BAB II
KAJIAN LITERATUR

2.1 Antrean

Antrean adalah kondisi ketika sejumlah orang menunggu untuk mendapatkan layanan karena kapasitas pelayanan lebih kecil dibandingkan jumlah permint unsigned aan. Antrean sering terjadi pada tempat pelayanan umum seperti rumah sakit, bank, dan restoran. Pengelolaan antrean yang tidak baik dapat menyebabkan waktu tunggu lama, kerumunan, dan menurunnya kepuasan pelanggan. Oleh karena itu, sistem antrean perlu diatur dengan baik agar pelayanan menjadi lebih terstruktur dan efisien. Sistem antrean berbasis teknologi informasi terbukti mampu meningkatkan kualitas layanan dan kepuasan pelanggan karena proses menjadi lebih cepat dan terorganisir [19] [20].

Gambar 2.1 Antrean Manual Restoran *[21]*

Pada gambar 2.1 dalam layanan restoran, antrean terjadi ketika kedatangan pelanggan lebih cepat daripada kemampuan petugas melayani, sehingga pelanggan harus menunggu dan barisan menjadi panjang. Karena itu, pengaturan jumlah loket atau petugas dan cara melayani sangat menentukan seberapa lama orang menunggu. Bukti lapangan di sebuah restoran cepat saji di Batam menunjukkan bahwa dua *server* dengan aturan "yang datang duluan, dilayani duluan" (FCFS) mampu menekan waktu tunggu rata-rata hingga ≈4,94 menit sambil menjaga utilisasi ≈84%, sehingga pelayanan tetap cepat dan efisien [22]. Dari sisi pemodelan, studi simulasi pada antrean restoran memperlihatkan bahwa jika sistem dibiarkan tanpa perbaikan, waktu tunggu

rata-rata dapat mencapai ±24,49 menit; namun dengan skenario perbaikan (misalnya menambah petugas atau menyeimbangkan beban antar loket), waktu tunggu dan panjang antrean turun nyata sehingga pelanggan merasa lebih cepat dilayani [23]. Untuk situasi yang lebih kompleks misalnya ada beragam jenis pelanggan dan beberapa jenis loket penelitian teori antrean menunjukkan bahwa meski FCFS terasa adil, cara menghubungkan jenis pelanggan ke loket harus cermat. Jika koneksi dibuat terlalu banyak tanpa perhitungan, waktu tunggu malah bisa naik (Fenomena mirip paradoks Braess), sehingga restoran perlu memantau kedatangan secara *real-time*, menyesuaikan penugasan petugas, dan menambah kapasitas pada waktu yang tepat [24]. Dengan demikian, penerapan sistem antrean digital pada restoran bertujuan meningkatkan efisiensi pelayanan dan kenyamanan pelanggan.

Teori antrean (*Queueing theory)* digunakan untuk menjelaskan bagaimana permint unsigned aan layanan diproses dalam sistem komputer, seperti *server*, jaringan, atau aplikasi digital. Teori ini membantu menganalisis proses kedatangan pelanggan, waktu pelayanan, dan urutan pemrosesan permint unsigned aan agar sistem dapat berjalan secara efisien. Salah satu metode yang sering digunakan adalah *First Come First Served* (FCFS), yaitu metode yang melayani permint unsigned aan berdasarkan urutan kedatangan sehingga pelanggan diproses secara adil dan transparan. Penelitian [25] menunjukkan bahwa model antrean seperti M/M/1 sering digunakan untuk menganalisis kinerja sistem dengan metode FCFS karena dapat mengukur waktu tunggu, panjang antrean serta efisiensi layanan secara matematis. Dalam kajian teori penjadwalan komputasi, disiplin antrean seperti FCFS juga menjadi dasar penting dalam memahami bagaimana kebijakan penjadwalan mempengaruhi keterlambatan layanan dan performa sistem komputasi [26]. FCFS juga banyak digunakan dalam sistem layanan digital karena memberikan mekanisme pengelolaan permint unsigned aan yang sederhana, transparan, dan mudah diterapkan dalam berbagai sistem antrean modern [27].

2.2 Metode *First Come First Serve*d (FCFS)

Metode *First Come First Served* (FCFS) merupakan metode penjadwalan layanan yang memprioritaskan pelanggan berdasarkan urutan kedatangan. Pelanggan yang datang terlebih dahulu akan dilayani terlebih dahulu tanpa adanya prioritas khusus. Tujuan penerapan metode ini adalah menciptakan sistem antrean yang adil, transparan, dan mudah dipahami oleh pelanggan [28] [1] [29].

Tahapan atau alur kerja metode FCFS dimulai ketika setiap pelanggan atau proses yang datang akan dimasukkan ke dalam antrean sesuai dengan waktu kedatangannya. Sistem kemudian akan memanggil entitas yang berada di posisi paling depan dalam antrean untuk dilayani terlebih dahulu. Setelah proses pelayanan selesai, entitas tersebut akan keluar dari sistem

dan posisi terdepan antrean akan ditempati oleh entitas berikutnya. Proses ini berlangsung secara berulang hingga seluruh pelanggan dalam antrean telah dilayani. Dengan demikian, tahapan utama dalam metode FCFS meliputi:

