Laporan dan Rancangan Website
Sistem Informasi Ekstrakulikuler (SIXKUL)
Untuk Suatu Sekolah Menengah Atas (SMA)

Laporan Tentang Project Sistem Informasi Manajemen (SIM)

Disusun Untuk Memenuhi Tugas Kelompok Mata Kuliah
SISTEM INFORMASI MANAJEMEN (SIM)

DANDY ARYA AKBAR	**1313623028**
DANAR PRIYO UTOMO	**1313623022**

KELAS A
PROGRAM STUDI ILMU KOMPUTER
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS NEGERI JAKARTA (UNJ)
RAWAMANGUN, JAKARTA TIMUR

TAHUN AJARAN GANJIL 2025/2026

KATA PENGANTAR

Puji Syukur kami panjatkan ke hadirat Tuhan Yang Maha Esa atas segala Rahmat dan karunia-Nya sehingga kami dapat menyelesaikan laporan tentang project website Sistem Informasi Manajemen (SIM) dengan judul "Laporan dan Rancangan Website Sistem Informasi Ekstrakulikuler (SIXKUL) Untuk Suatu Sekolah Menengah Atas (SMA)" ini dengan baik, benar, dan lancar. Serta tidak lupa juga kami mengucapkan terima kasih yang sebesar-besarnya kepada Yang Terhormat Ibu Ratna Widyati, S.Si., M.Kom. sebagai dosen pengampu mata kuliah Sistem Informasi Manajemen (SIM), yang telah membantu memberikan arahan dan pemahaman dalam penyusunan laporan pembuatan project ini, dan juga kepada pihak pihak lain yang terlibat secara langsung maupun tidak langsung yang nama-namanya tidak bisa kami sebutkan satu persatu namun perlu diingat disini bahwa hal tersebut tidak mengurangi rasa terima kasih kami.

Laporan pembuatan project ini disusun untuk memaparkan sekaligus membahas keseluruhan dari proses pembuatan dan pengembangan website Sistem Informasi Ekstrakulikuler (SIXKUL) yang kita tujukan untuk digunakan dan dimanfaatkan oleh para murid-murid atau para siswa dan siswi yang sedang menempuh masa-masa pendidikannya di suatu Sekolah Menengah Atas (SMA).

Dalam penerbitan laporan pembuatan project ini, kami menyadari bahwa mungkin masih terdapat banyak kekurangan dalam proses penyusunan laporan ini karena segala bentuk keterbatasan, baik dalam hal teknis maupun dalam hal fundamental. Oleh karena itu kami sangat mengharapkan kritik dan saran dari pembaca sekalian untuk membantu menyempurnakan laporan pembuatan project ini, dan kami harap, semoga apa yang tertulis, tersampaikan, dan tersirat di dalam laporan pembuatan project ini dapat memberikan manfaat bagi para pembaca sekalian.

DKI Jakarta, 10 November 2025

Anggota Kelompok 6

DAFTAR ISI

BAB I
PENDAHULUAN

1.1 Uraian Permasalahan

Kami menyadari bahwa dalam proses pengelolaan kegiatan ekstrakurikuler pada tingkat Sekolah Menengah Atas (SMA), masih terdapat berbagai kendala yang menyebabkan penyampaian informasi dan administrasi tidak berjalan secara optimal. Pengelolaan kegiatan yang umumnya masih dilakukan secara manual maupun melalui media komunikasi yang tidak terpusat sering kali menimbulkan keterlambatan informasi, kesalahan pencatatan, dan kurangnya koordinasi antara pihak siswa, pembina ekstrakurikuler, serta pihak sekolah. Kondisi ini membuat siswa kerap mengalami kebingungan mengenai jadwal kegiatan, prosedur pendaftaran, maupun perubahan agenda yang terjadi secara mendadak.

