BAB 2
TINJAUAN PUSTAKA

2.1 Penelitian Terdahulu

Penelitian mengenai sistem informasi dan pengelolaan klaim BPJS Ketenagakerjaan telah banyak dilakukan dengan berbagai fokus, seperti analisis sistem, prosedur klaim, sistem informasi akuntansi, dan tata kelola pembayaran klaim. Penelitian-penelitian tersebut menjadi referensi dalam pengembangan penelitian ini.

Penelitian oleh Rahma (2021) membahas analisis sistem dan prosedur pembayaran Jaminan Kecelakaan Kerja (JKK) pada BPJS Ketenagakerjaan Cabang Pematangsiantar. Hasil penelitian menunjukkan bahwa sistem akuntansi pembayaran klaim berperan dalam mendukung pengendalian intern melalui penerapan prosedur pembayaran klaim yang sesuai.

Penelitian Irdanasari dan Wijayanti (2021) menganalisis sistem informasi akuntansi pada prosedur pembayaran klaim JHT, JP, JKK, dan JKM di BPJS Ketenagakerjaan Cabang Madiun. Hasil penelitian menunjukkan bahwa sistem telah sesuai dengan teori, namun masih terdapat kendala berupa kelengkapan dokumen peserta dan lamanya proses klaim.

Penelitian Siregar et al. (2023) membahas prosedur klaim JKK menggunakan metode *library research*. Hasil penelitian menjelaskan bahwa prosedur klaim meliputi pendaftaran peserta, penanganan kecelakaan kerja, hingga pembayaran santunan sesuai prosedur yang berlaku.

Penelitian Putri dan Azizah (2024) membahas tata kelola tagihan pembayaran klaim JKK pada BPJS Ketenagakerjaan Cabang Surabaya Karimunjawa. Hasil penelitian menunjukkan bahwa tata kelola pembayaran klaim telah berjalan dengan baik melalui prosedur pelayanan, verifikasi data, dan pengelolaan tagihan secara efektif dan efisien.

Penelitian Oktapianti et al. (2024) menganalisis sistem klaim JHT pada BPJS Ketenagakerjaan Cabang Medan Kota. Hasil penelitian menunjukkan bahwa prosedur klaim telah berjalan secara sistematis dengan verifikasi data yang ketat,

meskipun masih terdapat kendala pada penggunaan aplikasi JMO dan pemahaman peserta.

Berdasarkan penelitian terdahulu, sebagian besar penelitian masih berfokus pada analisis sistem, prosedur, dan tata kelola klaim BPJS Ketenagakerjaan. Berbeda dengan penelitian tersebut, penelitian ini berfokus pada rancang bangun *back-end* Sistem Informasi Pengajuan dan Status Bayar Klaim JKK berbasis *web* menggunakan metode V-*Model* untuk mendukung pengelolaan data klaim, status pembayaran, dan pelaporan secara terintegrasi.

Tabel 2.1 Penelitian Terdahulu

Kategori	**Rahma (2021)**	**Irdanasari & Wijayanti (2021)**	**Siregar et al. (2023)**	**Putri & Azizah (2024)**	**Oktapianti et al. (2024)**
Fokus	Sistem dan prosedur pembayaran klaim JKK	Sistem informasi akuntansi pembayaran klaim	Prosedur klaim JKK	Tata kelola pembayaran klaim JKK	Sistem klaim JHT
Metode	Kualitatif	Kualitatif Deskriptif	*Library Research*	Kualitatif Deskriptif	Kualitatif Deskriptif
Objek	BPJS Ketenagakerjaan Cabang Pematangsiantar	BPJS Ketenagakerjaan Cabang Madiun	BPJS Ketenagakerjaan	BPJS Ketenagakerjaan Cabang Surabaya Karimunjawa	BPJS Ketenagakerjaan Cabang Medan Kota
Hasil	Prosedur pembayaran mendukung pengendalian intern	Sistem sesuai teori, masih terdapat kendala pada dokumen dan proses klaim	Menjelaskan prosedur klaim JKK	Tata kelola pembayaran klaim berjalan baik	Sistem klaim berjalan sistematis dan efektif

2.2 Landasan Teori

2.2.1 BPJS Ketenagakerjaan Pekanbaru Kota

BPJS Ketenagakerjaan merupakan lembaga yang menyelenggarakan program jaminan sosial bagi tenaga kerja di Indonesia yang resmi beroperasi sejak tahun 2014, berdasarkan amanat Undang-Undang Nomor 24 Tahun 2011 tentang Badan Penyelenggara Jaminan Sosial. BPJS Ketenagakerjaan Pekanbaru Kota sebagai salah satu kantor cabang memiliki peran dalam memberikan perlindungan kepada pekerja melalui berbagai program, salah satunya adalah Jaminan

Kecelakaan Kerja (JKK). Program ini bertujuan untuk memberikan jaminan perlindungan atas risiko kecelakaan kerja sehingga dapat meningkatkan rasa aman dan kesejahteraan tenaga kerja.

