BAB I
PENDAHULUAN

1.1 Latar belakang

Transformasi digital saat ini mengalami perkembangan yang sangat pesat. Di era digital, gaya hidup manusia telah berubah drastis dan tidak terlepas dari perangkat elektronik yang mendukung hampir seluruh aktivitas sehari-hari. Teknologi tidak hanya mempermudah berbagai tugas dan pekerjaan, tetapi juga telah menjadi motor penggerak peradaban yang membawa manusia ke dalam era digital yang lebih efisien dan terintegrasi (Setiawan, 2017).

Dalam dunia pendidikan, transformasi digital telah mengubah paradigma pembelajaran secara signifikan. Metode pembelajaran tradisional, yang sebelumnya mengandalkan ceramah dan ketergantungan pada dosen sebagai sumber utama pengetahuan, kini telah bergeser ke model pembelajaran yang lebih interaktif. Pemanfaatan teknologi informasi dan komunikasi memungkinkan mahasiswa untuk lebih aktif dalam proses belajar, di mana peran dosen berubah menjadi fasilitator yang mendukung kemandirian dan kreativitas (Cholik, 2017). Perubahan ini tidak hanya meningkatkan akses terhadap informasi, tetapi juga mendorong partisipasi aktif serta kolaborasi yang lebih luas antar mahasiswa.

Namun, di tengah berbagai kemudahan yang ditawarkan oleh teknologi digital, muncul tantangan-tantangan baru yang perlu diatasi. Mahasiswa, sebagai salah satu pengguna utama teknologi dalam kegiatan akademik, seringkali menghadapi dilema berupa distraksi digital yang dapat mengganggu fokus belajar. Sistem pengelolaan tugas dan proyek yang masih bersifat manual atau konvensional sering kali memicu fenomena prokrastinasi akademik, yaitu kecenderungan untuk menunda memulai atau menyelesaikan tugas-tugas akademik yang penting (Pratiwi & Sawitri, 2015). Fenomena ini telah mempengaruhi jutaan mahasiswa di seluruh dunia dan kerap menjadi akar dari kegagalan atau kinerja akademik yang tidak optimal (Wangid, 2019). Penelitian oleh Novitasari (2017) mengungkapkan bahwa rendahnya *self-efficacy* atau keyakinan diri dalam mencapai tujuan, dikombinasikan dengan lemahnya keterampilan manajemen waktu, berkontribusi signifikan terhadap meningkatnya kecenderungan prokrastinasi di kalangan mahasiswa. Hal ini diperkuat oleh temuan Sandra & Djalali (2013) yang menunjukkan bahwa prokrastinasi yang berakar dari kurangnya manajemen waktu dan ketidakjelasan target berdampak negatif pada performa akademik dan proses pembelajaran secara keseluruhan.

Berdasarkan hasil penelitian yang dilakukan oleh Syifa (2020) yang meneliti keterkaitan perilaku *phubbing* akibat penggunaan *smartphone* dengan prokrastinasi akademik memberikan bukti empiris yang memprihatinkan. Dari total 103 mahasiswa yang diteliti, ditemukan bahwa 66 mahasiswa (64,1%) menunjukkan tingkat prokrastinasi akademik yang tinggi, 25 mahasiswa (24,3%) berada pada kategori sedang, dan 16 mahasiswa (15,53%) bahkan menunjukkan tingkat prokrastinasi yang sangat tinggi. Hanya 12 mahasiswa (11,7%) yang memiliki tingkat prokrastinasi akademik sangat rendah (Syifa, 2020). Data ini mengonfirmasi bahwa distraksi digital

memiliki pengaruh yang sangat signifikan terhadap prokrastinasi akademik mahasiswa, sekaligus mengindikasikan bahwa pendekatan manajemen tugas dan proyek secara konvensional sudah tidak relevan dengan kebutuhan mahasiswa masa kini. Manajemen tugas secara konvensional memiliki beberapa keterbatasan dibandingkan dengan metode modern. Penggunaan formulir dan dokumen fisik dalam pengelolaan tugas dapat menyulitkan akses informasi secara cepat dan akurat (Julianto Simatupang & M. Muhammad, 2018). Monitoring proyek secara konvensional juga kurang efisien dibandingkan dengan penggunaan *software* manajemen proyek (M. Hakim, 2019).

Berdasarkan berbagai permasalahan yang telah diidentifikasi, terdapat kebutuhan mendesak untuk mengembangkan solusi inovatif yang sejalan dengan preferensi digital mahasiswa modern. Penelitian ini bertujuan untuk merancang dan mengimplementasikan sistem manajemen proyek berbasis website yang sederhana dan intuitif guna membantu mahasiswa dalam mengelola tugas dan proyek secara efektif. Sistem yang dikembangkan ini dirancang dengan serangkaian fitur yang bertujuan untuk memfasilitasi mahasiswa dalam pengelolaan tugas dan proyek akademik secara lebih efektif. Fitur-fitur seperti pengaturan waktu, penyusunan prioritas, dan pemantauan progres tugas secara *real-time*, diharapkan dapat menjadi alat bantu bagi mahasiswa dalam menyikapi tantangan terkait manajemen waktu dan penyelesaian tugas dan proyek, yang seringkali menjadi faktor pemicu prokrastinasi akademik. Dengan mengintegrasikan teknologi digital ke dalam manajemen proyek akademik, sistem ini diharapkan dapat menjadi alternatif solusi yang dapat membantu mahasiswa meningkatkan kinerja dan produktivitas akademik mereka.

Beberapa penelitian sebelumnya telah mengkaji penerapan sistem manajemen tugas dan proyek dalam berbagai konteks. Julianto Simatupang dan M. Muhammad (2018) mengembangkan aplikasi berbasis mobile untuk mendukung pengelolaan proyek tugas akhir mahasiswa, mencakup pengajuan judul dan akses informasi pelaksanaan tugas akhir. Sementara itu, Tuti Alawiyah et al. (2022) membangun sistem informasi manajemen proyek (SIMAPRO) berbasis web dengan menggunakan bahasa pemrograman PHP dan *framework* CodeIgniter untuk mengoptimalkan pengelolaan proyek pada perusahaan kontraktor yang sebelumnya masih menggunakan pencatatan manual. Selain itu, Dhuha, A. R. (2017) merancang sistem manajemen proyek berbasis web untuk PT. Swadaya Graha guna mendukung perencanaan, pengawasan, dan pelaporan proyek secara *real-time* dengan menerapkan model pengembangan perangkat lunak SDLC *waterfall*, pendekatan *Object Oriented Programming* (OOP), serta konsep *Model-View-Controller* (MVC).

Berbeda dengan penelitian-penelitian sebelumnya yang lebih berfokus pada konteks organisasi atau tugas akhir, penelitian ini, yang berjudul "Perancangan dan Implementasi Sistem Manajemen Proyek Mahasiswa Berbasis Website", mengembangkan sistem manajemen proyek yang secara khusus dirancang untuk memenuhi kebutuhan pengelolaan tugas dan proyek akademik sehari-hari mahasiswa. Sistem ini dikembangkan menggunakan teknologi modern yakni UI *framework* Svelte dan *framework* javascript SvelteKit. Pemilihan teknologi ini

didasarkan pada kebutuhan untuk menciptakan aplikasi web yang ringan dan responsif. Implementasi sistem mengikuti metodologi pengembangan SDLC model *waterfall* untuk memastikan proses pengembangan yang sistematis dan terstruktur. Pemilihan teknologi dan metodologi ini didasarkan pada pertimbangan untuk menciptakan sistem yang tidak hanya fungsional, tetapi juga memiliki antarmuka yang intuitif dan responsif sesuai dengan kebutuhan pengguna mahasiswa. Diharapkan, dengan adanya sistem ini, mahasiswa akan mendapatkan solusi digital terintegrasi yang tidak hanya mengurangi distraksi dan kecenderungan prokrastinasi, tetapi juga mendukung peningkatan kualitas dan produktivitas akademik mereka.

1.2 Rumusan Masalah

Berdasarkan permasalahan dan tantangan yang telah diuraikan, maka rumusan masalah dalam penelitian ini adalah:

1. Bagaimana merancang dan mengimplementasikan sistem manajemen proyek berbasis website yang sederhana dan intuitif untuk mendukung pengelolaan tugas dan proyek akademik mahasiswa secara efektif?
2. Fitur-fitur apa saja yang perlu diimplementasikan dalam sistem manajemen proyek mahasiswa untuk mendukung pengelolaan tugas dan proyek akademik yang lebih terstruktur dan efisien?
3. Bagaimana penerimaan dan kepuasan mahasiswa terhadap sistem manajemen proyek berbasis website yang dikembangkan?

1.3 Batasan Masalah

Untuk memastikan penelitian ini tetap fokus dan dapat dilaksanakan dengan efektif, berikut adalah batasan masalah yang diterapkan dalam penelitian "Perancangan dan Implementasi Sistem Manajemen Proyek Mahasiswa Berbasis Website":

1. Pengguna sistem dibatasi hanya untuk mahasiswa sebagai pengguna individual untuk mendukung pengelolaan tugas dan proyek akademik mahasiswa, tidak mencakup fitur kolaborasi antar mahasiswa atau interaksi dengan dosen.
2. Sistem hanya menyediakan fitur-fitur utama pengelolaan proyek seperti pembuatan tugas, pengaturan *deadline*, pemantauan progres, pengategorian tugas, dan sistem pengingat (*reminder*), tanpa menyediakan fitur analisis data atau pelaporan komprehensif.
3. Implementasi sistem menggunakan *UI framework* Svelte dan *framework* javascript SvelteKit dengan *database* PostgreSQL, tanpa mengintegrasikan sistem dengan platform pembelajaran atau sistem akademik universitas yang sudah ada.