Di sisi lain, proses pendaftaran ekstrakurikuler yang masih menggunakan formulir fisik menimbulkan permasalahan seperti menumpuknya berkas, potensi kehilangan data, serta kesulitan bagi pembina dalam melakukan rekapitulasi anggota secara cepat dan akurat. Keterbatasan ini diperparah dengan minimnya transparansi terkait akses data keanggotaan, absensi kegiatan, dan dokumentasi aktivitas yang membuat evaluasi kinerja setiap ekstrakurikuler menjadi kurang efektif. Selain itu, tidak adanya platform terintegrasi juga menyulitkan siswa untuk mencari informasi mengenai jenis-jenis ekstrakurikuler yang tersedia, visi dan misi masing-masing kegiatan, serta aktivitas yang sedang dijalankan.

Berdasarkan kondisi tersebut, diperlukan suatu sistem informasi yang mampu mengintegrasikan seluruh kebutuhan pengelolaan ekstrakurikuler dalam satu platform yang mudah diakses dan digunakan. Kehadiran Sistem Informasi Ekstrakurikuler (SIXKUL) diharapkan dapat menjadi solusi untuk meningkatkan efisiensi administrasi, mempercepat penyebaran informasi, serta membantu siswa dalam memperoleh informasi dan mengikuti kegiatan ekstrakurikuler secara lebih terstruktur dan terorganisir.

1.2 Analisis Kebutuhan

Dalam rangka mengatasi berbagai permasalahan yang muncul pada proses pengelolaan kegiatan ekstrakurikuler di tingkat SMA, diperlukan sebuah sistem informasi yang mampu menyediakan fitur-fitur yang relevan, terintegrasi, dan mudah digunakan baik oleh siswa maupun pihak sekolah. Sistem Informasi Ekstrakurikuler (SIXKUL) harus mampu menjadi wadah utama untuk penyediaan informasi, administrasi kegiatan, serta interaksi antara pembina dan peserta

ekstrakurikuler. Oleh karena itu, analisis kebutuhan dilakukan untuk mengidentifikasi fungsi-fungsi inti yang perlu dikembangkan dalam sistem ini.

Kebutuhan utama dari sistem ini adalah kemampuan untuk menampilkan informasi ekstrakurikuler secara jelas dan lengkap, mencakup deskripsi kegiatan, jadwal latihan, pembina yang bertanggung jawab, serta dokumentasi aktivitas. Sistem juga perlu menyediakan mekanisme pendaftaran online yang memudahkan siswa untuk memilih ekstrakurikuler yang diminati tanpa harus mengisi formulir fisik. Selain itu, diperlukan fitur manajemen data anggota yang memungkinkan pembina melakukan pencatatan, perubahan, dan pemantauan anggota secara terstruktur.

Di samping itu, sistem harus mampu menyediakan modul absensi digital untuk mendukung pencatatan kehadiran siswa secara akurat dan efisien. Kemampuan mengirimkan pengumuman atau notifikasi secara terpusat juga menjadi kebutuhan penting agar setiap informasi terkini dapat diterima oleh siswa dengan cepat. Sementara itu, dari sisi administrator atau pihak sekolah, diperlukan akses terhadap laporan kegiatan, statistik keanggotaan, serta data aktivitas ekstrakurikuler sebagai bahan evaluasi dan pengambilan keputusan.

Dengan adanya analisis kebutuhan ini, pengembangan SIXKUL dapat diarahkan untuk menyediakan solusi yang komprehensif, efektif, dan sesuai dengan kebutuhan nyata dalam pengelolaan ekstrakurikuler di lingkungan sekolah. Sistem yang dihasilkan diharapkan mampu meningkatkan efisiensi administrasi, memperkuat komunikasi, serta memberikan pengalaman yang lebih mudah dan terorganisir bagi seluruh pengguna.

