BAB II
TINJAUAN PUSTAKA DAN LANDASAN TEORI

2.1 Tinjauan Pustaka

Dalam penelitian ini, tinjauan pustaka telah dilakukan terhadap beberapa jurnal penelitian terbaru yang relevan dengan metode penelitian yang akan diteliti. Berikut penjelasan lebih lanjut dari hasil tinjauan penelitian sebelumnya.

Dalam tinjauan pustaka yang pertama, Peeyush Pareek dan Swati Chande di tahun 2021 melakukan penelitian dengan judul *"Grey Box Approach for Mobile Application Testing*". Penelitian ini dilakukan untuk menguji sebuah aplikasi *mobile* menggunakan pendekatan metode *grey-box testing*. *Grey-box testing* merupakan metode yang menggabungkan antara *white-box testing* dengan *black-box testing*. Pengujian ini membutuhkan pengetahuan mengenai sistem kerja internal aplikasi dan juga memiliki pengetahuan tentang aspek-aspek fundamental dari sistem. Secara objektif penelitian ini berfokus kepada fungsionalitas *User Interface*, akurasi, dan penyajian. Berdasarkan penelitian tersebut diperoleh hasil bahwa metode *grey-box testing* membutuhkan *effort* dan waktu yang lebih sedikit dibandingkan *black-box testing*. Selain itu, *Grey-box testing* sangat cocok untuk *requirement* yang cukup detail mengenai sistem internal. Meskipun pengujian pada aplikasi berbasis mobile memerlukan *test case* yang lebih spesifik, tetapi pengujian *grey-box* testing masih sangat efektif dibandingkan metode *white-box testing* maupun *black-box testing* [28].

Dalam tinjauan pustaka kedua, Mengqing TanLi, Ying Zhang, Yulin Wang, dan Yan Jiang di tahun 2021 melakukan penelitian dengan judul "*Grey-Box Technique of Software Integration Testing Based on Message*". Penelitian ini dilakukan untuk menguji sebuah sistem yang berfokus kepada sebuah fitur pesan dengan mengkombinasikan teknik *integration testing* dengan metode *grey-box testing*. *Integration testing* merupakan pengujian yang dilakukan untuk memvalidasi kesesuaian integrasi data dengan tampilan (GUI). Berdasarkan penelitian tersebut diperoleh hasil bahwa *grey-box testing* dapat meningkatkan efisiensi hingga 400% waktu pengujian yang dijalankan [29].

Dalam tinjauan pustaka ketiga, Samah Abu Salim, Rosziati Ibrahim, dan Jahari Abdul Wahab di tahun 2021 melakukan penelitian dengan judul "*Comparative Analysis of Software Testing Techniques for Mobile Applications*". Penelitian ini dilakukan untuk menganalisis komparasi teknik-teknik pengujian perangkat lunak untuk menguji aplikasi berbasis *mobile*. Pada penelitian ini, dilakukan komparasi terhadap beberapa metode dan teknik pengujian, antara lain; *black-box testing, white-box testing, grey-box testing, mutation testing, fuzzy testing,* dan *regression testing*. Objektif dari penelitian ini adalah menguji berbagai karakteristik aplikasi Android seperti *performance*, *reliability*, *memory*, *energy*, *security* dan *Interface Testing*. Berdasarkan penelitian tersebut diperoleh hasil perbandingan bahwa metode *white-box testing* meliputi cakupan pengujian yang menyeluruh dan mendalam terhadap kode dan struktur kode program. Sedangkan metode pengujian *black-box* tidak dapat melakukan pengujian yang menyeluruh dan mendalam terhadap kode dan struktur kode program, namun pengujian ini masih dapat membantu untuk menguji dari sisi pengguna dan memenuhi *requirement dari aplikasi*. Sedangkan metode pengujian *grey-box* sangat jauh lebih efektif dibandingkan teknik/metode lainnya untuk digunakan dalam pengujian integrasi sehingga pengembang memiliki lebih banyak waktu dalam memperbaiki *bug*. Selain itu, pengujian *grey-box* dapat membantu pengembang untuk mendapatkan *bug* yang

lumayan spesifik. Sedangkan teknik pengujian *mutation testing* merupakan teknik pengujian yang memakan banyak waktu dan sumber daya untuk menulis *test case* dan mutasi hasil uji cukup banyak dihasilkan dalam jumlah besar. Pengujian *mutation testing*, cocok digunakan terhadap kasus uji baru serta mendapatkan skor cakupan pernyataan hasil uji. Sedangkan teknik *regression testing* lebih cocok digunakan pada kondisi sistem yang memiliki perubahan/penambahan *requirement* fitur dan digabungkan dengan kondisi fitur sistem lama (*existing*) sehingga pengujian dilakukan untuk memastikan tidak terjadi perubahan pada fitur lama. Sedangkan pada teknik *fuzzy testing* dilakukan untuk mengidentifikasi semua *bug* dan masalah dalam sistem perangkat lunak yang dapat memengaruhi keamanan dan ancaman yang berbeda untuk mengatasi suatu program [30].

Dalam tinjauan pustaka keempat, Condro kartiko di tahun 2019 melakukan penelitian dengan judul "Evaluasi Kualitas Aplikasi Web Pemantau Menggunakan Model Pengujian Perangkat Lunak ISO/IEC 9126". Penelitian ini dilakukan untuk menguji kualitas sebuah aplikasi menggunakan pendekatan sebuah standardisasi ISO/IEC 9126. Standardisasi diperlukan sebagai tolak ukur seberapa jauh produk yang diuji memenuhi karakteristik dan kualitas sesuai standar tersebut. Standarisasi ISO/IEC 9126 merupakan salah satu standardisasi internasional yang digunakan sebagai standar kualitas pengujian sebuah *platform* digital. Terdapat enam aspek pengukuran kualitas berdasarkan ISO/IEC 9126 antara lain; 1) *functionality*, 2) *reliability*, 3) *usability*, 4) *efficiency*, 5) *maintainability*, 6) *portability*. Pada penelitian ini, penilaian hasil uji dibagi beberapa kelompok berdasarkan tiga kategori penilaian, yaitu 1) *unsatisfactory* antara 0-40%, 2) *marginal* antara 40-60%, 3) *satisfactory* antara 60-100%. Berdasarkan penelitian ini diperoleh hasil bahwa aspek *functionality* mendapatkan penilaian kategori *satisfactory* (76,55%), aspek *reliability* mendapatkan penilaian kategori *satisfactory* (99,14%), aspek *usability* mendapatkan penilaian kategori *marginal* (61,81%), aspek *efficiency* mendapatkan penilaian kategori *satisfactory* (66,8%). Sedangkan