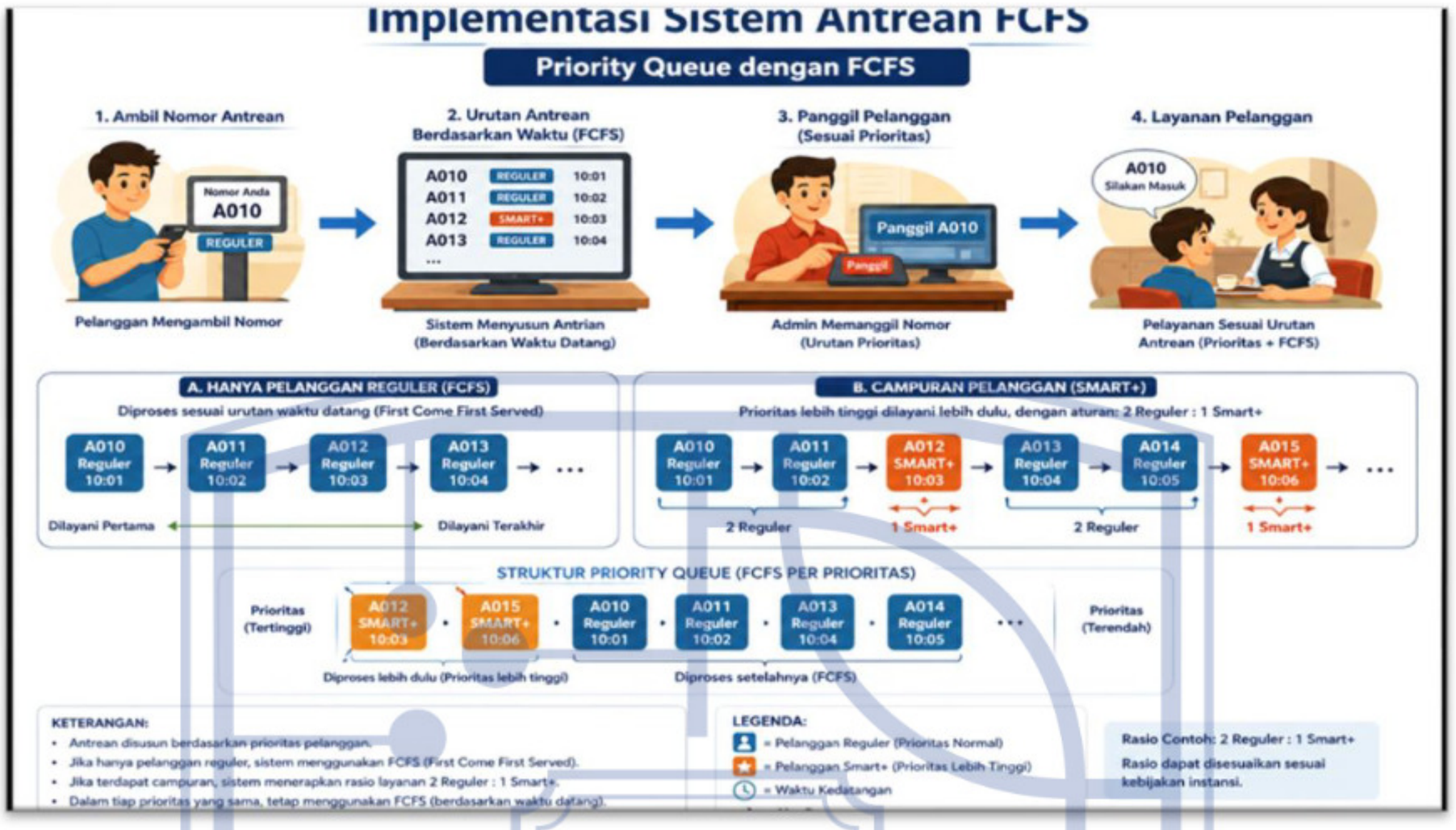
1. Pelanggan atau proses masuk ke dalam antrean berdasarkan waktu kedatangan.
2. Sistem memilih entitas pertama dalam antrean untuk dilayani.
3. Pelayanan dilakukan hingga selesai tanpa int unsigned erupsi.
4. Sistem melanjutkan pelayanan ke entitas berikutnya sampai antrean kosong atau habis [30] [31]

Meskipun metode FCFS bersifat adil dan mudah diterapkan, terdapat kondisi di mana sistem antrean memerlukan penanganan khusus terhadap pelanggan yang memiliki kebutuhan berbeda. Untuk mengakomodasi hal tersebut, sistem Antre menerapkan pendekatan FCFS yang diperkuat dengan algoritma *Priority Queue* (PQ), di mana entitas dengan prioritas lebih tinggi akan diproses terlebih dahulu dibandingkan entitas dengan prioritas lebih rendah, dan apabila terdapat dua atau lebih entitas dengan tingkat prioritas yang sama maka penentuan urutan pelayanan dikembalikan kepada prinsip FCFS yaitu entitas yang lebih dahulu tiba akan dilayani lebih dahulu [14].

Penggabungan kedua metode ini membentuk pendekatan *hybrid* antara *First Come First Served* (FCFS) dengan *Priority Queue*, di mana ketika pelanggan VIP memasuki sistem, nomor antreannya akan disisipkan ke dalam antrian FCFS sesuai tingkat prioritasnya sehingga pelayanan tetap terstruktur tanpa mengganggu urutan pelanggan yang sedang berjalan [14], dan pendekatan ini terbukti mampu menghasilkan penjadwalan yang lebih teratur karena proses dengan prioritas lebih tinggi diutamakan sementara proses dengan prioritas sama diselesaikan berdasarkan urutan kemunculannya sesuai prinsip FCFS [15]. Dengan demikian, penerapan algoritma *Priority Queue* dalam sistem Antre bertujuan memastikan pelanggan VIP mendapatkan layanan yang lebih cepat sekaligus tetap menjaga keadilan bagi pelanggan reguler yang telah mengambil nomor antrean terlebih dahulu, sehingga efisiensi dan kenyamanan layanan secara keseluruhan dapat ditingkatkan [14].

Pada penelitian ini, implementasi *Priority Queue* tidak menerapkan prioritas mutlak, melainkan menggunakan aturan bisnis (*business rule*) berupa rasio pelayanan 2 pelanggan reguler : 1 pelanggan Antre Smart+. Rasio tersebut dirancang sebagai kompromi antara pemberian keuntungan kepada pelanggan prioritas dan menjaga keadilan pelayanan bagi pelanggan reguler. Dengan demikian, pelanggan Antre Smart+ tetap memperoleh prioritas layanan tanpa menyebabkan pelanggan reguler mengalami waktu tunggu yang terlalu lama

(*starvation*). Pemilihan rasio ini merupakan kebijakan implementasi sistem yang ditetapkan peneliti karena teori *Priority Queue* sendiri tidak menentukan rasio pelayanan tertentu [32].



Gambar 2.2 Implementasi Antrean FCFS *[16]*

Gambar 2.2 menunjukkan implementasi mekanisme *Priority Queue* dengan prinsip FCFS pada sistem antrean digital. Setiap pelanggan yang mengambil nomor antrean melalui aplikasi akan dicatat berdasarkan waktu pengambilan dan status kategorinya. Sistem secara otomatis menyusun antrean berdasarkan prioritas kategori terlebih dahulu, kemudian mengurutkan pelanggan dalam kategori yang sama berdasarkan *timestamp* reservasi menggunakan prinsip FCFS. Ketika Admin menekan tombol "proses", sistem akan memanggil pelanggan VIP terlebih dahulu (jika ada), diikuti oleh pelanggan reguler sesuai urutan kedatangan. Implementasi ini memastikan layanan Antre Smart+ tetap mendapat prioritas sambil mempertahankan keadilan FCFS di dalam setiap kelompok pelanggan.

