201910370311197
Ivan Maulana Arrasid
Prodi Informatika

BAB II
TINJAUAN PUSTAKA

Bab ini menjelaskan terkait kajian, landasan, atau dasar teori untuk mendukung penelitian yang sedang dilakukan. Guna memberikan dasar pemahaman yang kokoh, merinci penelitian sebelumnya yang telah dilakukan, dan menunjukkan kebutuhan akan penelitian baru.

2.1 Sistem Pengelolaan Apotek

Pemanfaatan teknologi digital dalam pengelolaan apotek saat ini menjadi faktor penting dalam meningkatkan efisiensi operasional dan mutu layanan. Sistem pengelolaan apotek dikembangkan untuk memfasilitasi penyebaran informasi terkait apotek secara lebih luas. Selain itu, sistem ini berperan dalam mengelola data obat-obatan, mulai dari proses pengumpulan hingga penyimpanan, serta mengatur penggunaannya sesuai dengan kebutuhan perawatan pasien [23]. Sistem pengelolaan apotek memainkan peran penting dalam mengelola inventaris, meminimalkan kesalahan medis, dan mengotomatisasi proses manajemen rantai pasok. Serta dengan adanya sistem pengelolaan apotek mendukung kepatuhan terhadap regulasi, khususnya dalam mendokumentasikan data pasien dan resep obat secara akurat.

Selain itu pengelolaan stok obat menjadi lebih akurat melalui pemantauan real-time, notifikasi stok rendah, dan penyederhanaan proses stok opname, sehingga mencegah kekurangan maupun kelebihan stok yang sering terjadi pada sistem manual. Manajemen data pasien dan member juga menjadi lebih terorganisir, memungkinkan pencatatan riwayat resep. Dengan fitur laporan, apotek dapat menghasilkan data penjualan, keuangan, dan kinerja secara cepat dan akurat, yang berguna untuk pengambilan keputusan strategis. Selain itu, sistem otomatis membantu memenuhi persyaratan regulasi terkait pencatatan obat, sehingga mengurangi risiko kesalahan administratif.

2.2 *Agile Software Development Methods*

Agile Development Methods merupakan salah satu metode pengembangan software yang didasarkan pada kesamaan prinsip ataupun bersifat jangka pendek yang membutuhkan proses penyesuaian yang cepat dari pengembang dalam

merespon setiap perubahan. menggunakan pendekatan iteratif, membagi tugas menjadi bagian-bagian kecil dengan siklus 1-4 minggu. Setiap siklus mencakup analisis kebutuhan, desain, implementasi, pengujian, dan umpan balik. *Agile* memungkinkan perubahan di setiap tahap untuk diterapkan pada iterasi berikutnya, menghasilkan output secara bertahap. Metode ini menggabungkan praktik terbaik dari manajemen perubahan dan adaptasi metodologi, dengan fokus pada kepuasan pelanggan melalui keterlibatan langsung mereka [8].

2.3 *Personal Extreme Programming (PXP)*

Personal Extreme Programming (PXP) merupakan metode pengembangan perangkat lunak yang ditujukan bagi pengembang tunggal. Metode ini merupakan hasil integrasi antara *Extreme Programming (XP)* dengan *Personal Software Process (PSP)*, yang bertujuan untuk membantu pengembang individu mengelola proyek secara efektif. PSP mencatat bahwa pengembang tunggal sering kali kesulitan mengalokasikan waktu dan usaha secara optimal dalam menyelesaikan proyek yang kompleks karena kekhawatiran akan penundaan dan ketidaksesuaian dengan tenggat waktu yang ditetapkan [24].

Dalam penerapannya, PXP tetap mempertahankan prinsip-prinsip PSP, tetapi meminimalkan dokumentasi dan pemeliharaan yang berlebihan. Metode ini menggunakan pendekatan iteratif yang memungkinkan pengembang lebih responsif terhadap perubahan selama proses pengembangan. PXP juga mengintegrasikan elemen penting dari XP, seperti umpan balik, komunikasi, rasa hormat, dan kesederhanaan, untuk menjaga kualitas dan efisiensi pengembangan [24]. Berikut merupakan tahapan dari metode pengembangan Personal Extreme Programming (PXP)

2.3.1 Requirement

Tahap requirement merupakan tahap awal dalam metode *Personal Extreme Programming (PXP)* yang bertujuan untuk melakukan identifikasi dan pendefinisian terhadap kebutuhan sistem yang akan dibangun [25]. Tahap ini memiliki peran penting karena menjadi dasar dalam menentukan fungsi, fitur, serta ruang lingkup sistem yang akan dibangun.

Proses requirement pada personal extreme programming ini dilakukan melalui pengumpulan informasi dari pihak yang berkepentingan, seperti pengguna akhir atau pemilik bisnis, sehingga diperoleh pemahaman yang menyeluruh mengenai fungsi sistem, permasalahan yang terjadi, serta batasan yang harus diperhatikan dalam proses pengembangan sistem perangkat lunak [26]. Berbagai metode dapat dimanfaatkan, salah satunya melalui kegiatan observasi dan wawancara dengan pihak klien yang terlibat [27]. wawancara dilakukan kepada pihak yang berkepentingan untuk menggali kebutuhan dan kendala yang dialami. Hasil dari analisis tersebut kemudian disajikan dalam bentuk user story [28].