1.4 Tujuan Penelitian

Adapun tujuan penelitian ini adalah sebagai berikut:

1. Merancang dan mengimplementasikan sistem manajemen proyek berbasis website yang sederhana, intuitif, dan user-friendly untuk membantu mahasiswa dalam mengelola tugas dan proyek akademik.
2. Menilai tingkat kemudahan akses, kehandalan, dan kepuasan pengguna terhadap sistem yang dikembangkan, melalui pengujian dan evaluasi pengguna secara langsung.

1.5 Landasan Teori

1.5.1 Mahasiswa

Menurut Kamus Besar Bahasa Indonesia (KBBI), mahasiswa adalah orang yang menempuh pendidikan di perguruan tinggi. Secara etimologis, istilah "mahasiswa" berasal dari dua kata dalam bahasa Sanskerta, yaitu *mahat* yang berarti "tinggi" atau "agung" dan *sisya* yang berarti "murid" atau "orang yang diajari." Hartaji (2012) mendefinisikan mahasiswa sebagai individu yang tengah menimba ilmu dan terdaftar di salah satu jenis perguruan tinggi, baik itu akademi, politeknik, sekolah tinggi, institut, maupun universitas.

Berdasarkan definisi tersebut, dapat disimpulkan bahwa mahasiswa adalah individu yang tidak hanya menuntut ilmu secara formal, tetapi juga aktif dalam mengembangkan pengetahuan dan keterampilan melalui berbagai kegiatan akademik. Mereka berperan sebagai agen perubahan yang berpotensi membawa inovasi dalam berbagai aspek kehidupan, termasuk dalam memanfaatkan teknologi digital untuk mendukung proses pembelajaran.

1.5.2 Sistem Informasi

Menurut Kristanto, A. (2018), sistem informasi dapat didefinisikan sebagai integrasi antara perangkat keras, perangkat lunak, dan unsur manusia yang bersama-sama mengolah data. Tanpa adanya data untuk diproses, keberadaan sistem informasi menjadi tidak berarti. Data yang dimasukkan ke dalam sistem informasi mencakup berbagai bentuk, seperti formulir, prosedur, dan jenis informasi lainnya.

Dalam konteks akademik, sistem informasi berperan vital dalam memfasilitasi proses pembelajaran, pengelolaan administratif, dan kolaborasi antar stakeholder pendidikan. Implementasi sistem informasi yang efektif dapat meningkatkan efisiensi operasional, transparansi informasi, dan kualitas layanan pendidikan, mendukung proses pengambilan keputusan secara tepat dan strategis, sekaligus menciptakan lingkungan yang kondusif bagi mahasiswa untuk mengoptimalkan potensi akademik mereka melalui pengelolaan tugas dan proyek yang sistematis dan terstruktur.

1.5.3 Sistem Informasi Manajemen

Sistem Informasi Manajemen (SIM) merupakan sistem terintegrasi yang dirancang untuk menyediakan informasi guna mendukung kegiatan operasional, pengelolaan, dan pengambilan keputusan dalam suatu organisasi. SIM bekerja dengan mengolah masukan (*input*) melalui serangkaian proses untuk menghasilkan keluaran (*output*)

yang mendukung pencapaian tujuan manajerial tertentu. Secara akademis, istilah ini mencakup berbagai metode dan alat yang mendukung otomasi serta pengambilan keputusan strategis, seperti sistem pendukung keputusan, sistem pakar, dan sistem informasi eksekutif (Rohman & Brilian, 2023).

Selain itu, penerapan SIM tidak hanya meningkatkan efisiensi operasional, tetapi juga memungkinkan organisasi untuk melakukan analisis data secara akurat dan real-time. Dengan mengintegrasikan data dari berbagai departemen, SIM membantu dalam menyusun strategi yang lebih tepat dan responsif terhadap perubahan lingkungan. Dalam konteks pendidikan, sistem informasi manajemen dapat diterapkan untuk memantau kinerja akademik, mengelola administrasi, serta mendukung proses evaluasi dan perencanaan yang menyeluruh. Dengan demikian, SIM berperan penting dalam meningkatkan produktivitas dan efektivitas pengambilan keputusan di lingkungan organisasi maupun institusi pendidikan.

1.5.4 Manajemen Proyek

Manajemen proyek merupakan suatu proses sistematis yang mencakup perencanaan, pengorganisasian, dan penjadwalan tugas serta pengelolaan sumber daya untuk mencapai tujuan proyek secara efisien. Proses ini melibatkan penyusunan rencana kerja yang matang dan pengaturan aktivitas dengan memperhatikan faktor waktu dan biaya (Teguh, R., 2019). Konsep manajemen proyek dapat diterapkan pada berbagai jenis proyek, baik yang berskala kecil maupun proyek besar dan kompleks. Fokus utamanya adalah memastikan pencapaian seluruh sasaran proyek dalam batasan waktu dan anggaran yang telah ditetapkan (Darmawan, D., & Ratnasari, A., 2020).

Menurut *Project Management Body of Knowledge* (PMBOK), terdapat enam kendala utama yang harus dikelola dalam setiap proyek, yaitu: waktu (*time*), biaya (*cost*), ruang lingkup (*scope*), kualitas (*quality*), sumber daya (*resources*), dan risiko (risk) (A Guide to the Project Management Body of Knowledge, 2013). Dalam setiap proyek, beban atau nilai dari masing-masing kendala ini bisa berbeda-beda, sehingga peran manajer proyek sangat krusial dalam menyeimbangkan elemen-elemen tersebut agar tujuan akhir proyek dapat tercapai secara optimal (Mahmoudi & Feylizadeh, 2017).

Selain itu, manajemen proyek juga memanfaatkan berbagai teknik dan alat bantu, seperti diagram Gantt, metode jalur kritis (*Critical Path Method*), dan perangkat lunak manajemen proyek yang memungkinkan pemantauan progres secara real-time. Pendekatan ini tidak hanya memfasilitasi koordinasi antar tim, tetapi juga meningkatkan transparansi dan akuntabilitas dalam pelaksanaan proyek. Dengan penerapan manajemen proyek yang efektif, organisasi dapat mengoptimalkan penggunaan sumber daya, meminimalkan risiko kegagalan, dan memastikan setiap tahap proyek berjalan sesuai rencana.

1.5.5 Website

Website merupakan perangkat lunak yang dirancang untuk menampilkan dokumen digital secara online, sehingga memungkinkan pengguna mengakses informasi melalui jaringan internet (Mara Destiningrum, 2017). Dalam pengertian yang lebih luas, website adalah sekumpulan halaman yang saling terhubung dan menyajikan konten atau data tertentu kepada pengguna. Website pertama kali diperkenalkan pada tahun 1992 oleh CERN (*Conseil Européen pour la Recherche Nucléaire*) di Swiss, menandai awal dari era *World Wide Web*. Penting untuk dicatat bahwa internet dan web memiliki perbedaan mendasar. Internet adalah infrastruktur global yang menyediakan jaringan komunikasi dan konektivitas, sedangkan web merupakan layanan atau aplikasi yang berjalan di atas infrastruktur tersebut, menggunakan teknologi web untuk menyajikan halaman-halaman informasi yang saling terkait. Dengan kata lain, *World Wide Web* (WWW) adalah subsistem dari internet yang memungkinkan navigasi dan interaksi secara visual melalui browser (Rifani et al., 2020).

Berdasarkan penjelasan yang telah diuraikan, dapat disimpulkan bahwa website merupakan kumpulan halaman digital yang terorganisir dan saling terkait, diakses melalui jaringan internet, dan menyajikan berbagai informasi dalam beragam format untuk memenuhi kebutuhan penggunanya. Website telah menjadi komponen integral dalam ekosistem digital modern, berperan sebagai jembatan komunikasi dan pertukaran informasi dalam skala global.

1.5.6 JavaScript

JavaScript adalah bahasa skrip yang sangat populer dalam pengembangan web karena kemampuannya untuk menciptakan halaman yang interaktif dan responsif terhadap berbagai peristiwa (event) yang terjadi di dalamnya. Menurut Sianipar, R.H. (2015), JavaScript berperan sebagai "perekat" yang menyatukan berbagai elemen halaman web, sehingga memungkinkan pengalaman pengguna yang lebih dinamis.

Selain itu, JavaScript termasuk dalam kategori bahasa pemrograman tingkat tinggi (*high-level programming language*) yang memiliki sintaks dan struktur yang mudah dipahami, mirip dengan bahasa sehari-hari. Bahasa ini berjalan di sisi klien (*client-side*), artinya hanya membutuhkan browser untuk mengeksekusi kode, tanpa perlu pengolahan di server. JavaScript juga mendukung paradigma pemrograman berorientasi objek (*object-oriented programming*), yang memudahkan pengorganisasian dan pemeliharaan kode. Selain itu, sebagai bahasa yang *loosely typed*, JavaScript tidak mengharuskan deklarasi tipe variabel secara eksplisit, sehingga memberikan fleksibilitas dalam penulisan kode (Enterprise, J., 2017).