2.3 Aplikasi

Aplikasi adalah perangkat lunak yang digunakan untuk membantu pelanggan menyelesaikan suatu pekerjaan secara otomatis. Dalam sistem antrean digital, aplikasi dibagi menjadi dua yaitu aplikasi *mobile* untuk pelanggan dan aplikasi web untuk Admin atau pihak restoran. Sistem informasi yang baik dapat meningkatkan kepuasan pelanggan karena layanan menjadi lebih cepat, transparan, dan mudah diakses [19]. Tingkat kepuasan pelanggan juga dapat diukur menggunakan metode *Customer Satisfaction Index* (CSI) untuk mengetahui kualitas layanan yang diberikan oleh sistem [33].

Pada penelitian ini, aplikasi yang dikembangkan terdiri dari aplikasi *mobile* dan aplikasi web yang saling terint unsigned egrasi dalam satu basis data. Aplikasi *mobile* digunakan oleh pelanggan untuk mengambil nomor antrean, melihat posisi antrean, serta menerima notifikasi, sedangkan aplikasi web digunakan oleh Admin atau Karyawan untuk mengelola data antrean, memanggil pelanggan, dan melihat riwayat pelayanan. Int unsigned egrasi kedua aplikasi ini bertujuan untuk menciptakan sistem antrean digital yang efisien serta mempermudah proses pengelolaan layanan di restoran [34] [35].

2.3.1 Aplikasi *Mobile*

Aplikasi *mobile* merupakan aplikasi yang berjalan pada perangkat *smartphone* dan digunakan untuk memberikan kemudahan akses layanan kepada pelanggan secara langsung. Pada penelitian ini, aplikasi *mobile* digunakan oleh pelanggan untuk melakukan proses antrean secara digital tanpa harus datang ke lokasi, melihat estimasi waktu tunggu, serta menerima notifikasi ketika nomor antrean mendekati giliran.

Aplikasi *mobile* pada penelitian ini dikembangkan menggunakan Flutter dengan bahasa pemrograman Dart karena mampu membangun aplikasi *multiplatform* dengan satu basis kode, menghasilkan tampilan yang konsisten, serta memiliki performa yang cepat [36]. Proses pengembangan dilakukan menggunakan Visual Studio Code sebagai kode editor karena ringan, mendukung ekstensi Flutter, serta memiliki fitur *debugging* yang memudahkan pengembang dalam menulis dan menguji kode [37] [34]. Data antrean disimpan pada MySQL yang dihubungkan melalui *API* sehingga data dapat diakses secara *real-time* dan terint unsigned egrasi dengan aplikasi *web* [35] [29]. Pembuatan teknologi ini bertujuan untuk memberikan kemudahan bagi pelanggan dalam melakukan proses antrean serta meningkatkan efisiensi pelayanan restoran.

Android adalah sistem operasi untuk perangkat *mobile* seperti *smartphone* dan tablet yang dikembangkan berdasarkan kernel Linux dan bersifat *open-source*, sehingga dapat digunakan dan dikembangkan oleh banyak pengembang aplikasi. Android menyediakan berbagai komponen penting seperti kernel Linux untuk mengelola perangkat keras, *Android Runtime* untuk menjalankan aplikasi, serta *application framework* yang membantu pengembang membuat aplikasi dengan mudah. Dengan struktur ini, Android mampu menjalankan berbagai aplikasi dan mengelola sumber daya perangkat seperti memori, jaringan, dan sensor secara efisien. Karena fleksibilitas dan kemudahan pengembangnya, Android menjadi salah satu sistem operasi yang paling banyak digunakan dalam pengembangan aplikasi *mobile* di berbagai bidang teknologi dan layanan digital [38] [39] [40].

Pada penelitian ini, aplikasi *mobile* yang dikembangkan menggunakan Flutter dengan bahasa Dart menargetkan Android versi 8.0 (Oreo) ke atas. Pemilihan Android sebagai platform didasarkan pada tingginya pangsa pasar Android di Indonesia, sehingga aplikasi dapat menjangkau lebih banyak pelanggan [41] [42]. Flutter menyediakan antarmuka pengembangan yang memungkinkan pembuatan aplikasi Android dengan performa tinggi karena kode dikompilasi langsung ke kode mesin *native*, bukan diint unsigned erpretasi, sehingga animasi dan int unsigned eraksi antarmuka berjalan dengan lancar dan responsif [8] [27].

Dalam pengembangan aplikasi *mobile*, komunikasi antara aplikasi klien dengan *server* dilakukan melalui *REST API* (*Representational State Transfer Application Programming Int unsigned erface*), yaitu antarmuka pemrograman aplikasi yang menggunakan gaya arsitektur REST untuk memungkinkan sistem yang berbeda saling berkomunikasi melalui protokol HTTP. REST API menggunakan metode standar seperti GET, POST, PUT, dan DELETE untuk melakukan operasi pada sumber daya, serta mengembalikan data dalam format JSON yang ringan dan mudah diproses oleh berbagai platform, termasuk aplikasi *mobile*. Karakteristik utama REST API meliputi *statelessness* di mana setiap permint unsigned aan bersifat independen dan tidak bergantung pada sesi sebelumnya, arsitektur *client-server* yang memisahkan antarmuka pelanggan dengan penyimpanan data, serta kemampuan *caching* untuk meningkatkan performa [43]. Penelitian menunjukkan bahwa REST API merupakan gaya arsitektur yang paling banyak digunakan untuk merancang *API*, terutama pada platform *mobile*, karena kemudahan int unsigned egrasi dan kesederhanaan pelanggan dibandingkan dengan protokol lain seperti SOAP [44]. Pada penelitian ini, REST API digunakan pada aplikasi *mobile* berbasis Flutter untuk mengambil dan mengirim data antrean ke *server* secara *real-time*, sehingga pelanggan dapat memantau posisi antrean dan menerima notifikasi tanpa penundaan. Pendekatan ini memungkinkan int unsigned egrasi data yang mulus antara aplikasi *mobile* pelanggan dengan aplikasi Admin dalam satu basis data MySQL yang tersentralisasi [45].

2.3.2 Aplikasi Web

Aplikasi web adalah perangkat lunak yang berjalan melalui browser int unsigned ernet dan dapat diakses menggunakan jaringan tanpa harus diinstal secara langsung pada perangkat. Aplikasi web dikembangkan menggunakan teknologi seperti HTML, CSS, dan JavaScript serta didukung oleh *server* yang memproses permint unsigned aan pelanggan dan mengelola data. Sistem ini menggunakan *client-server*, yang memungkinkan pelanggan berint unsigned eraksi melalui browser sementara proses pengolahan data dilakukan di *server* sehingga aplikasi dapat diakses dari berbagai perangkat yang terhubung ke int unsigned ernet [46].