1.3. Model Pengembangan Perangkat Lunak

Proses pengembangan perangkat lunak website Sistem Informasi Ekstrakurikuler (SIXKUL) untuk Sekolah Menengah Atas (SMA) menggunakan model pengembangan **Agile SCRUM**. Pemilihan model ini didasarkan pada pertimbangan bahwa proyek SIXKUL memerlukan fleksibilitas tinggi, iterasi cepat, dan kolaborasi intensif antara tim pengembang, stakeholder, serta pengguna akhir (*end*-users, yang berupa siswa dan pembina ekstrakurikuler).

Agile SCRUM adalah metodologi pengembangan perangkat lunak yang berfokus pada delivery inkremental, feedback berkelanjutan, dan adaptasi terhadap perubahan kebutuhan. Model ini dipilih karena beberapa keuntungan strategis yang sesuai dengan karakteristik proyek SIXKUL:

1. **Fleksibilitas dan Adaptabilitas**: Agile SCRUM memungkinkan tim untuk menyesuaikan fitur dan fungsionalitas SIXKUL berdasarkan feedback dari pengguna selama proses pengembangan. Hal ini penting mengingat kebutuhan pengelolaan ekstrakurikuler dapat berkembang dan berubah seiring waktu.

2. **Iterasi Pendek (Sprint)**: Pengembangan dilakukan dalam siklus pendek yang disebut sprint (biasanya 1-4 minggu). Setiap sprint menghasilkan increment atau bagian dari produk yang dapat diuji dan dievaluasi. Pendekatan ini memastikan progress yang terukur dan risiko yang dapat dikelola dengan lebih baik.

3. **Kolaborasi Intensif**: Agile SCRUM menekankan komunikasi langsung dan kolaborasi antara anggota tim, Product Owner, dan stakeholder. Daily standup meetings, sprint planning, dan sprint review memastikan bahwa semua pihak tetap selaras dengan tujuan proyek.

4. **Feedback Berkelanjutan**: Melalui sprint review dan user testing di setiap akhir sprint, tim dapat mengumpulkan feedback langsung dari pengguna SIXKUL. Masukan ini kemudian diprioritaskan dalam sprint berikutnya, memastikan bahwa sistem yang dikembangkan benar-benar memenuhi kebutuhan nyata.

5. **Delivery Inkremental**: Agile SCRUM menghasilkan delivery produk secara bertahap dan incremental. Ini berarti fitur-fitur inti SIXKUL dapat dirilis lebih awal untuk penggunaan terbatas, sementara fitur tambahan dikembangkan dalam sprint berikutnya. Pendekatan ini mengurangi waktu tunggu untuk mendapatkan value dari sistem.

Struktur SCRUM dalam proyek SIXKUL terdiri dari:

- **Product Backlog**: Daftar lengkap fitur dan requirement yang harus dikembangkan dalam SIXKUL.

- **Sprint Backlog**: Subset dari product backlog yang dipilih untuk dikerjakan dalam sprint tertentu.

- **Daily Standup**: Pertemuan harian untuk memantau progress dan mengidentifikasi hambatan.

- **Sprint Review**: Presentasi increment produk yang selesai kepada stakeholder dan pengguna.

- **Sprint Retrospective**: Refleksi tim tentang proses pengembangan untuk perbaikan berkelanjutan.

Dengan menggunakan model proses Agile SCRUM, proyek SIXKUL ini dapat dikembangkan dengan efisien, responsif terhadap perubahan, dan menghasilkan produk yang sesuai dengan ekspektasi para pengguna-pengguna akhir di suatu Sekolah Menengah Atas (SMA).

1.4. Metode Pemrograman

Metode pemrograman yang digunakan dalam pengembangan website Sistem Informasi Ekstrakurikuler (SIXKUL) menggabungkan beberapa paradigma pemrograman modern yang saling melengkapi untuk menciptakan kode yang modular, maintainable, dan efisien. Pemilihan paradigma-paradigma ini dilakukan dengan mempertimbangkan karakteristik dari teknologi yang akan digunakan (Next.js, React.js, TypeScript, Tailwind CSS, dan Prisma ORM) serta best practices dalam software development kontemporer. Integrasi multi-paradigma ini memastikan bahwa setiap aspek dari pengembangan SIXKUL, mulai dari penulisan komponen UI hingga manajemen data, mengikuti prinsip-prinsip pemrograman yang solid dan proven.

