuk memastikan bahwa semua fitur, seperti pemilihan nasi, pengaturan level pedas, penambahan *topping*, dan finalisasi pesanan, berfungsi sesuai dengan rancangan.

### 3.5 Revisi Produk dan Implementasi Awal

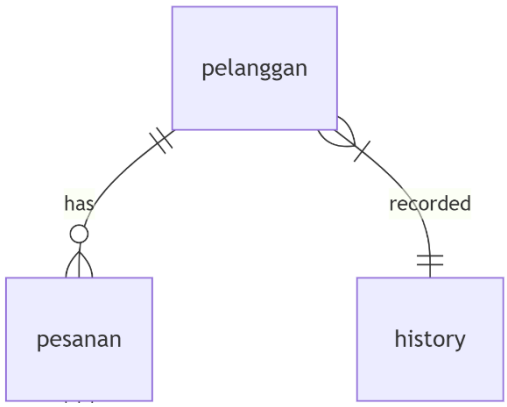
Berdasarkan hasil pengujian pada tahap sebelumnya, dilakukan perbaikan *bug* dan penyempurnaan fitur. Setelah revisi selesai, produk dianggap siap untuk implementasi skala kecil atau sebagai dasar untuk pengembangan lebih lanjut.

Kelima tahapan di atas dilaksanakan secara iteratif dan berkesinambungan. Setiap tahapan memberikan umpan balik yang digunakan untuk menyempurnakan tahapan berikutnya, sehingga produk akhir yang dihasilkan benar-benar memenuhi kebutuhan fungsional yang telah ditetapkan. Pendekatan ini selaras dengan karakteristik model R&D yang menekankan validasi berkelanjutan terhadap produk yang dikembangkan [10]. Secara keseluruhan, kombinasi antara metode R&D dan implementasi *Builder Pattern* memberikan kerangka kerja yang terstruktur dan dapat direproduksi untuk pengembangan sistem pemesanan dengan tingkat variasi produk yang tinggi.

## 4 HASIL DAN PEMBAHASAN

### 4.1 Desain Sistem

Struktur database dirancang untuk mendukung fleksibilitas produk. Tabel utama terdiri dari products, variants (misalnya, jenis nasi, level pedas), options (misalnya, nasi biasa, level 5), dan orders. Desain ini memungkinkan penambahan varian baru tanpa harus mengubah struktur kode secara signifikan. Berikut adalah ERD dari sistem yang diusulkan:




*Gambar 1. Entity Realitionship Diagram*

Arsitektur perangkat lunak berpusat pada implementasi Builder Pattern. Terdapat sebuah interface OrderBuilder yang mendefinisikan metode-metode untuk membangun pesanan, seperti buildNasi(), buildLevel(), dan buildTopping(). Kelas AyamGeprekBuilder adalah implementasi konkret dari interface tersebut. Sebuah kelas Director digunakan untuk mengelola proses konstruksi, memastikan urutan pembuatan pesanan berjalan dengan benar.

### 4.2 Implementasi *Builder Pattern*

Berikut adalah contoh implementasi *Builder Pattern* dalam kode PHP yang disederhanakan untuk merepresentasikan logika inti:

```
// Product
class Pesanan {
    public $nasi = '';
    public $levelPedas = 0;
    public $toppings = [];

    public function show() {
        // Logika untuk menampilkan detail pesanan
    }
}

// Builder Interface
interface PesananBuilder {
    public function setNasi(string $jenisNasi);
    public function setLevel(int $level);
    public function addTopping(string $topping);
    public function getPesanan(): Pesanan;
}

// Concrete Builder
class AyamGeprekBuilder implements PesananBuilder {
```