Dalam pengembangan sistem informasi, aplikasi web banyak digunakan karena mampu menyediakan layanan secara *online*, fleksibel, dan mudah diperbarui. Aplikasi web biasanya terdiri dari beberapa komponen utama seperti antarmuka pelanggan (*frontend)* yang berint unsigned eraksi dengan pelanggan, *server* aplikasi *(backend)* yang memproses logika sistem, serta basis data yang menyimpan informasi. Struktur ini memungkinkan aplikasi web untuk mengelola data, menjalankan proses bisnis, serta memberikan layanan digital kepada banyak pelanggan secara efisien [47].

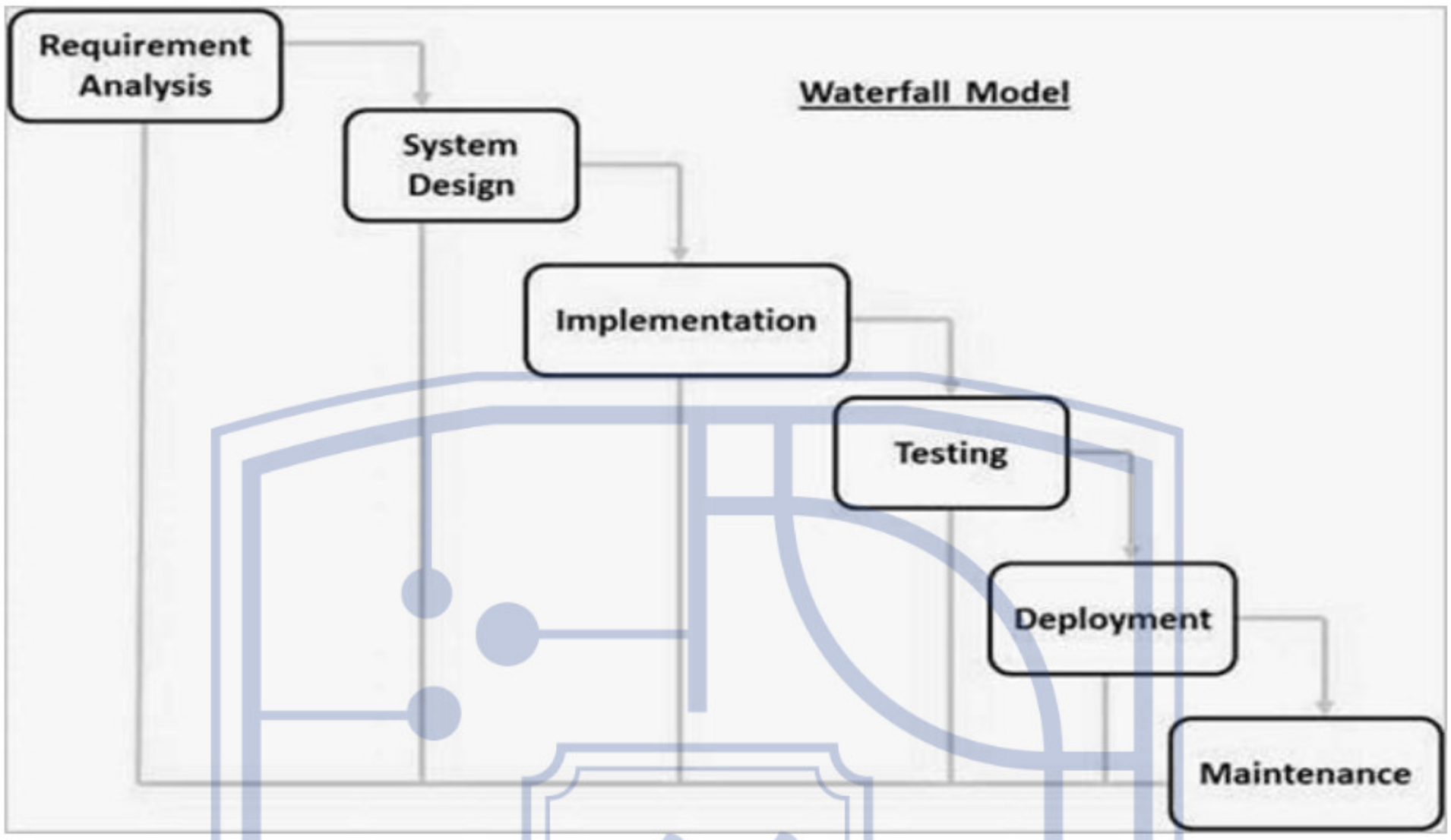
Seiring perkembangan teknologi, aplikasi web kini dirancang menggunakan arsitektur yang lebih modern seperti *framework* dan sistem modular untuk meningkatkan kinerja dan kemudahan pengembangan. Pendekatan ini memungkinkan aplikasi web menjadi lebih skalabel, mudah dipelihara, serta mampu menangani banyak pelanggan secara bersamaan, sehingga banyak digunakan dalam berbagai sistem informasi dan layanan digital saat ini [48].

Int unsigned egrasi data antara aplikasi web dan aplikasi *mobile* dalam sistem antrean digital pada penelitian ini dilakukan menggunakan mekanisme REST API. Int unsigned egrasi data merupakan proses penggabungan data dari berbagai sumber atau platform yang berbeda sehingga dapat diakses dan dikelola secara terpadu dalam satu sistem. Melalui REST API, aplikasi yang digunakan oleh Admin dapat mengirimkan dan menerima data antrean dari basis data MySQL yang sama dengan yang diakses oleh aplikasi *mobile* pelanggan, sehingga perubahan data yang terjadi pada satu platform akan langsung tercermin pada platform lainnya secara *real-time* [43]. Penelitian terdahulu membuktikan bahwa REST API berhasil diimplementasikan sebagai solusi int unsigned egrasi data antara sistem yang berbeda platform dengan menunjukkan waktu respons yang rendah dan pelanggan CPU yang efisien dalam pengiriman dan penerimaan data berformat JSON [44]. Dalam konteks sistem antrean digital, pelanggan REST API pada aplikasi memungkinkan Admin untuk memantau, memanggil, dan memperbarui status antrean secara langsung melalui browser, dengan data yang tersinkronisasi secara otomatis dengan aplikasi pelanggan di perangkat *mobile* [45].

2.4 Metodologi *Waterfall*

Metodologi *Waterfall* merupakan metode pengembangan sistem yang dilakukan secara berurutan mulai dari tahap analisis kebutuhan, perancangan, implementasi, pengujian, hingga pemeliharaan. Setiap tahap harus diselesaikan terlebih dahulu sebelum masuk ke tahap berikutnya sehingga proses pengembangan menjadi lebih terstruktur dan terdokumentasi dengan baik. Metode ini banyak digunakan pada pengembangan sistem informasi karena mudah

dipahami, memiliki alur kerja yang jelas, serta sesuai untuk kebutuhan sistem yang sudah ditentukan sejak awal [49].