Dalam pelaksanaannya, pengelolaan klaim Jaminan Kecelakaan Kerja (JKK) masih menghadapi beberapa kendala, terutama pada proses administrasi, verifikasi dokumen, serta pemantauan status pembayaran klaim. Proses yang belum sepenuhnya terintegrasi menyebabkan terjadinya keterlambatan serta kurangnya transparansi informasi bagi peserta maupun pihak terkait. Kondisi ini menunjukkan perlunya dukungan sistem informasi yang mampu mengelola proses klaim secara lebih efektif dan efisien.

Oleh karena itu, BPJS Ketenagakerjaan Pekanbaru Kota terus berupaya meningkatkan kualitas pelayanan melalui pemanfaatan teknologi informasi. Pengembangan sistem informasi berbasis *web* diharapkan dapat membantu proses pengajuan, verifikasi, serta pemantauan status bayar klaim secara terintegrasi dan *real-time*. Dengan adanya sistem tersebut, pelayanan diharapkan menjadi lebih cepat, akurat, dan transparan sehingga dapat meningkatkan kepuasan serta kepercayaan peserta terhadap layanan BPJS Ketenagakerjaan.

Gambar 2.1 BPJS Ketenagakerjaan Pekanbaru Kota

2.2.2 Jaminan Kecelakaan Kerja (JKK)

Jaminan Kecelakaan Kerja (JKK) adalah program perlindungan sosial yang diselenggarakan oleh BPJS Ketenagakerjaan untuk memberikan jaminan kepada pekerja terhadap risiko kecelakaan kerja atau penyakit akibat kerja. Berdasarkan

Peraturan Presiden Republik Indonesia Nomor 109 Tahun 2013, JKK mencakup pembiayaan pengobatan dan perawatan, santunan cacat, serta santunan kematian akibat kecelakaan kerja.

BPJS Ketenagakerjaan bekerja sama dengan fasilitas kesehatan yang tergabung dalam Pusat Layanan Kecelakaan Kerja (PLKK), seperti rumah sakit dan klinik mitra. PLKK bertugas memberikan layanan medis serta mengajukan klaim atas biaya perawatan kepada BPJS Ketenagakerjaan.

Namun, dalam praktiknya, proses pengajuan klaim masih banyak dilakukan secara manual melalui *spreadsheet* atau pengisian dokumen fisik, yang sangat rentan terhadap kesalahan, duplikasi data, serta keterlambatan proses validasi. Untuk itu, dibutuhkan *sistem informasi* yang dapat menangani pengajuan klaim JKK secara elektronik, terstruktur, serta dapat dipantau statusnya secara *real-time*.

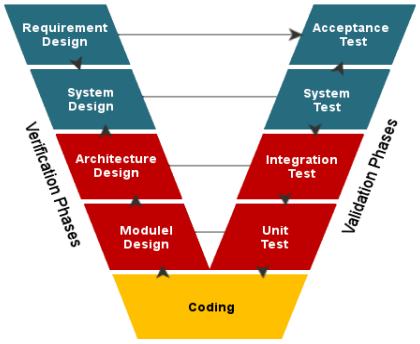
2.2.3 Sistem Informasi

Sistem informasi merupakan suatu kesatuan yang terdiri dari berbagai komponen seperti manusia, data, perangkat lunak, perangkat keras, serta prosedur yang saling berinteraksi dalam mengolah data menjadi informasi yang berguna. Komponen-komponen tersebut bekerja secara terintegrasi untuk mendukung kegiatan operasional dan proses pengambilan keputusan dalam organisasi. Menurut Fila et al. (2025), sistem informasi manajemen terdiri atas komponen manusia, data, perangkat keras, perangkat lunak, dan prosedur bisnis yang saling berinteraksi secara terintegrasi untuk mendukung efisiensi operasional dan pengambilan keputusan yang lebih efektif. Dengan adanya sistem informasi, pengelolaan data dapat dilakukan secara lebih terstruktur sehingga menghasilkan informasi yang dapat digunakan untuk mendukung kebutuhan organisasi.