pada aspek *maintainability* dan *portability* tidak dilakukan pengujian, karena aspek tersebut tidak terkait secara langsung dengan pengguna aplikasi. Sehingga dapat disimpulkan bahwa aplikasi yang diuji dapat dikatakan layak dan dapat digunakan sesuai pendekatan standardisasi ISO/IEC 9126 [31].

Dalam tinjauan pustaka yang kelima, Limia Kristiani, Clara Hetty Primasari, dan Alaysius Bagas P I di tahun 2021 melakukan penelitian dengan judul "*Improvement Recommendation for Functionality, Usability, and Efficiency Aspect in Credit Surveyor Application Using ISO 9126*". Dalam penelitian ini, dilakukan pengujian untuk mengetahui kualitas aplikasi *surveyor* kredit apakah sudah memenuhi standar ISO 9126 pada aspek fungsionalitas, kegunaan, dan efisiensi dari sudut pandang *surveyor* sebagai pengguna. Penelitian ini menggunakan metode *USE Questionnaire* untuk mendapatkan hasil *usability* yang meliputi kegunaan, kemudahan penggunaan, kemudahan pembelajaran, dan parameter kepuasan pengguna. Berdasarkan hasil analisis yang diperoleh dari pengguna, aspek *functionality* mendapatkan persentase kelayakan sebesar 81,98%, aspek *usability* mendapatkan persentase kelayakan sebesar 77,63%, dan aspek *efficiency* mendapatkan persentase kelayakan sebesar 75,54%. Hal ini dapat disimpulkan bahwa aplikasi *surveyor* kredit yang dikembangkan telah memenuhi standar aspek ISO 9126 serta terdapat beberapa rekomendasi seperti penambahan data pada fungsi *customer information*, meminimalisir sumber daya pengguna menetapkan standardisasi peralatan, dan melakukan pemeliharaan [32].

Tabel 2.1 Ringkasan Tinjauan Penelitian-Penelitian Sebelumnya

No	Tahun	Judul	Hasil
1.	2021	*Grey Box Approach for Mobile Application Testing*	Berdasarkan pengujian UI *Functionality* dan akurasi penyajian. Meskipun pengujian pada aplikasi berbasis *mobile* memerlukan *test case* yang lebih spesifik, tetapi

No	Tahun	Judul	Hasil
			pengujian *grey-box testing* masih sangat efektif dibandingkan metode *white-box* testing maupun *black-box testing*
2	2021	*Grey-Box Technique of Software Integration Testing Based on Message*	Berdasarkan pengujian metode *grey-box testing* yang dikombinasikan dengan *integration testing*, dapat meningkatkan efisiensi hingga 400% waktu pengujian yang dijalankan.
3	2021	*Comparative Analysis of Software Testing Techniques for Mobile Applications*	Metode *white-box testing* meliputi cakupan pengujian yang menyeluruh dan mendalam terhadap kode dan struktur kode program. Sedangkan metode *black-box testing* tidak dapat melakukan pengujian yang menyeluruh dan mendalam terhadap kode dan struktur kode program. Tetapi *black-box testing* masih dapat membantu untuk menguji dari sisi pengguna dan memenuhi *requirement dari aplikasi*. Sedangkan metode pengujian *grey-box* sangat jauh lebih efektif dibandingkan teknik/metode lainnya untuk digunakan dalam pengujian integrasi sehingga pengembang memiliki lebih banyak waktu dalam memperbaiki *bug*. Selain itu, pengujian *grey-box* dapat membantu pengembang untuk mendapatkan *bug* yang lumayan spesifik. Sedangkan teknik pengujian *mutation testing* merupakan teknik pengujian yang memakan banyak waktu dan sumber daya untuk menulis *test case* dan mutasi hasil uji cukup banyak dihasilkan dalam jumlah besar. Pengujian *mutation testing*, cocok digunakan terhadap kasus uji baru serta mendapatkan skor cakupan pernyataan hasil uji.

No	Tahun	Judul	Hasil
			Sedangkan teknik *regression testing* lebih cocok digunakan pada kondisi dimana sistem memiliki perubahan/penambahan *requirement* fitur dan digabungkan dengan kondisi fitur sistem lama (*existing*) sehingga pengujian dilakukan untuk memastikan tidak terjadi perubahan pada fitur lama. Sedangkan pada teknik *fuzzy testing* dilakukan untuk mengidentifikasi semua *bug* dan masalah dalam sistem perangkat lunak yang dapat memengaruhi keamanan dan ancaman yang berbeda untuk mengatasi suatu program.
4	2019	Evaluasi Kualitas Aplikasi Web Pemantau Menggunakan Model Pengujian Perangkat Lunak ISO/IEC 9126	Terdapat enam aspek yang diuji berdasarkan pendekatan standardisasi ISO/IEC 9126. 1) Aspek *functionality* mendapatkan penilaian kategori *satisfactory* (76,55%), 2) Aspek *reliability* mendapatkan penilaian kategori *satisfactory* (99,14%), 3) Aspek *usability* mendapatkan penilaian kategori *marginal* (61,81%), 4) Aspek *efficiency* mendapatkan penilaian kategori *satisfactory* (66,8%). Sedangkan pada aspek *maintainability* dan *portability* tidak dilakukan pengujian, karena aspek tersebut tidak terkait secara langsung dengan pengguna aplikasi. Sehingga dapat disimpulkan bahwa aplikasi yang diuji dapat dikatakan layak dan dapat digunakan sesuai pendekatan standardisasi ISO/IEC 9126.
5	2021	*Improvement Recommendation for Functionality, Usability, and Efficiency Aspect in Credit Surveyor*	Pengujian dilakukan untuk mengetahui kualitas aplikasi *surveyor* kredit apakah sudah memenuhi pendekatan standardisasi ISO 9126. Terdapat tiga aspek yang diuji, yaitu

