

BAB II

TINJAUAN PUSTAKA

2.1 Sistem Informasi

Menurut Adham, (2024) Sistem informasi adalah serangkaian komponen yang bekerja bersama untuk mengumpulkan, mengelola, menyimpan, memproses, dan menyebarkan informasi yang diperlukan untuk mendukung pengambilan keputusan dalam suatu organisasi atau entitas. Salah satu komponen utamanya adalah pemanfaatan teknologi berbasis perangkat keras (*hardware*) dan perangkat lunak (*software*).

Menurut Nindya Filza M, dkk (2024) Pengembangan sistem informasi merupakan proses penting di era modern karena sistem informasi memainkan peran krusial dalam mengelola, memproses, menyimpan, dan mengambil data, serta membantu pengambilan keputusan yang diperlukan bagi organisasi.

2.2 Aplikasi Berbasis Web

Menurut Johan & Chandra, (2024) menyatakan bahwa aplikasi web merupakan program berbasis internet yang memanfaatkan teknologi browser sebagai antarmuka pengguna. Pengembangan aplikasi web modern telah berkembang pesat dengan hadirnya berbagai *framework* dan *library* yang memudahkan para pengembang dalam membangun sistem yang kompleks namun tetap efisien dan mudah dikelola.

Menurut Nugroho & Voutama, (2023) Aplikasi berbasis web merupakan salah satu perkembangan teknologi informasi yang signifikan, dirancang untuk meningkatkan efisiensi, mengurangi hal yang tidak penting, meningkatkan kualitas

layanan, serta mengkoordinasikan setiap bagian dalam suatu organisasi agar tercapainya peningkatan kualitas manajemen.

2.3 Pelatihan *Online* (*E-learning*)

Menurut Muhammad Saifur Rijal, (2022) *E-learning* merupakan sistem pelatihan dan pengembangan berbasis *online* yang memanfaatkan teknologi digital dalam proses penyampaian materi pembelajaran. Sistem ini memberikan keunggulan dalam hal kecepatan akses informasi serta fleksibilitas instruksi, karena peserta dapat mengakses materi kapan saja dan di mana saja tanpa terikat oleh batasan waktu dan lokasi fisik. Selain itu, *e-learning* juga terbukti mampu meningkatkan produktivitas karena peserta dapat belajar sesuai dengan kecepatan belajar masing-masing.

Menurut Rifqi dkk., (2023) menjelaskan bahwa pelatihan *online* berbasis web memiliki karakteristik utama berupa kemudahan akses, efisiensi biaya, serta kemampuan untuk melacak perkembangan belajar peserta secara otomatis dan *real-time*. Dengan memanfaatkan platform digital, lembaga pelatihan dapat menjangkau lebih banyak peserta tanpa terbatas oleh kapasitas ruangan maupun lokasi geografis.

2.4 Uji Kompetensi *Online*

Menurut Gamaliel dkk., (2021) menyatakan bahwa ujian *online* merupakan salah satu bagian dari sistem pendidikan jarak jauh yang memanfaatkan media teknologi internet. Konsep ujian *online* yang tidak terikat oleh ruang dan waktu dapat menjadi solusi atas berbagai kendala yang ada pada metode ujian konvensional. Sistem ini dilengkapi dengan fasilitas manajemen soal, manajemen ujian, pengelolaan data pengguna, serta proses pelaksanaan ujian yang

menghasilkan *Output* nilai secara otomatis, sehingga proses evaluasi menjadi lebih efisien dan terukur.

Menurut Arif Budiman Sidiq, (2021) Sistem informasi ujian *online* dirancang untuk memberikan layanan pelaksanaan ujian yang lebih praktis dan mengikuti perkembangan teknologi. Dalam perancangannya, sistem ini memerlukan fasilitas terkait pengelolaan soal, pengelolaan ujian, pengelolaan guru, dan pengelolaan siswa. Sistem dibangun menggunakan *framework CodeIgniter* dengan bahasa pemrograman PHP yang mampu mengintegrasikan seluruh layanan ke dalam satu platform berbasis web, sehingga pelaksanaan ujian *online* di sekolah dapat berjalan lebih sistematis dan terstruktur.