2.3.1.1 Observasi

Observasi merupakan metode pengumpulan kebutuhan yang dilakukan melalui pengamatan langsung terhadap aktivitas kerja atau proses bisnis yang sedang berjalan di lingkungan pengguna tanpa mengganggu jalannya proses tersebut [29]. Dalam konteks pengembangan sistem informasi, metode ini digunakan untuk memperoleh pemahaman mengenai alur kerja sistem yang ada serta untuk mengidentifikasi permasalahan yang terjadi di lapangan. Observasi juga memungkinkan pengembang untuk melihat secara langsung permasalahan, hambatan, dan inefisiensi yang terjadi dalam proses yang sedang berjalan sehingga sistem yang dirancang benar-benar mampu menjawab permasalahan nyata di lapangan [30]. Menurut Karl Wiegers [31] Observasi menghasilkan informasi mengenai aktivitas pengguna, permasalahan sistem, serta kebutuhan yang dapat digunakan sebagai dasar dalam penyusunan kebutuhan sistem.

2.3.1.2 Wawancara

Wawancara merupakan metode elisitasi kebutuhan yang dilakukan melalui kominikasi dua arah secara langsung antara pengembang dengan pengguna atau pemangku kepentingan untuk memperoleh pemahaman yang lebih mendalam mengenai kebutuhan sistem [30]. egiatan ini dilakukan dengan melibatkan pihak yang berhubungan langsung dengan permasalahan yang diteliti guna memperoleh data dan informasi yang relevan [32].

Wawancara memungkinkan peneliti untuk mengumpulkan informasi secara komprehensif dan mendalam mengenai permasalahan yang sedang dikaji [33]. Dalam Agile Software Development, kebutuhan sistem umumnya digali dari product owner atau pihak yang mewakili kepentingan bisnis dan memiliki pengetahuan menyeluruh terhadap sistem yang berjalan [34]. Selain itu, menurut Ian Sommervile [35] menyatakan bahwa requirement dapat diperoleh dari *key stakeholder* yang memiliki pemahaman mendalam terhadap domain sistem. Oleh karena itu, penggunaan satu responden dalam wawancara dapat dianggap memadai selama responden tersebut merupakan pihak yang memahami keseluruhan proses operasional sistem.

Jika wawancara hanya menggunakan satu pertanyaan umum, maka informasi yang diperoleh cenderung terlalu luas, kurang terarah, dan berpotensi tidak mencakup seluruh kebutuhan sistem. Oleh karena itu, penyusunan beberapa pertanyaan dilakukan untuk meningkatkan kualitas data yang diperoleh. Menurut John W. Creswell (2014) [36], wawancara kualitatif menggunakan beberapa pertanyaan terbuka yang disusun untuk mengeksplorasi fenomena secara mendalam dan terarah. Jumlah pertanyaan dalam wawancara tidak ditentukan secara baku, melainkan disesuaikan dengan kebutuhan informasi yang ingin digali. Penggunaan beberapa pertanyaan ini bertujuan untuk memastikan bahwa data yang diperoleh tidak bersifat umum, tetapi spesifik dan mencakup seluruh kebutuhan sistem. Hal ini sejalan dengan pendapat Lexy J. Moleong (2018) [37] yang menyatakan bahwa wawancara perlu dikembangkan ke dalam beberapa pertanyaan agar menghasilkan data yang lebih rinci dan mendalam. Pendekatan ini memungkinkan perolehan data kualitatif yang komprehensif, yang sangat penting pada tahap awal pengembangan sistem informasi, khususnya dalam proses identifikasi kebutuhan [38].

Wawancara tersebut tidak hanya bertujuan untuk memahami proses bisnis yang sedang berjalan, tetapi juga untuk mengidentifikasi permasalahan serta fungsionalitas yang diharapkan oleh pengguna agar sistem yang dihasilkan adaptif dan mudah digunakan [39]. Dalam konteks analisis kebutuhan sistem, pertanyaan wawancara disusun berdasarkan topik dalam requirements engineering seperti aksi (proses), konsep (data), permasalahan, batasan, dan tujuan sistem. Menurut Ian Sommerville [40] dan Roger S. Pressman [30], analisis kebutuhan mencakup

pemahaman terhadap proses bisnis, data yang dikelola, kendala yang terjadi, aturan, serta kebutuhan dan harapan pengguna. Berdasarkan konsep tersebut, peneliti mengadaptasi beberapa aspek yang digunakan sebagai dasar dalam penyusunan wawancara, yaitu (1) aspek proses bisnis untuk memahami alur kerja sistem, (2) aspek kebutuhan data untuk mengidentifikasi data yang dikelola, (3) aspek identifikasi permasalahan untuk mengetahui kendala pada sistem yang sedang berjalan, (4) aspek aturan bisnis untuk memahami kebijakan yang berlaku, (5) serta aspek kebutuhan pengguna dan ekspektasi sistem untuk menggali kebutuhan tambahan dan harapan pengguna terhadap sistem yang akan dikembangkan. Oleh karena itu, aspek-aspek tersebut digunakan sebagai dasar dalam penyusunan pertanyaan wawancara agar informasi yang diperoleh bersifat terarah dan sesuai dengan kebutuhan sistem.