No	Tahun	Judul	Hasil
		Application Using ISO 9126	*functionality*, *usability*, dan *efficiency* dari sudut pandang *surveyor* sebagai pengguna. Penelitian ini menggunakan metode *USE Questionnaire* untuk mendapatkan hasil *usability test*. Berdasarkan hasil analisis yang diperoleh dari pengguna, aspek *functionality* mendapatkan persentase kelayakan sebesar 81,98%, karakteristik *usability* mendapatkan persentase kelayakan sebesar 77,63%, dan karakteristik *efficiency* mendapatkan persentase kelayakan sebesar 75,54%. Hal ini dapat disimpulkan bahwa aplikasi *surveyor* kredit yang dikembangkan telah memenuhi pendekatan standardisasi ISO 9126 pada aspek tersebut dan terdapat beberapa rekomendasi seperti penambahan data pada fungsi *customer information*, meminimalisir sumber daya pengguna menetapkan standarisasi peralatan, dan melakukan pemeliharaan.

2.2 Landasan Teori

Berikut merupakan kajian mengenai teori-teori yang digunakan pada penelitian ini.

2.2.1 *Platform Marketplace* BUMDesa

Platform Marketplace BUMDesa merupakan *platform* yang dikembangkan oleh Unit Sentra Inovasi dan HAKI Institut Teknologi Telkom Purwokerto. *Platform* ini dikembangkan berdasarkan permintaan BUMDesa XYZ dengan tujuan untuk mendigitalisasikan proses bisnis di desa supaya terpantau sehingga dapat mengupayakan peningkatan bisnis bersama masyarakat di

Desa. Guna memberikan layanan kepada pengguna, *platform* ini dapat diakses melalui *website*, beberapa kategori pengguna dapat mengunduhnya melalui perangkat *mobile* android. Pengguna *platform marketplace* BUMDesa terdiri dari operator BUMDesma, operator BUMDesa serta para pemilik warung maupun distributor produk BUMDesa. Masing-masing pengguna memiliki fitur yang sudah disesuaikan dengan kebutuhan dan hak aksesnya. Tabel di bawah ini merupakan fitur dan hak akses *platform marketplace* BUMDesa.

Tabel 2.2 Fitur Platform Marketplace BUMDesa

Pengguna	***Platform***	**Fitur**
Operator BUMDesma	*Website*	*Login*
		Mendaftarkan BUMDesa
		Mengaktifkan BUMDesa
		Mendaftarkan admin BUMDesa
		Mengaktifkan admin BUMDesa
		Menambahkan data UMKM
		Mendaftarjan Warung (distributor)
		Mengaktifkan akun warung
		Membuat kategori produk
Operator BUMDesa	*Website*	Pendaftaran
		Login
		Menambahkan data UMKM
		Menambahkan data produk
		Membuat data kurir
		Mengatur harga ongkos kirim
		Menerima pesanan
		Mengirim pesanan
Pemilik Warung (Distributor)	Aplikasi *Mobile* Android	Pendaftaran
		Login
		Membeli produk BUMDesa
		Mengelola riwayat pembeli
		Menambah data inventory
		Menjual barang (*point of sales*)
		Menambah data member
		Mengelola riwayat penjualan

2.2.2 BUMDesa

Badan Usaha Milik Desa (BUMDesa) merupakan badan hukum yang diatur oleh Peraturan Pemerintah No.11 Tahun 2021. BUMDesa didirikan oleh desa dan/atau bersama desa-desa guna mengelola usaha, memanfaatkan aset, dan mengembangkan investasi [7]. BUMDesa memiliki wewenang untuk meningkatkan urusan ekonomi dan kegiatan usaha di desa [8].

Pembentukan BUMDesa didasari pada kebutuhan, potensi, serta kapasitas desa sebagai upaya peningkatan ekonomi dan kesejahteraan masyarakat. Oleh karena itu BUMDesa harus mencapai perubahan yang lebih baik bagi desa dalam meningkatkan perekonomian desa, meningkatkan pendapatan asli desa, meningkatkan pengelolaan potensi desa. BUMDesa memiliki peran besar dalam pertumbuhan dan pemerataan ekonomi desa [12].

2.2.3 *Marketplace*

Marketplace merupakan sebuah istilah yang berasal dari Bahasa Inggris, yakni dari kata “market” yang bermakna “pasar”, yang dapat diartikan sebagai sebuah tempat pertemuan dimana penjual dengan beragam produk yang dijual dan pembeli yang memiliki minat terhadap produk tersebut. Saat ini istilah *marketplace* sering digunakan untuk menggambarkan sebuah *platform online* yang menyediakan kegiatan jual/beli suatu barang ataupun jasa. Sedangkan proses transaksi secara elektronik ketika menggunakan *marketplace* dapat diartikan sebagai *e-commerce* [33].

Marketplace dapat dikategorikan ke dalam 3 jenis, antara lain; 1) *Marketplace Vertical*, yaitu *marketplace* yang menjual produk dari berbagai sumber namun produk yang dijual hanya terdiri satu jenis. Contohnya *marketplace* yang hanya menjual bensin; 2) *Marketplace Horizontal*, yaitu *marketplace* yang menjual produk

dari berbagai sumber dengan jenis-jenis produk yang masih keterkaitan satu dengan lainnya. Contohnya *marketplace* yang menjual produk komputer, aksesoris komputer, dan *spare-part* komputer*; 3) Marketplace Global*, yaitu *marketplace* yang menjual berbagai produk yang tidak berkaitan satu sama lain hingga penyediaan layanan jasa. Contohnya seperti Shopee dan Tokopedia [34].