2.5 Sertifikat Digital

Menurut Ira Zulfa, Eliyin, (2025) (2025), menyatakan bahwa penerbitan sertifikat digital secara otomatis dapat menghemat waktu, mengurangi biaya produksi, sekaligus memastikan konsistensi format dan data yang tercantum dalam sertifikat. Sertifikat digital juga lebih ramah lingkungan dan dapat diarsipkan secara sistematis dalam basis data untuk keperluan verifikasi di masa mendatang.

Menurut Putra dkk., (2019) menyatakan bahwa sertifikat kompetensi digital merupakan bentuk pengakuan resmi atas kemampuan teknis seseorang yang tidak cukup hanya diwakili oleh ijazah akademik semata, melainkan harus dibuktikan melalui mekanisme uji kompetensi yang terstandarisasi dan diakui oleh lembaga berwenang seperti Badan Nasional Sertifikasi Profesi (BNSP). Dalam penelitiannya, aplikasi sertifikasi *online* yang dikembangkan terbukti mampu meningkatkan efektivitas dan efisiensi kegiatan uji kompetensi, terutama dalam hal

kemudahan pendaftaran, pengelolaan soal ujian, penilaian oleh *asesor*, hingga penerbitan sertifikat secara digital. Selain itu, sistem berbasis paperless ini turut berkontribusi dalam mengurangi ketergantungan pada dokumen fisik yang rentan terhadap kerusakan dan kehilangan data.

2.6 Node.js

Menurut Eli Nurhayati, (2023) menjelaskan bahwa Node.js merupakan platform server-side yang memungkinkan pengembang untuk menggunakan *JavaScript* baik di sisi klien maupun server sekaligus, sehingga mempermudah proses pengembangan *full-stack*. Dalam penelitiannya tentang pengembangan *back-end* API menggunakan Node.js dan Express, disebutkan bahwa arsitektur Node.js yang berbasis *event loop* memungkinkan penanganan banyak koneksi secara bersamaan tanpa membutuhkan banyak sumber daya memori.

Menurut Calista Chandra dkk., (2024) menyatakan bahwa Node.js dengan *framework* Express JS merupakan teknologi yang digunakan untuk membangun web API yang berfungsi sebagai komponen *back-end* dari sebuah aplikasi *mobile*. Dalam penelitiannya, Node.js dipilih karena kemampuannya dalam mendukung arsitektur REST (*Representational State Transfer*) yang memungkinkan komunikasi antara *client* dan server secara efisien melalui pola *request-response*. Sistem yang dibangun menggunakan Node.js berhasil memberikan performa optimal bahkan ketika diakses oleh hingga 50 pengguna secara bersamaan, yang membuktikan keandalan Node.js sebagai platform *back-end* dalam lingkungan pengembangan aplikasi berskala nyata.

2.7 TypeScript

Menurut Wijaya, (2025) menyatakan bahwa *TypeScript* menawarkan keunggulan berupa deteksi kesalahan kode pada saat kompilasi (*compile-time*), bukan pada saat eksekusi (*runtime*), sehingga meminimalkan *bug* yang lolos ke lingkungan produksi. Dengan menggunakan *TypeScript*, pengembang mendapatkan dukungan *tooling* yang lebih baik seperti *autocomplete*, *refactoring*, dan navigasi kode yang akurat di berbagai code editor modern.

Menurut Kesuma (2024) menyatakan bahwa *TypeScript* telah menjadi pilihan utama dalam pengembangan aplikasi *frontend* maupun *backend* modern karena kemampuannya dalam meningkatkan keterbacaan dan keterpeliharaan kode (*code maintainability*). *TypeScript* dapat dikompilasi menjadi *JavaScript* biasa sehingga kompatibel dengan semua platform yang mendukung *JavaScript*, menjadikannya pilihan yang fleksibel untuk berbagai proyek pengembangan perangkat lunak.

