BAB I
PENDAHULUAN

1.1. Latar Belakang

Perkembangan teknologi digital telah mengubah perilaku konsumen dalam bertransaksi. Era digital menuntut kemudahan, kecepatan, dan kenyamanan, sehingga terjadi pergeseran besar dari belanja konvensional ke *platform* daring. Konsumen mengharapkan akses 24 jam, alur pembelian yang ringkas, dan pelacakan layanan tanpa batasan geografis. Nilai transaksi e-commerce Indonesia pada Januari sampai Juli 2025 mencapai Rp44,4 triliun yang menegaskan kuatnya pergeseran tersebut [1].

Sistem informasi berperan menyatukan alur kerja dari penerimaan pesanan, pencatatan, hingga pelaporan agar data konsisten, proses lebih cepat, dan informasi mudah ditelusuri. Perancangan yang baik mengurangi ketergantungan pada pencatatan manual, menekan salah *input*, serta meningkatkan visibilitas kinerja harian sehingga keputusan operasional dapat diambil lebih sigap [2].

UMKM berperan penting bagi perekonomian lokal. Pada sektor makanan, ekspektasi kemudahan pesan dan bayar meningkat seiring kebiasaan digital masyarakat. Banyak pelaku sudah hadir secara daring namun masih berhenti pada etalase sehingga transaksi ujung ke ujung belum tertangani konsisten dan kualitas layanan belum seragam [3].

Nusantara Kuliner merupakan usaha kuliner yang menyediakan produk frozen food, misalnya rica-rica bebek yang dikemas secara vakum dalam kemasan siap kirim, serta melayani pemesanan makanan. Nusantara Kuliner sendiri sebenarnya telah memiliki kehadiran digital dasar. Saat ini proses pemesanan, konfirmasi pembayaran, dan pelacakan layanan belum terintegrasi penuh. Pencatatan masih manual sehingga konsistensi data pesanan, transparansi status layanan, dan kemudahan audit belum optimal.

Pada 2025, biaya kanal pihak ketiga menekan margin. Di ShopeeFood terdapat komisi sekitar 15 sampai 20 persen ditambah biaya layanan 3 sampai 5 persen per transaksi [4]. Di Shopee untuk produk di *marketplace* terdapat biaya

proses pesanan Rp1.250 per transaksi yang selesai dan biaya admin yang bervariasi menurut kategori dan status penjual [5]. Langganan sistem kasir komersial menambah biaya tetap, misalnya Moka POS paket Basic Rp299.000 per outlet per bulan [6]. Di sisi lain, kanal milik sendiri dapat berjalan dengan biaya *server* sekitar Rp70.000 per bulan dan biaya transaksi melalui QRIS yaitu 0,7% per transaksi. Adopsi QRIS pada semester pertama 2025 menjangkau 57 juta pengguna dan 39,3 juta merchant yang layak dijadikan dasar pemilihan kanal pembayaran utama [7].

Melihat tingginya biaya operasional *platform* pihak ketiga dan kebutuhan integrasi berbagai kanal transaksi, Nusantara Kuliner memerlukan solusi *website* milik sendiri yang dapat mengakomodasi seluruh kebutuhan bisnis secara terpadu. *Website* tersebut dirancang untuk mendukung pemesanan produk layaknya *marketplace*, pengelolaan katering berlangganan dengan sistem subscription, fungsi *Point of Sale* (POS) untuk transaksi di lokasi usaha, serta pengelolaan operasional yang terpusat melalui satu panel administrasi. Dengan pendekatan ini, efisiensi biaya dapat ditingkatkan, konsistensi data terjaga, dan kontrol penuh atas sistem berada di tangan pelaku usaha sehingga pengembangan fitur dapat disesuaikan dengan kebutuhan tanpa terikat pada keterbatasan *platform* pihak ketiga.

Sistem diusulkan dipisah antara *backend* dan *frontend* agar tanggung jawab pengembangan jelas, perubahan tampilan tidak mengganggu logika bisnis, dan pengembangan antarmuka dapat bergerak cepat mengikuti kebutuhan pengguna [8]. Pemisahan ini membentuk kontrak data yang tegas, meningkatkan ketahanan layanan saat jumlah pengguna bertambah, memungkinkan kerja paralel antar tim, serta mempermudah integrasi dengan berbagai kanal layanan melalui API yang sama [8].