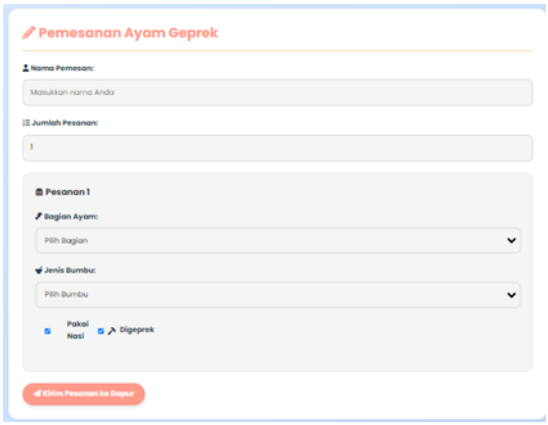
```
    private $pesanan;

    public function __construct() {
        $this->pesanan = new Pesanan();
    }

    public function setNasi(string $jenisNasi)
{
        $this->pesanan->nasi = $jenisNasi;
        return $this;
    }

    public function setLevel(int $level) {
        $this->pesanan->levelPedas = $level;
        return $this;
    }

    public function addTopping(string
$topping) {
        $this->toppings[] = $topping;
        return $this;
    }

    public function getPesanan(): Pesanan {
        return $this->pesanan;
    }
}

// Penggunaan (Client Code)
$builder = new AyamGeprekBuilder();
$pesananKustom = $builder-
>setNasi('Uduk')
                ->setLevel(5)
                ->addTopping('Keju')
                ->getPesanan();
```

*Potongan Kode 1. Implementasi Sederhana Builder Pattern*

Dari potongan kode di atas, terlihat bahwa klien dapat membangun objek Pesanan secara bertahap (*step-by-step*) dengan cara yang sangat deskriptif (*fluent interface*). Pendekatan ini secara signifikan menyederhanakan kode klien dan menyembunyikan logika konstruksi yang kompleks di dalam kelas *Builder*.

### 4.3 Antarmuka Pengguna

Antarmuka pengguna dirancang agar sederhana dan interaktif. Pengguna dipandu melalui beberapa langkah untuk menyesuaikan pesanan mereka.




*Gambar 2. Tampilan Antarmuka Pemilih Varian*

Setiap pilihan yang dibuat oleh pengguna akan memanggil metode yang sesuai pada objek *Builder* di sisi *backend*, secara dinamis membangun objek pesanan hingga siap untuk diselesaikan. Desain antarmuka yang terstruktur ini merupakan manifestasi dari proses konstruksi langkah demi langkah yang difasilitasi oleh *Builder Pattern*.

Alur penggunaan sistem secara keseluruhan dapat dijabarkan sebagai berikut: pertama, pengguna mengakses halaman pemesanan dan memilih jenis nasi yang diinginkan; kedua, pengguna menentukan tingkat kepedasan; dan ketiga, pengguna memilih topping tambahan jika dikehendaki. Setiap langkah ini berkorespondensi langsung dengan pemanggilan metode pada objek *AyamGeprekBuilder* di sisi backend. Setelah semua langkah selesai, metode *getPesanan()* dipanggil untuk menghasilkan objek pesanan final yang kemudian disimpan ke dalam basis data MySQL. Pendekatan bertahap ini tidak hanya memastikan kelengkapan data pesanan, tetapi juga memberikan umpan balik visual kepada pengguna di setiap tahap, sehingga menurunkan risiko kesalahan konfigurasi pesanan secara signifikan.

### 4.4 Hasil Pengujian

Pengujian fungsional dengan metode *black-box* dilakukan pada beberapa skenario pesanan, seperti:

1. Memesan paket dasar tanpa kustomisasi.
2. Memesan dengan mengubah jenis nasi dan level pedas.
3. Memesan dengan menambahkan beberapa *topping*.

4. Mencoba kombinasi input yang tidak valid.

Hasil pengujian menunjukkan bahwa sistem mampu menangani semua skenario dengan benar. Data pesanan yang tersimpan di database akurat dan sesuai dengan kustomisasi yang dipilih oleh pengguna. Dengan demikian, dapat disimpulkan bahwa implementasi *Builder Pattern* berhasil menjawab kebutuhan fungsional sistem untuk mengelola pesanan produk multivarian secara fleksibel dan akurat.

Rincian hasil pengujian fungsional disajikan pada Tabel 1 berikut, yang merangkum setiap skenario uji beserta kondisi masukan, keluaran yang diharapkan, dan status hasil pengujian.

