BAB II
TINJAUAN PUSTAKA

2.1 Tinjauan Pustaka

Pertama, penelitian dengan judul "Perancangan Sistem Informasi Resep Pulang Farmasi Rawat Inap Dengan Metode *E-Prescribing* Pada *RS XYZ*" oleh Tb. Dedifuadi, Ahmad Surahmat, Rizki Fatullah [9]. Dalam penelitian ini, peneliti menggunakan metode pengembangan sistem *waterfall* yang menyusun langkah-langkah terstruktur dalam pembuatan perangkat lunak, dimulai dari tahap analisis, desain, pembuatan kode, pengujian, hingga tahap dukungan secara berurutan menggunakan metode *E-Prescribing* untuk memperluas ketersediaan layanan medis dan meningkatkan standar serta efisiensi dari layanan yang telah disediakan. Sistem yang dikembangkan oleh peneliti mencakup penggunaan elektronik dalam penulisan resep untuk obat yang bertujuan untuk meningkatkan kualitas serta memperbaiki efektifitas layanan kesehatan yang diberikan. Hasil peneliti dimana menunjukan adanya perbedaan signifikan dalam waktu tunggu pasien sebelum dan setelah penggunaan resep elektronik. Ini menandakan bahwa perubahan menuju sistem peresepan elektronik memiliki dampak positif yang signifikan pada kualitas pelayanan terkait terhadap obat yang mereka butuhkan. Hal ini berpengaruh pada pelayanan pasien karena obat yang diberikan lebih cepat dan juga menghindari kesalahan dalam membaca resep.

Kedua penelitian dengan judul "Peneran Sistem *E-Prescribing* dan *Barcode System* di Rumah Sakit Pertamina Menggunakan *PowerBuilder* Dengan *Arsitektur Client - Server System*" oleh Try Viananda Nova M, S. Kom., M. Kom, dan Rina Alfah, S. Kom., M. Kom [10]. Hasil dari penelitian terkait pengembangan di Rumah Sakit Pertamina Tanjung, aplikasi *E-Prescribing* dan sistem *Barcode* dibuat menggunakan *Power Builder v12.6* dan memanfaatkan *database stbase* menunjukkan bahwa aplikasi *E-Prescribing* tersebut memberikan manfaat yang signifikan bagi instalasi farmasi di rumah sakit tersebut. Aplikasi ini menampilkan kenyamanan dalam proses menerima resep yang diberikan oleh dokter kepada para

pasien yang ditangani, dan juga mengurangi risiko kesalahan dalam pemberian obat yang mungkin terjadi. Peneliti menggunakan metode *Rapid Aplication Development (RAD)* dalam pengembangan aplikasi, pendekatan yang digunakan juga memiliki kecepatan secara visual, kekuatan yang signifikan, dan kemudahan penggunaan yang optimal. Penelitian yang dilakukan melibatkan eksperimen dan penerapan guna menciptakan sistem yang menyatukan dokter dengan asisten atau apoteker, mengatur informasi tentang ketersediaan stok obat, memberikan informasi tentang indikasi dan kegunaan obat.

Ketiga, penelitian dengan judul "Rancang Bangun Aplikasi Elektronik Resep yang Terhubung ke Kasir Menggunakan *Website*" oleh Helfy Susilawati, Tri Arif Wiharso, Teddy Mulyadi Hidayat [11]. Penelitian ini menggunakan resep elektronik yang terhubung secara langsung ke sistem kasir, merupakan langkah peningkatan dalam layanan kesehatan karena memungkinkan pasien untuk melakukan pembayaran hanya sekali saja. Dalam praktik konvensional, pasien hanya memberikan resep ke kasir, menunggu perhitungan jumlah pembayaran, dan kemudian melakukan pembayaran ke kasir. Tetapi, melalui penerapan resep elektronik yang saling terkait secara langsung ke kasir, pasien hanya perlu menunggu hingga nomor antrian atau nomor pasien mereka dipanggil sebelum melakukan pembayaran untuk obatnya. Peneliti menerapkan metode pembuatan aplikasi tersebut menggunakan metode *prototype*, yang sesuai untuk sistem atau perangkat yang akan mengalami perkembangan lebih lanjut. *Prototype* akan berkembang sesuai dengan keinginan pengguna dan akan membantu pengembangan oleh para pengembang.

Keempat, penelitian dengan judul "Rancang Bangun Sistem *Electronic Prescribing* Dokter dengan Menggunakan *Codeigniter*" oleh Dian Rakasiwi, Ria Arafiyah, Fariani Hermin Indiyah [12]. Tujuan dari penelitian ini adalah menerapkan sistem *electronic prescribing* guna meningkatkan aksesibilitas layanan kesehatan serta meningkatkan mutu dan efisiensi layanan yang disediakan pada Rumah Sakit Gigi dan Mulut *TNI-AU* Jakarta. Pengembangan sistem tersebut melibatkan pendekatan metode *waterfall*, yang melibatkan langkah-langkah dari analisis kebutuhan, perancangan sistem, penulisan kode program, implementasi

perangkat lunak, dan pemeliharaan sistem. Peneliti memanfaatkan Bahasa pemrograman *PHP* dengan framework *Codeigniter* yang mengadopsi pola *MVC (Model View Controller).* Penelitian ini bertujuan untuk mengalihkan cara tradisional pencatatan resep dokter menjadi sistem resep elektronik bertujuan untuk meningkatkan efektivitas dan efisiensi, terutama dengan meminimalkan waktu tunggu saat pengambilan obat dan menghindari kesalahan dalam proses pemberian obat.