2.2.4 Perangkat Lunak

Perangkat lunak merupakan suatu komponen yang tidak dapat disentuh secara fisik. Perangkat lunak dikembangkan dari sekumpulan kode yang diprogram sehingga dapat menerima permintaan, menjalankan perintah, dan memberikan informasi yang diminta oleh pengguna perangkat lunak tersebut [35]. Sedangkan tempat untuk menjalankan perangkat lunak dapat disebut juga sebagai sebuah *platform* [36].

2.2.5 Pengujian Perangkat Lunak

Pengujian perangkat lunak merupakan salah satu fase di dalam *software process models* (SPM). Fase tersebut disebut dengan *Umbrella Activities*, dimana pada fase ini dilakukan *software project and tracking control, risk management, software quality assurance, technical reviews, measurement, software configuration management, reusability management, work product preparation and production* [37]. Dalam proses pengembangan perangkat lunak terdapat beberapa persyaratan dan kebutuhan tertentu yang harus dipenuhi. Sehingga sangat penting untuk melakukan proses pengujian pada pengembangan perangkat lunak dengan melakukan untuk memastikan bahwa perangkat lunak yang dikembangkan sudah sesuai dan tidak ada kondisi kesalahan yang menyebabkan kegagalan saat digunakan oleh pengguna [17]. Pengujian perangkat lunak merupakan sebuah metode untuk mencari *error* di dalam

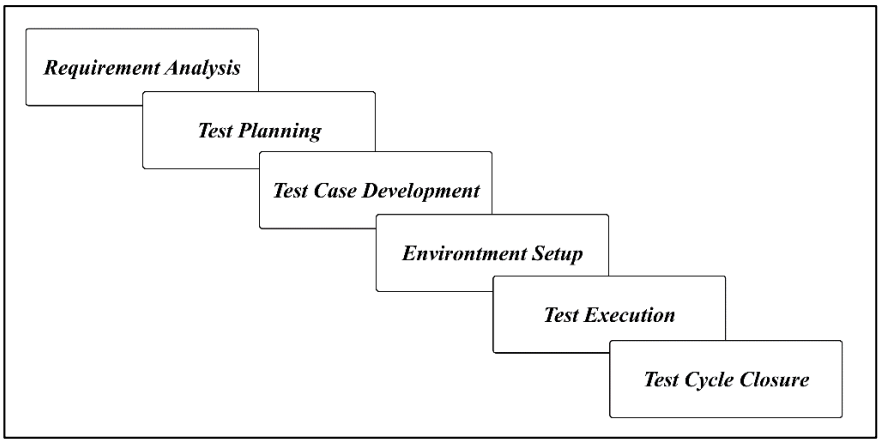
sebuah *software* dengan upaya tidak akan terjadi *bug* di masa mendatang [22].

2.2.6 Kualitas Perangkat Lunak

Kualitas merupakan sebuah konsep yang tidak terwujud, tidak mudah didefinisikan, tetapi setiap orang bisa merasakan ketika kualitas itu tidak ada [38]. Organisasi Internasional untuk standarisasi (ISO) mendefinisikan kualitas sebagai serangkaian karakteristik yang menyeluruh dan memenuhi persyaratan. Kualitas dapat diartikan juga menggunakan kata sifat, seperti buruk, baik, dan sangat baik. Kualitas juga dapat diartikan sebagai penilaian kesesuaian untuk digunakan dengan kesesuaian pengguna [20].

2.2.4 *Software Testing Life Cycle*

Software Testing Life Cycle (STLC) merupakan tahapan proses pengujian perangkat lunak yang dilakukan secara sistematis dan terencana untuk memastikan kualitas produk [39]. Setiap tahap STLC memiliki; 1) *Entry Criteria*, ketentuan yang harus dimiliki sebelum pengujian dimulai; 2) *Exit Criteria*, ketentuan yang harus diselesaikan sebelum menyelesaikan pengujian; 3) *Activities dan Deliverable*, hasil yang didapat setelah menyelesaikan pengujian. Gambar di bawah ini menunjukan langkah-langkah STLC.



Gambar 2.1 Tahapan Software Testing Life Cycle [39]

2.2.4.1 *Requirement Analysis*

Pada tahapan ini, dokumen *requirement* mengenai perangkat lunak yang akan diuji sudah diperoleh. Kemudian dokumen tersebut dilakukan analisis dan dipelajari secara detail. Hal ini dilakukan untuk mengidentifikasi persyaratan detail yang dapat diuji secara manual dan automasi serta menganalisa cakupan fungsi atau fitur mana saja yang akan diuji secara fungsional dan non-fungsional.

2.2.4.2 *Test Planning*

Test Planning atau dapat disebut *Test Strategy* merupakan persiapan terstruktur yang digunakan untuk mempersiapkan rencana untuk melakukan pengujian berdasarkan *requirement analysis*. Sehingga diharapkan dapat mengungkap dengan detail anggaran waktu dan biaya yang dialokasikan dengan memaksimalkan proses pengujian [24]. Berikut ini merupakan langkah-langkah untuk menyusun strategi pengujian:

1. Menentukan *quality objective*
2. Menentukan *acceptance objective*
3. *Product objective*
4. Tipe Pengujian
5. Bagaimana cara uji
6. Bagaimana cara *regression test*
7. *Criteria successful*
8. *Mechanism for defect*
9. *Progress reporting*
10. *Defect analysis*

2.2.4.3 *Test Case Development*

Pada tahap ini dokumen *test case*, *test scenario*, *test data*, dan *test script* dipersiapkan untuk masing-masing modul. Dokumen tersebut dibuat sesuai kebutuhan dengan mengacu kepada dokumen *requirement* dan dokumen *test plan*.

2.2.4.4 *Environment Setup*

Pada tahap ini, *environment test* dipastikan berjalan dengan baik dan sesuai harapan. Sebagai contoh, pengujian *software* dipastikan dapat berjalan sesuai dan sama di setiap *environment development*, *staging*, maupun *production*.