2.8 Redis

Menurut Prasetyo (2024) menyatakan bahwa Redis merupakan basis data berbasis memori (*in-memory database*) yang memungkinkan penyimpanan dan akses data secara sangat cepat. Redis bekerja dengan menyimpan data dalam bentuk pasangan *key-value* di dalam RAM, sehingga kecepatan baca dan tulisnya jauh lebih tinggi dibandingkan *database* berbasis *disk* seperti *MySQL*. Implementasi Redis terbukti efektif sebagai solusi untuk mempercepat operasi pengambilan data pada sistem yang memiliki volume transaksi besar.

Menurut Dipraja & Rahman, (2025) mendefinisikan Redis sebagai teknologi *distributed caching* yang bekerja dengan menyimpan sementara data yang sering

diakses di memori, sehingga sistem tidak perlu berulang kali melakukan kueri langsung ke *database* utama yang lebih lambat. Berbeda dari sistem basis data konvensional yang bergantung pada penyimpanan berbasis *disk*, Redis memanfaatkan RAM sebagai media penyimpanan utama, yang memungkinkan kecepatan baca-tulis jauh melebihi teknologi *database* tradisional.

2.9 Websocket

Menurut Sena dkk., (2024) menjelaskan bahwa *WebSocket communication protocol* merupakan teknologi yang memungkinkan komunikasi dua arah (*full-duplex*) secara persisten antara *client* dan *server* dalam satu saluran koneksi tunggal, berbeda dari HTTP konvensional yang hanya mendukung pola *request-response* searah. Keunggulan ini menjadikan *WebSocket* sebagai solusi utama dalam sistem daring yang membutuhkan pembaruan data secara *real-time*, seperti penilaian kuis, notifikasi langsung, dan manajemen sesi pengguna secara simultan.

Menurut Hizbullah dkk., (2017) menyatakan bahwa *WebSocket* hadir menggantikan metode *HTTP polling* yang tidak efisien, dengan mempertahankan satu koneksi TCP yang tetap terbuka selama sesi berlangsung sehingga pertukaran data terjadi secara langsung tanpa *overhead* koneksi berulang. Protokol ini sangat relevan untuk aplikasi modern seperti layanan pesan instan, *Dashboard monitoring*, dan sistem pengujian daring yang menuntut latensi rendah dan *throughput* tinggi secara bersamaan.

2.10 Framework

Menurut Gunawan dkk., (2024) mendefinisikan *framework* sebagai kerangka kerja terstruktur dan teruji yang mempermudah proses pengembangan

aplikasi secara efisien dan efektif, di mana pengembang dapat mengurangi waktu pengembangan, mengadopsi praktik terbaik, serta memanfaatkan modul dan komponen yang telah ada untuk mengatasi tantangan teknis yang kompleks. Pemanfaatan *framework* terbukti meningkatkan produktivitas, konsistensi, dan kualitas perangkat lunak secara signifikan, sekaligus memastikan keberlanjutan dan skalabilitas solusi teknologi yang dihasilkan untuk memenuhi kebutuhan pasar yang terus berkembang.

Menurut Sinlae dkk., (2024) menyatakan bahwa *Framework* merupakan kerangka kerja yang membantu pengembang membangun aplikasi *website* dengan struktur kode yang bersih, terorganisir, dan dapat diatur dengan baik, serta menyediakan berbagai fitur bawaan yang mendukung pengembangan modern seperti *routing*, *middleware*, dan manajemen *database*.

2.10.1 NextJS (*Frontend*)

Menurut Hidayatullah & Suartana, (2025) menjelaskan bahwa Next.js adalah *framework* yang menerapkan teknik *Server Side Rendering* (SSR), yaitu metode *rendering* halaman yang dilakukan di sisi server sehingga HTML yang dikirim ke klien telah berisi konten lengkap dan siap ditampilkan tanpa menunggu proses *rendering* di sisi klien. Keunggulan ini menjadikan Next.js unggul dibanding pendekatan konvensional berbasis *Client Side Rendering* (CSR) dalam hal kecepatan tampil dan optimasi mesin pencari (SEO).