2.3.1.3 Analisis Kebutuhan Sistem

Analisis kebutuhan adalah proses mengolah dan menginterpretasikan seluruh informasi yang telah dikumpulkan melalui observasi dan wawancara untuk menghasilkan definisi kebutuhan sistem yang jelas, lengkap, dan tidak ambigu [40]. Pendekatan analisis kebutuhan dalam penelitian ini dilakukan dengan mengidentifikasi masalah, menentukan solusi tindakan, menetapkan tujuan atau manfaat solusi, serta merumuskan kebutuhan sistem [34]. Pendekatan ini bertujuan untuk menjamin bahwa sistem yang dikembangkan selaras dengan kebutuhan pengguna dan mampu mengatasi permasalahan yang ada.

Analisis kebutuhan dilakukan melalui identifikasi permasalahan pada sistem yang sedang berjalan, kemudian dilanjutkan dengan perumusan kebutuhan sistem sebagai solusi yang diharapkan. Dalam konteks agile, solusi difokuskan pada fitur yang memberikan nilai langsung bagi pengguna [34]. Manfaat atau tujuan tindakan menjadi hasil yang diharapkan dari penerapan solusi dalam sistem. Penentuan tujuan membantu proses pengembangan sistem agar lebih terarah sesuai dengan kebutuhan pengguna. Penetapan tujuan ini dilakukan agar solusi yang dikembangkan tidak hanya menyelesaikan permasalahan yang ada, tetapi juga meningkatkan kualitas kinerja sistem yang dapat dirasakan secara langsung oleh pengguna [31]. Selanjutnya Tahap akhir adalah merumuskan kebutuhan sistem

yang menggambarkan fungsi atau layanan yang harus disediakan oleh sistem berdasarkan hasil identifikasi masalah dan solusi yang telah ditentukan [30]. Kebutuhan sistem yang telah dikumpulkan kemudian diubah menjadi user story, yang menggambarkan fungsi dan hasil yang diharapkan dari sistem [41].

2.3.1.4 Penentuan User Story

User story merupakan salah satu teknik dalam pengembangan perangkat lunak berbasis *agile* yang digunakan untuk menggambarkan kebutuhan sistem dari perspektif pengguna [30]. Dalam metode PXP, kebutuhan sistem dituliskan dalam bentuk user story dengan format "Sebagai <jenis pengguna>, saya ingin <melakukan tindakan tertentu> sehingga <mendapatkan manfaat dari tindakan tersebut>" [9]. Hasil penyusunan user story ini menjadi acuan bagi pengembang dalam merancang sistem agar sesuai dengan kebutuhan stakeholder. Penggunaan user story membantu pengembang dalam memahami kebutuhan pengguna secara lebih terstruktur [42]. Keluaran dari tahap requirement adalah daftar kebutuhan sistem yang telah terdokumentasi dalam bentuk user story, yang kemudian digunakan sebagai dasar dalam tahap planning pengembangan sistem.

2.3.2 Planning

Tahap Planning dalam Personal Extreme Programming (PXP) adalah merupakan tahap penting dalam menerjemahkan kebutuhan sistem menjadi jadwal kerja yang dapat dicapai secara realistis. Berbeda dengan pendekatan perencanaan tradisional, planning dalam PXP lebih bersifat fleksibel dan menyesuaikan dengan kemampuan serta estimasi produktivitas dari satu orang pengembang [12].

2.3.2.1 Estimasi Waktu Penyelesaian User Story

Salah satu kegiatan utama dalam tahap planning adalah menentukan estimasi waktu pengembangan. Estimasi waktu untuk setiap User Story dihitung menggunakan pendekatan Story Point, yang menggambarkan tingkat usaha yang diperlukan untuk menyelesaikan setiap fitur [25]. Estimasi waktu dilakukan untuk memperkirakan durasi pengerjaan setiap kebutuhan sistem atau user story sehingga pengembang dapat mengatur jadwal pengerjaan dan mengontrol kemajuan proyek pada setiap iterasi. Dalam menentukan beban kerja setiap User Story, PXP

menggunakan satuan Story Points [43]. Penentuan nilai Story Point dilakukan dengan mempertimbangkan tingkat kesulitan dan durasi pengerjaan yang diperkirakan oleh pengembang [25]. Story Point bukanlah representasi jam kerja absolut, melainkan satuan relatif yang menggambarkan tingkat kompleksitas, risiko, dan upaya yang dibutuhkan untuk menyelesaikan sebuah User Story [44]. Nilai story point memiliki hubungan yang sebanding dengan durasi pengerjaan [45]. Penentuan nilai tersebut didasarkan pada penilaian subjektif pengembang yang dipengaruhi oleh pengalaman sebelumnya. Besaran story point dapat dinyatakan dalam satuan waktu, seperti jam atau hari, dan umumnya menggunakan asumsi bahwa 1 story point setara dengan 2 hari kerja ideal [45]. Dengan adanya estimasi tersebut, peneliti dapat menentukan jumlah pekerjaan yang dapat diselesaikan dalam satu iterasi secara lebih terukur [24].

2.3.2.2 Penentuan Prioritas

Prioritisasi kebutuhan merupakan proses penting dalam pengembangan perangkat lunak yang bertujuan untuk menentukan urutan implementasi kebutuhan berdasarkan tingkat kepentingannya. Teknik prioritisasi digunakan untuk membantu pengembang dalam memilih kebutuhan yang harus diimplementasikan terlebih dahulu agar sesuai dengan tujuan sistem [21]. Dalam pengembangan berbasis Agile, prioritas kebutuhan menjadi sangat penting karena proses pengembangan dilakukan secara iteratif dan bertahap. Pendekatan Agile menekankan pentingnya pengembangan fitur secara bertahap berdasarkan nilai yang diberikan kepada pengguna [7] Salah satu pendekatan yang digunakan dalam menentukan prioritas kebutuhan adalah kombinasi metode MoSCoW dan Analytical Hierarchy Process (AHP). Penelitian [17] menunjukkan bahwa kombinasi teknik kategorisasi kebutuhan dan metode perbandingan berpasangan dapat menghasilkan prioritas yang lebih akurat dalam pengambilan keputusan.