Pengembangan sistem dilakukan secara kolaboratif, di mana *frontend* dan *backend* dipandang sama penting serta saling melengkapi. Pada penelitian ini, ruang lingkup difokuskan pada sisi *backend* karena *backend* berperan menyediakan layanan inti, aturan bisnis, dan kontrak data yang menjadi acuan integrasi antarmuka, termasuk *frontend web* yang dikerjakan oleh peneliti lain [9]. Meskipun implementasi saat ini berupa *web*, rancangan *backend* berbasis *RESTful API* disiapkan agar dapat digunakan kembali oleh kanal lain di masa depan, misalnya

aplikasi *mobile*, tanpa perlu menulis ulang logika bisnis, cukup mengonsumsi *endpoint* API yang sama. Dengan fondasi *backend* yang kuat, proses bisnis seperti autentikasi, manajemen produk, pemesanan, pembayaran, dan pelaporan dapat berjalan konsisten serta mudah diperluas ketika kebutuhan *platform* bertambah.

Backend menjadi tulang punggung layanan yang menjaga konsistensi data dan menyediakan *endpoint* yang stabil bagi seluruh fitur sistem. Pengelolaan data menggunakan basis data relasional dengan pemetaan objek agar skema terjaga rapi dan akses data terkendali. Kinerja baca ditingkatkan melalui mekanisme singgahan pada bagian yang tepat [10]. Siklus rilis diotomatisasi melalui *pipeline* CI/CD yang mencakup proses *build* dan *deployment* ke VPS, sehingga setiap perubahan kode dapat dirilis secara konsisten dengan langkah yang sama, hasilnya dapat dilacak melalui *log*, dan prosedurnya terdokumentasi dengan baik [11].

Untuk memastikan kebutuhan nonfungsional seperti responsivitas dan ketahanan layanan pada berbagai tingkat beban terpenuhi, pengujian performa dilakukan menggunakan k6 sebagai alat *load testing* yang dapat mengotomasi skenario uji dan mengukur respons layanan pada berbagai kondisi pengguna [12]. Skenario pengujian dapat mencakup *load test*, *spike test*, *soak test*, dan *performance test*, dengan metrik utama seperti *response time*, *throughput*, *error rate*, dan *requests per second* untuk menggambarkan kecepatan respons, kapasitas pemrosesan, serta reliabilitas layanan pada kondisi beban yang berbeda [12].

Pemilihan bahasa pemrograman untuk *backend* perlu mempertimbangkan performa layanan, efisiensi CPU dan memori, serta kemudahan *deployment* karena sistem direncanakan berjalan pada VPS dan harus melayani permintaan konkuren dari berbagai kanal layanan. Studi performance *testing* REST API lintas bahasa yang membandingkan Spring Boot (Java), Flask (Python), Express.js (JavaScript), Laravel FrankenPHP (PHP), dan Gin (Golang) dengan 20 *virtual users* selama 10 menit menunjukkan bahwa pada skenario GET 10.000 data, Gin mencatat *response time* rata-rata 84 ms dengan *throughput* 236.99 req/s. Pada skenario yang sama, Express.js mencatat *response time* 174 ms dengan *throughput* 144.08 req/s, sedangkan Spring Boot membutuhkan memori jauh lebih besar yaitu 2082.52 MB dibanding Gin yang sebesar 193.99 MB [13]. Pada skenario GET 100.000 data, Gin tetap memberikan *response time* 742 ms dengan *throughput* 26.93 req/s.

Sebaliknya, Spring Boot meningkat menjadi *response time* 48 s dengan *error rate* 99% dan penggunaan memori 2273.91 MB pada konfigurasi uji tersebut [56]. Selain metrik performa layanan, kajian efisiensi lintas 27 bahasa melaporkan Go memiliki konsumsi energi 3.23× dan waktu eksekusi 2.83× terhadap baseline C, yaitu 57.86 J dan 2019.26 ms. Go juga memiliki penggunaan memori 1.05× terhadap baseline Pascal sebesar 65.96 MB. Sebagai pembanding, JavaScript tercatat 4.45× lebih boros energi dan 6.52× lebih lambat dari baseline tersebut [14]. Berdasarkan pertimbangan tersebut, pemilihan bahasa diarahkan pada solusi yang responsif, stabil saat trafik meningkat, dan tetap efisien dalam pemakaian sumber daya.