Gambar 2.3 Metodologi *Waterfall [50]*

Gambar 2.3 merupakan tahapan metodologi *Waterfall* dengan proses pengembangan sistem dilakukan secara bertahap dan berurutan, yaitu:

1. *Requirement Analysis*, yaitu tahap untuk mengumpulkan dan memahami kebutuhan pelanggan dengan cara mengidentifikasi masalah, melakukan wawancara, atau mengumpulkan data sehingga kebutuhan sistem dapat dirumuskan dengan jelas.
2. *System Design*, yaitu tahap membuat rancangan sistem berdasarkan kebutuhan tadi, termasuk merancang alur kerja, desain tampilan, dan struktur *database* agar proses pembuatan sistem lebih mudah dan terarah.
3. *Implementation*, yaitu mengubah desain menjadi program nyata dengan membuat modul-modul kecil sesuai fungsi masing-masing dan memastikan setiap modul berjalan dengan baik.
4. *Testing*, yaitu tahap menggabungkan semua modul menjadi satu sistem lengkap, memeriksa apakah terdapat kesalahan, dan memastikan sistem sudah bekerja sesuai kebutuhan pelanggan sebelum digunakan secara resmi.
5. *Deployment*, yaitu tahap menerapkan sistem agar bisa digunakan oleh pelanggan dalam lingkungan sebenarnya.

6. *Maint unsigned enance*, yaitu pemeliharaan untuk memperbaiki *bug*, menambah perbaikan kecil, dan melakukan penyesuaian agar sistem tetap berjalan stabil dan sesuai kebutuhan pelanggan dari waktu ke waktu.

Penelitian ini menggunakan metodologi *Waterfall* yang dibatasi hingga tahapan pengujian (*Testing).* Tahapan yang dilakukan meliputi analisis kebutuhan untuk mengidentifikasi fitur dan kebutuhan sistem, perancangan sistem untuk membuat desain arsitektur dan antarmuka, implementasi untuk membangun aplikasi berbasis *mobile* dan web , serta pengujian untuk memastikan sistem berjalan sesuai dengan fungsi yang diharapkan. Pembatasan hingga tahap pengujian dilakukan karena berfokus pada proses perancangan, pembangunan dan validasi sistem yang dikembangkan. Selain itu, tahap pemeliharaan (*Maint unsigned enance*) tidak termasuk dalam cakupan penelitian karena memerlukan waktu penerapan sistem secara berkelanjutan dalam jangka waktu yang lebih panjang, sehingga berada di luar batasan waktu dan ruang lingkup penelitian skripsi.

2.5 Unified Modeling Language (UML)

Unified Modeling Language (UML) merupakan bahasa pemodelan visual standar yang digunakan untuk melakukan spesifikasi, visualisasi, serta dokumentasi arsitektur perangkat lunak melalui berbagai jenis diagram. UML berfungsi untuk menggambarkan struktur dan perilaku sistem sehingga pengembang dan pemangku kepentingan memiliki pemahaman yang sama mengenai bagaimana logika sistem akan bekerja sebelum proses pengkodean dilakukan. Tujuan pelanggan UML adalah mendefinisikan kebutuhan sistem, mengidentifikasi aktor yang terlibat, serta memetakan int unsigned eraksi antara pelanggan dan sistem secara terstruktur agar kesalahan perancangan dapat diminimalkan sejak tahap awal pengembangan [20].

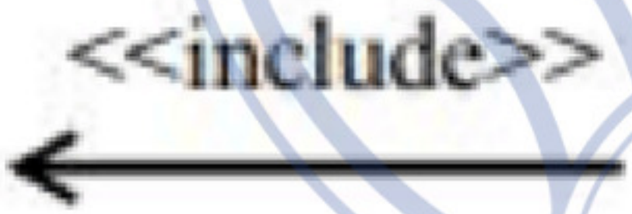
Dalam penelitian ini, UML yang digunakan hanya terdiri dari *Use Case* Diagram dan *Activity Diagram*. *Use Case* Diagram digunakan untuk menggambarkan hubungan antara aktor dengan sistem antrean digital, sedangkan *Activity Diagram* digunakan untuk menggambarkan alur proses pengambilan antrean hingga pemanggilan pelanggan. Pelanggan diagram ini bertujuan agar proses perancangan sistem lebih terarah sebelum masuk ke tahap pengembangan aplikasi [51] [52].

2.5.1 *Use Case* Diagram

Use Case Diagram merupakan diagram yang digunakan untuk menggambarkan int unsigned eraksi antara aktor dengan sistem serta fungsi-fungsi yang tersedia pada sistem. Diagram ini membantu dalam menentukan kebutuhan sistem berdasarkan sudut pandang

pelanggan seperti pelanggan dan Admin pada sistem antrean digital. Berikut tampilan pada Tabel 2.1 dibawah ini.

Tabel 2.1 Simbol *Use Case [53]*

Simbol	Nama	Deskripsi
	Aktor	Mewakili peran orang, sistem yang lain, atau alat ketika berkomunikasi dengan *use case*
	*Use Case*	Abstraksi dan int unsigned eraksi antara sistem dan aktor
	*Association*	Abstraksi dari penghubung antara aktor dengan *use case*
	*Generalisasi*	Menunjukkan spesialisasi aktor untuk dapat berpartisipasi dengan *use case*
	*Include*	Menunjukkan bahwa suatu *use case* seluruhnya merupakan fungsionalitas dari *use case* lainnya
	*Extend*	Menunjukkan bahwa suatu *use case* merupakan tambahan fungsional dari *use case* lainnya jika suatu kondisi terpenuhi

Berikut merupakan contoh sederhana *use case* diagram dari sebuah restoran yang dapat dilihat pada gambar 2.4

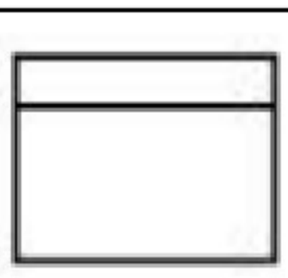
Gambar 2.4 *Use Case* Diagram Restoran *[53]*

Gambar 2.4 menggambarkan bagaimana beberapa peran (aktor) seperti waiter, client, cashier, dan chef berint unsigned eraksi dengan sistem restoran. Setiap aktor memiliki tugas tertentu, misalnya waiter menerima pesanan dan mengantarkannya, client memesan makanan dan melakukan pembayaran, chef memasak makanan atau menyiapkan minuman, sedangkan cashier menerima pembayaran. Diagram ini menunjukkan alur layanan restoran mulai dari pelanggan melakukan pesanan, makanan disiapkan, pelanggan makan, hingga proses pembayaran selesai. Hubungan *include* dan *extend* juga terlihat, yang menunjukkan bahwa beberapa aktivitas merupakan bagian dari aktivitas lainnya atau hanya terjadi jika kondisi tertentu terpenuhi (Misalnya penyajian wine hanya dilakukan jika pelanggan memesan wine). Secara keseluruhan, diagram ini membantu memperjelas bagaimana proses pelayanan restoran berlangsung dan siapa saja yang terlibat dalam setiap langkahnya.