2.2.4.5 *Test Execution*

Pada tahap ini, setelah perangkat lunak siap diuji dan persiapan pengujian sudah selesai disiapkan, pengujian dilaksanakan berdasarkan *test plan*, *test case*, dan *test scenario*. Selama proses eksekusi pengujian berjalan, setiap hasil uji diperhatikan untuk memastikan berjalan dengan sukses atau terdapat kegagalan. Jika sukses maka bukti hasil pengujian akan didokumentasikan, sedangkan jika terjadi kegagalan, maka dokumen *bug report* untuk disampaikan saran perbaikan kepada tim terkait.

2.2.4.6 *Test Cycle Closure*

Tahap ini merupakan tahapan akhir penyelesaian pengujian perangkat lunak. Pada tahap ini dibuat dokumen pelaporan penyelesaian pengujian kemudian menganalisis artefak pengujian

untuk mengidentifikasi strategi pengujian untuk *test cycle* berikutnya.

2.2.5 Standardisasi pengujian ISO/IEC 9126

Standardisasi diperlukan sebagai tolak ukur seberapa jauh produk yang diuji memenuhi karakteristik dan kualitas sesuai standar yang telah ditentukan. ISO/IEC 9126 merupakan salah satu standardisasi internasional yang ditetapkan sebuah lembaga yang bernama *International Organization for Standardization* (ISO) bersama *International Electrotechnical Commision* (IEC). ISO/IEC 9126 digunakan sebagai standar mendefinisikan kualitas produk perangkat lunak, model, karakteristik mutu, maupun matriks terhadap serangkaian evaluasi yang dilakukan untuk menetapkan kualitas sebuah produk software. Terdapat enam aspek pengujian, antara lain; 1) *functionality*, 2) *reliability*, 3) *usability*, 4) *efficiency*, 5) *maintainability*, 6) *portability*. Masing-masing aspek uji memiliki karakteristik kualitas [24] yang ditunjukan pada tabel 2.3.

Tabel 2.3 Karakteristik Aspek Standardisasi Pengujian ISO/IEC 9126

Aspek	Karakteristik	Deskripsi	Parameter
Functionality	*Suitability*	*Software* mampu menyediakan fungsi yang memadai pengguna melakukan tugas untuk mencapai tujuan penggunaan.	1. Fungsi input data 2. Fungsi proses data 3. Fungsi *output* data
	Accuracy	*Software* mampu menyediakan state positif (benar) dan negatif (salah) berdasarkan kesepakatan	1. Keakuratan pengolahan data 2. Keakuratan menampilkan data
	Security	*Software* mampu mencegah akses yang tidak diinginkan dan menolak serangan yang disengaja yang dimaksudkan untuk mendapatkan akses yang tidak sah ke informasi rahasia atau untuk membuat modifikasi yang	Keamanan data dan akses. Top 10: 2021 *List*: 1. A01 *Broken Access Control* *2.* A02 *Cryptographic Failures* 3. A03 *Injection*

Aspek	Karakteristik	Deskripsi	Parameter
		tidak sah pada informasi atau program sehingga memberikan penyerang beberapa keuntungan atau untuk menolak layanan kepada pengguna yang sah.	4. A04 *Insecure Design* 5. A05 *Security Misconfiguration* 6. A06 *Vulnerable and Outdated Components* 7. A07 *Identification and Authentication Failures* 8. A08 *Software and Data Integrity Failures* 9. A09 *Security Logging and Monitoring Failures* 10. A10 *Server-Side Request Forgery* (SSRF)
	Interoperability	*Software* mampu berinteraksi dengan salah satu atau beberapa sistem tertentu.	Kemampuan komponen *software* untuk berinteraksi dengan komponen-komponen atau sistem lainnya
Reliability	*Maturity*	*Software* mampu menghindari kegagalan software akibat kesalahan internal software	Manajemen risiko
	Fault Tolerance	*Software* mampu mempertahankan *performance* jika terjadi kesalahan atau pelanggaran antarmuka yang ditentukan	Kesalahan dalam penggunaan
	Recoverability	*Software* mampu meningkatkan kembali *performance* dan *recovery* pemulihan data jika terjadi kegagalan	Data Maintenance
Usability	*Understandability*	Sebuah *software* mampu dipahami oleh *user* dan *user* paham untuk mengerjakan tugas-tugas dalam kondisi tertentu.	Fitur-fitur mudah dimengerti pengguna
	Learnability	Kemampuan sebuah *software* dipelajari dengan mudah	Kemudahan mempelajari *software* tanpa dan atau dengan manual *book*
	Operability	Kemampuan sebuah *software* dioperasikan dengan mudah	Pengoperasian mudah

Aspek	Karakteristik	Deskripsi	Parameter
Efficiency	*Time Behavior*	Kemampuan sebuah *software* untuk menyediakan respon, waktu *processing*, dan *throughput* yang sesuai ketika menjalankan sebuah fungsi dalam kondisi tertentu	*Load time*
	Resource Utilization	Kemampuan sebuah *software* untuk menyediakan *resource* secara maksimal agar dapat memfasilitasi waktu eksekusi yang lebih optimal ketika menjalankan sebuah fungsi dalam kondisi tertentu	*Memory* dan penyimpanan data yang terpakai tidak besar kapasitasnya.
Maintainability	*Analyzability*	Kemampuan sebuah *software* untuk mendiagnosis defisiensi atau penyebab kegagalan agar dapat diidentifikasi dan dimodifikasi	Analisis penyebab jika terjadi kesalahan
	Changeability	Kemampuan sebuah *software* untuk dimodifikasi dan diimplementasikan	*Software* dapat dikembangakan untuk penambahan fitur ke versi selanjutnya
	Stability	Kemampuan sebuah *software* untuk meminimalisir hasil tidak terduga dari modifikasi yang dilakukan	Stabilitas *software* ketika *regression*
	Testability	Sebuah *software* tetap mampu diverifikasi dan divalidasi setelah dimodifikasi	*Software* dapat diverifikasi
Portability	*Adaptability*	Kemampuan sebuah *software* untuk dimodifikasi ke dalam spesifikasi *environment* yang berbeda tanpa menerapkan	Peluang untuk beradaptasi pada lingkup sistem yang berbeda
	Installability	Sebuah *software* mampu di instal di berbagai versi *environment* perangkat keras yang berbeda-beda	Kemampuan perangkat lunak untuk diinstal dalam spesifikasi perangkat keras yang berbeda-beda
	Coexistence	Sebuah *software* mampu untuk berjalan berdampingan bersamaan dengan *software* lainnya.	Kemampuan perangkat lunak untuk berdampingan dengan perangkat lunak lainnya dalam satu