Kelima, penelitian dengan judul "Sistem Informasi Klinik Berbasis *Website* Menggunakan Metode *Extreme Programming* Pada Klinik Karunia Bunda" oleh Donda Banjarnahor, Nurulfaizah, Kurnia gusti Ayu [13]. Berdasarkan penelitian yang dilakukan, peneliti membangun sistem informasi klinik berbasis *web* dengan menerapkan metode *Extreme Programming*. Hasil dari aplikasi yang dikembangkan oleh peneliti adalah aplikasi yang mampu menghasilkan laporan dan informasi sesuai kebutuhan, memungkinkan manajemen untuk mengambil keputusan yang mendukung perkembangan klinik. Aplikasi yang dikembangkan juga mempermudah seluruh petugas dalam memberikan layanan optimal kepada pasien, terutama dalam pengelolaan iniformasi medis pasien. Peneliti memanfaatkan bahasa pemrograman *PHP* dan menggunakan *database MySQL* untuk mengembangkan aplikasi tersebut.

Keenam, penelitian dengan judul "*Impact of the e-prescribing system on the incidence and nature of drug-related problems in children in a Saudi hospital*" oleh Aeshah A. Alazmia, Hani Alhamdana, Omaima Ahmeda, Stephen Tomlin, Asia N. Rashed [14]. Berdasarkan penelitian yang dilakukan oleh peneliti penggunaan sistem *electronic prescribing* berpengaruh pada anak-anak dan saat menggunakan sistem *electronic prescribing* ini mengurangi angka kesalahan peresepan obat pada anak-anak khususnya pada rumah sakit saudi dan menggunakan sistem *CPOE* untuk menurunkan insiden *DRP* pada anak-anak yang dirawat di *KAMC-J*. Menggabungkan panduan klinis berbasis komputer yang lebih khusus untuk pediatrik ke dalam *CPOE* diperlukan untuk lebih mengurangi masalah ini dan meningkatkan perawatan yang diberikan kepada anak-anak.

Ketujuh, penelitian dengan judul "*Mixed methods study of medication-related decision support alerts experienced during electronic prescribing for inpatients at an English hospital*" oleh Helen Bell, Sara Garfield, Sonia Khosla, Chimnay Patel, Bryony Dean Franklin [15]. Berdasarkan penelitian yang dilakukan oleh peneliti menemukan bahwa sebagian besar peringatan dukungan keputusan diabaikan, namun peringatan untuk *VTE (venous thromboembolism)* jauh lebih kecil kemungkinannya untuk diabaikan ketika muncul di luar putaran *ward*. Temuan ini menunjukkan bahwa peringatan memiliki kemungkinan lebih kecil untuk diabaikan jika mereka terintegrasi dalam alur kerja yang tepat bagi para penulis resep, baik dari segi waktu peringatan maupun para profesional kesehatan yang dituju. Temuan juga menyarankan bahwa peringatan seharusnya mencakup tautan ke tindakan yang diperlukan. Maka dari itu, peneliti menggunakan sistem *electronic prescribing* untuk mengurangi kesalahan penulisan resep obat dari dokter. Bertujuan untuk membantu para penulis resep dalam melakukan preskripsi secara aman, seperti kamus obat, saran dosis default, peringatan interaksi obat-obatan, peringatan alergi obat, dan peringatan terhadap hasil laboratorium yang relevan. Peringatan pop-up sering digunakan untuk menyoroti peringatan kepada penulis resep.

Pada Tabel 2.1 berisikan ringkasan penelitian terdahulu yang relevan dengan penelitian yang akan dilakukan oleh peneliti.

Tabel 2.1 Ringkasan Penelitian Terdahulu

No	Judul	Penulis (tahun)	Hasil	Perbedaan
1	Perancangan Sistem Informasi Resep Pulang Farmasi Rawat Inap Dengan Metode *E-Presscribing* Pada *RS XYZ* [9]	Tb. Dedifuadi, Ahmad Surahmat, Rizki Fatullah	Mengembangkan sebuah sistem informasi untuk resep obat di farmasi rawat inap menggunakan pendekatan/metode *E-Prescribing* dengan bertujuan meningkatkan aksebilitas layanan kesehatan serta meningkatkan efisiensi dari pelayanan yang disediakan.	*Website* kali ini akan dibuat dengan tampilan yang lebih hidup, dengan menggunakan *framework react.js*
2	Penerapan Sistem *E-Prescribing* dan *Barcode system* di Rumah Sakit Pertamina menggunakan *PowerBuilder* dengan *Arsitektur Client - Server System* [10]	Try Viananda Nova M, S.Kom.,M.Kom, Rina Alfah, S.Kom.,M.Kom	Menerapkan sistem *E-Prescribing* dan *Barcode System* di Rumah Sakit Pertamina yang bertujuan mempermudah instalasi farmasi di rumah sakit agar lebih efisien dalam menerima dan memproses resep dari dokter dan mengurangi risiko *medician error*.	Penelitian sebelumnya menggunakan pengembangan sistem *Rapid Aplication Development (RAD)*, sedangkan penelitian kali ini menggunakan pengembangan sistem yaitu *extreme programming*