Dalam perkembangannya, sistem informasi berbasis *web* menjadi salah satu solusi yang banyak digunakan karena mampu memberikan kemudahan akses informasi secara *real-time*. Sistem berbasis *web* memungkinkan pengguna untuk mengakses data kapan saja dan di mana saja selama terhubung dengan jaringan internet, sehingga dapat meningkatkan efisiensi dan efektivitas dalam pengelolaan data (Lubis & Husaini, 2025). Selain itu, sistem ini juga mampu mendukung integrasi antar bagian dalam organisasi sehingga aliran informasi menjadi lebih cepat dan terkoordinasi.

Penerapan sistem informasi dalam suatu organisasi, khususnya pada bidang pelayanan publik, memiliki peran penting dalam meningkatkan kualitas layanan. Sistem informasi dapat membantu mempercepat proses administrasi, mengurangi kesalahan akibat proses manual, serta meningkatkan transparansi dalam penyampaian informasi kepada pengguna (Wahyudin, 2025). Oleh karena itu, penggunaan sistem informasi *web-based* (berbasis *web*) sangat relevan untuk diterapkan dalam pengelolaan pengajuan dan pemantauan status klaim Jaminan Kecelakaan Kerja (JKK) agar proses menjadi lebih efektif dan efisien.

2.2.4 Metode Pengembangan *V-Model*



Gambar 2.2 *V-Model*
Sumber: BINUS Bekasi (2024)

Metode *V-Model* merupakan salah satu *model* pengembangan perangkat lunak yang menekankan hubungan antara tahapan pengembangan dan tahapan pengujian secara sistematis. *Model* ini menggambarkan proses secara terstruktur, di mana setiap tahap pengembangan memiliki pasangan tahap pengujian yang dilakukan secara berurutan dan saling berkaitan. Pendekatan ini memungkinkan proses verifikasi dan validasi dilakukan sejak awal pengembangan sehingga kualitas sistem dapat lebih terjamin (Pavlíčková et al., 2024).

Keunggulan metode *V-Model* terletak pada kemampuannya dalam mendeteksi kesalahan pada setiap tahapan pengembangan melalui proses pengujian yang terencana. Setiap fase seperti analisis kebutuhan, desain sistem, hingga implementasi memiliki tahapan pengujian yang sesuai sehingga kesalahan dapat diidentifikasi lebih dini. Dengan demikian, metode ini mampu meningkatkan keandalan sistem serta mempermudah pengembang dalam memastikan kesesuaian antara kebutuhan pengguna dan sistem yang dibangun (Sarhadi et al., 2022).

Dalam penelitian ini, metode *V-Model* digunakan untuk merancang dan membangun sistem informasi pengajuan serta pemantauan status bayar klaim Jaminan Kecelakaan Kerja (JKK) pada PLKK BPJS Ketenagakerjaan Pekanbaru Kota. Setiap tahapan pengembangan disusun secara sistematis mulai dari analisis kebutuhan hingga pengujian sistem untuk memastikan bahwa sistem berjalan sesuai dengan kebutuhan pengguna. Dengan penerapan metode ini, diharapkan sistem yang dihasilkan memiliki kualitas yang baik, mampu meminimalkan kesalahan, serta mendukung efisiensi dan transparansi dalam pengelolaan data klaim.

2.2.5 *Web*-Based Application

Aplikasi berbasis *web* merupakan perangkat lunak yang dijalankan melalui *browser* dan dapat diakses melalui jaringan internet maupun intranet tanpa memerlukan instalasi pada perangkat pengguna. Seluruh proses pengolahan data dilakukan pada *server* sehingga pengguna dapat mengakses sistem secara fleksibel dari berbagai perangkat. Sistem berbasis *web* mampu meningkatkan efisiensi dalam pengelolaan data serta mempermudah akses informasi secara *real-time* (Hermiati et al., 2021).