Aspek	Karakteristik	Deskripsi	Parameter
			lingkungan dengan berbagi sumber daya.
	Replace ability	Sebuah *software* mampu untuk digunakan sebagai pengganti perangkat lunak tertentu lainnya di lingkungan perangkat lunak tersebut	Kemampuan perangkat lunak untuk digunakan sebagai sebagai pengganti perangkat lunak lainnya.

2.2.6 Metode *Grey-box testing*

Grey-box testing merupakan salah metode pengujian yang menggabungkan metode *white-box testing* dengan *black-box testing*. Pada metode ini sistem kerja internal aplikasi dan pengetahuan tentang aspek-aspek fundamental sistem diharuskan dapat diketahui. Tetapi tidak perlu mengetahui internal pengkodean aplikasi tersebut secara mendalam. Berdasarkan penelitian sebelumnya teknik pengujian ini dinilai lebih efektif, efisien, dan mendukung implementasi pengeluaran biaya pengembangan yang cenderung lebih sedikit dibandingkan metode pengujian lainnya [22], [28]–[30]. Implementasi pengujian dengan metode *grey-box testing* masih akan tetap melaksanakan pengujian layaknya *black-box testing* tetapi pengujian akan dilakukan dengan mengkombinasikannya dengan *white-box testing* meskipun tidak sepenuhnya *white-box testing* diterapkan [40].

Dibandingkan *black-box testing*, *grey-box testing* memberikan informasi yang lebih mendalam tentang bagaimana sistem bekerja dan mengapa sistem berperilaku seperti itu. *Grey-box testing* membantu untuk memfokuskan upaya pengujian pada area uji yang paling kritis dari sistem dan menghindari pengujian yang tidak perlu pada area yang tidak akan menyebabkan masalah. Di sisi lain, dibandingkan dengan *white-box testing*, *grey-box testing* memberikan pandangan yang lebih luas terhadap sistem untuk melihat sistem dari sudut pandang pengguna. Secara keseluruhan,

pengujian *grey-box* memberikan pendekatan yang seimbang yang menggabungkan kekuatan dari pengujian *white-box* dan *black-box* dan membantu mengetahui untuk memastikan kualitas dan keandalan perangkat lunak. Tabel di bawah ini menunjukan komparasi metode pengujian *Black-box, White-Box, dan Grey-box* [28].

Tabel 2.4 Komparasi Metode Pengujian *Black-box, White-box,* dan *Grey-box*

Black-Box Testing	***White-Box Testing***	***Grey-Box Testing***
Tidak perlu mengetahui struktur internal kode.	Wajib mengetahui struktur internal kode.	Setidaknya sebagian struktur internal kode diketahui.
Disebut juga sebagai *closed-box testing*, *data driven testing, functional testing*.	Disebut juga sebagai *clear-box testing, structural testing, closed testing*	Disebut juga sebagai *translucent testing* (pengujian tertutup tetapi tembus pandang)
Tidak dapat dilakukan pengujian algoritma	Direkomendasikan untuk melakukan pengujian algoritma	Tidak dapat dilakukan pengujian algoritma
Kesalahan yang tersembunyi sulit dideteksi.	Kesalahan yang tersembunyi mudah dideteksi	Menemukan kesalahan yang tersembunyi tidak sesulit *black-box testing* tetapi tidak secepat *white-box testing*
Pengujian dilakukan oleh seorang *tester*, *developer*, dan *user*	Pengujian dilakukan oleh seorang *tester* dan *developer*	Pengujian dilakukan oleh seorang *tester*, *developer*, dan *user*
Prosesnya sedikit memakan waktu	Prosesnya paling banyak memakan waktu	Prosesnya membutuhkan effort waktu yang lebih sedikit dibanding *black-box testing*
Jenis pengujian yang paling tidak menyeluruh.	Bentuk pengujian yang paling lengkap.	Bentuk pengujian yang lumayan lengkap
Memiliki ruang pengujian input terbanyak.	Memiliki ruang pengujian input terkecil.	Ruang pengujian lebih besar dari *white-box testing* tetapi lebih kecil dari *black-box testing*.

Pengujian *grey-box* biasanya digunakan ketika perangkat lunak yang diuji bersifat kompleks dan membutuhkan pemahaman yang mendetail mengenai perilaku internal dan eksternalnya. Tujuan dari pengujian *grey-box* adalah untuk mengidentifikasi masalah potensial dengan perangkat lunak seperti *bug*, masalah kinerja, dan kerentanan keamanan dengan *effort* yang lebih mudah dan waktu yang lebih cepat. Metode *grey-box testing* memiliki beberapa teknik pengujian [41], antara lain:

1) *Matrix Testing*

   Teknik ini dilakukan dengan mendefinisikan seluruh variabel yang ada untuk diuji berdasarkan potensi risiko yang akan timbul. Teknik ini membantu untuk mengidentifikasi variabel yang tidak digunakan atau tidak dioptimalkan.