No	Judul	Penulis (tahun)	Hasil	Perbedaan
3	Rancang Bangun Aplikasi Elektronik Resep yang Terhubung ke Kasir Menggunakan *Website* [11]	Helfy Susilawati, Tri Arif Wiharso, Teddy Mulyadi Hidayat	Membuat aplikasi elektronik resep yang terhubung kasir dikembangkan dengan metode *prototype,* tujuan dari penelitian ini adalah meminimalkan waktu tunggu pasien untuk membayar, sehingga hanya dibutuhkan satu kali proses pembayaran setelah panggilan di kasir. Selain itu, resep obat juga didesain agar mudah dibaca oleh pasien dan apoteker.	Pengembangkan sistem sebelumnya menggunaan metode *prototype*, pada penelitian kali ini menggunakan metode *extreme programming*
4	Rancang Bangun Sistem *Electronic Prescribing* Dokter dengan Menggunakan *Codeigniter* [12]	Dian Rakasiwi, Ria Arafiyah, Fariani Hermin Indiyah	Pembuatan sistem e*lectronic prescribing* dokter dengan menggunakan *framework Codeigniter* yang bertujuan meningkatkan ketersediaan layanan kesehatan dan meningkatkan standar serta efisiensi yang diberikan oleh Rumah Sakit.	Penelitian ini sebelumnya menggunakan *framework codeigniter*, sedangkan penelitian kali ini menggunakan bahasa *javascript* dengan *framework express.js*

No	Judul	Penulis (tahun)	Hasil	Perbedaan
5	Sistem Informasi Klinik Berbasis *Website* Menggunakan Metode *Extreme Programming* Pada Klinik Karunia Bunda [13]	Donda Banjarnahor, Nurulfaizah, Kurnia Gusti Ayu	Pengembangan sistem informasi klinik berbasis *website* yang menerapkan metode *extreme programming* dengan tujuan untuk menyediakan kemudahan bagi semua petugas dalam memberikan layanan yang optima kepada pasien di klinik tersebut.	Peneliti sebelumnya menggunakan bahasa pemrograman *PHP*, sedangkan Penlitian kali ini menggunakan bahasa pemrograman *javascript*
6	*Impact of the e-prescribing system on the incidence and nature of drug-related problems in children in a Saudi Hospital* [14]	Aeshah A. AlAzmia, Hani AlHamdana, Omaima Ahmeda, Stephen Tomlin, Asia N. Rashed	Penelitian dengan peresepan obat dengan sistem *electronic prescribing* pada anak-anak dengan menggabungkan panduan klinis berbasis komputer yang lebih khusus untuk pediatrik ke dalam *CPOE.*	Penelitian kali ini menggunakan *javascript* untuk bahasa pemrogramannya dan menggunakan *react.js* untuk bagian *frontend*
7	*Mixed methods study of medication-related decision support alerts experienced during electronic prescribing for inpatients at an English hospital* [15]	Helen Bell, Sara Garfield, Sonia Khosla, Chimnay Patel, Bryony Dean Franklin	Membangun sistem *electronic prescribing* yang dilakukan oleh peneliti bertujuan untuk membantu para penulis resep dalam melakukan preskripsi secara aman, saran dosis default, peringatan interaksi obat-obatan, dan peringatan alergi obat.	Penelitian kali ini membuat sistem *electronic prescribing* menggunakan *express.js* untuk bagian *backend* dan *MySQL* untuk *database*

2.2 Landasan Teori

Pada penelitian kali ini, diperlakukan pemahaman mengenai beberapa teori yang relevan dan akan digunakan selama proses pengembangan aplikasi sistem *electronic prescribing* dalam penelitian ini.

2.2.1 Rancang Bangun

Rancang bangun merupakan proses konstruksi sistem yang berorientasi pada pembuatan atau perbaikan sistem yang sudah ada, baik secara menyeluruh maupun sebagian, bertujuan untuk meningkatkan efisiensi atau kinerja secara keseluruhan. Rancang bangun merupakan langkah pertama dalam menciptakan gambaran atau sketsa sistem yang belum pernah ada sebelumnya, dan kemudian mengelolanya hingga menciptakan representasi visual atau sketsa yang sesuai dengan kebutuhan fungsional yang diinginkan [16].

2.2.2 *Website*

Website merupakan koleksi halaman-halaman yang memuat informasi dalam bentuk teks, gambar yang bisa diam atau bergerak, animasi, serta elemen suara, dan kombinasi dari semua elemen tersebut, yang bisa bersifat tetap atau dinamis. Ini juga membentuk suatu struktur bangunan yang saling terkait. Setiap bagian dari struktur ini dihubungkan dengan jaringan halaman, yang terdiri dari sejumlah halaman *web* yang saling terhubung dengan topik yang berhubungan satu sama lain. Di beberapa kasus, halaman *web* juga bisa memuat berkas-berkas seperti gambar, video, atau jenis berkas lainnya [17].