**Tabel 1. Hasil Pengujian Black-Box**

| No | Skenario Uji | Keluaran yang Diharapkan | Keluaran Aktual | Status |
|---|---|---|---|---|
| 1 | Pemesanan paket dasar tanpa kustomisasi | Pesanan tersimpan dengan nilai default | Pesanan tersimpan dengan nilai default | Berhasil |
| 2 | Mengubah jenis nasi menjadi nasi uduk dan level pedas level 7 | Pesanan mencatat nasi=uduk dan levelPedas=7 | Pesanan mencatat nasi=uduk dan levelPedas=7 | Berhasil |
| 3 | Menambahkan tiga topping: keju, telur, sosis | Array topping berisi ketiga item yang dipilih | Array topping berisi ketiga item yang dipilih | Berhasil |
| 4 | Memasukkan level pedas di luar rentang valid (misalnya, level 0 atau level 11) | Sistem menampilkan pesan validasi dan pesanan tidak diproses | Sistem menampilkan pesan validasi dan pesanan tidak diproses | Berhasil |
| 5 | Pemesanan dengan kombinasi penuh: nasi uduk, level 5, topping keju dan telur | Objek pesanan memuat semua atribut sesuai pilihan | Objek pesanan memuat semua atribut sesuai pilihan | Berhasil |

## *5 Analisis Perbandingan Pendekatan*

Untuk mempertegas manfaat penggunaan *Builder Pattern*, dilakukan perbandingan konseptual antara pendekatan konvensional dan pendekatan berbasis pola desain. Pada pendekatan konvensional tanpa pola desain, pembuatan objek pesanan yang memiliki banyak parameter opsional umumnya dilakukan melalui konstruktor dengan banyak argumen (telescoping constructor) atau dengan menggunakan objek yang bersifat mutable dan dapat dimodifikasi langsung. Pendekatan ini menghasilkan kode yang sulit dibaca, rawan kesalahan dalam pengurutan argumen, dan menyulitkan proses pemeliharaan ketika terdapat penambahan varian baru.

Sebaliknya, pendekatan berbasis *Builder Pattern* yang diimplementasikan dalam penelitian ini memisahkan secara tegas proses konstruksi objek dari representasinya. Penambahan varian baru, misalnya jenis nasi baru atau topping tambahan, hanya memerlukan perubahan pada kelas *ConcreteBuilder* tanpa harus mengubah kode di bagian lain sistem. Hal ini secara langsung mendukung prinsip *Open/Closed Principle* dalam SOLID, yaitu kelas terbuka untuk perluasan namun tertutup untuk modifikasi. Dengan demikian, sistem yang dikembangkan memiliki tingkat skalabilitas yang lebih tinggi dibandingkan pendekatan konvensional, serta meminimalkan risiko pengenalan bug baru ketika dilakukan pengembangan fitur di masa mendatang [14].

Selain keunggulan dari sisi arsitektur perangkat lunak, penggunaan *Builder Pattern* juga memberikan dampak positif pada pengalaman pengguna. Alur pemesanan yang terstruktur secara langkah demi langkah, sebagaimana difasilitasi oleh *fluent interface*

pada implementasi kelas *AyamGeprekBuilder*, berkorespondensi secara langsung dengan desain antarmuka yang memandu pengguna melalui tahap pemilihan jenis nasi, penentuan level pedas, hingga penambahan topping. Hal ini menciptakan konsistensi antara logika bisnis dan pengalaman pengguna, yang pada akhirnya berpotensi meningkatkan kepuasan dan kepercayaan pelanggan terhadap sistem pemesanan mandiri [1].

Meskipun demikian, penelitian ini memiliki beberapa keterbatasan yang perlu diakui. Pertama, pengujian dilakukan dalam lingkungan terkontrol dengan data simulasi dan belum melibatkan pengguna akhir secara langsung dalam skenario penggunaan nyata. Kedua, prototipe yang dihasilkan belum diintegrasikan dengan sistem pembayaran digital maupun modul manajemen stok, sehingga siklus transaksi belum sepenuhnya tertutup. Ketiga, implementasi saat ini bersifat monolitik dan belum mempertimbangkan arsitektur berbasis layanan (microservices) yang mungkin diperlukan pada skala yang lebih besar. Keterbatasan-keterbatasan ini membuka ruang bagi penelitian lanjutan untuk memperkuat validitas dan cakupan sistem yang dikembangkan. Terlepas dari keterbatasan tersebut, kontribusi utama penelitian ini terletak pada demonstrasi yang konkret bahwa *Builder Pattern* merupakan pendekatan arsitektur yang tepat sasaran untuk domain permasalahan pemesanan produk multivarian, dan bahwa penerapannya tidak memerlukan teknologi yang kompleks sehingga dapat diakses oleh pengembang dengan tingkat kemampuan yang beragam [12].