2.3.2.2.1 Menerapkan Metode MoSCoW

Metode MoSCoW Method adalah metode prioritisasi dalam pengembangan perangkat lunak yang mengelompokkan kebutuhan menjadi empat kategori yaitu Must-Have (M), Should-Have (S), Could-Have (C), dan Won’t-Have (W)[46]

a. *Must-Have* (M): Fitur atau kebutuhan yang harus ada dalam produk. Ini adalah elemen yang kritis dan menjadi inti utama dari proyek[46].
b. *Should-Have* (S): Fitur atau kebutuhan yang diinginkan dan akan memberikan nilai tambah jika dimasukkan, tetapi bukan keharusan. kebutuhan ini memberikan kontribusi signifikan terhadap keberhasilan proyek, tetapi ketiadaannya tidak akan menyebabkan kegagalan sistem secara keseluruhan[46]
c. *Could-Have* (C): Fitur atau kebutuhan yang diinginkan, tetapi tidak kritis. Kebutuhan ini biasanya berupa fitur tambahan (nice-to-have) yang dapat meningkatkan kualitas sistem, namun dapat ditunda apabila terdapat keterbatasan waktu atau sumber daya[46]
d. *Won't-Have* (W): Fitur atau Kebutuhan tersebut dianggap tidak penting atau belum diperlukan, sehingga dapat dipertimbangkan untuk pengembangan di masa mendatang atau pada proyek terpisah [46].

4 kategori prioritas ini disematkan pada tiap *user story*. Prioritas *Must have* artinya *user story* tersebut adalah kebutuhan yang sangat penting dan harus ada. Kemudian *Should hav*e diberikan untuk kebutuhan yang penting namun masih dapat diabaikan dalam waktu singkat. *Could have* untuk kebutuhan yang dapat memberikan manfaat tambahan namun tidak esensial. *Won't have* untuk kebutuhan yang telah dibahas namun disepakati untuk diabaikan dalam *deliverable* saat ini. Metode ini membantu tim untuk fokus pada kebutuhan yang paling krusial terlebih dahulu dalam pengembangan perangkat lunak, sehingga memastikan pengembangan yang efisien dan efektif [9]. Penentuan prioritas MoSCoW dapat dilakukan dengan melibatkan pemilik bisnis yang mempunyai pemahaman tetang kebutuhan bisnis [40]. Pendekatan ini bertujuan untuk memperoleh penilaian langsung dari pihak yang memahami kebutuhan bisnis secara mendalam [40]

2.3.2.2.2 Menyusun Pairwise Comparison Matrix

Setelah kebutuhan dikelompokkan menggunakan metode MoSCoW, langkah selanjutnya adalah melakukan perbandingan antar user story dalam kategori yang sama menggunakan metode AHP. Perbandingan dilakukan dengan teknik pairwise comparison, yaitu membandingkan setiap kebutuhan secara berpasangan untuk

menentukan tingkat kepentingannya relatif terhadap kebutuhan lainnya [50]. Kebutuhan tersebut kemudian digunakan dalam metode AHP dengan membentuk matriks perbandingan antar kriteria yang nilainya ditentukan berdasarkan skala penilaian [47].

Tabel 2.1 Skala *Pairwise comparison [47]*

Intensitas Kepentingan	**Definisi**	**Penjelasan**
1	Kedua elemen memiliki tingkat kepentingan yang setara	Kedua elemen memberikan kontribusi atau pengaruh yang sama terhadap tujuan yang dinilai.
3	Salah satu elemen sedikit lebih penting dibandingkan elemen lainnya	Berdasarkan pengalaman dan penilaian, terdapat kecenderungan yang sedikit lebih mendukung satu elemen dibandingkan elemen lainnya.
5	Salah satu elemen lebih penting dibandingkan elemen lainnya	Pengalaman dan pertimbangan menunjukkan bahwa satu elemen memiliki pengaruh yang lebih kuat dibandingkan elemen lainnya.
7	Salah satu elemen jauh lebih penting dibandingkan elemen lainnya	Satu elemen menunjukkan dominasi yang kuat dalam memengaruhi tujuan dibandingkan elemen lainnya.
9	Salah satu elemen memiliki tingkat kepentingan yang sangat dominan	Satu elemen dianggap memiliki keunggulan mutlak dibandingkan pasangannya dengan tingkat keyakinan yang sangat tinggi.
2, 4, 6, 8	Nilai antara dua tingkat kepentingan yang berurutan	Digunakan sebagai nilai kompromi ketika tingkat kepentingan berada di antara dua skala utama.

Kebalikan	Apabila elemen i dibandingkan dengan elemen j menghasilkan suatu nilai tertentu, maka perbandingan j terhadap i adalah nilai kebalikannya.