2) *Regression Testing*

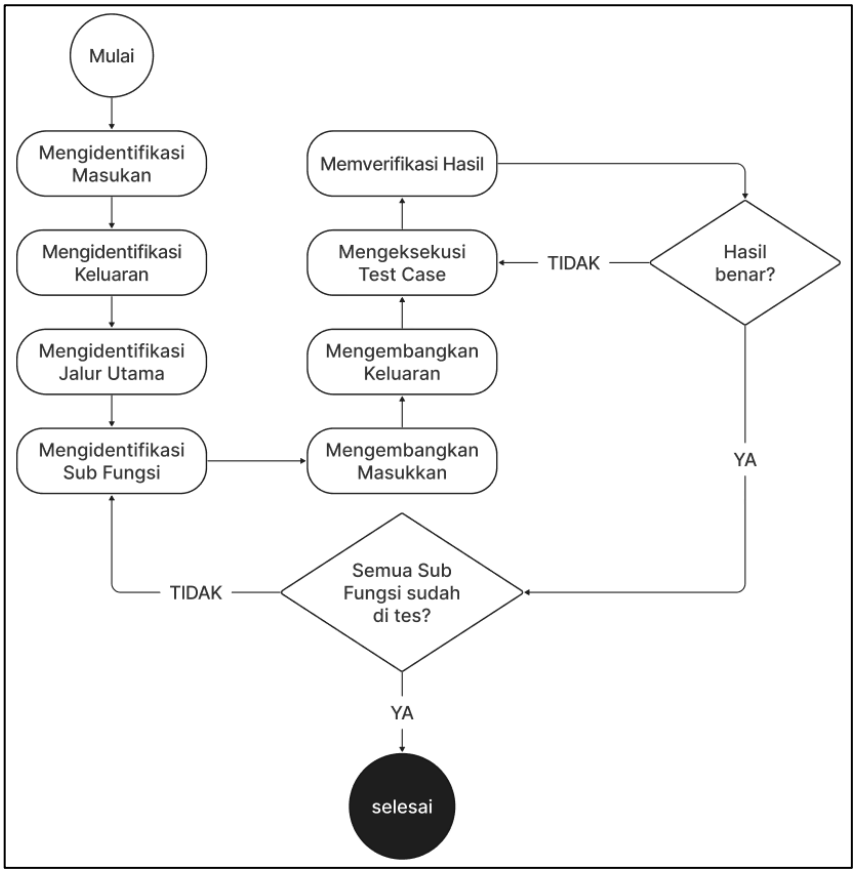
   Teknik ini dilakukan dengan mengecek ulang seluruh pengujian terhadap perubahan yang dilakukan pada aplikasi untuk mengetahui apakah perubahan tersebut menghasilkan kesalahan baru atau konsekuensi. Tujuan dari teknik pengujian ini adalah untuk memverifikasi bahwa perbaikan atau modifikasi pada aplikasi tidak berdampak negatif pada keseluruhan fungsinya. Hal ini memerlukan penyesuaian input dan *output* tergantung pada skenario yang diuji.

3) *Pattern Testing*

   Teknik ini dilakukan dengan mengidentifikasi pola cacat pada aplikasi dengan menganalisis kejadian di masa lalu. Teknik ini dapat memprediksi cacat di masa depan berdasarkan pola yang ditemukan dan mengevaluasi efektivitas perbaikan sebelumnya. Informasi yang diperoleh dari teknik ini dapat digunakan untuk pengembangan dan pengujian aplikasi di masa depan.

Teknik-teknik tersebut dapat dilakukan dengan langkah berikut ini [41]:

1) Mengidentifikasi masukan (input)
2) Mengidentifikasi keluaran (*output*)
3) Mengidentifikasi jalur utama
4) Mengidentifikasi sub fungsi
5) Mengembangkan masukkan untuk setiap sub fungsi
6) Mengembangkan keluaran untuk setiap sub fungsi
7) Mengeksekusi *test case* di setiap sub fungsi
8) Memverifikasi hasil
9) Mengulangi langkah-langkah untuk setiap sub fungsi



Gambar 2.2 Langkah-langkah pengujian *grey-box testing*

2.2.7 API

Application Program Interface (API) merupakan sebuah perangkat lunak yang berguna untuk mengintegrasikan dua aplikasi atau lebih agar dapat berkomunikasi dan bekerjasama dengan baik. API membantu sebuah aplikasi dapat mengakses data maupun fitur dari aplikasi lainnya yang saling terintegrasi, tanpa harus memahami bagaimana aplikasi tersebut dibuat dan tanpa memerlukan akses pengkodean internal dari aplikasi tersebut. API kerap digunakan sebagai alternatif opsi untuk mempercepat pengembangan perangkat lunak dan memperluas jangkauan aplikasi. API terdiri dari berbagai jenis, seperti REST API, SOAP API, GraphQL API. Jenis API sendiri dapat digunakan sesuai kebutuhan pengembang perangkat lunak dalam memfasilitasi interaksi dan integrasi data antar aplikasi [42].

2.2.8 Rest API

Rest API merupakan sebuah client yang dapat mengakses data dari server dengan meskipun *source client* dan server berbeda. Secara proses, *client* akan meminta sumber data kepada server. Sumber data akan dibedakan berdasarkan *universal resource identifiers* (URI) oleh server. Kemudian server akan memberikan respon data yang dapat diterima dalam format *text, Json,* maupun *xml*. Terdapat beberapa metode HTTP yang secara umum sering digunakan dalam Rest API, antara lain; *GET*, *POST*, *PUT*, *FETCH*, *DELETE,* dan *OPTIONS*. Rest API berperan untuk memastikan *request* dan *response* memiliki format yang sama dan dapat dipahami oleh semua sistem yang terintegrasi. Sehingga Rest API memiliki keunggulan di aspek fleksibilitas integrasi antar *platform* [42].

2.2.9 *End-to-end Test*

End-to-end test merupakan salah satu teknik pengujian *software* yang validasi keberhasilannya berdasarkan kesuksesan menjalankan fungsi dari awal sampai akhir berdasarkan skenario tertentu [43]. Pelaksanaan *end-to-end testing* terbagi menjadi dua cara peimplementasian, yaitu 1) Horizontal *end-to-end testing*, misalnya pada web penjualan buku, maka semua proses yang mencakup pengisian data pembeli, detail pembelian, termasuk detail pembelian yang dilakukan pembeli harus dilakukan pengujian dari awal sampai akhir. 2) Vertikal *end-to-end testing*, melakukan pengujian pada termasuk API dan SQL.