1. Paradigma Object-Oriented Programming (OOP)

Object-Oriented Programming (OOP) merupakan salah satu paradigma utama yang diterapkan dalam pengembangan SIXKUL, terutama dalam konteks pemodelan data dan struktur aplikasi. Dalam OOP, sistem dipandang sebagai koleksi object yang saling berinteraksi, di mana setiap object memiliki state (data) dan behavior (method). Pada SIXKUL, paradigma OOP digunakan untuk mendefinisikan entities seperti User, Ekstrakurikuler, Anggota, Absensi, dan Notifikasi sebagai class atau type yang memiliki properties dan methods tertentu.

Implementasi OOP dalam SIXKUL terutama terlihat dalam penggunaan **TypeScript**, yang mendukung class-based OOP dengan fitur inheritance, encapsulation, dan polymorphism. Sebagai contoh, dapat didefinisikan class User dengan subclass Student, Pembina, dan Administrator (Admin) yang masing-masing mewarisi properti dan method dari parent class tetapi dengan behavior yang lebih spesifik. Dengan OOP, data yang berkaitan dengan ekstrakurikuler dapat diorganisir secara hierarki dan logis, sehingga code menjadi lebih terstruktur dan mudah dipahami. Selain itu, OOP memungkinkan implementasi prinsip SOLID (Single Responsibility, Open/Closed, Liskov Substitution, Interface

Segregation, Dependency Inversion), yang menjamin bahwa setiap komponen dalam SIXKUL memiliki tanggung jawab yang jelas dan dapat diperluas tanpa merusak existing code.

2. Paradigma Functional Programming (FP)

Functional Programming (FP) adalah paradigma yang sangat dominan dalam pengembangan frontend SIXKUL, khususnya dalam penggunaan React.js. FP menekankan pada konsep pure functions, immutability, dan function composition, di mana setiap function dirancang untuk menerima input dan menghasilkan output yang konsisten tanpa melakukan side effects yang tidak terduga. Dalam React, paradigma FP diterapkan melalui **functional components** (bukan class components), yang merupakan JavaScript functions yang return JSX untuk merepresentasikan UI.

Setiap komponen React dalam SIXKUL, seperti ExtrakurikulerCard, FormPendaftaran, DashboardAbsensi, dan NotificationPanel, ditulis sebagai functional components yang menerima props sebagai parameter dan return elements UI yang sesuai. Dengan pendekatan ini, setiap komponen bersifat pure dalam arti bahwa component dengan props yang sama akan selalu merender output UI yang sama. Konsep immutability dalam FP juga diterapkan melalui penggunaan **React Hooks** seperti useState dan useReducer, di mana state updates dilakukan dengan membuat state baru daripada memodifikasi state yang sudah ada. Pendekatan ini mencegah bugs yang sulit ditelusuri akibat state mutations yang tidak terduga. Selain itu, function composition dalam FP memungkinkan pembuatan reusable utility functions dan custom hooks yang dapat menggabungkan logic dari multiple functions, sehingga code reusability dan DRY (Don't Repeat Yourself) principle terjaga dengan baik.

3. Paradigma Declarative Programming

Declarative Programming adalah paradigma yang fokus pada **apa** yang ingin dicapai, bukan **bagaimana** cara mencapainya. Dalam konteks SIXKUL, paradigma deklaratif diterapkan di berbagai layer, terutama dalam React dan Tailwind CSS. Dalam React, developer menulis deklarasi tentang apa yang seharusnya di-render di UI tanpa perlu menspesifikasi step-by-step DOM manipulations. Sebagai contoh, jika ingin menampilkan list ekstrakurikuler dengan status tertentu, developer cukup menulis deklarasi kondisional dan filtering data, sementara React secara otomatis menangani rendering logic dan DOM updates.