Seiring dengan perkembangan teknologi, JavaScript telah menjadi fondasi utama dalam pengembangan aplikasi web modern. *Framework* dan pustaka seperti React, Angular, dan Vue.js memanfaatkan JavaScript untuk menciptakan antarmuka pengguna yang responsif dan dinamis. Selain itu, dengan munculnya platform seperti Node.js, JavaScript kini juga digunakan untuk pengembangan aplikasi sisi server, memungkinkan pembuatan aplikasi *full-stack* yang terintegrasi (Shukla, 2023).

1.5.7 TypeScript

TypeScript adalah bahasa pemrograman yang merupakan perluasan (superset) dari JavaScript, dirancang khusus untuk pengembangan aplikasi web berskala besar. Keunggulan utama TypeScript terletak pada sistem pengetikan statisnya yang memaksa pengembang untuk mendefinisikan tipe data secara eksplisit. Fitur ini membantu mendeteksi kesalahan sejak tahap pengembangan, sehingga meningkatkan kualitas dan stabilitas kode (Jansen, R.H., 2015). Selain itu, TypeScript mudah dipelajari berkat sintaksisnya yang mirip dengan JavaScript dan didukung oleh ekosistem yang kaya akan berbagai *library* dan *framework*.

TypeScript juga mendukung prinsip-prinsip pemrograman berorientasi objek, seperti kelas, pewarisan, enkapsulasi, dan polimorfisme, yang memungkinkan penulisan kode yang lebih terstruktur dan mudah dikelola. Kelebihan lainnya, setiap kode JavaScript yang valid juga merupakan kode TypeScript, sehingga memudahkan transisi dan integrasi dalam proyek yang sudah ada. Dengan demikian, TypeScript tidak hanya meningkatkan keterbacaan dan maintainability dari kode, tetapi juga mendukung proses refactoring dan kolaborasi tim dalam pengembangan aplikasi modern.

1.5.8 *Framework*

Framework merupakan kerangka kerja yang dirancang untuk mempermudah dan mempercepat proses pengembangan situs web. Di dalamnya terdapat berbagai komponen, pustaka kode, dan variabel yang disusun secara sistematis untuk membantu perancang web dalam menulis, merencanakan, menguji, serta memelihara kode sumber situs web. Dengan menggunakan *framework*, pengembangan situs web menjadi lebih terstruktur, konsisten, dan efisien, karena *framework* menyediakan fondasi yang telah distandarisasi sehingga memudahkan integrasi dengan modul-modul atau teknologi lain yang diperlukan (Renaldo Prasena & Sama, 2020).

Selain itu, *framework* juga mendukung pengembangan aplikasi web yang scalable dan mudah dipelihara, sehingga para pengembang dapat lebih fokus pada pembuatan fitur-fitur unik dan peningkatan kualitas aplikasi. Dengan adanya kerangka kerja yang tersusun rapi, proses debugging dan pengujian pun menjadi lebih mudah dilakukan, yang secara keseluruhan meningkatkan produktivitas serta efektivitas pengembangan situs web (Haniefardy et al., 2019).

Berdasarkan penjelasan tersebut, *framework* memegang peranan penting dalam mendukung pembuatan situs web yang modern, responsif, dan mudah dikelola, sehingga menjadi pilihan utama bagi pengembang dalam menciptakan solusi digital yang inovatif.

1.5.9 Svelte

Svelte adalah sebuah alat untuk membangun aplikasi web yang memungkinkan pengembang menciptakan aplikasi secara deklaratif dengan memanfaatkan komponen-komponen yang mengintegrasikan markup, gaya, dan perilaku. Komponen-komponen ini kemudian dikompilasi menjadi modul-modul JavaScript

kecil yang efisien, sehingga mengurangi overhead yang biasanya muncul pada *framework* antarmuka (UI) pengguna tradisional. Dengan Svelte, Anda dapat mengembangkan berbagai jenis aplikasi web, mulai dari komponen yang berdiri sendiri hingga aplikasi full-stack yang kompleks dengan bantuan *framework* aplikasi pendamping seperti SvelteKit (Svelte, 2025).

Berbeda dengan kerangka kerja *frontend* konvensional seperti React atau Vue yang melakukan sebagian besar pemrosesan di dalam browser saat runtime, Svelte memindahkan tugas-tugas tersebut ke tahap kompilasi. Dengan demikian, saat Anda membangun aplikasi menggunakan Svelte, komponen-komponennya dikompilasi menjadi kode JavaScript yang sangat efisien dan langsung memperbarui Document Object Model (DOM). Keunggulan utama dari Svelte adalah aplikasi yang dihasilkannya cenderung lebih cepat, responsif, dan memiliki ukuran bundel yang lebih kecil dibandingkan dengan aplikasi yang dibangun menggunakan kerangka kerja lain (Kurniawan, A., & Santoso, N., 2023).

Selain itu, Svelte menawarkan kemudahan pengembangan berkat sintaks yang sederhana dan intuitif, yang memudahkan proses penulisan kode serta pemeliharaan aplikasi. Pendekatan kompilasi yang digunakan tidak hanya mengurangi beban pada browser, tetapi juga meningkatkan performa aplikasi, terutama pada perangkat dengan sumber daya terbatas. Dengan segala keunggulan tersebut, Svelte menjadi pilihan menarik bagi pengembang yang menginginkan aplikasi web modern dengan performa optimal dan pengalaman pengguna yang superior.

1.5.10 SvelteKit

SvelteKit adalah *framework* yang dirancang untuk mempercepat pengembangan aplikasi web yang kuat dan berkinerja tinggi dengan menggunakan Svelte. Bagi pengguna React, SvelteKit dapat dianggap serupa dengan Next.js, sedangkan bagi pengguna Vue, *framework* ini mirip dengan Nuxt.js. Secara sederhana, Svelte digunakan untuk membuat komponen antarmuka pengguna—seperti menu navigasi, bagian komentar, atau formulir kontak—yang dapat dilihat dan diinteraksikan oleh pengguna melalui browser. Kompiler Svelte kemudian mengubah komponen-komponen ini menjadi JavaScript yang menghasilkan HTML dan CSS untuk halaman web (Svelte, 2025).

Meskipun Svelte mampu merender komponen UI secara efisien, untuk membangun sebuah aplikasi web yang utuh dibutuhkan lebih dari sekadar Svelte. Di sinilah SvelteKit berperan, dengan membantu pengembang membuat aplikasi web yang mengikuti praktik terbaik modern dan menyediakan solusi untuk berbagai tantangan umum dalam pengembangan. SvelteKit menawarkan fitur dasar seperti *router* yang secara otomatis memperbarui tampilan saat tautan diklik, serta kemampuan lanjutan seperti optimasi proses *build* untuk memuat hanya kode yang diperlukan, dukungan *offline*, pramuat halaman sebelum navigasi, dan konfigurasi *rendering* yang fleksibel—baik itu *rendering* di sisi server (SSR), di sisi klien, atau saat *build* dengan *prerendering*. Fitur-fitur tambahan seperti optimasi gambar juga turut disediakan (Svelte, 2025).

Dengan SvelteKit, proses membangun aplikasi web yang kompleks menjadi lebih sederhana, karena *framework* ini menangani tugas-tugas teknis yang rutin sehingga pengembang dapat lebih fokus pada aspek kreatif dan inovatif dari aplikasi yang sedang dikembangkan.

1.5.11 Tailwind CSS

Tailwind CSS adalah sebuah *framework* CSS yang berfokus pada pendekatan *utility-first* untuk membangun antarmuka pengguna aplikasi web. Artinya, *framework* ini menyediakan sekumpulan kelas siap pakai yang dapat langsung diterapkan ke markup untuk menciptakan berbagai desain tanpa perlu menulis CSS khusus dari awal. Tailwind CSS bekerja dengan memindai semua file HTML, komponen JavaScript, dan template lain untuk menemukan nama kelas, kemudian menghasilkan gaya yang sesuai dan menuliskannya ke dalam file CSS statis. Dengan proses ini, *framework* ini dikenal cepat, fleksibel, dan handal tanpa adanya beban runtime (TailwindCSS, 2025).

Selain itu, Tailwind CSS memungkinkan pengembang untuk dengan mudah membuat desain yang konsisten dan responsif melalui konfigurasi yang dapat disesuaikan. Fitur-fiturnya mendukung pembuatan tata letak yang kompleks sekaligus menjaga performa aplikasi, sehingga mengurangi waktu pengembangan dan meningkatkan efisiensi kerja. Tailwind CSS juga mendukung pengembangan dengan prinsip desain modular, memungkinkan integrasi yang lancar dengan berbagai workflow pengembangan modern.

1.5.12 PostgreSQL

PostgreSQL, yang juga dikenal sebagai Postgres, merupakan salah satu sistem manajemen basis data relasional (RDBMS) yang bersifat open-source dan terkenal karena skalabilitas, fleksibilitas, dan kinerjanya yang tinggi. PostgreSQL dirancang untuk menangani berbagai beban kerja, mulai dari aplikasi kecil hingga sistem enterprise yang kompleks, serta mendukung transaksi yang aman dan konsisten melalui kepatuhan terhadap standar ACID. Selain itu, PostgreSQL menawarkan berbagai fitur canggih seperti dukungan untuk tipe data kustom, prosedur tersimpan, dan replikasi, yang membuatnya sangat cocok untuk pengembangan aplikasi modern (Waruwu, 2019).