2.5.2 Activity Diagram

Activity Diagram merupakan diagram dalam *Unified Modeling Language* yang digunakan untuk menggambarkan alur proses sistem dari awal hingga akhir. Diagram ini menunjukkan langkah-langkah yang dilakukan oleh pelanggan maupun sistem secara berurutan [54] [55]. *Activity Diagram* membantu dalam memahami proses bisnis yang terjadi pada sistem antrean digital. Berikut tampilan pada Tabel 2.2 dibawah ini.

Tabel 2.2 Simbol *Activity Diagram [47]*

Simbol	Nama	Keterangan
	Status awal	Sebuah diagram aktivitas memiliki sebuah status awal
	Aktivitas	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja
	Percabangan / *Decision*	Percabangan di mana ada pilihan aktivitas yang lebih dari satu
	Penggabungan / *Join*	Penggabungan yang mana lebih dari satu aktivitas lalu digabungkan jadi satu
	Status akhir	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuat status akhir
	*Swimlane*	*Swimlane* memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi

Pada sistem antrean digital, *Activity Diagram* digunakan untuk menggambarkan proses registrasi, *login*, pengambilan nomor antrean, hingga pemanggilan pelanggan oleh Admin.

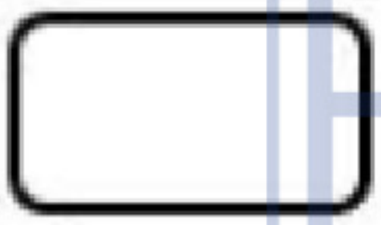
Diagram ini digunakan sebagai acuan dalam pembuatan logika program sehingga alur sistem yang dibuat sesuai dengan proses yang dirancang.

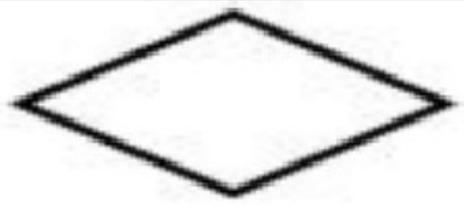
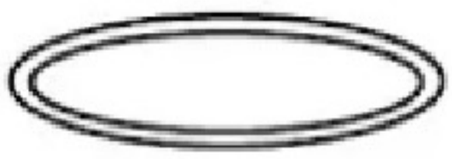
2.6 Entity Relationship Diagram (ERD)

Entity Relationship Diagram (ERD) merupakan model konseptual yang digunakan untuk merancang struktur basis data dengan menggambarkan entitas, atribut, serta hubungan antarentitas. Tujuan pelanggan ERD adalah menghasilkan rancangan basis data yang terstruktur, efisien, dan mampu mengurangi redundansi data sehingga mendukung kinerja sistem secara optimal [52] [34]

Dalam perancangan ERD, terdapat beberapa simbol standar yang digunakan untuk merepresentasikan komponen basis data. Pemahaman terhadap simbol-simbol ini penting agar rancangan ERD dapat dibaca dan dipahami secara konsisten oleh seluruh tim pengembang. Berikut adalah simbol-simbol utama yang digunakan dalam ERD pada Tabel 2.3 dibawah ini.

Tabel 2.3 Simbol ERD *[17]*

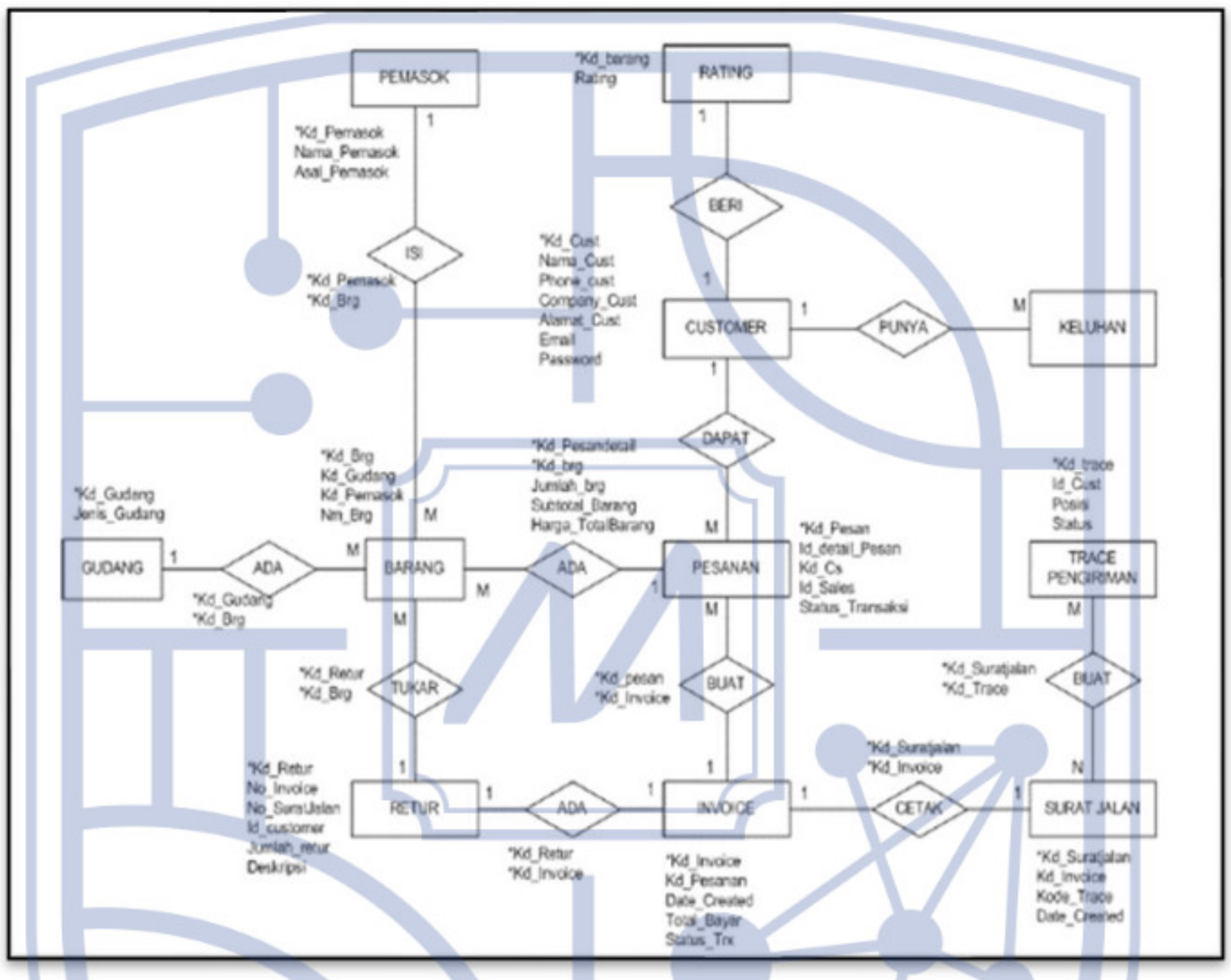
Simbol	Nama	Keterangan
	Entity (Entitas)	Entitas merupakan objek yang akan menjadi perhatian dalam suatu *database*. Entitas dapat berupa manusia, tempat, benda, atau kondisi mengenai data yang dibutuhkan.
	Attribute (Atribut)	Atribut merupakan informasi yang terdapat dalam entitas. Sebuah entitas harus memiliki *primary key* sebagai ciri khas entitas dan *foreign key* yang mereferensikan *primary key*. Atribut biasanya terletak dalam tabel entitas atau dapat juga terpisah dari tabel.