Selain itu, aplikasi berbasis *web* memiliki keunggulan dalam hal kemudahan penggunaan, pemeliharaan sistem, serta kemampuan untuk digunakan oleh banyak pengguna dalam satu lingkungan sistem. Sistem ini juga mendukung pengelolaan data secara terpusat sehingga meminimalisir kesalahan akibat proses manual. Hal tersebut menjadikan sistem berbasis *web* sebagai solusi yang efektif dalam meningkatkan efisiensi operasional (Lubis & Husaini, 2025).

Dalam penelitian ini, sistem informasi pengajuan dan pemantauan status bayar klaim Jaminan Kecelakaan Kerja (JKK) dikembangkan berbasis *web* dan digunakan secara internal oleh petugas BPJS Ketenagakerjaan. Sistem ini berfungsi untuk membantu proses pengelolaan data klaim secara terstruktur mulai dari pengajuan hingga pemantauan status pembayaran. Dengan adanya sistem ini, proses administrasi menjadi lebih cepat, akurat, serta memudahkan petugas dalam mengakses dan mengelola data klaim (Arizal et al., 2022).

2.2.6 Arsitektur *Back-end*

Arsitektur *back-end* merupakan struktur yang mengatur bagaimana komponen pada sisi *server* bekerja dalam memproses data, menjalankan logika

bisnis, mengelola autentikasi pengguna, serta berinteraksi dengan basis data. *Back-end* berfungsi sebagai pusat pengolahan data yang menerima permintaan (*request*) dari pengguna, memprosesnya sesuai aturan bisnis yang berlaku, kemudian mengirimkan hasil pemrosesan kembali kepada pengguna melalui antarmuka aplikasi. Dengan penerapan arsitektur *back-end* yang baik, sistem dapat memiliki tingkat keamanan, skalabilitas, dan kemudahan pemeliharaan yang lebih tinggi (Cherukuri, 2024).

Dalam pengembangan aplikasi *web* modern, arsitektur *back-end* umumnya menerapkan pemisahan tanggung jawab (*separation of concerns*) antara komponen yang menangani logika bisnis, akses data, dan layanan sistem lainnya. Pendekatan tersebut bertujuan untuk meningkatkan keteraturan kode program sehingga proses pengembangan, pengujian, dan pemeliharaan aplikasi dapat dilakukan secara lebih efisien. Selain itu, arsitektur yang terstruktur juga membantu pengembang dalam melakukan pengembangan fitur baru tanpa mengganggu fungsi sistem yang telah berjalan sebelumnya (Asqia, 2024).

Pada penelitian ini, arsitektur *back-end* diimplementasikan menggunakan *framework* Laravel yang bertugas menangani proses autentikasi pengguna, pengelolaan data klaim Jaminan Kecelakaan Kerja (JKK), pengolahan data rumah sakit atau klinik, serta komunikasi dengan basis data MySQL. Dengan penerapan arsitektur *back-end* tersebut, sistem diharapkan mampu mendukung proses bisnis secara terintegrasi dan menjaga konsistensi data yang digunakan dalam operasional sistem.

2.2.7 Laravel

Laravel merupakan salah satu *framework* PHP yang digunakan dalam pengembangan aplikasi berbasis *web* dan menerapkan arsitektur *Model-View-Controller* (MVC). Arsitektur tersebut memisahkan pengelolaan data, logika aplikasi, dan tampilan sehingga proses pengembangan sistem dapat dilakukan secara lebih terstruktur. Menurut Akbar et al. (2023), Laravel dapat digunakan dalam pengembangan *back-end* sistem informasi dengan menerapkan konsep MVC sehingga pengembangan aplikasi dapat dilakukan secara lebih terstruktur dan terorganisasi.

Laravel juga menyediakan berbagai fitur yang mendukung proses pengembangan aplikasi *web*, seperti *routing*, *middleware*, autentikasi, *Eloquent ORM*, serta *Migration* untuk membantu pengelolaan struktur basis data. Kadim et al. (2023) menjelaskan bahwa penerapan *framework* Laravel dalam pembangunan sistem informasi berbasis *web* dapat mendukung proses pengembangan aplikasi dan pengelolaan fungsi sistem secara terstruktur. Berbagai fitur yang tersedia pada Laravel membantu pengembang dalam membangun aplikasi sesuai dengan kebutuhan sistem.