Penggunaan PostgreSQL sangat luas dan dapat diintegrasikan dengan berbagai platform serta didukung oleh banyak bahasa pemrograman, seperti Python, Java, C/C++, dan lain-lain, sehingga memberikan fleksibilitas tinggi bagi pengembang (Sugiana, O. & Wiryana, I. M., 2001). Keunggulan ini menjadikan PostgreSQL sebagai pilihan utama bagi banyak organisasi yang membutuhkan sistem basis data yang handal, dapat diperluas, dan mampu beradaptasi dengan kebutuhan teknologi yang terus berkembang.

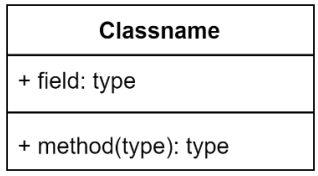
1.5.13 UML (*Unified Modeling Language*)

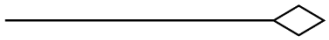
UML adalah bahasa standar yang digunakan untuk merancang cetak biru (*blueprint*) perangkat lunak. Bahasa ini memungkinkan para pengembang untuk memvisualisasikan, menentukan spesifikasi, membangun, dan mendokumentasikan berbagai komponen dari sistem yang kompleks. UML cocok digunakan untuk memodelkan sistem mulai dari sistem informasi skala perusahaan, aplikasi web terdistribusi, hingga sistem tertanam yang harus beroperasi secara real-time (Booch et al., 2005).

UML terinspirasi oleh konsep pemodelan berorientasi objek (*Object-Oriented*, OO), yang melihat sistem sebagai kumpulan objek yang merepresentasikan entitas di dunia nyata. Dengan menggunakan simbol-simbol yang khas, pemodelan OO menyediakan proses yang standar dan independen, sehingga memudahkan komunikasi dan integrasi antara berbagai komponen sistem (Haviluddin, H., 2016). Selain itu, UML juga berperan penting dalam membantu tim pengembang untuk merancang sistem secara kolaboratif dan mendokumentasikan desain perangkat lunak secara menyeluruh, yang pada akhirnya meningkatkan efisiensi pengembangan dan pemeliharaan sistem. UML memiliki beberapa jenis diagram namun bebapa yang umum digunakan diantaranya:

Class diagram. *Class* Diagram menampilkan kumpulan kelas, antarmuka, dan kolaborasi beserta hubungan-hubungan antar elemen tersebut. Diagram ini merupakan alat utama dalam pemodelan sistem berorientasi objek, karena menyajikan pandangan statis dari desain sistem. Diagram kelas yang mencakup kelas aktif juga dapat menggambarkan proses statis dalam sistem. Diagram kelas tidak hanya berfungsi untuk memvisualisasikan, menentukan spesifikasi, dan mendokumentasikan model struktural, tetapi juga berperan dalam pembuatan sistem yang dapat dieksekusi melalui proses *forward engineering* dan *reverse engineering* (Booch et al., 1999). Komponen-komponen utama yang membentuk sebuah *class diagram* dirangkum pada Tabel 1 berikut.

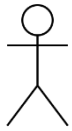
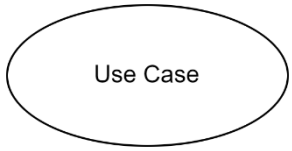
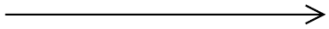
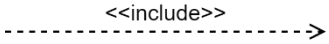
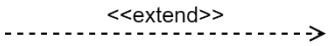
Tabel 1. Komponen *Class Diagram*

Komponen	Keterangan
	*Class*: terdiri dari dari 3 bagian, bagian atas merupakan bagian nama *class*, bagian merupakan atribut dari *class*, dan bagian merupakan *method* dari *class*
	*Association*: menunjukkan hubungan antar *class*
	*Composition*: hubungan "bagian-dari" yang kuat antara dua kelas. Jika objek parent dihapus, objek child juga akan dihapus

Komponen	Keterangan
	*Aggregation*: hubungan "bagian-dari" yang lemah antara dua kelas. Jika objek parent dihapus, objek child tetap ada
	*Generalization*: Menunjukkan hubungan antara *subclass* atau *child class* (kelas anak) mewarisi properti dan perilaku dari *superclass* atau *parent class* (kelas induk)
	*Dependency*: menunjukkan ketergantungan antar *class*
0…1	sebuah objek dapat memiliki 0 atau 1 instansi dari objek lain
1	sebuah objek harus memiliki tepat 1 instansi dari objek lain.
0...*	Menunjukkan bahwa sebuah objek dapat memiliki nol, satu, atau banyak instansi dari objek lain, sehingga hubungan tersebut tidak harus ada namun bisa ada secara tidak terbatas.
1…*	Menunjukkan bahwa sebuah objek harus memiliki setidaknya 1 instansi, dan bisa lebih banyak lagi dari objek lain.
1…n	Secara konseptual sama dengan 1…*, artinya sebuah objek harus memiliki minimal 1 instansi, dengan 'n' menunjukkan jumlah yang bisa berapa saja, lebih dari satu.

Use case diagram. *use case diagram* menampilkan serangkaian *use case* dan aktor (yang merupakan jenis kelas khusus) beserta hubungan-hubungan antar mereka. Diagram ini digunakan untuk menggambarkan tampilan statis dari *use case* dalam suatu sistem, sehingga memudahkan pemahaman tentang bagaimana sistem berinteraksi dengan pengguna atau entitas lain. *Use case diagram* sangat penting dalam mengorganisir dan memodelkan perilaku sistem, karena membantu tim pengembang untuk merencanakan dan mengkomunikasikan fungsi-fungsi utama yang harus dipenuhi oleh sistem (Booch et al., 1999). Tabel 2 berikut merinci komponen-komponen yang digunakan untuk membangun *use case diagram*.

Tabel 2. Komponen *Use Case Diagram*

Komponen	Keterangan
 Actor	Aktor: entitas (bisa berupa orang, kelompok, atau sistem lain) yang berinteraksi dengan sistem.
	*Use case*: abstraksi dan interaksi antara sistem dan aktor.
	*Association*: hubungan yang menghubungkan aktor dengan use case.
	*Generalization*: menggambarkan hubungan pewarisan antara aktor atau antara use case.
	*Include*: hubungan antar use case yang menunjukkan bahwa sebuah use case selalu memasukkan fungsionalitas dari use case lain sebagai bagian dari alur kerjanya.
	*Extend*: hubungan yang menggambarkan penambahan fungsionalitas opsional pada use case dasar.

Activity diagram. *activity diagram* menggambarkan alur kerja atau proses dari satu aktivitas ke aktivitas berikutnya dalam suatu sistem. Diagram ini menampilkan serangkaian aktivitas, baik yang berjalan secara berurutan maupun yang bercabang, serta memperlihatkan objek-objek yang berperan aktif atau dipengaruhi selama proses berlangsung. *activity diagram* digunakan untuk memberikan gambaran dinamis mengenai cara kerja sistem, dan sangat penting dalam memodelkan fungsi serta aliran kontrol antar objek (Booch et al., 1999). Dengan demikian, diagram ini membantu dalam analisis proses bisnis dan pengambilan keputusan, karena menyoroti bagaimana setiap aktivitas saling terkait dan mengalir satu sama lain. Simbol dan komponen yang digunakan dalam *activity diagram* dijelaskan pada Tabel 3.

Tabel 3. Komponen *Activity Diagram*

Komponen	Keterangan
	*Initial Node*: Titik awal dari alur aktivitas yang menandakan dimulainya proses.
	*End Node*: Titik akhir dari alur aktivitas yang menandakan selesainya proses.
	*Activity State*: Representasi dari sebuah aktivitas atau proses yang terjadi di dalam sistem.
	*Join*: Titik penggabungan alur aktivitas yang sebelumnya bercabang paralel menjadi satu alur tunggal.
	*Decision Node*: Titik cabang dalam alur aktivitas yang memungkinkan pengambilan keputusan berdasarkan kondisi tertentu.
	Kondisi Transisi *(Guard Condition)*: Kondisi atau syarat yang harus dipenuhi agar aliran dari satu aktivitas ke aktivitas berikutnya dapat berlangsung.

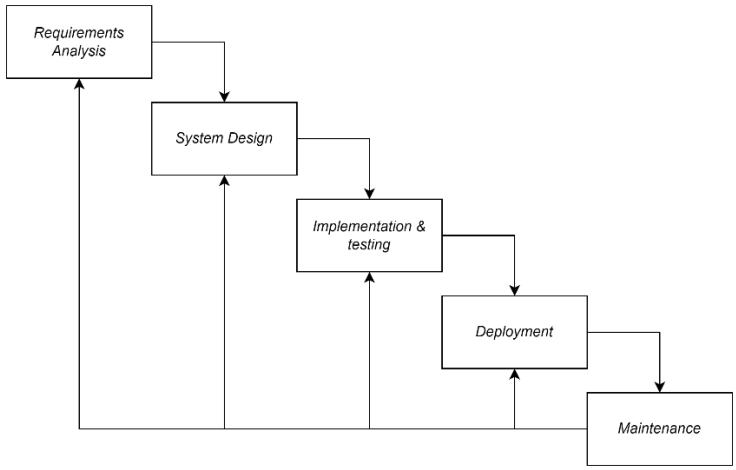
1.5.14 *System Development Life Cycle* (SDLC)

Systems Development Life Cycle (SDLC), atau Siklus Hidup Pengembangan Sistem, merupakan suatu proses terstruktur yang digunakan dalam rekayasa sistem dan pengembangan perangkat lunak. Proses ini mencakup tahap-tahap pembuatan, pemeliharaan, dan modifikasi sistem, serta penerapan berbagai model dan metodologi untuk mengembangkan sistem secara efektif (Susanto, R. & Andriana, A., 2016).