Berdasarkan hasil perbandingan performa yang mencakup *response time* dan *throughput* serta efisiensi penggunaan CPU dan memori pada studi terdahulu, dan dengan mempertimbangkan kebutuhan layanan yang stabil pada lingkungan VPS, penelitian ini menetapkan Golang sebagai bahasa utama dalam pengembangan *backend* Nusantara Kuliner. Pemilihan Golang didukung oleh karakteristiknya yang sederhana, memiliki performa eksekusi tinggi, serta penggunaan sumber daya yang relatif efisien sehingga sesuai untuk layanan yang harus responsif dan mudah dipelihara pada *server* dengan spesifikasi terbatas. Golang juga menyediakan dukungan bawaan untuk pengujian dan pengelolaan dependensi yang membantu menjaga kualitas rilis dan konsistensi pengembangan. Untuk membangun *RESTful API*, penelitian ini menggunakan GoFiber karena *framework* ini ringan dan memiliki *model* penanganan *request* yang jelas melalui *routing* dan *middleware*, sehingga memudahkan penerapan aturan bisnis secara terstruktur. Kombinasi Golang dan GoFiber mendukung pembuatan *endpoint* yang lebih cepat, menjaga keterbacaan kode, serta mempermudah pemeliharaan ketika fitur berkembang [15].

Sistem *backend* akan menggunakan basis data relasional PostgreSQL untuk menjaga integritas data. Akses data dibantu dengan ORM agar konsistensi skema dan portabilitas terjaga [11]. Otomasi rilis memanfaatkan CI dan CD menggunakan GitLab dengan Docker agar proses pembangunan, pengujian, dan penyebaran berjalan konsisten, dan aplikasi akan dideploy di VPS milik sendiri untuk menjaga

kendali operasional dan biaya [11]. Optimalisasi jalur baca memanfaatkan singgahan agar respons tetap cepat pada beban yang meningkat [10].

Clean Architecture adalah pendekatan arsitektur perangkat lunak yang menempatkan aturan bisnis sebagai pusat dan mengarahkan ketergantungan kode ke lapisan yang lebih dalam, sedangkan detail seperti *framework* dan basis data berada di lapisan terluar [16]. Pada konteks *backend*, pemisahan ini membuat logika layanan tetap independen dari mekanisme *input output*, sehingga inti sistem lebih mudah diuji tanpa bergantung pada infrastruktur dan perubahan teknologi tidak memaksa perubahan pada aturan bisnis [16]. Pendekatan ini dipilih karena lebih menjaga aturan bisnis tetap stabil saat terjadi perubahan teknologi dibanding pola yang menjadikan *framework* sebagai pusat struktur aplikasi [16]. Dalam implementasi API menggunakan Golang, pembagian *layer* seperti handler, *use case*, dan *repository* membantu membatasi dampak perubahan sekaligus memperjelas kontrak antarkomponen ketika fitur bertambah [17]. Dengan demikian *Clean Architecture* menjadi dasar yang kuat untuk membangun *backend* Nusantara Kuliner yang mudah dipelihara dan siap berkembang seiring perubahan kebutuhan [17].

Proses pengembangan dilakukan secara iteratif menggunakan Scrum dengan membagi pekerjaan ke dalam beberapa sprint yang memiliki tujuan dan keluaran yang jelas. Kebutuhan sistem dihimpun dan diprioritaskan ke dalam *product backlog*, kemudian dipilih menjadi sprint backlog pada awal sprint agar ruang lingkup pekerjaan terukur. Selama sprint berlangsung dilakukan pemantauan progres secara rutin untuk memastikan hambatan cepat teridentifikasi, sedangkan hasil pekerjaan dievaluasi pada akhir sprint melalui review bersama stakeholder untuk memvalidasi kesesuaian fitur dengan kebutuhan. Setelah itu dilakukan *sprint retrospective* untuk menilai efektivitas proses dan menetapkan perbaikan pada sprint berikutnya sehingga kualitas rilis tetap konsisten dan meningkat dari iterasi ke iterasi [18].

Berdasarkan latar belakang permasalahan tersebut, penelitian ini bertujuan untuk merancang dan membangun *Backend Website* Nusantara Kuliner Menggunakan Golang *Clean Architecture*. Diharapkan dengan adanya sistem ini mampu mengintegrasikan alur pemesanan, pembayaran, dan pelacakan layanan

secara konsisten, mengurangi ketergantungan pada *platform* pihak ketiga, serta menyediakan fondasi yang efisien dan mudah berkembang untuk mendukung pertumbuhan usaha Nusantara Kuliner.