Demikian pula, dalam **Tailwind CSS**, styling dilakukan secara deklaratif dengan menambahkan utility classes ke elemen HTML tanpa perlu menulis explicit CSS rules. Developer mendeklarasikan styling intent melalui class names seperti flex, gap-4, rounded-lg, shadow-md, sementara Tailwind secara otomatis menerjemahkannya menjadi CSS yang sesuai. Paradigma deklaratif ini meningkatkan readability kode, mengurangi boilerplate, dan membuat developer dapat fokus pada business logic daripada implementation details. Dalam Prisma ORM, query database juga ditulis secara deklaratif melalui Prisma Client API, di mana developer mendeklarasikan struktur data yang ingin diambil atau dimodifikasi, dan Prisma menggenerate SQL query yang sesuai secara otomatis. Sebagai contoh, prisma.ekstrakurikuler.findMany({ where: { status: 'aktif' }, include: { anggota: true } }) merupakan deklarasi yang jelas tentang apa yang diinginkan, bukan imperative SQL query.

4. Paradigma Component-Based Architecture

Component-Based Architecture merupakan evolusi dari konsep modularitas dalam programming, di mana sistem dibangun dari collection of reusable, self-contained components. Dalam SIXKUL, paradigma ini diterapkan melalui React.js, yang mendorong pemisahan UI menjadi smaller, manageable components yang dapat di-reuse di berbagai bagian aplikasi. Setiap component memiliki clear input (props) dan output (rendered UI), sehingga component dapat diprediksi, testable, dan mudah di-maintain.

Pada praktiknya, SIXKUL dibangun dari components seperti Header, Navigation, ExtrakurikulerList, StudentProfile, Attendance Form, NotificationBell, dan banyak lagi. Setiap component dapat dikembangkan secara independen, ditest secara isolated, dan diintegrasikan dengan components lain untuk membentuk pages atau features yang lebih besar. Component-Based Architecture juga mendukung **composition over inheritance**, di mana components yang kompleks dibangun dari kombinasi multiple smaller components, bukan dari inheritance hierarchy yang rumit. Hal ini membuat codebase lebih flexible, mudah di-refactor, dan mengurangi tight coupling antar komponen. Dalam konteks Next.js, component-based approach dikombinasikan dengan file-based routing, sehingga setiap page dalam SIXKUL dapat dibangun dari collection of components yang compose menjadi page yang utuh.

5. Paradigma Type-Driven Development dengan TypeScript

Type-Driven Development adalah pendekatan di mana type system dari bahasa pemrograman (dalam hal ini TypeScript) digunakan secara proaktif untuk mendefinisikan structure, contract, dan constraints dari data dan functions dalam aplikasi. Berbeda dengan JavaScript vanilla yang bersifat dynamically-typed, **TypeScript** memberikan static typing, yang berarti tipe dari setiap variable, parameter, dan return value harus dispesifikasi secara explicit atau diinfer oleh compiler.

Dalam pengembangan SIXKUL, Type-Driven Development diterapkan dengan mendefinisikan interfaces dan types yang merepresentasikan domain entities dan data structures. Sebagai contoh, dapat didefinisikan interface Student, interface Ekstrakurikuler, interface Attendance, dan sebagainya dengan properti-properti yang spesifik dan tipe data yang tepat. Dengan approach ini, setiap function yang bekerja dengan Student data akan memiliki type signatures yang jelas, misalnya function enrollStudent(student: Student, ekstrakurikuler: Ekstrakurikuler): Promise<Enrollment>. Type system TypeScript kemudian akan memastikan bahwa function hanya dipanggil dengan arguments yang tipe-nya sesuai, dan return value dapat diprediksi tipenya. Manfaat dari Type-Driven Development adalah: (1) **compile-time error detection** - banyak bugs terdeteksi saat development, bukan saat runtime; (2) **self-documenting code** - tipe dari setiap data menjadi dokumentasi implicit; (3) **IDE support** - autocomplete dan refactoring tools menjadi lebih akurat karena compiler tahu tipe dari setiap entity; (4) **safer refactoring** - mengubah type akan menghasilkan compilation errors di semua lokasi yang terpengaruh, memastikan tidak ada yang terlewat.