Tabel 2.*1* menunjukkan skala perbandingan berpasangan (pairwise comparison) yang digunakan dalam metode Analytical Hierarchy Process (AHP) untuk menilai tingkat kepentingan relatif antar elemen atau kriteria. Skala ini memungkinkan penilaian yang lebih sistematis dan terukur dalam proses penentuan prioritas menggunakan AHP. Hasil perbandingan tersebut disusun dalam bentuk matriks yang disebut matriks perbandingan berpasangan (pairwise comparison matrix), yang digunakan sebagai dasar dalam perhitungan prioritas kebutuhan [15]. Matriks n x n merupakan jumlah user story dalam satu kategori. Jumlah perbandingan yang dilakukan dapat dihitung menggunakan rumus:

Persamaan 1 *[19]* Jumlah pairwise-comparison $= \frac{n(n-1)}{2}$

Rumus ini menunjukkan bahwa jumlah perbandingan bergantung pada banyaknya kebutuhan yang dianalisis [19].

2.3.2.2.3 Menghitung Rata-Rata dan Normalisasi Matriks

Setelah matriks perbandingan terbentuk, dilakukan proses normalisasi untuk memperoleh bobot prioritas dari setiap kebutuhan. Normalisasi dilakukan dengan membagi setiap nilai matriks dengan jumlah kolomnya menggunakan persamaan:

Persamaan 2 [48] $w_{ij} = \frac{x_{ij}}{\sum_{i=1}^{n} x_{ij}}$

Dimana x_{ij} merupakan setiap nilai pada kolom terkait dan $\sum_{i=1}^{n} x_{ij}$ merupakan jumlah kolom terkait [48]. Selanjutnya, dilakukan perhitungan rata-rata setiap baris untuk memperoleh bobot prioritas masing-masing user story. Proses ini menghasilkan nilai yang merepresentasikan tingkat kepentingan relatif dari setiap kebutuhan dalam kategori yang sama [15]

2.3.2.2.4 Menghitunng Nilai Prioritas dalam Kategori yang sama

Nilai prioritas user story diperoleh dari hasil perhitungan bobot menggunakan metode AHP dengan persamaan:

Persamaan 3 [48] $$w_i = \frac{\sum_{i=1}^{n} w_{ij}}{n}$$

Dimana w_i merupakan nilai prioritas user storynya, n merupakan jumlah user story, dan $\sum_{i=1}^{n} w_{ij}$ merupakan jumlah setiap baris dari normalisasi matriks [48].

Proses ini dilakukan untuk setiap kategori dalam metode MoSCoW secara bertahap hingga seluruh kebutuhan memperoleh nilai prioritas. Dengan pendekatan ini, metode MoSCoW-AHP mampu menghasilkan prioritas kebutuhan yang lebih terstruktur dan mendukung pengambilan keputusan secara lebih efektif [17]

2.3.2.3 Release Planning

Tahap planning dilanjutkan dengan release planning. Pada setiap iterasi, terdapat sejumlah user story yang akan dikerjakan selama proses berlangsung[49]. Penentuan berapa banyak user story yang dimasukkan ke tiap iterasi umumnya didasarkan pada velocity, yaitu jumlah story point yang berhasil diselesaikan. Pada iterasi pertama, nilai *velocity* belum dapat ditentukan secara akurat karena belum tersedia data historis dari hasil pengembangan sebelumnya, Oleh karena itu, estimasi velocity ditentukan oleh pengembang berdasarkan pengalaman dalam menyelesaikan sejumlah story point dalam satu iterasi serta mempertimbangkan kebutuhan selama proses pengembangan sistem [45]. Nilai tersebut kemudian akan disesuaikan berdasarkan hasil aktual pada iterasi pertama sehingga diperoleh nilai velocity yang lebih representatif untuk iterasi selanjutnya [50]. Beberapa implementasi PXP dan studi terkait menyebut bahwa velocity dihitung dari total story point yang selesai setiap iterasi dan digunakan sebagai batas kapasitas untuk memilih user story pada iterasi berikutnya [25].

2.3.3 Iteration Initialization

Dalam metode Personal Extreme Programming (PXP), tahap *iteration initialization* digunakan untuk menandai dimulainya suatu iterasi dalam pengembangan sistem. Proses ini diawali dengan pemilihan user story yang akan menjadi fokus utama selama iterasi berlangsung. User story tersebut berasal dari hasil perencanaan pada tahap sebelumnya [51]. Pada tahap *iteration initialization*, pengembang menentukan user story yang akan dikerjakan dalam satu iterasi berdasarkan tahap *planning* [52]. Pemilihan user story dilakukan dengan

mempertimbangkan prioritas kebutuhan sistem serta kapasitas pengembang yang direpresentasikan dalam bentuk velocity.

2.3.4 Desain

Tahapan design merupakan proses perancangan struktur sistem yang akan dikembangkan berdasarkan kebutuhan yang telah diidentifikasi sebelumnya. Pada metode Personal Extreme Programming, proses design dilakukan secara sederhana dan berorientasi pada objek agar sistem mudah dikembangkan dan dipelihara. Pengembang harus merancang sistem agar hanya memenuhi kebutuhan pengguna saat ini tanpa mencoba memprediksi kebutuhan di masa depan. Metode perancangan ditentukan oleh pengembang dengan tetap mempertimbangkan penggunaan pendekatan yang sederhana [24].