2.2.3 Sistem *Electronic Prescribing*

Sistem *electronic prescribing* atau sistem peresepan electronik adalah sistem untuk membuat resep elektronik yang dapat langsung dikirimkan ke apotek atau sistem farmasi terkait. Sistem ini menggunakan teknologi informasi untuk menggantikan atau mengurangi penggunaan resep kertas dalam proses pemberian resep obat kepada pasien. Tujuan utama dari penerapan *electronic prescribing* yaitu mengurangi kesalahan medis, mengurangi biaya yang terkait dengan apotek, meningkatkan efisiensi

penulisan resep dan apotek, menghilangkan risiko kehilangan dalam menginterpretasikan tulis tangan, mengurangi komunikasi melalui telepon antara apoteker dan dokter, mengurangi pekerjaan *input* data, serta mempercepat permintaan salinan resep [18].

2.2.4 *User Interface*

User Interface (UI) adalah suatu cara atau antarmuka yang digunakan untuk memfasilitasi interaksi antara manusia dan sistem komputer. UI sering kali dianggap sebagai pengganti *Human Computer Interaction* (HCI), mencakup semua bentuk interaksi yang dilakukan oleh manusia terhadap komputer. Dalam konteks ini, UI mencakup elemen-elemen seperti tata letak, elemen grafis, dan kontrol yang memungkinkan pengguna berkomunikasi dengan sistem komputer secara efektif. Inti dari UI adalah memberikan pengalaman pengguna yang intuitif dan efisien, memastikan bahwa interaksi antara manusia dan komputer berjalan dengan lancar dan memenuhi kebutuhan pengguna [19].

2.2.5 *Database MySQL*

Database adalah kumpulan informasi data yang tersimpan dengan terstruktur di dalam komputer untuk memungkinkan pengambilan informasi. Istilah “basis data” pertama kali muncul dalam domain ilmu komputer. Artikel ini membahas basis data komputer, meskipun signifikansinya kemudian berkembang untuk mencakup hal-hal di luar konteks elektronik. Basis data telah ada sebelum periode revolusi industri, bentuk awal dari basis data terdapat dalam buku, kwitansi, dan koleksi informasi bisnis yang terstruktur [20]. *MySQL* ialah sebuah program manajemen basis data yang *open source*. Ini berasal dari konsep utama dalam manajemen basis data yang disebut *SQL (Structured Query Languange)*. *SQL* merupakan sebuah konsep yang digunakan untuk mengatur dan mengoperasikan basis data, terutama dalam pemilihan serta pengelompokkan data, sehingga memungkinkan pengoperasian data secara otomatis dengan mudah [21].

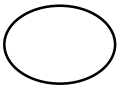
2.2.6 UML

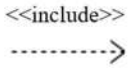
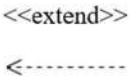
UML (Unified Modeling Language) adalah sebuah bahasa berbasis gambar yang digunakan untuk visualisasi, penjelasan, perancangan, dan dokumentasi sistem perangkat lunak yang menggunakan pendekatan berorientasi objek. UML juga memberikan standar untuk membuat panduan sistem, termasuk konsep proses bisnis, perancangan kelas dalam bahasa pemrograman tertentu, struktur basis data, serta elemen-elemen yang diperlukan dalam pengembangan perangkat lunak [22].

2.2.5.1 *Use Case* Diagram

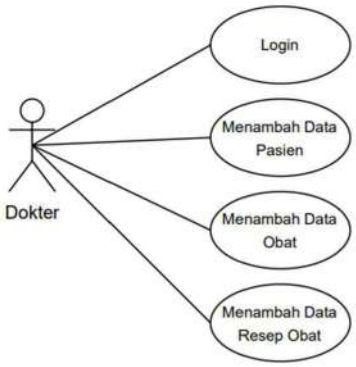
Diagram use case merupakan alat pemodelan yang digunakan untuk menggambarkan bagaimana satu atau lebih aktor berinteraksi dengan sistem informasi yang sedang dikembangkan. Diagram *use case* dipakai untuk menjelaskan bagaimana interaksi antara *actor* (pengguna atau *entitas ekternal*) dengan sistem yang sedang dibangun. Dengan menggunakan *Use case* diagram, proses pemetaan kebutuhan fungsional yang diperoleh dalam tahap analisis menjadi lebih mudah [23]. Nama simbol beserta fungsinya terdapat pada tabel 2.3 berikut ini:

Tabel 2.2 *Use-case* Diagram [23]

Gambar	Nama	Deskripsi
	Asosiasi	Asosiasi berguna sebagai penghubung antara objek satu dengan objek lainnya.
	*Use case*	Deskripsi dari suatu fungsi atau tindakan yang dapat dilakukan oleh sistem.
	*System*	Menampilkan sistem yang sedang dikembangkan atau sedangberjalan.