## 5 KESIMPULAN

Penelitian ini berhasil mengembangkan sebuah prototipe sistem pemesanan mandiri berbasis web dengan mengimplementasikan *Builder Pattern* menggunakan metode *Research and Development*. Penerapan *Builder Pattern* terbukti sangat efektif untuk mengatasi kompleksitas pembuatan objek pesanan ayam geprek yang memiliki banyak varian. Pola ini memungkinkan konstruksi objek dilakukan secara bertahap, memberikan fleksibilitas tinggi bagi pengguna untuk melakukan kustomisasi, serta meninsgkatkan keterbacaan dan pemeliharaan kode. Sistem yang dihasilkan mampu meminimalkan potensi kesalahan yang sering terjadi pada proses pemesanan manual. Untuk pengembangan selanjutnya, sistem ini dapat ditingkatkan dengan menambahkan fitur pembayaran digital, integrasi dengan sistem manajemen inventaris, notifikasi pesanan *real-time*, serta pengembangan versi aplikasi *mobile* untuk memperluas aksesibilitas dan jangkauan pengguna.

## UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada pihak-pihak terkait yang telah memberi dukungan terhadap penelitian ini.

## DAFTAR PUSTAKA

[1] A. P. Putra dan I. G. A. P. R. Agung, “The Role of Digital Ordering Systems in Enhancing Customer Experience in the Culinary Industry,” *Journal of Hospitality and Tourism Technology*, vol. 14, no. 1, hlm. 45–60, 2023.

[2] S. N. Hasanah dan F. R. Sari, “Analysis of Student Consumption Patterns towards Street Food Vendors around University Campuses,” *Indonesian Journal of Nutrition and Dietetics*, vol. 11, no. 2, hlm. 89–97, 2023.

[3] J. Liu, H. Zhang, dan L. Wei, “A Framework for Managing Product Customization Complexity in E-commerce Platforms,” dalam *Proceedings of the 2022 International Conference on E-Business and E-Government*, 2022, hlm. 112–118.

[4] M. Fowler, *Patterns of Enterprise Application Architecture*. Addison-Wesley, 2002.

[5] E. Gamma, R. Helm, R. Johnson, dan J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994.

[6] Shvets dan Alexander, “Dive Into Design Patterns,” https://refactoring.guru/design-patterns/builder.

[7] W. R. Borg dan M. D. Gall, *Educational Research: An Introduction*. New York: Longman, 1989.

[8] Sugiyono, *Metode Penelitian dan Pengembangan (Research and Development/R&D)*. Bandung: Alfabeta, 2019.

[9] R. A. Sukamto dan M. Shalahuddin, *Rekayasa Perangkat Lunak: Terstruktur dan Berorientasi Objek*. Bandung: Informatika, 2018.
[10] D. A. Saputra dan H. Haryanto, "Development of a Web-Based Academic Information System Using the Research and Development (R&D) Method," *Journal of Information Systems and Technology*, vol. 5, no. 3, hlm. 201–210, 2022.
[11] S. Maharani, "PEMBANGUNAN MODEL DATA SKALA BESAR DENGAN MENGGUNAKAN BUILDER DESIGN PATTERN," *Jurnal Informatika dan Teknik Elektro Terapan*, vol. 13, no. 3, hlm. 322–323, Jul 2025.
[12] K. S. Kumar dan S. Chandrashekar, "A Systematic Review of Creational Design Patterns for Software Maintainability," *Journal of Software Engineering and Applications*, vol. 16, no. 4, hlm. 150–168, 2023.
[13] F. A. Nugraha dan A. B. M. Z. Abidin, "Comparative Analysis of Builder and Factory Method Patterns for Dynamic Object Creation in Game Development," *IEEE Access*, vol. 11, hlm. 25432–25445, 2023.
[14] T. Chen, Y. Sun, dan Z. Ma, "The Impact of Design Patterns on Software Quality: An Empirical Study," dalam *2022 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2022, hlm. 345–355.
[15] I. P. A. E. D. Putra dan N. K. A. Werthi, "Web-Based Food Ordering System Development Using Prototyping Method," *International Journal of Engineering and Emerging Technology*, vol. 7, no. 1, hlm. 56–62, 2022.