Pada tahap ini, perancangan sistem bertujuan untuk memodelkan alur proses dan interaksi dalam sistem sebelum diimplementasikan dalam bentuk kode program [40]. Salah satu alat yang digunakan dalam tahap desain adalah *activity diagram*, yang digunakan untuk memvisualisasikan alur aktivitas dalam sistem secara terstruktur [30]. *Activity diagram* merupakan pengembangan dari *flowchart* yang dapat menggambarkan aliran kendali antar aktivitas [49]. Activity diagram digunakan untuk menggambarkan aktivitas yang berbeda dari sebuah sistem, subaktifitas, transisi, titik keputusan, aktivitas yang berjalan bersamaan, percabangan, penggabungan, fork penghubung, dll. Layaknya flowchart diagram, Diagram dimulai dari aktivitas awal dan diakhiri dengan aktivitas akhir [53]. Penggunaan activity diagram dalam PXP sejalan dengan prinsip kesederhanaan (simplicity), di mana desain dibuat secukupnya untuk mendukung proses coding tanpa menambah kompleksitas yang tidak diperlukan. Selain itu, activity diagram membantu dalam mengidentifikasi langkah-langkah proses secara sistematis sehingga dapat digunakan sebagai dasar dalam penyusunan unit testing dan implementasi kode [30].

2.3.5 Implementation

Tahapan implementasi merupakan proses pembangunan perangkat lunak berdasarkan hasil perancangan sistem yang telah dilakukan sebelumnya. Pada metode Personal Extreme Programming, implementasi dilakukan secara bertahap

sesuai dengan iterasi pengembangan yang telah direncanakan. Dalam kerangka kerja Personal Extreme Programming (PXP), tahap ini terdiri dari tiga sub-tahapan yang saling berkaitan [54].

2.3.5.1 Unit Testing

Tahapan pertama dilakukan unit testing sebelum proses coding dengan menerapkan pendekatan Test-Driven Development (TDD) [54]. unit testing merupakan proses pengujian terhadap bagian terkecil dari sistem seperti fungsi atau modul tertentu sebelum dilakukan integrasi sistem secara keseluruhan. Tujuan unit testing adalah memastikan bahwa setiap kebutuhan sistem telah diterjemahkan ke dalam skenario pengujian yang jelas. Hasil dari pengujian ini failed atau passed [49]. Dalam praktiknya, unit testing yang dibuat pada tahap awal akan menghasilkan failed karena kode belum tersedia [55]. Selanjutnya, pengembang menuliskan kode untuk memenuhi pengujian hingga seluruh pengujian berhasil dijalankan [55].

2.3.5.2 Code

Pada tahap code generation pengembang akan membuat kode program pada setiap modul dari user stories pada iterasi yang diimplementasikan [49]. Dalam pendekatan TDD, proses coding dilakukan untuk memenuhi pengujian yang telah dibuat sebelumnya sehingga kode yang dikembangkan sesuai dengan kebutuhan sistem [55].

2.3.5.3 Refactor

Setelah menyelesaikan implementasi kode selesai, maka akan dikerjakan pengoptimalan kode apabila dibutuhkan [49]. Pada tahap refactoring, yaitu merupakan proses perbaikan struktur kode program tanpa mengubah fungsi utama sistem. Refactoring dilakukan setelah kode berhasil memenuhi skenario unit testing dengan tujuan meningkatkan kualitas kode, mengurangi kompleksitas, serta mempermudah proses pengembangan sistem di masa mendatang [54].

2.3.6 System Testing

System testing merupakan tahap pengujian yang dilakukan untuk memastikan bahwa sistem yang telah dikembangkan berjalan sesuai dengan kebutuhan pengguna dan spesifikasi fungsional yang telah ditetapkan [30]. Dalam metode Personal Extreme Programming (PXP), system testing dilakukan sebagai bagian dari proses pengembangan yang bersifat iteratif dan dilakukan pada setiap increment sistem [12]. Black box testing merupakan salah satu metode dalam menguji perangkat lunak, dimana pengembang web bersama klien akan menguji dari segi spesifikasi fungsionalnya saja (tanpa melihat desain dan kode program) [49]. Pada metode ini, sistem diuji dengan memberikan input tertentu dan mengamati output yang dihasilkan, kemudian dibandingkan dengan spesifikasi fungsional yang telah ditentukan[40]. Black-box testing dilakukan oleh pengembang agar untuk memastikan kualitas sistem yang telah dibuat berjalan sebagaimana mestinya [49].

Dalam konteks PXP, pengujian sistem juga dapat melibatkan pengguna atau pemilik sistem untuk memastikan bahwa hasil pengembangan sesuai dengan kebutuhan nyata di lapangan [51]. Dengan dilakukannya system testing menggunakan metode black-box testing, pengembang dapat memastikan bahwa sistem yang dibangun telah memenuhi kebutuhan pengguna serta siap digunakan dalam operasional [56]

2.3.7 Retrospective

Pada tahap ini dilakukan peninjauan kembali terhadap setiap iterasi yang telah dilaksanakan sebelumnya. Pengembang harus mengevaluasi apakah estimasi waktu tugas sesuai dengan waktu aktual, serta mengidentifikasi penyebab keterlambatan untuk mencegah kesalahan estimasi (terlalu rendah atau terlalu tinggi) pada proyek berikutnya [24]. Apabila ditemukan kendala atau ketidaksesuaian dalam penerapan sistem, maka proses pengembangan akan kembali ke tahap iteration initialization untuk melakukan perbaikan, penyesuaian, maupun penyempurnaan yang dibutuhkan. Langkah ini bertujuan agar sistem yang dikembangkan benar-benar sesuai dengan harapan dan dapat berjalan dengan baik sebelum masuk ke tahap berikutnya. Proses evaluasi dan perbaikan yang dilakukan secara berkelanjutan

pada setiap iterasi juga membantu meningkatkan kualitas sistem yang dihasilkan [25].