Gambar	Nama	Deskripsi
	*Actor*	Aktor merupakan entitas yang berhubungan secara aktif dengan sistem, seperti pengguna atau sistem lainnya.
	*Include*	Hubungan antara *use case* yang memasukkan fungsionalitas dari *use case* lain.
	*Extend*	Hubungan antara *use case* yang menambahkan fungsionalitas ke *use case lain.*

Berikut merupakan contoh penggunaan *use case* diagram yang terdapat pada gambar 2.1.

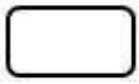
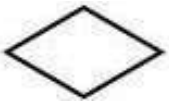
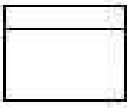


Gambar 2.1 Contoh Diagram *Use Case* [23]

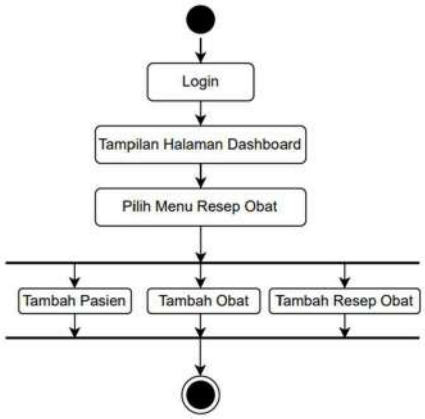
2.2.5.2 *Activity* Diagram

Activity diagram merupakan suatu bentuk pemodelan yang diterapkan pada sistem yang menggambarkan bagaimana aktivitas dalam sistem berjalan. *Activity* diagram dipakai untuk menguraikan kegiatan dalam sebuah program tanpa harus memeriksa kode atau tampilannya. Diagram ini merepresentasikan proses bisnis dan urutan kegiatan dalam suatu proses. Seperti *flowchart*, diagram ini menggambarkan aliran kerja dari satu aktivitas ke aktivitas lainnya atau dari aktivitas ke status tertentu dalam sistem [24]. Nama simbol beserta fungsinya terdapat pada tabel 2.4 berikut ini:

Tabel 2.3 *Activity* Diagram [24]

Simbol	Nama	Keterangan
	Aktivitas	Umumnya dimulai dengan kata kerja, menggambarkan aktivitas yang sedang dilakukan oleh sistem
	*Initial Node*	Mengawali suatu aktivitas.
	Status Akhir	Mengakhiri alur sistem.
	Penggabungan	Penggabungan yang mana lebih dari satu aktivitas akan digabungkan menjadi satu.
	*Decision*	Menggambarkan terjadinya percabangan atau ada pilihan aktivitas yang lebih dari satu.
	*Swimlane*	*Swimlane* memperlihatkan tanggung jawab pelaksanaan aktvitas dalam suatu diagram atau aktivitas yang terjadi.

Berikut merupakan contoh penggunaan *activity* diagram yang terdapat pada gambar 2.2.



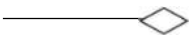
Gambar 2.2 Contoh *Activity* Diagram [24]

2.2.5.3 *Class* Diagram

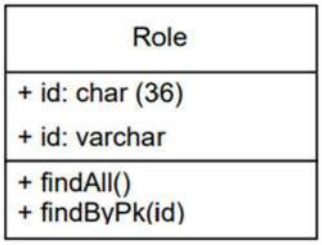
Class diagram merupakan tipe diagram dalam *UML* dipakai untuk menggambarkan struktur kelas dan paket yang terdapat dalam sistem yang direncanakan digunakan. Dalam kata lain, *UML* digunakan untuk mengilustrasikan kelas-kelas dan kelompok-kelompok yang ada dalam sistem tersebut, diagram ini memberikan representasi visual mengenai struktur sistem serta interaksi-interaksi yang ada di dalamnya. *Class* diagram merupakan cara untuk menunjukan kelas-kelas yang ada didalam sistem dan hubungan logis di antara mereka, dan ini memberikan gambaran tentang struktur statis sistem tersebut [25]. Nama simbol beserta fungsinya terdapat pada tabel 2.5 berikut ini:

Tabel 2.4 *Class* Diagram [25]

Gambar	Nama	Deskripsi
nama_kelas + attribut + operasi()	Kelas	Entitas yang merepresentasikan objek dalam suatu sistem atau proses

Gambar	Nama	Deskripsi
 **nama_interface**	*Interface*	Mirip dengan ide dari antarmuka dalam paradigma pemrograman berbasis objek
	Asosiasi berarah	Relasi antar kelas yang bermakna generalisasi - spesialisasi (umum khusus)
	Asosiasi	Hubungan antar kelas yang memiliki arti yang umum, sering kali terkait dengan asosiasi yang juga menyertakan konsep *multiplicity*
	Generalisasi	Hubungan antar kelas bisa terjadi melalui asosiasi dan generalisasi-spesialisasi
	*A ggregation*	Hubungan yang berarti antara kelas yang mencakup keseluruhan dan bagian-bagiannya *(whole-part)*
	*Dependecy*	Ketergantungan antarkelas

Berikut merupakan contoh penggunaan *class* diagram yang terdapat pada gambar 2.3.