Dalam penelitian ini, Laravel digunakan sebagai *framework* utama dalam pembangunan *back-end* Sistem Informasi Pengajuan dan Status Bayar Klaim Jaminan Kecelakaan Kerja (JKK) berbasis *web*. Laravel digunakan untuk menangani proses autentikasi pengguna, pengaturan *routing web* dan API, logika bisnis pada fitur admin dan pengguna, validasi data, pengelolaan basis data melalui *Eloquent ORM*, serta proses *server-side* seperti pengelolaan data RS/Klinik, unggah *file* klaim, pengelolaan status klaim, dan ekspor laporan. Dengan penggunaan Laravel, pengembangan *back-end* sistem dapat dilakukan secara lebih terstruktur serta memudahkan proses pengelolaan dan pemeliharaan fungsi-fungsi sistem.

2.2.8 *Inertia.js*

Inertia.js merupakan sebuah *library* yang digunakan sebagai penghubung antara aplikasi *back-end* dan antarmuka pengguna (*front-end*) dalam pengembangan aplikasi *web*. Penggunaan *Inertia.js* memungkinkan aplikasi tetap memanfaatkan mekanisme *routing*, *controller*, *middleware*, autentikasi, serta fitur lain yang disediakan oleh *framework* Laravel pada sisi *server*, sementara antarmuka pengguna dapat dikembangkan menggunakan teknologi *front-end* modern.

Dalam implementasinya, *Inertia.js* memungkinkan Laravel pada sisi *back-end* dihubungkan dengan antarmuka pengguna dalam satu arsitektur aplikasi. Tempariyawan et al. (2025) menerapkan Laravel Inertia dengan Vue.js dalam pengembangan aplikasi *web* dan membandingkan performanya dengan Laravel Blade. Hasil penelitian menunjukkan bahwa implementasi Laravel Inertia dengan Vue.js dapat memberikan performa yang baik pada aplikasi *web*, khususnya berdasarkan waktu eksekusi fitur yang diuji.

Selain itu, Munir (2026) menerapkan arsitektur *Single Page Application* (SPA) menggunakan Laravel, Vue.js, dan *Inertia.js* dalam pengembangan aplikasi berbasis *web*. Penerapan tersebut menunjukkan bahwa *Inertia.js* dapat digunakan untuk mendukung integrasi antara Laravel sebagai teknologi pada sisi *back-end* dan Vue.js sebagai teknologi antarmuka pengguna dalam pengembangan aplikasi *web*.

Dalam penelitian ini, *Inertia.js* dimanfaatkan sebagai penghubung pada sisi *back-end* Laravel untuk mengirimkan data dari *controller* menuju antarmuka pengguna. Pendekatan tersebut memungkinkan proses pengelolaan data dan logika bisnis tetap dilakukan melalui Laravel, sedangkan data yang dihasilkan diteruskan melalui *Inertia.js* untuk ditampilkan pada antarmuka pengguna. Oleh karena itu, pembahasan *Inertia.js* dalam penelitian ini difokuskan pada mekanisme pertukaran data yang dikelola oleh Laravel pada sisi *server*.

2.2.9 Composer

Composer adalah sebuah *dependency manager* untuk bahasa pemrograman PHP yang digunakan untuk mengelola pustaka atau paket yang dibutuhkan dalam pengembangan aplikasi. Dengan menggunakan *Composer*, pengembang dapat mengunduh, menginstal, dan memperbarui pustaka pihak ketiga sesuai dengan kebutuhan proyek. Seluruh dependensi sistem dicatat dalam *file composer.json* sehingga memudahkan proses pengelolaan dan pemeliharaan dependensi proyek.

Dalam pengembangan aplikasi berbasis *framework*, khususnya Laravel, *Composer* memiliki peran penting dalam pengelolaan komponen inti maupun pustaka tambahan yang digunakan oleh aplikasi. *Composer* memungkinkan berbagai *library* yang dibutuhkan sistem dikelola secara terstruktur tanpa harus melakukan pengelolaan dependensi secara manual. Islam et al. (2021) menjelaskan bahwa pengelolaan dependensi paket pihak ketiga merupakan bagian penting dalam berbagai *technology stack*, dengan *package manager* berperan dalam memastikan paket dapat dipasang, dikonfigurasi, maupun dihapus dari suatu aplikasi dengan tepat.