2.4 *System Usability Scale (SUS)*

System Usability Scale (SUS) adalah alat evaluasi yang banyak digunakan untuk mengukur tingkat kebergunaan sebuah sistem atau produk. SUS dikembangkan oleh John Brooke pada tahun 1986 sebagai metode praktis untuk menilai kegunaan [57]. System Usability Scale (SUS) merupakan metode yang banyak digunakan untuk mengevaluasi tingkat kegunaan suatu sistem karena memiliki beberapa keunggulan [57], antara lain:

1. Perhitungannya relatif sederhana dan menghasilkan skor dalam rentang 0 hingga 100 sehingga mudah untuk dipahami.
2. Tidak membutuhkan biaya tambahan dalam penerapannya.
3. Tetap memberikan hasil yang valid dan reliabel meskipun menggunakan jumlah responden yang relatif sedikit.

Pada tahap evaluasi menggunakan SUS, responden diminta untuk menjawab sepuluh pernyataan yang telah disediakan. Berikut merupakan daftar pertanyaan SUS.

Tabel 2.2 Daftar pertanyaan metode SUS [58]

No	Pertanyaan	STS	TS	N	S	ST
1.	Saya berpikir akan menggunakan sistem ini lagi	1	2	3	4	5
2.	Saya merasa sistem ini rumit untuk digunakan	1	2	3	4	5
3.	Saya merasa sistem ini mudah digunakan	1	2	3	4	5
4.	Saya membutuhkan bantuan dari orang lain atau teknisi dalam menggunakan sistem ini	1	2	3	4	5
5.	Saya merasa fitur-fitur sistem ini berjalan dengan semestinya	1	2	3	4	5
6.	Saya merasa ada banyak hal yang tidak konsisten (tidak serasi) pada sistem ini	1	2	3	4	5
7.	Saya merasa orang lain akan memahami cara menggunakan sistem ini dengan cepat	1	2	3	4	5

8.	Saya merasa sistem ini membingungkan	1	2	3	4	5
9.	Saya merasa tidak ada hambatan dalam menggunakan sistem ini	1	2	3	4	5
10.	Saya perlu membiasakan diri terlebih dahulu sebelum menggunakan sistem ini	1	2	3	4	5

Tahap perhitungan data dilakukan setelah proses penyebaran kuesioner selesai dan seluruh respons telah diperoleh. Perhitungan skor pada kuesioner System Usability Scale (SUS) mengikuti sejumlah aturan tertentu untuk menentukan tingkat kegunaan aplikasi. Adapun aturan dalam perhitungan skor SUS adalah sebagai berikut:

Tabel 2.3 Aturan perhitungan skor SUS [58]

No	Aturan-aturan perhitungan skor SUS
1.	Untuk setiap pernyataan bernomor ganjil, skor diperoleh dengan mengurangi nilai jawaban responden dengan 1.
2.	Untuk setiap pernyataan bernomor genap, skor dihitung dengan mengurangkan nilai jawaban dari angka 5.
3.	Seluruh skor dari pernyataan ganjil dan genap kemudian dijumlahkan, lalu hasil total tersebut dikalikan dengan 2,5.

Setelah skor masing-masing responden diperoleh, langkah berikutnya adalah menghitung nilai rata-rata dengan menjumlahkan seluruh skor responden, kemudian dibagi dengan jumlah responden yang terlibat.

Persamaan 4 [59] $$\bar{x} = \frac{\sum x}{n}$$

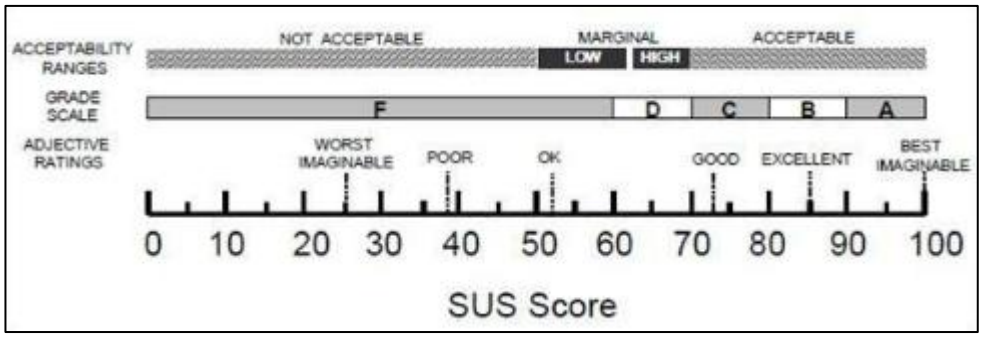
Keterangan:

$\bar{x}$: Skor rata-rata

$\sum x$: Jumlah Skor SUS

n : Jumlah Responden

Skor SUS tidak diinterpretasikan sebagai persentase, melainkan sebagai skor usability relatif. Interpretasi skor SUS umumnya dilihat dari tiga sudut pandang yaitu berdasarkan **Gambar 2.*1***



Gambar 2.1 SUS Score [58]

Nilai rata-rata global SUS adalah sekitar 68. Sistem dengan skor di atas 68 dikategorikan memiliki usability di atas rata-rata, sedangkan skor di atas 80 menunjukkan tingkat usability yang sangat baik.