Menurut Jirava, P. (2004), SDLC adalah serangkaian langkah yang diikuti untuk membangun sebuah sistem informasi; pada intinya, ini adalah proses yang ditempuh oleh perusahaan dalam mengembangkan sistem informasi baru. Metodologi pengembangan sistem menyediakan kerangka kerja yang diperlukan untuk mencapai kesuksesan dalam pembuatan sistem informasi.

Dengan menerapkan SDLC, pengembang dapat memastikan setiap tahap mulai dari analisis kebutuhan, perancangan, implementasi, pengujian, hingga pemeliharaan dijalankan dengan terstruktur dan efisien. Pendekatan ini membantu dalam mengidentifikasi kebutuhan pengguna, mengurangi risiko kegagalan, dan mengoptimalkan penggunaan sumber daya, sehingga hasil akhir sistem lebih dapat diandalkan dan sesuai dengan harapan. Berdasarkan uraian di atas, dapat disimpulkan bahwa SDLC merupakan fondasi penting dalam pengembangan sistem

modern yang mendukung keberhasilan proyek teknologi informasi. Salah satu model SDLC yang paling umum digunakan adalah model *waterfall*, yang alur kerjanya diilustrasikan pada Gambar 1.



Gambar 1. Metode *waterfall*

Model *waterfall* merupakan salah satu metode pengembangan perangkat lunak yang paling populer dan banyak digunakan. Metode ini mengikuti alur kerja yang linier, dimulai dari tahap perencanaan awal hingga tahap pemeliharaan akhir. Setiap fase harus diselesaikan secara tuntas sebelum melanjutkan ke fase berikutnya, tanpa adanya iterasi kembali ke fase sebelumnya (Susanto, R. & Andriana, A., 2016). Tahapan utama dalam metode *waterfall* adalah sebagai berikut:

1. Analisis Kebutuhan (*Requirements Analysis*)
   Mengidentifikasi dan mendokumentasikan kebutuhan sistem dari pengguna dan pemangku kepentingan, sehingga pengembangan didasarkan pada pemahaman yang jelas mengenai apa yang harus dibangun.
2. Perancangan Sistem (*System Design*)
   Menyusun blueprint sistem dengan merancang arsitektur, basis data, antarmuka pengguna, serta struktur program. Desain ini akan menjadi pedoman bagi pengembang selama proses implementasi.
3. Implementasi dan Pengujian (*Implementation and Testing*)
   Pada tahap ini, pengembang mulai menulis kode berdasarkan desain yang telah dibuat. Selama proses ini, sistem juga diuji secara menyeluruh untuk mendeteksi dan memperbaiki kesalahan (bug) serta memastikan bahwa semua kebutuhan telah terpenuhi.
4. Penerapan (*Deployment*)
   Sistem yang telah dikembangkan dan diuji kemudian disebarkan ke lingkungan produksi agar dapat digunakan oleh pengguna.

5. Pemeliharaan (*Maintenance*)
   Setelah sistem diterapkan, dilakukan perbaikan, pembaruan, dan penyesuaian berdasarkan umpan balik pengguna dan perubahan kebutuhan seiring waktu.

1.5.15 *Blackbox Testing*

Pengujian *blackbox* berfokus pada evaluasi fungsionalitas eksternal suatu sistem tanpa memeriksa atau mengetahui struktur kode program di dalamnya. Pendekatan ini menggunakan metode pengujian berbasis spesifikasi dan parameter untuk mendeteksi kesalahan, dengan mengacu pada kebutuhan sistem serta spesifikasi produk yang telah ditetapkan (Kurniawan, A. & Santoso, N., 2023).

Pendekatan pengujian *blackbox* bertujuan untuk mengidentifikasi berbagai kategori kesalahan, seperti fungsi yang salah atau tidak ada, kesalahan pada antarmuka pengguna, masalah pada struktur data atau akses basis data eksternal, kesalahan perilaku atau performa, serta kesalahan pada proses inisialisasi dan terminasi sistem (Pressman, 2015). Dengan demikian, pengujian *blackbox* memastikan bahwa sistem memberikan output yang sesuai dengan input yang diberikan oleh pengguna, tanpa memerlukan pemahaman terhadap logika atau implementasi internal kode, sehingga dapat memastikan bahwa sistem beroperasi sesuai dengan harapan dalam lingkungan operasionalnya.

Salah satu tool yang dapat digunakan untuk implementasi *blackbox testing* pada aplikasi web adalah Playwright Test, yaitu *framework end-to-end testing* yang memungkinkan simulasi interaksi pengguna nyata dengan *browser* untuk memvalidasi fungsionalitas sistem.

1.5.16 Playwright Test

Playwright Test adalah sebuah kerangka kerja *open-source* (sumber terbuka) yang ditujukan untuk melakukan pengujian ujung ke ujung (*end-to-end*) pada aplikasi web. Kerangka kerja ini dikembangkan oleh Microsoft dan resmi diluncurkan pada bulan Januari tahun 2020 (Playwright, 2024). Tujuan utama dari Playwright Test adalah untuk memfasilitasi proses otomatisasi pengujian yang tidak hanya handal dan efisien, tetapi juga dapat diandalkan ketika dijalankan pada berbagai jenis peramban web modern. Beberapa peramban yang didukung meliputi Chromium, yang menjadi dasar bagi Chrome dan Edge, serta Firefox dan WebKit yang digunakan oleh Safari. Salah satu keunggulan Playwright Test adalah tersedianya Antarmuka Pemrograman Aplikasi (API) tunggal yang dapat digunakan untuk berinteraksi dengan semua peramban yang didukung. Kemampuan ini memungkinkan pelaksanaan pengujian lintas-peramban (*cross-browser*) yang konsisten dan memiliki kecepatan tinggi, menjadikannya sebagai salah satu pilihan yang populer di kalangan pengembang maupun penguji perangkat lunak (Playwright, 2024).

Pengembangan Playwright Test dilatar belakangi oleh upaya untuk mengatasi berbagai keterbatasan yang ada pada kerangka kerja pengujian generasi sebelumnya, seperti Selenium dan Puppeteer. Sebagai contoh, berbeda dengan Selenium yang masih memerlukan penggunaan *driver* (penggerak) spesifik untuk

setiap jenis peramban, Playwright mengadopsi protokol WebSocket. Protokol ini memungkinkan komunikasi secara langsung dengan peramban, yang pada gilirannya berkontribusi pada peningkatan kecepatan dan efisiensi proses pengujian (Priyanka J., 2024). Lebih lanjut, Playwright juga menawarkan dukungan untuk berbagai bahasa pemrograman populer, di antaranya adalah JavaScript, TypeScript, Python, Java, dan C#. Fleksibilitas dalam pilihan bahasa pemrograman ini memberikan keleluasaan bagi tim pengembang untuk memilih bahasa yang paling sesuai dengan keahlian mereka (Playwright, 2024).

1.5.17 *System Usability Scale* (SUS)

System Usability Scale (SUS) adalah alat ukur standar yang digunakan untuk menilai kegunaan (*usability*) dari suatu sistem, aplikasi, atau produk digital. SUS terdiri dari 10 pernyataan yang direspons menggunakan skala Likert 5 poin, dari "sangat tidak setuju" hingga "sangat setuju". Alat ini memberikan skor keseluruhan yang mencerminkan seberapa mudah dan menyenangkan suatu sistem untuk digunakan.

SUS dirancang agar mudah diterapkan dan cepat untuk diisi, sehingga memungkinkan pengumpulan data dari pengguna secara efisien. Skor yang dihasilkan tidak hanya memberikan indikasi tingkat kepuasan pengguna, tetapi juga membantu mengidentifikasi area yang perlu ditingkatkan dari segi antarmuka dan interaksi pengguna. Dalam konteks pengembangan sistem, SUS sering digunakan sebagai metode evaluasi *usability* dalam proses iteratif, sehingga desain sistem dapat disesuaikan dengan kebutuhan dan preferensi pengguna. Dengan kemampuannya untuk mengukur *usability* secara konsisten, SUS menjadi salah satu alat andalan dalam penelitian dan pengembangan produk digital, serta memberikan dasar empiris untuk perbaikan antarmuka dan peningkatan pengalaman pengguna (Lewis, 2018).

1.5.18 *Technology Acceptance Model* (TAM)

Technology Acceptance Model (TAM) adalah model teoritis yang dikembangkan oleh Fred D. Davis pada tahun 1989 untuk menjelaskan dan memprediksi penerimaan pengguna terhadap teknologi informasi. Model ini mengidentifikasi dua konstruk utama yang menentukan penerimaan teknologi: *Perceived Usefulness* (PU) yaitu tingkat kepercayaan seseorang bahwa menggunakan teknologi tertentu akan meningkatkan kinerja pekerjaannya, dan *Perceived Ease of Use* (PEOU) yaitu tingkat kepercayaan bahwa menggunakan teknologi tersebut akan bebas dari usaha yang berlebihan. Kedua konstruk tersebut secara langsung mempengaruhi *Attitude Toward Using* (ATU), yang selanjutnya berdampak pada *Behavioral Intention* (BI) dan akhirnya menentukan penggunaan sistem aktual.