Gambar 2.3 Contoh *Class* Diagram [25]

2.2.9 Pemrograman *Web*

Pemrograman *web* merupakan hasil penggabungan dua kata, yaitu pemrograman dan *web*. Pemrograman merujuk pada proses atau metode pembuatan program. Sedangkan, *web* merujuk pada suatu rangkaian komputer yang terhubung di jaringan, terdiri dari berbagai situs *internet* yang menyuguhkan teks, gambar, dan sumber daya animasi menggunakan protokol transfer *hypertext*. Istilah umum yang digunakan untuk *web adalah "WWW" (World Wide Web)*, yang merupakan kumpulan halaman *website* yang terhubung satu sama lain melalui hyperlink. *WWW* beroperasi dengan menggunakan *protocol HyperText Transfer Protocol (HTTP)*. Sebuah halaman *web* adalah dokumen teks yang berisi kode *HTML* dan bisa diakses, dilihat atau ditampilkan oleh peramban *internet*. Kode *HTML* memfasilitasi penampilan konten seperti teks, gambar, audio, video, dan animasi [26].

2.2.10 *JavaScript*

JavaScript adalah bahasa pemrograman *web* yang sering dipakai untuk pengembangan di sisi klien *(Client-Side Programming Language).* Merupakan jenis Bahasa pemrograman yang dilakukan oleh *client*, pada konteks ini mengacu pada perangkat seperti *Google Chrome*, *Mozilla Firefox*, *Opera Mini*, dan lainnya. Memungkinkan banyak orang untuk melihat dan mengakses isi *Source Code JavaScript*. Seiring dengan perkembangan waktu, *JavaScript* telah berkembang lebih lanjut melalui

kontribusi para pemrogram dan pengembang, sehingga saat ini *JavaScript* dapat digunakan tidak hanya pada *web browser*, tetapi juga dalam *server* ataupun *game* [27].

2.2.11 *Framework*

Framework adalah sekumpulan instruksi yang dikelompokkan dalam kelas-kelas dan fungsi-fungsi dengan tujuan untuk mempermudah cara penggunaannya oleh para pengembang. Hal ini memungkinkan para pengembang untuk tidak mengulangi penulisan sintaks program serupa secara berulang, menghemat waktu dan usaha [28]. Terdapat jenis-jenis *framework* yang digunakan untuk pengembangan *website*, terdapat dua yang sering digunakan oleh *web developer* adalah *ReactJS* dan *ExpressJS*.

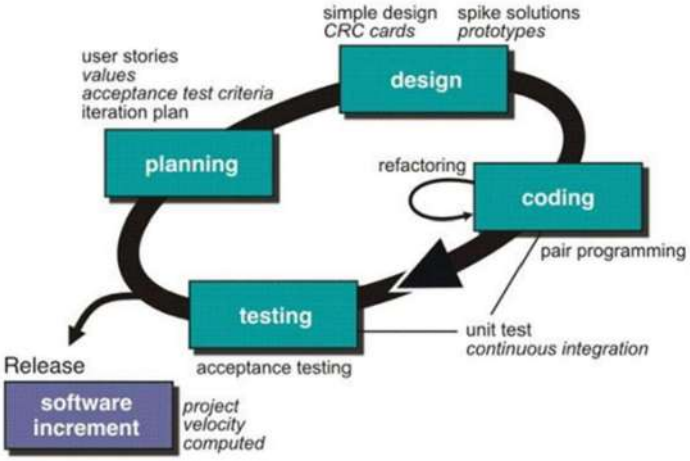
Dalam pembuatan situs *web*, penggunaan *framework* dapat menjadi sarana untuk mempermudah proses pengembangan. Dalam konteks ini, penulis memanfaatkan *framework ReactJS* sebagai alat bantu untuk mengelompokkan *code* program. *ReactJS* merupakan sebuah perpustakaan *JavaScript* sumber terbuka yang diciptakan oleh *Facebook*, yang digunakan untuk konstruksi antarmuka pengguna *(User Interface)*. *ReactJS* difokuskan pada pengelolaan seluruh aspek yang terkait dengan tampilan, memungkinkan penggunaannya dalam pembangunan serta pengembangan aplikasi berbasis *web* [29].

ExpressJS adalah sebuah *framework* aplikasi *web* untuk *NodeJS* yang ditulis dalam Bahasa pemrograman *javascript*. *ExpressJS* digunakan untuk membuat aplikasi sisi belakang *(backend)* secara efisien dan optimal. Sebagai *backend web* aplikasi *framework* yang bersifat *open soure* dibawah lisensi *IMT* untuk *NodeJS*, *ExpressJS* didesain khusus untuk membangun aplikasi *web* dan *API* [30].

2.2.12 Metode *Extreme Programming*

Extreme Programming (XP) adalah sebuah metode pengembangan perangkat lunak yang sederhana dan melibatkan praktik-praktik dari metode yang fleksibel yang dipelipori oleh Kent Beck, Ron Jeffries, dan Ward Cunningham. *XP* dikenal sebagai salah satu pendekatan metode

tangkas yang sangat terkenal dan umum digunakan. Metode ini menerapkan pendekatan rekayasa perangkat lunak yang cenderung menggunakan pendekatan yang berfokus pada objek [31]. Metode *Extreme Programming* adalah metode yang berfokus pada pembuatan kode aplikasi dengan tujuan menyederhanakan pengembangan sistem agar lebih fleksibel [32]. Pada gambar 2.4 merupakan tahapan dari *extreme programming* yang terdiri dari empat tahapan dan juga penjelasan dari tahapan-tahapan tersebut.