Menurut Hanafi dkk., (2024) menyatakan bahwa Next.js merupakan *framework* yang dirancang untuk mengatasi keterbatasan React dalam *server-side rendering*, dengan menyediakan tiga metode *rendering* sekaligus yaitu CSR, SSR,

dan SSG (*Static Site Generation*) yang dapat dipilih sesuai kebutuhan halaman. Penelitian ini mengungkap bahwa SSG terbukti paling cepat pada pemuatan awal (*initial load*) dengan waktu 180 ms, sementara SSR memberikan stabilitas performa meski dengan konsumsi sumber daya server lebih besar.

2.10.2 ExpressJS (Backend)

Menurut Hadinata dkk., (2024) menyatakan bahwa Express.js adalah *framework* backend berbasis Node.js yang digunakan untuk membangun *RESTful* API, yaitu antarmuka yang mampu mengintegrasikan data dan menghubungkan aplikasi yang berjalan di berbagai *platform* secara efisien. Pemilihan Express.js sebagai teknologi *backend* dinilai sangat krusial karena berpengaruh langsung terhadap kinerja server dalam menangani permintaan.

Menurut Bachtiar dkk., (2024) menjelaskan bahwa Express.js merupakan *framework* yang sudah mapan dan banyak digunakan untuk membangun komponen *backend* berbasis arsitektur REST, di mana setiap *endpoint* API menerima *Input* dan menghasilkan *Output* dalam format JSON yang dapat dikonsumsi oleh aplikasi *frontend* maupun *mobile*.

2.11 TailwindCSS

Menurut Azhariyah & Mukhlis, (2023) mendefinisikan Tailwind CSS sebagai *framework* CSS berbasis utilitas yang memungkinkan pengembang membangun antarmuka pengguna yang responsif dan konsisten dengan lebih cepat, tanpa memerlukan banyak kode CSS manual sehingga menyederhanakan dan mempercepat seluruh proses pengembangan *frontend*. Berbeda dari *framework* konvensional yang menyediakan komponen siap pakai dengan tampilan yang

terbatas, Tailwind CSS memberikan kebebasan penuh kepada pengembang dalam menentukan gaya dan tata letak sesuai kebutuhan melalui kelas utilitas yang dapat dikombinasikan secara fleksibel.

Menurut Rifandi dkk., (2022) menjelaskan bahwa Tailwind CSS merupakan *framework* yang mengutamakan pendekatan *utility-first*, di mana pengembang tidak bergantung pada komponen yang sudah dibuat sebelumnya, melainkan menggabungkan kelas-kelas utilitas yang dapat dikustomisasi untuk menciptakan desain yang unik dan konsisten.

2.12 MySQL

Menurut Dachi dkk., (2025) mendefinisikan *MySQL* sebagai salah satu *Relational Database Management System* (RDBMS) yang berperan krusial dalam sistem informasi modern karena memungkinkan penyimpanan, pengelolaan, dan pengambilan data terstruktur secara efisien melalui operasi CRUD (*Create, Read, Update, Delete*).

Menurut Jurisman Waruwu, dkk., (2024) menyatakan bahwa *MySQL* merupakan *database* pilihan utama dalam pengembangan sistem informasi berbasis web karena kemampuannya menyimpan dan mengelola data secara terstruktur sesuai kebutuhan sistem.

2.13 Laragon

Menurut Panjaitan dkk., (2025) menjelaskan bahwa Laragon merupakan *local development environment* yang digunakan sebagai lingkungan pengembangan sistem berbasis web, di mana seluruh komponen server seperti PHP, *MySQL*, dan *web server* dapat dijalankan secara lokal tanpa memerlukan koneksi internet.

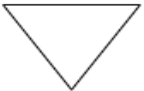
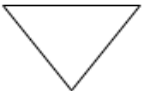
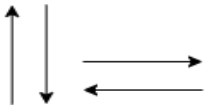
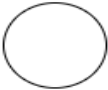
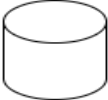
Menurut Mair dkk., (2025) menyatakan bahwa Laragon digunakan sebagai *local server* dalam proses pengembangan sistem informasi berbasis web, berfungsi menyediakan lingkungan pengembangan yang stabil dan mendukung berbagai teknologi seperti PHP, *MySQL*, serta *web server* sebelum sistem dipublikasikan secara *online*.