TAM telah menjadi salah satu kerangka kerja yang paling banyak digunakan dalam penelitian sistem informasi karena kesederhanaannya dan kemampuannya memberikan prediksi yang konsisten terhadap penerimaan teknologi di berbagai konteks sehingga memungkinkan pengembang untuk merancang sistem yang lebih sesuai dengan kebutuhan dan ekspektasi pengguna (Davis, 1989; Venkatesh & Davis, 2000).

BAB II
METODE PENELITIAN

2.1 *Design Science Research* (DSR)

Design Science Research (DSR) merupakan paradigma penelitian yang berfokus pada pengembangan dan evaluasi artefak teknologi inovatif yang dirancang untuk menyelesaikan permasalahan praktis. Hevner et al. (2004) mendefinisikan DSR sebagai metodologi penelitian yang menekankan pada penciptaan artefak yang bertujuan khusus, memiliki kontribusi keilmuan yang jelas, dan dapat dievaluasi secara sistematis. Dalam konteks sistem informasi, DSR menyediakan kerangka kerja yang memungkinkan peneliti untuk mengembangkan solusi teknologi yang tidak hanya inovatif namun juga relevan dalam mengatasi permasalahan dunia nyata.

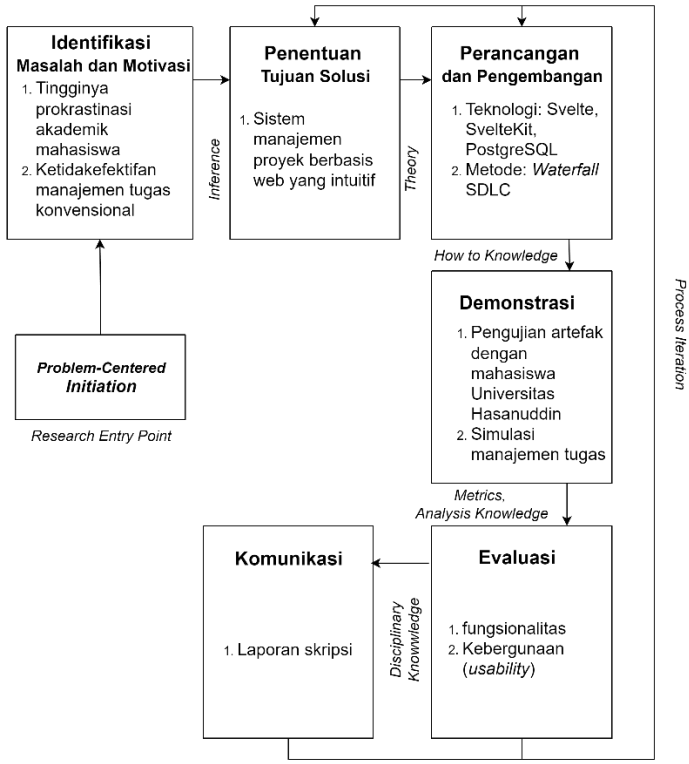
Peffers et al. (2007) mengajukan model proses DSR yang terstruktur dan diterima secara luas dalam komunitas penelitian sistem informasi. Model ini terdiri dari enam aktivitas penelitian yang dapat dilaksanakan secara sekuensial, meskipun urutan pelaksanaannya dapat disesuaikan berdasarkan pendekatan spesifik yang digunakan:

1. Identifikasi Masalah dan Motivasi
   Tahap ini melibatkan perumusan masalah penelitian secara spesifik dan menetapkan nilai dari solusi yang diusulkan. Peneliti mengidentifikasi permasalahan yang dihadapi mahasiswa dalam manajemen proyek akademik, termasuk kesulitan dalam mengelola tenggat waktu, mengorganisasi tugas, dan memantau progres proyek. Pemahaman mendalam terhadap masalah ini menjadi dasar yang kuat untuk pengembangan sistem, sekaligus memberikan motivasi bagi seluruh rangkaian penelitian.
2. Penentuan Tujuan Solusi
   Pada tahap ini, peneliti menetapkan tujuan sistem yang akan dikembangkan berdasarkan definisi masalah dan pengetahuan tentang apa yang mungkin dan layak dilakukan. Tujuan dapat bersifat kuantitatif, seperti peningkatan efisiensi pengelolaan proyek, atau kualitatif, seperti peningkatan pengalaman pengguna. Tujuan harus terukur dan dapat dievaluasi untuk menilai keberhasilan artefak yang dikembangkan.
3. Perancangan dan Pengembangan
   Aktivitas ini meliputi penciptaan artefak, yaitu sistem manajemen proyek berbasis website untuk mahasiswa. Tahap ini mencakup penentuan fungsionalitas sistem, arsitektur, dan perancangan antarmuka pengguna. Peneliti menggunakan prinsip perancangan sistem yang berorientasi pada pengguna untuk memastikan bahwa sistem yang dikembangkan memenuhi kebutuhan mahasiswa dalam mengelola proyek akademik. Implementasi dilakukan dengan memanfaatkan teknologi TypeScript, SvelteKit, dan PostgreSQL.

4. Demonstrasi
   Pada tahap ini, peneliti mendemonstrasikan penggunaan artefak dalam menyelesaikan satu atau lebih kasus yang merepresentasikan masalah yang telah diidentifikasi. Demonstrasi ini dapat dilakukan melalui eksperimen, simulasi, studi kasus, bukti formal, atau aktivitas lain yang sesuai. Mahasiswa sebagai pengguna target dilibatkan dalam demonstrasi untuk memvalidasi fungsionalitas sistem.
5. Evaluasi
   Evaluasi mencakup pengamatan dan pengukuran seberapa baik artefak mendukung solusi terhadap masalah. Aktivitas ini melibatkan perbandingan antara tujuan solusi yang telah ditetapkan dengan hasil observasi penggunaan artefak dalam konteks demonstrasi. Dalam penelitian ini, evaluasi dilakukan menggunakan metode *blackbox testin*g untuk memverifikasi fungsionalitas sistem, *System Usability Scale* (SUS) dan wawancara untuk menilai kegunaan sistem dari perspektif pengguna.
6. Komunikasi
   Tahap final melibatkan komunikasi masalah dan relevansinya, artefak yang dikembangkan, kebaruan, desain, dan efektivitasnya kepada peneliti dan praktisi yang relevan. Komunikasi ini umumnya dilakukan melalui publikasi penelitian, presentasi, atau dokumentasi teknis. Dalam konteks penelitian ini, skripsi menjadi sarana utama untuk mengkomunikasikan hasil penelitian secara komprehensif.

Model DSR yang diajukan oleh Peffers et al. (2007) menawarkan fleksibilitas dalam penerapannya. Peneliti dapat memulai dari berbagai titik masuk, bergantung pada kondisi spesifik penelitian. Misalnya, penelitian dapat dimulai dari masalah teridentifikasi (*problem-centered initiation*), tujuan yang telah ditetapkan (*objective-centered solution*), desain dan pengembangan (*design & development-centered initiation*), atau berdasarkan konteks klien/pengguna (*client*/*context initiated*).

Dalam konteks pengembangan sistem manajemen proyek untuk mahasiswa, pendekatan DSR sangat relevan karena memungkinkan peneliti untuk menciptakan solusi teknologi yang inovatif, memiliki dasar teoritis yang kuat, dan dapat dievaluasi secara sistematis. Metodologi ini juga memfasilitasi integrasi antara kebutuhan pengguna, prinsip perancangan sistem, dan evaluasi empiris, sehingga menghasilkan artefak teknologi yang tidak hanya berfungsi secara teknis namun juga memiliki kontribusi keilmuan yang signifikan. Alur penelitian DSR yang diadaptasi dalam penelitian ini divisualisasikan pada Gambar 2.



Gambar 2. *Design science research diagram* menurut Ken Peffers

2.2 Pendekatan Penelitian

Penelitian ini menerapkan pendekatan campuran (*mixed method*) yang mengintegrasikan elemen kualitatif dan kuantitatif. Pada tahap awal yaitu identifikasi masalah dan penentuan tujuan solusi, pendekatan kualitatif digunakan melalui observasi dan studi literatur untuk mendapatkan pemahaman komprehensif tentang kebutuhan mahasiswa dalam manajemen proyek akademik. Pendekatan ini memungkinkan peneliti mengungkap konteks dan kompleksitas permasalahan yang dihadapi oleh pengguna (Creswell, 2014). Sementara itu, untuk tahap evaluasi artefak atau pengujian, pendekatan kuantitatif diaplikasikan dengan menggunakan metode *System Usability Scale* (SUS) dan *blackbox testing* untuk mengukur kebergunaan dan fungsionalitas sistem secara objektif.

Pendekatan campuran ini selaras dengan karakteristik DSR yang berfokus pada pengembangan dan evaluasi sistematis artefak teknologi untuk menyelesaikan permasalahan praktis (Hevner et al., 2004; Venable et al., 2016). Dengan mengadopsi pendekatan ini, penelitian tidak hanya menghasilkan sistem manajemen proyek yang fungsional, tetapi juga pemahaman komprehensif tentang bagaimana sistem tersebut dapat mengoptimalkan pengelolaan proyek akademik mahasiswa, serta memberikan triangulasi data yang meningkatkan validitas dan reliabilitas hasil penelitian

2.3 Objek Penelitian

Objek penelitian ini adalah mahasiswa sebagai pengguna utama dari sistem manajemen proyek berbasis website yang dikembangkan. Penelitian ini berfokus pada analisis kebutuhan, pengalaman, dan persepsi mahasiswa dalam menggunakan sistem untuk mengelola tugas dan proyek akademik.