	Relationship (Relasi)	Relationship merupakan hubungan antara dua atau lebih entity dalam suatu sistem. Relasi yang dapat dimiliki oleh ERD ada beberapa macam, yaitu: 1.One-to-One Satu anggota entitas dapat berelasi dengan satu anggota entitas lain. 2.One-to-Many Satu anggota entitas dapat berelasi dengan beberapa anggota entitas lain. 3.Many-to-Many Beberapa anggota entitas dapat berelasi dengan beberapa anggota entitas lain.
	Weak Entity (Entitas Lemah)	Entitas lemah adalah entitas yang keberadaannya bergantung pada entitas lain (entitas kuat). Entitas lemah tidak memiliki *primary key* sendiri, melainkan menggunakan partial key yang digabungkan dengan *primary key* entitas kuat. Dilambangkan dengan persegi panjang bergaris ganda.
	Multi-valued Attribute (Atribut Bernilai Banyak)	Atribut bernilai banyak adalah atribut yang dapat memiliki lebih dari satu

		nilai untuk satu instance entitas. Dilambangkan dengan elips bergaris ganda (double oval). Contohnya, seorang pelanggan dapat memiliki lebih dari satu nomor telepon.
	Line / Link (Garis Penghubung)	Garis penghubung digunakan untuk menghubungkan entitas dengan atribut, atau entitas dengan relasi. Garis ini menunjukkan keterhubungan antar komponen dalam diagram ERD.
	*Primary Key* (Atribut Kunci)	Field atau kolom data yang butuh disimpan dalam suatu entitas dan digunakan sebagai kunci akses record yang diinginkan, biasanya berupa id kombinasi dari beberapa kolom yang bersifat unik (Berbeda tanpa ada yang sama).
	Cardinality (Kardinalitas)	Batasan jumlah maksimum instans entitas yang dapat dikaitkan dengan instans entitas lain melalui suatu relasi.

	Derived Attribute (Atribut Derivatif)	Atribut yang nilainya dihasilkan dari kalkulasi atau manipulasi atribut lain, bukan disimpan langsung dalam *database*.

Berikut merupakan contoh sederhana diagram ERD yang dapat dilihat pada gambar 2.5



Gambar 2.5 Diagram ERD *[56]*

Gambar 2.5 menggambarkan rancangan basis data untuk sistem pengelolaan persediaan dan transaksi barang. Dalam sistem ini terdapat beberapa entitas utama seperti pemasok, gudang, barang, customer, pesanan, *invoice*, retur, surat jalan, *trace* pengiriman, keluhan, dan *rating*. Entitas pemasok berhubungan dengan barang karena pemasok menyediakan barang yang akan disimpan di gudang. Barang yang tersedia kemudian dapat dipesan oleh customer melalui entitas pesanan, yang berisi informasi jumlah barang, subtotal, dan harga total. Dari pesanan tersebut akan dibuat *invoice* sebagai bukti transaksi pembayaran. Setelah itu sistem dapat membuat surat jalan untuk proses pengiriman barang yang dilengkapi dengan *trace* pengiriman untuk memantau status pengiriman. Selain itu, sistem juga menyediakan fitur retur apabila barang perlu dikembalikan, serta keluhan dan *rating* dari customer sebagai bentuk umpan balik terhadap barang atau layanan. Secara keseluruhan, ERD ini menunjukkan hubungan antar data dalam sistem yang mendukung proses mulai dari pemasokan barang, penyimpanan di gudang, pemesanan oleh pelanggan, hingga pengiriman dan layanan purna jual.

2.7 *Black Box Testing*

Black Box Testing merupakan metode pengujian perangkat lunak yang berfokus pada pengujian fungsi sistem tanpa melihat struktur kode program. Pengujian dilakukan dengan memberikan input tertentu dan mengevaluasi *output* yang dihasilkan sesuai dengan spesifikasi kebutuhan. Tujuan metode ini adalah memastikan setiap fitur sistem berjalan sesuai dengan yang dirancang.

Terdapat beberapa teknik utama dalam *Black Box Testing* yang umum digunakan dalam pengujian perangkat lunak [57] [58] [59], antara lain:

1. *Equivalence Partitioning* (Partisi Ekuivalen): Teknik ini membagi domain input menjadi beberapa kelas ekuivalen di mana semua nilai dalam satu kelas diharapkan menghasilkan perilaku yang sama. Dengan mengambil satu nilai representatif dari setiap kelas, jumlah *test case* dapat diminimalkan tanpa mengurangi cakupan pengujian. Kelas ekuivalen terdiri dari kelas *valid* (data yang diterima sistem) dan kelas tidak *valid* (data yang seharusnya ditolak sistem).
2. *Boundary Value Analysis* (Analisis Nilai Batas): Teknik ini berfokus pada pengujian nilai-nilai di batas atau tepi dari kelas ekuivalen, karena kesalahan sering terjadi di sekitar nilai batas. Nilai yang diuji meliputi nilai tepat di batas, satu nilai di bawah batas, dan satu nilai di atas batas.
3. *Decision Table Testing* (Pengujian Tabel Keputusan): Teknik ini digunakan untuk sistem yang memiliki kombinasi kondisi logika kompleks. Setiap kombinasi kondisi input dipetakan ke tindakan sistem yang sesuai dalam sebuah tabel keputusan.
4. *State Transition Testing* (Pengujian Transisi Status): Teknik ini menguji perubahan status sistem sebagai respons terhadap input atau kejadian tertentu, sangat cocok untuk sistem yang memiliki berbagai status seperti sistem antrean digital.
5. *Use Case Testing* (Pengujian *Use Case*): Teknik ini menggunakan skenario pelanggan sistem untuk merancang *test case* yang mencerminkan int unsigned eraksi nyata antara pelanggan dan sistem.