Pada penelitian ini, *Composer* digunakan sebagai alat utama dalam pengelolaan dependensi untuk memastikan bahwa Sistem Informasi Pengajuan dan Status Bayar Klaim JKK dapat berjalan menggunakan pustaka dan versi yang sesuai dengan kebutuhan pengembangan. Dengan demikian, penggunaan *Composer*

membantu proses pengelolaan pustaka menjadi lebih terstruktur dan memudahkan pengelolaan dependensi yang digunakan dalam pengembangan sistem.

2.2.10 MySQL

MySQL adalah salah satu sistem manajemen basis data relasional (*Relational Database Management System/RDBMS*) yang digunakan dalam pengembangan aplikasi berbasis *web*. Dalam pengoperasiannya, *MySQL* menggunakan *Structured Query Language* (SQL) untuk melakukan berbagai operasi pengelolaan data, seperti pengambilan (*select*), penambahan (*insert*), pembaruan (*update*), dan penghapusan (*delete*) data di dalam basis data.

Menurut Choina dan Skublewska-Paszkowska (2022), sistem manajemen basis data memiliki peran penting dalam aplikasi modern dan pemilihan sistem basis data beserta pustaka yang digunakan dapat memengaruhi kecepatan operasi data. Penelitian tersebut melakukan pengujian terhadap *MySQL*, PostgreSQL, dan Oracle melalui operasi *insert*, *update*, *delete*, dan *select*, serta menunjukkan adanya perbedaan performa berdasarkan sistem basis data dan mekanisme pengelolaan data yang digunakan.

Selain itu, Schab (2023) melakukan analisis perbandingan performa terhadap *MySQL*, PostgreSQL, dan Microsoft SQL *Server* dengan mengukur waktu eksekusi operasi *select*, *update*, dan *insert* pada beberapa jumlah data. Hasil penelitian menunjukkan bahwa performa masing-masing sistem manajemen basis data dapat berbeda bergantung pada jenis operasi dan jumlah data yang diproses. Hal tersebut menunjukkan bahwa pemilihan sistem basis data perlu mempertimbangkan karakteristik operasi dan kebutuhan aplikasi.

Dalam penelitian ini, *MySQL* dimanfaatkan sebagai basis data utama untuk menyimpan seluruh informasi terkait proses pengajuan dan status pembayaran klaim JKK. Integrasi antara *MySQL* dengan *framework* Laravel dilakukan melalui penggunaan fitur *Eloquent ORM*, yang memungkinkan pengelolaan data dilakukan secara lebih terstruktur melalui *model* pada aplikasi. Pemilihan *MySQL* juga disesuaikan dengan kebutuhan pengembangan sistem SIJAKKA serta dukungannya terhadap lingkungan pengembangan aplikasi berbasis PHP dan Laravel.

2.2.11 *Unit testing*

Unit testing merupakan proses pengujian yang dilakukan terhadap *Unit* terkecil dari perangkat lunak, seperti fungsi, metode, atau modul, untuk memastikan bahwa setiap *Unit* telah bekerja sesuai dengan logika yang dirancang. Pengujian ini dilakukan sebelum proses integrasi sehingga kesalahan dapat ditemukan sejak tahap awal pengembangan dan mengurangi risiko terjadinya kesalahan pada tahap pengembangan berikutnya.

Salah satu pendekatan yang dapat digunakan untuk mengevaluasi kualitas *Unit testing* adalah *Mutation Testing*. Menurut Sánchez et al. (2024), *Mutation Testing* merupakan teknik yang dapat digunakan untuk mengevaluasi kemampuan *test suite* dalam mendeteksi kesalahan dengan memasukkan kesalahan sintetis ke dalam perangkat lunak. Perubahan yang dihasilkan disebut *mutant*, kemudian *test suite* dijalankan terhadap *mutant* tersebut untuk mengetahui apakah perubahan yang diberikan dapat dideteksi oleh pengujian. Apabila pengujian gagal setelah perubahan diberikan, *mutant* dinyatakan berhasil dideteksi atau *killed*. Sebaliknya, apabila pengujian tetap berhasil, *mutant* dinyatakan *survived*.

Moradi Dakhel et al. (2024) menggunakan *Mutation Testing* untuk mengevaluasi efektivitas *Test Case* dalam mendeteksi kesalahan pada kode program. Efektivitas pengujian dapat dievaluasi melalui *mutation score*, yaitu nilai yang menunjukkan kemampuan *test suite* dalam mendeteksi *mutant* yang dihasilkan. Semakin banyak *mutant* yang berhasil dideteksi, semakin tinggi kemampuan *test suite* dalam mengidentifikasi perubahan atau kesalahan yang disisipkan pada kode program.