2.14 Aliran Sistem Informasi

Menurut Ridho, (2021) menyatakan bahwa aliran sistem informasi merupakan suatu alat bantu sistem yang digunakan untuk merancang sebuah sistem dari awal sampai akhir perancangan. Lebih lanjut disebutkan pula bahwa menurut Jogyanto, aliran sistem informasi merupakan sebuah bagan untuk menggambarkan arus kerja dari awal sampai akhir pengerjaan program tersebut.

Menurut Asril, (2020) menyatakan bahwa dalam perancangan sistem informasi, Aliran Sistem Informasi digunakan untuk menggambarkan kondisi sistem yang sedang berjalan (*current system*) dan sistem yang diusulkan (*proposed system*). ASI digambarkan menggunakan simbol-simbol standar seperti simbol dokumen, proses, *database*, keputusan, dan penghubung antar proses, yang disusun secara berurutan dari atas ke bawah dan dari kiri ke kanan untuk merepresentasikan alur data secara logis dan terstruktur.

Tabel 2.1 Simbol-simbol pada Aliran Sistem Informasi

Simbol	Nama	Keterangan
	Proses Komputerisasi	Menunjukkan proses yang dilakukan secara otomatis oleh sistem komputer
	Dokumen	Menggambarkan dokumen masukan atau keluaran dari suatu proses.
	Proses Manual	Menunjukkan aktivitas yang masih dilakukan secara manual oleh pengguna.
 (A)	Penyimpanan Data Alfabetik	Menggambarkan arsip atau <i>file</i> yang disimpan berdasarkan urutan abjad
 (B)	Penyimpanan Data Kronologis	Menggambarkan arsip yang disusun berdasarkan urutan waktu atau tanggal.
	Garis Alir (<i>Flow Line</i>)	Menunjukkan arah aliran data atau proses dalam sistem
	Penghubung (Connector)	Digunakan untuk menghubungkan alur yang terpisah dalam satu halaman atau antar halaman.
	Basis Data	Menunjukkan tempat penyimpanan data secara elektronik di dalam sistem

2.15 Unified Modeling Language (UML)

Menurut Binangkit dkk., (2023) UML adalah metode pemodelan visual yang digunakan sebagai sarana perancangan sistem berorientasi objek, membantu pengembang memvisualisasikan struktur, perilaku, dan interaksi komponen sistem sebelum masuk tahap implementasi. Penerapan UML mencakup beberapa jenis

diagram seperti *Use Case*, *Activity*, *Sequence*, dan *Class Diagram*, yang masing-masing merepresentasikan aspek berbeda dari sistem yang dikembangkan.

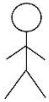
Menurut Voutama, (2022) UML adalah model umum yang paling banyak digunakan dalam proses perancangan sistem, membantu menggambarkan model proses (*blueprint*) dari suatu rekayasa perangkat lunak berorientasi objek termasuk struktur aktor, aktivitas, dan mekanisme sistem berbasis *website*.

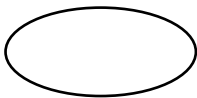
2.15.1 *Use Case Diagram*

Menurut Rhifky dan Fahmi (2023) *Use Case Diagram* menjelaskan sebuah fungsi yang disediakan oleh sistem yang menghasilkan *Output* terlihat bagi aktor, di mana aktor menjelaskan setiap entitas yang berinteraksi dengan sistem — menghasilkan definisi batas sistem (*system boundary*) yang memisahkan tugas aktor dan sistem.

Menurut Khairunnisa dkk., (2024) *Use Case Diagram* menggambarkan fungsionalitas sistem dan interaksi antara aktor dan sistem, membantu pengembang mengidentifikasi semua fungsi yang harus disediakan berdasarkan kebutuhan pengguna.