2.2.10 *Behavior Driven Development*

Behavior Driven Development (BDD) merupakan sebuah *framework* pengujian untuk penulisan *test case* yang didasarkan didasarkan pada *requirement* yang dikumpulkan dari *stakeholders* [44]. Salah satu *framework* BDD yang banyak digunakan adalah Cucumber Gherkins *Syntax*. Terdapat beberapa *syntax* utama penulisan BDD seperti; 1) *Feature*, digunakan untuk untuk memberikan deskripsi tingkat tinggi dari fitur perangkat lunak, dan untuk mengelompokkan skenario terkait; 2) *Rule*, digunakan untuk mewakili suatu aturan bisnis yang harus diterapkan. Kata kunci ini memberikan informasi tambahan untuk sebuah fitur; 3) *Example*, dapat digunakan sebagai sekumpulan contoh data yang dapat dieksekusi saat pengujian dijalankan; 4) *Steps*, digunakan untuk mengidentifikasi langkah-langkah pengujian seperti *Given, When, And, Then, But*; 5) *Scenario Outline*, digunakan untuk mendeskripsikan skenario *test* yang dapat dijalankan untuk skenario yang sama beberapa kali, dengan kombinasi nilai yang berbeda.

2.2.11 *Open Web Application Security Project*

Open Web Application Security Project (OWASP) adalah suatu organisasi internasional non-profit yang didirikan oleh *OWASP foundation* pada tahun 2004. Tujuan utama OWASP yaitu memelihara, meningkatkan, dan menjamin keamanan perangkat lunak [45]. Terdapat 10 aspek penjagaan keamanan menurut OWASP di tahun 2021 [46], yang ditunjukan pada tabel 2.5.

Tabel 2.5 Aspek OWASP Top 10: 2021

Aspek	Parameter
A01 *Broken Access Control*	Memberlakukan kebijakan kontrol akses sedemikian rupa sehingga pengguna tidak dapat bertindak di luar izin yang diberikan. Kegagalan biasanya menyebabkan pengungkapan informasi yang tidak sah, modifikasi, atau penghancuran semua data atau melakukan fungsi bisnis di luar batas pengguna.
A02 *Cryptographic Failures*	Memberlakukan perlindungan data seperti kata sandi, nomor kartu kredit, catatan kesehatan, informasi pribadi, dan rahasia bisnis yang memerlukan ekstra perlindungan.
A03 *Injection*	Memberlakukan perlindungan untuk 1) mencegah injeksi SQL; 2) menggunakan API yang aman; 3) menggunakan limit dan kontrol SQL; dan sebagainya.
A04 *Insecure Design*	Memberlakukan desain yang aman, terdapat desain kontrol yang baik dan efisien.
A05 *Security Misconfiguration*	Memberlakukan perlindungan pada proses *deployment*
A06 *Vulnerable and Outdated Components*	Memberlakukan perlindungan dengan melakukan *versioning software*; Menghapus dependensi, fitur, komponen, berkas, dan dokumentasi yang tidak digunakan; Memperbarui *patch* dan komponen yang digunakan.
A07 *Identification and Authentication Failure*s	Memberlakukan perlindungan dengan menerapkan otentikasi multi-faktor untuk mencegah pengisian kredensial otomatis, *brute force*, dan dan serangan penggunaan kembali kredensial yang dicuri; Tidak mengirim atau menyebarkan dengan kredensial bawaan apa pun; Menerapkan pemeriksaan kata sandi yang lemah.
A08 *Software and Data*	Mampu menjaga integritas data.

Integrity Failures	
A09 *Security Logging and Monitoring Failures*	Memberlakukan pendeteksian respon terhadap penjebolan yang sedang aktif.
A10 *Server-Side Request Forgery (SSRF)*	Mampu menjaga dari permintaan *remote resource* tanpa melakukan validasi URL. Mampu menerapkan skema URL, port, dan destinasi dengan daftar izin positif. Menonaktifkan pengalihan HTTP.

2.2.12 *User Acceptance Test*

User Acceptance Test (UAT) merupakan sebuah proses pengujian yang dilakukan oleh para *stakeholder* sesuai dokumen langkah-langkah pengujian yang hasilnya dapat digunakan sebagai bukti apakah perangkat lunak yang dikembangkan dapat dikatakan mampu memenuhi persyaratan yang diperlukan [47].

2.2.13 Uji Kelayakan

Uji kelayakan dilakukan untuk menentukan kategori penilaian kelayakan terhadap suatu sistem yang dikembangkan. Uji ini sangat dibutuhkan untuk menentukan apakah suatu sistem yang sudah dikembangkan tersebut sudah layak digunakan, atau perlu dilakukan perbaikan, atau dapat dilanjutkan ke pengembangan selanjutnya [48]. Berikut ini merupakan rumus perhitungan [48] yang digunakan untuk mengetahui hasil uji kelayakan:

$$Skor\ Kelayakan = \frac{skor\ aktual\ (f)}{skor\ ideal\ (n)} \times 100\ \% \qquad (2.1)$$

Berdasarkan rumus diatas, uji kelayakan dapat diperoleh dengan cara menghitung hasil skor aktual (*f*) yang dibagi dengan skor ideal (*n*) kemudian dikali 100%. Skor aktual didapatkan berdasarkan jumlah kasus pengujian yang sukses yang sesuai dengan ekspektasi. Sedangkan skor ideal didapatkan berdasarkan banyaknya keseluruhan kasus pengujian yang diekspektasikan.

Setelah mendapatkan hasil perhitungan, hasil tersebut dibandingkan dengan mengelompokkan nilai berdasarkan lima kategori penilaian seperti yang ditunjukan pada tabel di bawah ini.

Tabel 2.6 Kategori Penilaian Kelayakan [49]

Persentase Pencapaian (%)	**Interpretasi**
$x \geq 90$	Sangat Baik
$90 > x \geq 80$	Baik
$80 > x \geq 70$	Cukup
$70 > x \geq 60$	Kurang
$60 > x$	Sangat Kurang