Gambar 2.4 Metode *Extreme Programming* [32]

1. *Planning*

   Tahap *planning atau* perencanaan dimulai dengan langkah awal dalam pembangunan sistem dengan metode *XP*, dimana beberapa kegiatan perencanaan dilakukan, seperti identifikasi permasalahan, analisis kebutuhan, hingga penetapan jadwal pelaksanaan pembangunan sistem [32].

   Pada tahap *planning,* perencanaan dilakukan melalui *planning meeting*. Pertemuan dimulai dengan meninjau kebutuhan pengguna, selanjutnya, akan dilakukan penetapan estimasi waktu penyelesaian. Di akhir pertemuan, pengguna membuat keputusan mengenai fitur mana yang akan diimplementasikan [33]. Luaran yang diharapkan dari tahap *planning meeting* dapat berupa *list* masalah ataupun daftar

kebutuhan fitur pengguna [32]. Pada tabel 2.5 merupakan contoh *output* dari tahapan *planning meeting*.

Tabel 2.5 Contoh Kebutuhan Pengguna [34]

Jenis Kebutuhan	
Kebutuhan Pengguna	**Kebutuhan Sistem**
Pengguna dapat mengakses halaman *website* dan mengisi *form login*	Sistem mengalihkan ke halaman *dashboard* seteleh *login*
Pengguna dapat menambahkan data pasien	Sistem menampilkan data pasien yang ada
Pengguna dapat mencetak resep	Sistem akan kembali halaman *login* jika menekan tombol keluar

Dalam setiap iterasi, tim pengembangan dan pelanggan bertemu untuk mendefinisikan pekerjaan yang akan diselesaikan melalui pengembangan "*user stories*". *User Story* merupakan deskripsi pengalaman atau aksi pengguna pada situs *web*. Setiap deskripsi *user story* umumnya terdiri dari satu hingga tiga kalimat. Kegiatan ini merupakan kelanjutan dari perencanaan yang dilakukan pada tahap awal, dimana user stories memberikan landasan konkret untuk pekerjaan yang akan dilakukan dalam setiap iterasi. Berikut contoh kalimat *user story* yang terdapat pada tabel 2.6.

Tabel 2.6 Contoh *User Story* [33]

No	***User Story***
1	Sistem yang dibuat harus ada validasi atau konfirmasi setelah data pasien atau resep obat ditambahkan
2	Sistem yang dibuat harus mampu update otomatis jika proses pembuatan resep obat selesai

2. *Design*

Pada tahap *design* dilakukan perencanaan representasi sistem yang akan digunakan untuk memudahkan *developer* dalam proses pengembangan aplikasi yang umumnya dengan bantuan *Unified Modelling Language (UML)* sebagai modelnya, berbagai jenis diagram seperti diagram *use case*, *activity* diagram dan *class* diagram digunakan untuk mewakili informasi secara visual mengenai bagaimana sistem atau proses bekerja, sementara untuk memodelkan basis data digunakan *class* diagram [35]. Contoh *use case* diagram terdapat pada gambar 2.1, contoh *class* diagram pada gambar 2.2 dan contoh *class* diagram pada gambar 2.3.

3. *Coding*

Tahapan *coding* adalah tahapan yang melibatkan langsung *developer* sistem untuk langsung mengimplementasikan hasil dari desain yang telah sebelumnya telah dibuat. Aktivitas yang terlibat dalam tahap ini mencakup pembuatan basis data dan antarmuka pengguna [34].

Tahapan *coding* dalam pengembangan suatu aplikasi biasannya sistem akan dibagi menjadi dua, yaitu untuk *frontend* dan *backend*. Pada tahapan pengkodean dari sisi *frontend* atau coding akan dilakukan proses penerjemahan dari perancangan *UML* menjadi sebuah *user interface* dan dari sisi *backend* akan merancang basis data menggunakan bahasa pemrograman. Sehingga *output* dari tahapan ini akan berupa tampilan dan fitur dari sebuah sistem yang akan dibangun [36].

4. *Testing*

Pada tahap ini, aplikasi yang dibuat akan dilakukan pengujian atau *testing*. Tahap ini dipengaruhi oleh pengguna sistem yang memprioritaskan fitur dan fungsi keseluruhan sistem, serta dinilai oleh mereka sendiri. Metode yang digunakan untuk menguji

aplikasi sistem *electronic prescribing* dokter berbasis *website* adalah *Blackbox Testing.*

Ada beberapa cara untuk melakukan pengujian sistem, salah satunya dengan menggunakan metode *blackbox*. Keuntungan dari metode ini adalah penentuan penguji atau *tester* tidak harus memiliki pengetahuan akan bahasa pemrograman [36]. Pada tabel 2.7 merupakan contoh halaman dan fitur yang akan dilakukan pengujian menggunaan *blackbox testing.*

Tabel 2.7 Contoh Pengujian Fitur Pada Halaman *Login* [36]