Pada penelitian ini, teknik *Black Box Testing* yang digunakan adalah *Equivalence Partitioning*. Pemilihan teknik ini didasarkan pada beberapa pertimbangan [57] [58] [59], antara lain:

1. Aplikasi antrean digital yang dikembangkan memiliki banyak formulir input seperti registrasi, *login*, pengambilan nomor antrean, dan pengelolaan data restoran, sehingga teknik

Equivalence Partitioning sangat tepat untuk mengelompokkan data uji ke dalam kelas-kelas ekuivalen yang representatif.

2. Teknik ini mampu meminimalkan jumlah *test case* yang diperlukan sambil tetap memaksimalkan cakupan pengujian, sehingga proses pengujian menjadi lebih efisien.
3. *Equivalence Partitioning* terbukti efektif untuk mendeteksi kesalahan validasi input dan alur logika pada aplikasi berbasis web dan *mobile*.
4. Teknik ini sesuai dengan tahap pengujian (*Testing*) dalam metodologi *Waterfall* yang digunakan pada penelitian ini.

Tabel 2.4 Contoh Hasil Pengujian *Black Box Testing Equivalence Partitioning [58]*

ID	Test Steps	Hasil yang Diharapkan	Hasil Pengujian
I.2-B01	1. Go to Form Upload File Submission 2. Click Add Submission 3. Select the file to be uploaded with Size file 5 MB 4. Click Save Changes	Sistem menerima request user dan mengubah Submission status dari "No attemps" menjadi "Submitted for grading".	Sistem menerima request user dan mengubah Submission status dari "No attemps" menjadi "Submitted for grading".
I.2-B02	1. Go to Form Upload File Submission 2. Click Add Submission 3. Select the file to be uploaded: Size file 79 MB and totally one file 4. Click Save Changes	Sistem menolak unggah file dan menampilkan alert notification "The file you tried to upload is too large for the server to process".	Sistem menolak unggah file dan menampilkan alert notification "The file you tried to upload is too large for the server to process".
I.2-B03	1. Go to Form Upload File Submission 2. Click Add Submission 3. Select the file to be uploaded: Size file 5 MB and totally 20 file 4. Click Save Changes	Sistem menolak unggah file dan sistem akan menghilangkan menu upload file karena sudah mencapai 20 file.	Sistem menolak unggah file dan sistem akan menghilangkan menu upload file karena sudah mencapai 20 file.
I.2-B04	1. Go to Form Upload File Submission 2. Click Add Submission 3. Select the file to be uploaded: Size file 79 MB and totally 20 file 4. Click Save Changes	Sistem menolak unggah file dan menampilkan alert notification "The file you tried to upload is too large for the server to process".	Sistem menolak unggah file dan menampilkan alert notification "The file you tried to upload is too large for the server to process".
I.2-B05	1. Go to Form Upload File Submission 2. Click Add Submission 3. Not uploading any files 4. Click Save Changes	Sistem menolak unggah file dan menampilkan alert notification "Nothing was Submitted".	Sistem menolak unggah file dan menampilkan alert notification "Nothing was Submitted
I.2-B06	1. Go to Form Upload File Submission 2. Click Add Submission 3. Select the file to be uploaded: with typefile(.pdf) 4. Click Save Changes	Sistem menerima request user dan mengubah Submission status dari "No attemps" menjadi "Submitted for grading".	Sistem menerima request user dan mengubah Submission status dari "No attemps" menjadi "Submitted for grading".
I.2-B07	1. Go to Form Upload File Submission 2. Click Add Submission 3. Select the file to be uploaded: not with typefile(.pdf) 4. Click Save Changes	Sistem menolak unggah file dan menampilkan alert notification ("Word 2007 Document filetype cannot be accepted")	Sistem menolak unggah file dan menampilkan alert notification ("Word 2007 Document filetype cannot be accepted")
I.3-C01	1. Go to Form Upload File Submission 2. Click Submission Comments 3. Write your comments ("Halo"). 4. Click Save Comment	Sistem akan menerima *comments* dan menampilkan *comments* di halaman *submission*.	Sistem akan menerima *comments* dan menampilkan *comments* di halaman *submission*.
I.3-C02	1. Go to Form Upload File Submission 2. Click Submission Comments 3. Write your comments. 4. Click Save Comment	Sistem menerima dan *comments* kosong tidak akan ditampilkan.	Sistem menerima dan *comments* kosong tidak akan ditampilkan.

Tabel 2.4 menampilkan hasil pengujian sistem menggunakan metode *Black Box Testing* pada fitur unggah file submission dan komentar. Setiap pengujian memiliki kode ID yang menunjukkan skenario uji, langkah pengujian (*test steps*), hasil yang diharapkan, serta hasil pengujian yang diperoleh dari sistem. Pengujian dilakukan dengan berbagai kondisi, seperti mengunggah file dengan ukuran yang sesuai (5 MB), mengunggah file yang melebihi batas ukuran (79 MB), mengunggah jumlah file yang melebihi batas maksimum (lebih dari 20 file), tidak mengunggah file sama sekali, serta mengunggah file dengan format yang tidak sesuai. Selain itu, dilakukan juga pengujian pada fitur *submission comments* untuk memastikan sistem dapat menerima dan menampilkan komentar yang diberikan oleh pelanggan, serta memastikan

komentar kosong tidak ditampilkan. Berdasarkan hasil pengujian yang ditampilkan pada tabel, seluruh skenario menunjukkan bahwa sistem memberikan respons yang sesuai dengan hasil yang diharapkan, seperti menampilkan notifikasi kesalahan ketika file tidak memenuhi ketentuan, serta menerima dan memproses input yang *valid* dengan benar. Hal ini menunjukkan bahwa fitur unggah file dan komentar pada sistem telah berjalan sesuai dengan fungsi yang dirancang.