2.4 Metode Pengumpulan Data

Metode pengumpulan data merupakan elemen krusial yang menentukan keberhasilan penelitian, karena kesesuaian dan ketepatan teknik yang digunakan sangat memengaruhi validitas hasil yang diperoleh. Dalam penelitian ini, metode pengumpulan data dibagi menjadi dua fase sesuai dengan tahapan penelitian:

Fase Awal (Analisis Kebutuhan):

1. Observasi
   Observasi merupakan teknik pengumpulan data melalui pengamatan langsung terhadap subjek penelitian. Metode ini digunakan untuk mendokumentasikan perilaku, interaksi, dan fenomena yang terjadi secara alami tanpa adanya intervensi (Ardiansyah et al., 2023). Dalam penelitian ini, peneliti melakukan observasi terhadap mahasiswa di Universitas Hasanuddin untuk memahami cara mereka melakukan manajemen proyek dan tugas serta dampaknya terhadap kegiatan akademik
2. Studi Literatur
   Studi literatur melibatkan pencarian, pengumpulan, dan analisis sumber-sumber tertulis yang relevan dengan topik penelitian. Teknik ini memungkinkan peneliti untuk mendapatkan landasan teori yang kuat, mengidentifikasi temuan-temuan sebelumnya, serta mengintegrasikan berbagai perspektif dan konsep yang berkaitan dengan sistem manajemen proyek dan pengelolaan tugas akademik. Sumber yang digunakan mencakup jurnal ilmiah, buku, artikel, dan dokumen-dokumen lainnya.

Fase Evaluasi (Pengujian sistem):

3. Pengujian Fungsionalitas
   Setelah sistem selesai dikembangkan, dilakukan pengujian *blackbox* untuk mengevaluasi apakah setiap fitur sistem bekerja sesuai dengan spesifikasi yang telah ditentukan. Metode ini berfokus pada pengujian fungsionalitas sistem tanpa memperhatikan struktur internal kode. Pengujian dilakukan dengan skenario input-output untuk memastikan keandalan dan kestabilan sistem dalam menjalankan tugas-tugas yang dirancang.
4. Pengujian Kebergunaan
   Selain pengujian fungsional, evaluasi *usability* juga dilakukan dengan menggunakan metode *System Usability Scale (SUS)*. Instrumen ini digunakan untuk mengukur sejauh mana sistem mudah digunakan, dapat dipahami, dan memberikan pengalaman pengguna yang memuaskan. Kuesioner SUS terdiri dari 10 pernyataan yang dinilai oleh pengguna untuk menghasilkan skor kuantitatif yang merefleksikan tingkat kegunaan sistem secara keseluruhan. Wawancara dilakukan setelah pengujian SUS

menggunakan pendekatan berbasis konstruk TAM untuk menggali pemahaman yang lebih mendalam mengenai pengalaman, kesulitan, dan persepsi pengguna yang tidak dapat terungkap hanya melalui skor kuantitatif.

2.5 Waktu dan Lokasi Penelitian

Penelitian ini dilaksanakan selama empat bulan, mulai dari Desember 2024 hingga Maret 2025, dan berlangsung di lingkungan Universitas Hasanuddin. Proses penelitian dibagi ke dalam beberapa tahap, yang dirangkum secara rinci dalam tabel berikut.

Tabel 4. Jadwal penelitian

Tahapan Penelitian	2024				2025											
	Desember				Januari				Februari				Maret			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Pengumpulan data																
Analisis Kebutuhan																
Perancangan sistem																
Implementasi																
Penerapan dan Pengujian sistem																

2.6 Metode Pengembangan Sistem

Dalam penelitian ini, proses pengembangan sistem informasi manajemen proyek mahasiswa berbasis website mengadopsi salah satu metodologi klasik dan fundamental dalam *Systems Development Life Cycle* (SDLC), yaitu metode *waterfall*. Pemilihan metode *waterfall* didasarkan pada pendekatannya yang sistematis dan sekuensial, di mana setiap tahapan pengembangan harus diselesaikan secara menyeluruh dan tuntas sebelum dapat melanjutkan ke tahapan berikutnya. Karakteristik utama dari metode ini adalah alur kerja yang linear dan terstruktur, dimulai dari perencanaan awal hingga pemeliharaan akhir, tanpa adanya iterasi atau pengulangan kembali ke fase yang telah selesai. Berikut adalah rincian tahapan-tahapan yang dilakukan dalam pengembangan sistem ini sesuai dengan metode *waterfall*:

1. Analisis Kebutuhan *(Requirements Analysis)*
   Tahap ini merupakan fondasi awal dan krusial dalam keseluruhan proses pengembangan sistem. Pada fase ini, peneliti melakukan identifikasi secara

mendalam terhadap permasalahan yang dihadapi oleh mahasiswa terkait manajemen proyek akademik. Selain itu, dilakukan analisis komprehensif untuk menentukan kebutuhan-kebutuhan fungsional dan non-fungsional yang harus dipenuhi oleh sistem yang akan dibangun. Proses pengumpulan data esensial dilakukan melalui observasi langsung untuk mengamati secara langsung bagaimana mahasiswa mengelola tugas dan proyek mereka serta tantangan yang mereka hadapi, dan studi literatur untuk menelaah berbagai sumber pustaka, penelitian terdahulu, dan dokumentasi terkait sistem manajemen proyek guna mendapatkan landasan teoritis dan praktik terbaik. Hasil dari tahap ini adalah sebuah daftar rinci mengenai fitur-fitur dan fungsi-fungsi spesifik yang wajib diimplementasikan dalam sistem agar dapat menjawab permasalahan dan memenuhi kebutuhan pengguna secara efektif.

2. Perancangan Sistem (*System Design*)
   Setelah kebutuhan sistem teridentifikasi dengan jelas, tahap selanjutnya adalah perancangan sistem. Pada fase ini, peneliti fokus untuk menyusun cetak biru (*blueprint*) atau arsitektur detail dari sistem yang akan dibangun. Proses perancangan ini mencakup perancangan diagram UML (*Unified Modeling Language*) untuk membuat model visual sistem menggunakan diagram UML seperti *use case diagram*, *class diagram*, dan *activity diagram* guna memvisualisasikan interaksi pengguna, struktur data, dan alur kerja sistem, serta perancangan antarmuka pengguna (*User Interface Design*) untuk mendesain tampilan visual dan pengalaman interaksi pengguna dengan sistem, memastikan antarmuka yang intuitif, mudah digunakan, dan menarik. Tahap perancangan sistem ini menghasilkan dokumen spesifikasi desain yang komprehensif dan akan menjadi panduan utama bagi tim pengembang pada tahap implementasi.
3. Pembangunan Sistem (*Implementation*)
   Pada tahap implementasi, rancangan sistem yang telah disusun sebelumnya diwujudkan menjadi sebuah aplikasi website yang fungsional. Peneliti menerjemahkan model dan desain ke dalam kode program yang sebenarnya. Teknologi yang digunakan dalam pembangunan sistem ini meliputi bahasa pemrograman TypeScript sebagai bahasa utama untuk pengembangan logika aplikasi yang menawarkan keunggulan static typing untuk mengurangi *error* dan meningkatkan kualitas kode, *framework* SvelteKit yang dipilih sebagai *framework full-stack* yang memanfaatkan keunggulan *framework* JavaScript Svelte untuk membangun komponen antarmuka pengguna (UI) yang sangat responsif dan efisien karena Svelte mengkompilasi komponen menjadi kode JavaScript imperatif yang optimal pada saat *build-time* bukan *run-time* di browser, dan PostgreSQL yang digunakan sebagai sistem manajemen basis data relasional (RDBMS) untuk menyimpan dan mengelola seluruh data terkait pengguna, proyek, dan tugas yang dipilih karena keandalan, skalabilitas, dan fitur-fitur lanjut yang dimilikinya. Proses pembangunan ini dilakukan secara iteratif dalam lingkup

internal tahap implementasi hingga seluruh modul dan fitur yang dirancang berhasil dikembangkan.

4. Penerapan dan Pengujian Sistem (*Deployment and Testing*)
   Setelah tahap pembangunan sistem *(Implementation)* selesai, aplikasi memasuki tahap pengujian *(Testing)* yang komprehensif untuk memastikan kualitas dan kelayakannya sebelum diterapkan secara publik. Pada fase ini, dilakukan pengujian fungsional melalui metode *blackbox testing* untuk memverifikasi bahwa setiap fungsi sistem bekerja sesuai dengan spesifikasi kebutuhan yang telah ditetapkan, dengan fokus pada validasi input dan output yang dihasilkan. Selanjutnya, untuk mengevaluasi aspek kebergunaan dari perspektif pengguna, diterapkan pendekatan metode campuran yang menggabungkan evaluasi kuantitatif menggunakan kuesioner *System Usability Scale* (SUS) dan evaluasi kualitatif melalui wawancara semi-terstruktur. Kuesioner SUS bertujuan untuk mengukur seberapa mudah, efektif, dan memuaskan sistem ini bagi mahasiswa, sementara wawancara digunakan untuk mendapatkan umpan balik yang lebih mendalam mengenai pengalaman mereka. Hasil dari seluruh rangkaian pengujian ini menjadi dasar untuk menentukan apakah sistem telah layak dan bebas dari kesalahan kritis. Setelah sistem dinyatakan lulus tahap pengujian, barulah dilanjutkan ke tahap penerapan *(deployment)*, di mana sistem dipasang pada platform *hosting cloud* Vercel agar dapat diakses secara publik oleh pengguna (mahasiswa) melalui internet.