No	Proses	***Expexted Output***
1	Mengakses halaman *website* di *browser*	Menampilkan Halaman *Login*
2	Mengisi *form login* dan klik tombol *login*	Mengalihkan pada halaman *login*

2.2.13 *Blackbox Testing*

Blackbox Testing adalah metode pegujian perangkat lunak yang berfokus pada pengujian fungsionalitas aplikasi tanpa memerlukan pemahaman mendalam tentang struktur internal atau bagaimana aplikasi bekerja didalamnya. Dalam pengujian ini, informasi yang digunakan untuk mengembangkan uji kasus berasal dari deskripsi eksternal perangkat lunak, seperti spesifikasi, dan desain. Pengujian ini dapat mencakup baik pengujian fungsional. Perancang uji akan memilih input yang valid dan invalid serta menetapkan hasil yang diharapkan. Pendekatan pengujian ini dapat diterapkan pada berbagai tahapan pengujian perangkat lunak, mulai dari pengujian unit, integrasi, fungsional, sistem, hingga tahap penerimaan [37].

Salah satu metode *Blackbox* yang cukup pupular adalah *Equivalance partitioning*, metode ini didasarkan pada pemisahan masukan dan keluaran suatu komponen sesuai spesifikasi. Asumsinya adalah masukan yang sama akan menghasilkan respons yang sama [38]. Pada tabel 2.8 merupakan

contoh *test case* pada pengujian dengan *Equivalance partitioning* pada pengujian *blackbox*.

Tabel 2.8 Contoh *Test Case* [38]

No	Skenario	Ekspektasi Output	Hasil Pengujian
1	Mengakses halaman *website* di *browser*	Menampilkan halaman *login*	Berhasil
2	Mengisi *form login* dan klik tombol *login*	Mengalihkan ke halaman *dashboard*	Berhasil

Hasil dari metode ini akan melihat apakah fungsi pada suatu halaman sudah bekerja dengan dan sesuai harapan. Dengan demikian akan diketahui fitur apa saja yang belum sesuai dengan ekspektasi yang diharapkan dan akan diperbaiki [39]. Pada Persamaan 2.1 merupakan perhitungan nilai kesuksesan dalam pengujian diperoleh dengan perhitungan sebagai berikut [38].

$$\text{Nilai Kesuksesan} = \frac{\text{Pengujian sukses}}{\text{Total pengujian}} \times 100 \quad (2.\ 1)$$

Berdasarkan ketentuan perhitungan diatas akan diketahui presentase kesuksesan pengujian halaman dan fitur didalamnya dengan cara membandingkan jumlah pengujian sukses dengan banyaknya pengujian yang dilakukan [38].

2.2.14 Rasio Efisiensi

Pada perhitungan hasil efisiensi tahap pertama menggunkan perhitungan rata rata dari setiap waktu atau barang yang digunakan untuk menghitung rata rata efisiensi yang akan dihitung nantinya. Berikut untuk membandingkan rasio efisiensi dengan menghitung rata-rata sebelum dan

rata-rata setelah digunakan [40]. Yang pertama menghitung rata-rata waktu yang dapat dilihat pada Persamaan 2.2.

$$\text{Rata} - \text{rata waktu} = \frac{\text{Total waktu minimum} + \text{Total waktu maksimum}}{2} \quad (2.2)$$

Setelah menghitung rata-rata waktu, selanjutnya bisa menghtung perbedaan rata-rata untuk melihat perbedaan sesuai rata-rata yanga ada dengan melihat Persamaan 2.3.

$$\text{Perbedaan rata} - \text{rata} = \text{rata} - \text{rata sebelum} - \text{rata} - \text{rata setalah} \quad (2.3)$$

Setelah itu bisa menghitung untuk persentase perbedaan untuk melihat persentase yang didapatkan yang sesuai dengan melihat Persamaan 2.4.

$$\text{Persentase perbedaan} = \left(\frac{\text{Perbedaan rata}-\text{rata}}{\text{Rata}-\text{rata sebelum}}\right) \times 100\% \quad (2.4)$$

Setelah mendapatkan hasilanya, selanjutnya bisa menghitung rasio efisiensi untk melihat rasio efisiensi yang sesuai dengan persamaan 2.5

$$\text{Rasio efisiensi} = \frac{\text{Rata}-\text{rat sebelum}}{\text{rata}-\text{rata setelah}} \quad (2.5)$$

Setelah menemukan rasio efisiensi maka selanjutnya menghitung persentase peningkatan efisensi, pada Persamaan 2.6 untuk menghitung persentase peningkatan efisiensi dengan rasio efisiensi dan rasio efisiensi sebelum untuk melihat berapa persen meningkat untuk tingkat efisiensinya [40].

$$persentase\ peningkatan\ efisiensi = \left(\frac{rasio\ efisiensi - rasio\ efisiensi\ sebelum}{rasio\ efisiensi\ sebelum}\right) \times 100\% \quad (2.6)$$

Setelah dihitung maka terlihat apakah persentase meningkat atau menurum dalam perhitungan persentase peningkatan efisiensi dan peningkatan akan terlihat secara signifikan menggunakan rumus tersebut [40].