2.5 Penelitian Terdahulu

Penelitian terdahulu digunakan sebagai rujukan guna menunjang penelitian baru yang akan dikerjakan. Penelitian terdahulu juga digunakan sebagai dasar dalam penentuan algoritma atau metode yang sesuai untuk diaplikasikan dalam penelitian. Adapun penulis menggunakan penelitian terdahulu sebagai referensi dalam penelitian terkait Implementasi Metode *Personal Extreme Programming* dengan Metode Prioritas *MoSCoW-AHP* dalam Pengembangan Sistem Pengelolaan Apotek.

Tabel 2.4 Penelitian Terdahulu

No.	Nama Peneliti	Tahun	Judul	Pembahasan
1.	Cindy E Ch Lahama1, Chriestie Ellyanne Juliet Clara Montolalu, dan Edwin Tenda	2024	*Web-Based Pharmacy Information System Using Personal Extreme Programming (PXP) Method with MVC Architecture*	Hasil penelitian berupa sistem informasi apotek berbasis web yang dikembangkan dengan metode *Personal Extreme Programming*. Sistem tersebut mampu mengelola data produk, transaksi penjualan dan pembelian, serta menyediakan laporan transaksi. Perancangan sistem menggunakan arsitektur

				Model-View-Controller (MVC) untuk memisahkan aspek logika bisnis, antarmuka pengguna, dan interaksi sistem ke dalam komponen yang berbed.[23]
2.	Ahmad,.M. F. dan I Putu Deny, A. S. P.	2020	Rancang Bangun Sistem Informasi Buku Tamu Pada Dinas Pemuda, Olahraga dan Pariwisata Kota Balikpapan Dengan Metode Personal Extreme Programming	Penelitian ini menghasilkan sistem informasi buku tamu (SI-KUTA) berbasis web yang dimanfaatkan oleh pegawai, khususnya pada subbagian umum di Dinas Pemuda, Olahraga, dan Pariwisata Kota Balikpapan. Sistem ini digunakan untuk merekap, mengelola, dan menyimpan data pengunjung DPOP. Proses pengembangan dilakukan dengan metode Personal Extreme Programming (PXP), sementara pengujian sistem menggunakan metode *black box testing*. Sistem yang dibangun dirancang agar fleksibel terhadap perubahan kebutuhan klien. Selain itu, pendekatan ini mampu meningkatkan efektivitas komunikasi antara pengembang dan klien, serta meningkatkan efisiensi kerja dan meminimalkan kesalahan dalam pengelolaan data yang

				sebelumnya dilakukan secara konvensional.[13]
3.	Tegar Palyus Fiqar, Nur Fajri Azhar, dan Fathima Azzahra	2022	Implementasi Metode Personal Extreme Programming dalam Pengembangan Sistem Informasi Manajemen Hak Paten pada Sentra HKI ITK	Sistem Informasi Manajemen (SIM) Hak Paten telah dikembangkan menggunakan metodologi *Personal Extreme Programming* (PXP) dengan penentuan prioritas fitur yang ditentukan melalui metode MoSCoW berdasarkan user story. Sistem ini mencakup 14 fitur yang dikembangkan dalam 5. Pelatihan kepada pemilik aplikasi menunjukkan bahwa pengguna dapat mengoperasikan aplikasi sesuai kebutuhan. Ke depannya, disarankan untuk mengadakan sosialisasi kepada seluruh civitas akademika mengenai aplikasi HKI yang telah dirancang.[60]
4.	M. S. Jahan, F. Azam, M. W. Anwar, A. Amjad, dan K. Ayub,	2019	*A Novel Approach for Software Requirement Prioritization*	Penelitian ini memperkenalkan teknik Prioritisasi Kebutuhan Perangkat Lunak (Software Requirements Prioritization) yang ditingkatkan dengan studi kasus sistem manajemen perpustakaan, yang menggabungkan keuntungan dari metode *MoSCoW* dan *AHP*. Bagian validasi menunjukkan

				bahwa akan diperlukan lebih sedikit perbandingan pasangan, jika menerapkan *AHP* dalam metode *MoSCoW* kita hanya memerlukan 45 perbandingan pasangan dibandingkan dengan 210 jika menggunakan *AHP* secara terpisah. Hal ini mengurangi kompleksitas perbandingan pasangan dan memberikan hasil yang lebih cepat. [17]

Berdasarkan **Tabel 2.*4***, penelitian terdahulu menunjukkan bahwa metode Personal Extreme Programming (PXP) banyak digunakan dalam pengembangan sistem informasi berbasis web karena fleksibel terhadap perubahan kebutuhan dan efektif meningkatkan efisiensi pengelolaan data. Namun, penentuan prioritas kebutuhan pada penelitian sebelumnya umumnya masih dilakukan secara kualitatif menggunakan metode MoSCoW tanpa pembobotan kuantitatif. Sementara itu, penelitian mengenai MoSCoW–AHP membuktikan bahwa kombinasi kedua metode tersebut mampu mengurangi kompleksitas *pairwise comparison* dibandingkan AHP murni. Berdasarkan celah tersebut, belum terdapat penelitian yang mengintegrasikan PXP dengan MoSCoW–AHP dalam pengembangan sistem pengelolaan apotek, sehingga penelitian ini dilakukan untuk mengisi kekosongan tersebut