2.7 Tahapan Penelitian

Penelitian ini dilaksanakan dengan mengikuti tahapan-tahapan sistematis sesuai dengan metodologi *waterfall* yang telah ditetapkan, dimulai dari fase perencanaan hingga evaluasi akhir sistem. Tahap awal penelitian diawali dengan proses pengumpulan data komprehensif melalui dua pendekatan utama, yaitu observasi langsung dan studi literatur ekstensif. Observasi langsung dilakukan untuk memperoleh gambaran nyata tentang bagaimana mahasiswa saat ini mengelola tugas dan proyek akademik mereka, termasuk mengidentifikasi kendala-kendala yang dihadapi dalam sistem konvensional serta pola perilaku yang berkaitan dengan prokrastinasi akademik. Proses observasi ini melibatkan pengamatan terhadap kebiasaan mahasiswa dalam mengorganisir jadwal, memprioritaskan tugas, dan menggunakan alat bantu manajemen yang telah ada. Sementara itu, studi literatur dilakukan secara mendalam untuk mengkaji penelitian-penelitian terdahulu yang relevan dengan topik manajemen proyek, prokrastinasi akademik, serta pengembangan sistem informasi berbasis website, guna memperoleh landasan teoritis yang kuat dan mengidentifikasi *best practices* dalam pengembangan sistem serupa.

Informasi dan data yang diperoleh dari kedua pendekatan tersebut kemudian dianalisis secara menyeluruh untuk menyusun analisis kebutuhan yang detail dan komprehensif. Proses analisis ini mencakup identifikasi kebutuhan fungsional sistem seperti fitur pembuatan tugas, pengaturan *deadline*, kategorisasi proyek, sistem

notifikasi, dan pemantauan progres, serta kebutuhan non-fungsional seperti aspek performa, kegunaan, dan kompatibilitas lintas platform. Hasil analisis kebutuhan ini selanjutnya digunakan untuk menghasilkan *blueprint* atau cetak biru sistem yang berfungsi sebagai panduan utama dalam seluruh proses pengembangan. *Blueprint* ini mencakup spesifikasi teknis, arsitektur sistem, serta spesifikasi kebutuhan yang menjadi acuan dalam tahapan-tahapan selanjutnya.

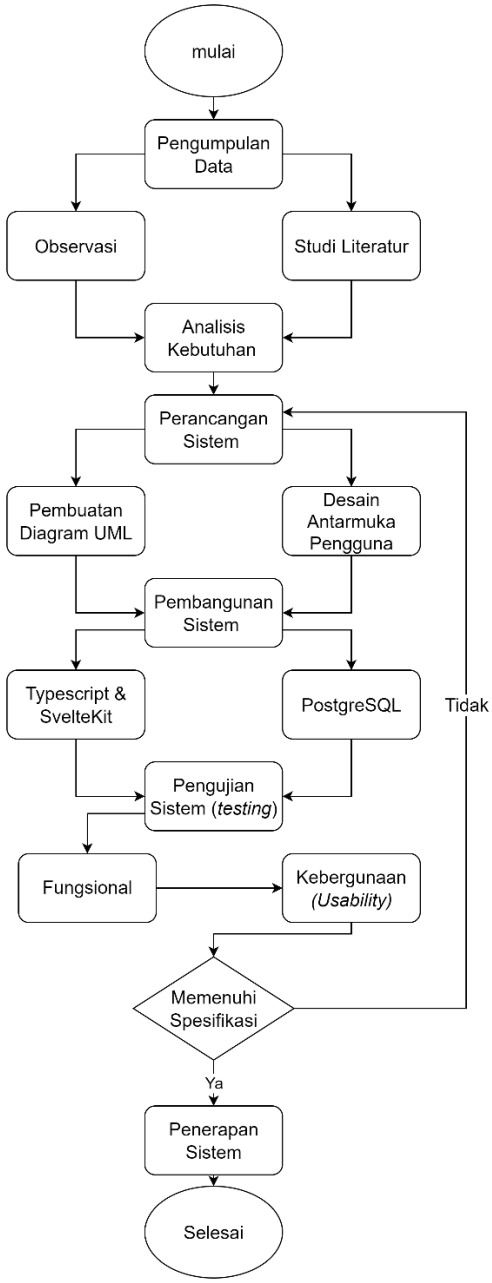
Berdasarkan *blueprint* yang telah disusun, peneliti kemudian melanjutkan ke tahap perancangan sistem yang detail dan terstruktur. Pada fase ini, dilakukan penyusunan berbagai diagram UML yang mencakup *use case diagram* untuk menggambarkan interaksi antara pengguna dengan sistem, *class diagram* untuk merepresentasikan struktur data dan relasi antar entitas dalam sistem, serta *activity diagram* untuk mengilustrasikan alur kerja dan proses bisnis yang terjadi dalam sistem. Selain pembuatan diagram UML, tahap perancangan juga meliputi proses desain antarmuka pengguna (*User Interface*) yang intuitif, responsif, dan *user-friendly*. Proses desain UI/UX ini mempertimbangkan prinsip-prinsip desain modern, aksesibilitas, serta kebutuhan spesifik mahasiswa sebagai target pengguna utama sistem. mockup dibuat untuk memvisualisasikan tata letak, navigasi, dan elemen-elemen visual sistem sebelum masuk ke tahap implementasi.

Tahap implementasi sistem dilaksanakan berdasarkan desain dan spesifikasi yang telah ditetapkan pada fase sebelumnya. Pengembangan aplikasi website dilakukan menggunakan teknologi modern yang terdiri dari TypeScript sebagai bahasa pemrograman utama yang menyediakan keunggulan *static typing* untuk meningkatkan kualitas dan maintainability kode, SvelteKit sebagai *framework full-stack* yang memberikan performa optimal dan *developer experience* yang baik, serta PostgreSQL sebagai sistem manajemen basis data relasional yang handal dan scalable untuk menyimpan dan mengelola data aplikasi. Proses implementasi dilakukan secara bertahap, dimulai dari pengembangan struktur *database*, implementasi *backend* API, pembuatan komponen *frontend*, hingga integrasi seluruh modul sistem. Setiap komponen yang dikembangkan melalui proses testing internal untuk memastikan *functionality* dan *compatibility* sebelum diintegrasikan dengan komponen lainnya.

Setelah seluruh fitur aplikasi berhasil dikembangkan dan terintegrasi, dilakukan serangkaian pengujian komprehensif untuk memastikan kualitas dan kesesuaian sistem dengan requirement yang telah ditetapkan. Pengujian dilakukan menggunakan metode *blackbox testing* yang fokus pada verifikasi input dan output sistem tanpa mempertimbangkan struktur internal kode, mencakup testing fungsionalitas setiap fitur, validasi form input, pengujian navigasi antar halaman, serta skenario penggunaan normal dan *edge cases*. Selain *blackbox testing*, dilakukan juga pengujian kebergunaan (*usability*) menggunakan metode *System Usability Scale* (SUS) untuk mengevaluasi tingkat kegunaan dan kepuasan pengguna terhadap sistem. Pengujian SUS melibatkan sejumlah responden mahasiswa yang diminta untuk menggunakan sistem dan memberikan penilaian terhadap berbagai aspek *usability* seperti kemudahan penggunaan, efisiensi,

kepuasan pengguna, dan *learning curve* sistem. Untuk memperdalam temuan dari kuesioner, beberapa responden perwakilan akan diwawancarai secara semi-terstruktur menggunakan pendekatan berbasis konstruk TAM untuk menggali pengalaman dan mendapatkan umpan balik kualitatif. Hasil dari seluruh metode pengujian ini kemudian dianalisis secara mendalam untuk menentukan kelayakan sistem.

Hasil dari kedua metode pengujian tersebut kemudian dianalisis secara mendalam untuk menentukan apakah sistem telah memenuhi semua kriteria dan standar yang ditetapkan. Jika hasil pengujian menunjukkan bahwa sistem telah berfungsi optimal, memenuhi requirement fungsional dan non-fungsional, serta mendapat respon positif dari pengguna, maka aplikasi akan diterapkan ke lingkungan produksi melalui *deployment* pada platform *cloud hosting*. Proses *deployment* meliputi konfigurasi server produksi, pengaturan *database cloud*, dan optimasi performa. Setelah sistem berhasil di-*deploy* dan dapat diakses secara publik, penelitian dianggap telah mencapai tahap penyelesaian dengan sistem yang siap digunakan oleh target pengguna. Namun, apabila dalam proses pengujian ditemukan kekurangan, *bug*, atau aspek yang belum memenuhi standar yang ditetapkan, maka akan dilakukan iterasi perbaikan dengan kembali ke tahap yang relevan dalam siklus pengembangan, hingga tercapai hasil yang optimal dan memenuhi seluruh kriteria keberhasilan yang telah ditetapkan. Keseluruhan proses dan alur tahapan penelitian ini dirangkum secara visual dalam diagram alir pada Gambar 3.



Gambar 3. Tahapan penelitian