Tabel 2.2 Simbol-simbol *Use Case Diagram*

Simbol	Nama Simbol	Keterangan
	<i>Actor</i>	Menunjukkan entitas eksternal yang berinteraksi langsung dengan sistem, seperti pengguna atau perangkat lain.

	<i>Use Case</i>	Menggambarkan fungsi atau layanan utama yang disediakan oleh sistem. Biasanya ditulis dalam bentuk elips.
	<i>System Boundary</i>	Batas sistem yang menunjukkan ruang lingkup fungsional dari sistem yang akan dikembangkan. Semua <i>Use Case</i> berada di dalam batas ini.
	Asosiasi	Garis penghubung antara aktor dan <i>Use Case</i> yang menunjukkan adanya interaksi.
	<i>Include</i>	Menunjukkan bahwa satu <i>Use Case</i> selalu memanggil <i>Use Case</i> lain sebagai bagian dari alurnya.
	<i>Extend Relationship</i>	Menunjukkan <i>Use Case</i> tambahan yang dapat dijalankan bila kondisi tertentu terpenuhi.

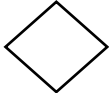
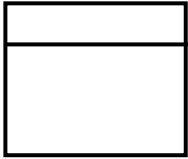
2.15.2 Activity Diagram

Menurut M. Rhifky Wahyudi (2023) *Activity Diagram* adalah diagram dalam *Unified Modeling Language* (UML) yang digunakan untuk memodelkan alur kerja atau aktivitas dalam sebuah sistem secara dinamis. Diagram ini menampilkan urutan aktivitas, kondisi percabangan, aktivitas paralel, dan pengendalian aliran proses, sehingga memudahkan pemahaman bagaimana proses atau sistem bekerja secara keseluruhan. *Activity Diagram* juga sering dipakai untuk memodelkan proses bisnis maupun alur kerja dalam pengembangan sistem informasi.

Menurut Gina dan Apriade (2024) *Activity Diagram* digunakan untuk merancang alur proses sistem mulai dari manajemen inventaris, peminjaman,

verifikasi, hingga pelaporan, menggambarkan setiap langkah yang dilakukan oleh admin maupun pengguna secara visual.

Tabel 2.3 Simbol-simbol *Activity Diagram*

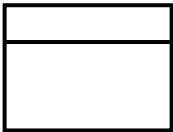
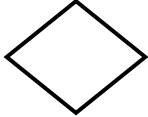
Simbol	Nama	Keterangan
	<i>Initial Node</i>	Titik awal dimulainya suatu proses atau aktivitas dalam sistem.
	<i>Activity/Action</i>	Menunjukkan aktivitas atau tindakan yang dilakukan dalam sistem.
	<i>Decision Node</i>	Titik pengambilan keputusan yang menyebabkan alur bercabang berdasarkan kondisi tertentu.
	<i>Final Node</i>	Titik akhir dari suatu alur aktivitas.
	<i>Control Flow</i>	Menunjukkan arah aliran dari satu aktivitas ke aktivitas lainnya.
	<i>Swimlane</i>	Digunakan untuk membagi tanggung jawab aktivitas berdasarkan pelaku atau entitas tertentu (misalnya admin, peserta, sistem).
	<i>Join Node</i>	Menunjukkan penggabungan dua atau lebih alur paralel menjadi satu alur aktivitas berikutnya.

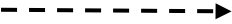
2.15.3 Class Diagram

Menurut Mustofa dkk., (2022) *Class Diagram* merupakan inti dari pengembangan dan desain berorientasi objek yang menggambarkan struktur kelas beserta atribut, metode, dan hubungan antar kelas seperti *containment*, pewarisan, dan asosiasi, sehingga pengembang dapat memahami arsitektur sistem secara menyeluruh sebelum implementasi.

Menurut Aqilah dkk., (2023) *Class Diagram* dalam UML digunakan untuk merancang antarmuka sistem secara visual dengan menggambarkan kelas-kelas, atribut, dan relasi antar komponen sistem sebelum memasuki tahap implementasi kode. Penggunaan *Class Diagram* bersama *Use Case Diagram* memungkinkan pengembang merancang fitur-fitur sistem secara terstruktur seperti manajemen pengguna dengan hak akses berbeda, pelacakan data secara *real-time*, dan pembuatan laporan.