Pada penelitian ini, *Mutation Testing* digunakan untuk mengevaluasi kualitas *Unit testing* pada beberapa fungsi utama *back-end* sistem, seperti autentikasi pengguna, pengelolaan data pengguna, pengelolaan data rumah sakit atau klinik, pengelolaan data klaim, perubahan status pembayaran klaim, serta pembuatan laporan. Pelaksanaan *Mutation Testing* didukung oleh *PHPUnit* sebagai *framework Unit testing* dan *Infection PHP* sebagai alat untuk menghasilkan serta mengevaluasi *mutant* pada kode program. *Infection PHP* menjalankan kumpulan *Unit testing* yang telah dibuat menggunakan *PHPUnit*, kemudian menghasilkan

nilai *Mutation Score Indicator* (MSI) berdasarkan *mutant* yang berhasil dideteksi maupun yang tidak terdeteksi.

2.2.12 *Integration Testing*

Gambar 2.3 *Integration Testing*

Integration testing merupakan tahap pengujian yang dilakukan untuk memverifikasi interaksi dan keterhubungan antara komponen atau modul yang telah diintegrasikan dalam suatu sistem. Pengujian ini bertujuan untuk memastikan bahwa komponen yang telah berfungsi secara individual tetap dapat bekerja dengan baik ketika saling berinteraksi. Melalui *integration testing*, kesalahan yang muncul pada proses komunikasi, pertukaran data, maupun hubungan antar komponen dapat diidentifikasi sebelum sistem diuji secara menyeluruh.

Menurut Hellhake et al. (2022), *integration testing* pada sistem berbasis komponen berfokus pada pengujian interaksi antarkomponen untuk menemukan kesalahan yang muncul ketika komponen-komponen tersebut diintegrasikan. Pengujian integrasi menjadi penting karena hubungan dan tingkat keterkaitan antarkomponen dapat memengaruhi proses pengujian serta kemungkinan ditemukannya kesalahan pada sistem.

Selain itu, Chrisna et al. (2024) menerapkan *integration testing* untuk mengevaluasi integrasi komponen dalam aplikasi melalui pendekatan *white-box* dan *black-box testing*. Pengujian tersebut menunjukkan bahwa *integration testing* dapat digunakan untuk memverifikasi keterhubungan dan interaksi antarbagian aplikasi sehingga fungsi yang melibatkan lebih dari satu komponen dapat berjalan sesuai dengan hasil yang diharapkan.

Dalam pengembangan Sistem Informasi Pengajuan dan Status Bayar Klaim JKK BPJS Ketenagakerjaan Pekanbaru Kota, *integration testing* dilakukan menggunakan PHP*Unit* melalui *Feature Test* untuk menguji keterkaitan antarproses dan komponen pada *back-end*. Pengujian mencakup autentikasi,

registrasi akun, pengelolaan kata sandi, pengelolaan profil, validasi hak akses, pengelolaan data klaim, serta pengunduhan dokumen PDF. Pengujian ini bertujuan untuk memastikan bahwa setiap komponen yang terlibat dapat saling terintegrasi dengan baik serta menghasilkan keluaran sesuai dengan hasil yang diharapkan.

2.2.13 *System Testing*

System Testing merupakan pengujian perangkat lunak secara menyeluruh untuk memverifikasi bahwa seluruh fungsi yang telah terintegrasi dapat berjalan sesuai dengan kebutuhan sistem. Pengujian ini dilakukan dengan menjalankan skenario penggunaan sistem untuk memastikan bahwa setiap fungsi dapat bekerja sebagai satu kesatuan. Tahap *System Testing* dilakukan setelah *Unit testing* dan *integration testing* selesai dilakukan.