1.2. Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan, rumusan masalah penelitian ini adalah bagaimana merancang dan membangun *backend website* Nusantara Kuliner menggunakan Golang berbasis *Clean Architecture* dengan penerapan metode Scrum.

1.3. Batasan Masalah

Batasan masalah digunakan agar penelitian lebih terarah dan terorganisasi dengan baik sesuai apa yang diinginkan. Adapun batasan masalah dalam penelitian ini sebagai berikut:

1. Penelitian ini berfokus pada pembuatan *RESTful API* dan *Website* Admin.
2. Bahasa Pemrograman yang digunakan adalah Golang dengan menggunakan *framework* GoFiber
3. Basis data yang digunakan adalah PostgreSQL.
4. Pengujian sistem dilakukan menggunakan metode *black box testing* untuk menguji fungsionalitas API.
5. Menggunakan k6 untuk menguji kebutuhan nonfungsional.

1.4. Tujuan Penelitian

Berdasarkan rumusan masalah diatas, maka tujuan penelitian ini adalah sebagai berikut:

1. Merancang arsitektur *backend website* Nusantara Kuliner menggunakan prinsip *Clean Architecture* dengan Golang yang mampu memisahkan concerns antar *layer* dan memenuhi kebutuhan fungsional sistem.
2. Membangun sistem *backend website* Nusantara Kuliner dengan menerapkan metode Scrum untuk memastikan proses pengembangan berjalan terstruktur, iteratif, dan responsif terhadap perubahan kebutuhan.

1.5. Manfaat Penelitian

Adapun manfaat yang diharapkan dapat diperoleh dari hasil pelaksanaan penelitian ini adalah sebagai berikut:

1. Meningkatkan efisiensi operasional Nusantara Kuliner melalui *backend* yang mampu mengelola proses bisnis secara terintegrasi.
2. Menyediakan rancangan teknis *backend* berbasis Golang dengan *Clean Architecture* yang modular dan mudah dikembangkan sebagai acuan sistem serupa.
3. Menghasilkan dokumentasi lengkap arsitektur dan proses bisnis sebagai referensi pengembangan lanjutan dan penelitian sistem transaksi UMKM.

1.6. Sistematika Penulisan

Sistematika penulisan berfungsi sebagai panduan dalam menyusun laporan secara terstruktur dan fokus pada topik penelitian yang dibahas. Hal ini bertujuan agar penulisan skripsi tetap konsisten dengan tujuan yang ingin dicapai. Dengan adanya sistematika penulisan, proses penyusunan laporan penelitian ini memiliki acuan atau kerangka yang jelas, sehingga keseluruhan isi dapat disusun sesuai rencana awal. Sistematika penulisan ini dibagi ke dalam beberapa BAB yaitu:

BAB I PENDAHULUAN

Bab ini berisi gambaran umum penelitian yang mencakup latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, serta sistematika penulisan.

BAB II TINJAUAN PUSTAKA

Bab ini berisi dasar teori dan tinjauan penelitian terdahulu sebagai bahan perbandingan dengan penelitian yang sedang dilakukan, landasan teori yang relevan dengan isu yang akan dibahas, serta alat atau *tools* yang akan digunakan dalam penelitian ini.

BAB III METODOLOGI PENELITIAN

Bab ini berisi tentang setiap langkah yang ditempuh untuk mencapai tujuan penelitian, termasuk pengumpulan data, analisis kebutuhan sistem, perancangan arsitektur, penerapan metodologi Scrum, implementasi, serta penyusunan laporan.

BAB IV HASIL DAN PEMBAHASAN

Bab ini memuat hasil dari setiap tahapan metodologi yang telah dilakukan serta pembahasan mengenai proses perancangan, pengembangan, dan pengujian sistem.

BAB V KESIMPULAN DAN SARAN

Bab ini berisikan kesimpulan dari hasil penelitian yang telah dilakukan serta saran yang dapat digunakan sebagai bahan pertimbangan untuk pengembangan sistem di masa mendatang.

DAFTAR PUSTAKA

Bab ini memuat hasil dari setiap tahapan metodologi yang telah dilakukan serta pembahasan mengenai proses perancangan, pengembangan, dan pengujian sistem.

LAMPIRAN

Bab lampiran berisi data atau pelengkap yang menunjang dalam pembuatan skripsi.