Tabel 2.4 Simbol-simbol *Class Diagram*

Simbol	Nama	Keterangan
	<i>Class</i>	Menunjukkan objek dalam sistem beserta atribut dan metodenya.
	<i>Nary Association</i>	Menunjukkan hubungan antara tiga atau lebih kelas.
	<i>Realization</i>	Hubungan antara interface dan kelas yang mengimplementasikannya.

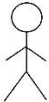
	<i>Dependency</i>	Menunjukkan ketergantungan satu kelas terhadap kelas lain.
	<i>Generalization</i>	Menunjukkan hubungan pewarisan (inheritance) antara kelas induk dan anak.
	<i>Association</i>	Hubungan langsung antara dua kelas yang saling berinteraksi.

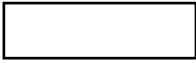
2.15.4 Sequence Diagram

Menurut Rohmanto & Setiawan, (2022) *Sequence Diagram* adalah pemodelan UML yang digunakan untuk menggambarkan dinamika interaksi antar objek dalam sebuah *Use Case* secara berurutan berdasarkan waktu melalui pesan (*message*) antar *lifeline*, sehingga alur logika sistem dapat dipahami secara mendetail sebelum implementasi.

Menurut Khilda & Lisna, (2022) *Sequence Diagram* adalah salah satu diagram UML yang menggambarkan urutan interaksi antar objek dalam sistem secara kronologis, memperlihatkan pesan yang dikirim dan diterima antar objek untuk mengimplementasikan suatu fungsi tertentu.

Tabel 2.5 Simbol-simbol *Sequence Diagram*

Simbol	Nama	Keterangan
	<i>Actor</i>	Entitas eksternal (misalnya peserta, instruktur, atau admin) yang berinteraksi dengan sistem.

	<i>Object</i>	Menggambarkan <i>object</i> apa saja yang terlibat dan berpartisipasi secara berurutan dengan mengirim atau menerima pesan
	<i>Lifeline</i>	Garis vertikal yang menunjukkan keberlangsungan <i>object</i> selama proses berlangsung
	<i>Message Call</i>	Pesan dikirim dan menunggu respon (fungsi dipanggil).
	<i>Return Message</i>	Pesan balasan dari fungsi yang dipanggil.
	<i>Activation</i>	Menunjukkan kapan objek sedang aktif melakukan proses.

2.16 Kesimpulan

Berdasarkan uraian landasan teori yang telah dipaparkan pada bab ini, dapat disimpulkan bahwa pengembangan sistem informasi pelatihan dan uji kompetensi berbasis web membutuhkan pemahaman yang komprehensif terhadap berbagai konsep, teknologi, dan metode perancangan yang saling mendukung. Konsep sistem informasi, aplikasi berbasis web, pelatihan *online*, uji kompetensi *online*, dan sertifikat digital menjadi landasan fungsional yang menentukan arah pengembangan sistem secara keseluruhan.

Dari sisi teknologi, *Node.js*, *TypeScript*, *Redis*, *WebSocket*, *Next.js*, *Express.js*, *TailwindCSS*, *MySQL*, dan *Laragon* memberikan fondasi teknis yang kuat untuk membangun sistem yang efisien, responsif, dan skalabel, di mana masing-masing

teknologi memiliki peran spesifik yang saling melengkapi mulai dari pengelolaan *backend*, *caching*, komunikasi *real-time*, antarmuka pengguna, hingga penyimpanan data.

Dalam aspek perancangan, Aliran Sistem Informasi (ASI) dan diagram-diagram UML yang meliputi *Use Case Diagram*, *Activity Diagram*, *Class Diagram*, dan *Sequence Diagram* digunakan untuk memodelkan sistem secara menyeluruh dari sisi kebutuhan, proses, struktur, hingga interaksi antar objek, sehingga sistem yang dibangun dapat terstruktur, terukur, dan dapat dipertanggungjawabkan secara ilmiah.