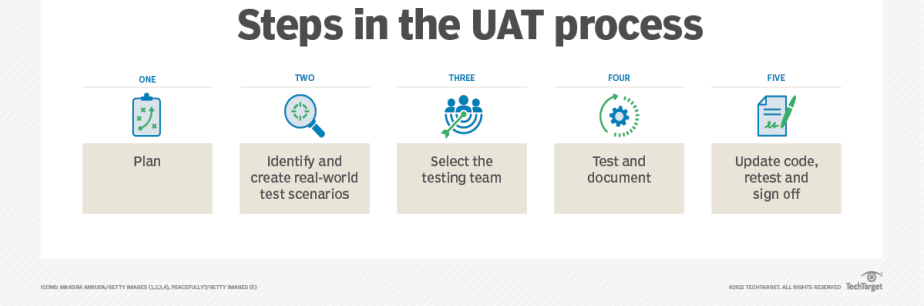
Dalam pelaksanaannya, *System Testing* dapat dilakukan dengan pendekatan *End-to-End (E2E) Testing*, yaitu pengujian yang memvalidasi alur aplikasi secara menyeluruh berdasarkan interaksi pengguna. Menurut Talakola (2022), *End-to-End Testing* memungkinkan pengujian dilakukan dengan mereplikasi interaksi pengguna dalam kondisi penggunaan nyata untuk memvalidasi aplikasi secara menyeluruh. Pendekatan ini juga membantu mengidentifikasi permasalahan integrasi antarbagian pada aplikasi yang mungkin tidak ditemukan ketika setiap komponen diuji secara terpisah.

Pengujian aplikasi *web* secara menyeluruh juga dapat didukung melalui penggunaan *framework* pengujian otomatis berbasis *browser*. Bhimanapati, Goel, dan Jain (2024) menjelaskan bahwa penggunaan *framework* pengujian otomatis pada aplikasi *web* dapat mendukung pengujian *End-to-End* dengan menjalankan interaksi pengguna secara otomatis serta membantu mengevaluasi fungsi aplikasi secara menyeluruh.

Pada penelitian ini, *System Testing* dilakukan menggunakan Laravel Dusk dengan pendekatan *End-to-End Testing*. Laravel Dusk digunakan untuk menjalankan skenario pengujian melalui *browser* sehingga interaksi pengguna dengan sistem dapat disimulasikan secara otomatis. Pengujian dilakukan terhadap alur utama sistem SIJAKKA, mulai dari proses *login*, mengakses *dashboard*, memilih RS/Klinik, mengunggah data klaim, melihat laporan, melakukan *filter* data, mengunduh laporan, hingga *logout*.

Melalui pengujian tersebut, seluruh fungsi yang telah terintegrasi dapat diverifikasi dalam satu alur penggunaan sistem. Hasil *System Testing* digunakan untuk memastikan bahwa sistem telah berjalan sesuai dengan kebutuhan dan siap dilanjutkan ke tahap *User Acceptance Testing* (UAT) oleh pengguna.

2.2.14 *User Acceptance Testing (UAT)*



Gambar 2.4 *User Acceptance Testing (UAT)*

User Acceptance Testing (UAT) adalah tahap pengujian yang dilakukan oleh pengguna akhir untuk memastikan bahwa sistem yang dikembangkan dapat memenuhi kebutuhan dan harapan mereka. UAT dilaksanakan setelah proses pengembangan selesai dan sistem dianggap stabil untuk diuji oleh pengguna. Tahap ini berfungsi untuk memverifikasi bahwa seluruh fungsi sistem berjalan sesuai dengan persyaratan yang telah ditentukan sebelumnya, serta untuk memastikan sistem dapat diterima dengan baik oleh pengguna.

Menurut Priyatna, Hananto, & Nova (2022), UAT merupakan bagian penting dalam pengembangan perangkat lunak karena melibatkan pengguna langsung untuk mengevaluasi kesesuaian sistem dengan kebutuhan operasional. Melalui UAT, pengguna dapat menilai apakah sistem yang dibangun benar-benar mampu mendukung aktivitas mereka, sekaligus mengidentifikasi potensi masalah sebelum implementasi penuh dilakukan.

Dalam sistem pengajuan dan status bayar klaim JKK pada BPJS Ketenagakerjaan Pekanbaru Kota, *User Acceptance Testing* (UAT) dilakukan oleh petugas BPJS Ketenagakerjaan sebagai pengguna sistem. Pengujian ini mencakup berbagai proses bisnis, mulai dari pengelolaan data rumah sakit atau klinik, unggah data klaim, pengelolaan data klaim, hingga pemantauan status pembayaran dan penyajian laporan. Hasil UAT digunakan untuk memastikan bahwa sistem telah memenuhi kebutuhan operasional pengguna, mudah digunakan, serta layak diterapkan pada lingkungan kerja BPJS Ketenagakerjaan Pekanbaru Kota.