6. Paradigma Data-First Design dengan Prisma ORM

Paradigma Data-First Design menempatkan database schema dan data model sebagai starting point dari application development. Dalam SIXKUL, paradigma ini diterapkan melalui **Prisma ORM**, di mana developer mendefinisikan data model dalam file schema.prisma sebelum menulis business logic atau UI. File schema ini merupakan single source of truth yang mendeskripsikan struktur data, relationships, dan constraints dalam SIXKUL.

Dalam schema.prisma, dapat didefinisikan models seperti User, Ekstrakurikuler, Anggota, Attendance,
dan Notification dengan fields, types, dan relationships yang spesifik. Dari schema ini, Prisma secara otomatis menggenerate: (1) **Prisma Client** - type-safe database client dengan autocomplete support; (2) **Migration files** - database migration scripts yang dapat di-version control dan di-

deploy; (3) **TypeScript types** - type definitions yang auto-sync dengan database schema, memastikan type consistency antara application code dan database. Data-First Design approach ini memastikan bahwa data model dalam aplikasi selalu konsisten dengan database, mencegah mismatch yang sering terjadi dalam development cycle. Selain itu, dengan Prisma, query database dapat ditulis secara type-safe dan readable, misalnya prisma.ekstrakurikuler.findUnique({ where: { id: ekskul_id }, include: { anggota: true, pembina: true } }), daripada raw SQL query yang mudah error dan sulit untuk di-refactor.

7. Integrasi Multi-Paradigma dalam Pengembangan SIXKUL

Ketujuh paradigma di atas tidak berdiri sendiri, tetapi terintegrasi secara harmonis dalam setiap layer dari aplikasi SIXKUL. Pada **layer presentasi (frontend)**, React.js menggunakan kombinasi Functional Programming, Declarative Programming, dan Component-Based Architecture untuk membangun UI yang interaktif dan maintainable. TypeScript memberikan type safety pada setiap komponen dan hook. Pada **layer business logic (backend)**, Next.js API routes menggunakan kombinasi OOP untuk pemodelan domain entities dan Functional Programming untuk request handlers. Type-Driven Development dengan TypeScript memastikan request dan response contracts yang jelas. Pada **layer data (database)**, Prisma ORM menggunakan Data-First Design untuk mendefinisikan schema, dan kemudian TypeScript types auto-generated dari schema memberikan type safety ketika mengakses database dari business logic.

Integrasi ini menciptakan end-to-end type safety dari database hingga UI, sehingga type errors dapat terdeteksi lebih awal dalam development cycle. Selain itu, masing-masing paradigma memiliki kontribusi unik: OOP memberikan struktur dan organization; FP memberikan predictability dan composability; Declarative programming mengurangi boilerplate dan meningkatkan readability; Component-Based Architecture memberikan modularity dan reusability; Type-Driven Development memberikan safety dan developer experience; Data-First Design memberikan data consistency dan clarity.

Dengan menggabungkan paradigma-paradigma ini secara konsisten, pengembangan SIXKUL dapat berlangsung dengan efisien, menghasilkan kode yang berkualitas tinggi, mudah di-maintain, dan robust terhadap berbagai jenis error. Tim pengembang juga dapat dengan mudah berkolaborasi karena setiap paradigma memiliki clear rules dan best

practices yang membuat code patterns menjadi predictable dan familiar bagi semua anggota tim.

1.5. Perancangan Arsitektur

Perancangan arsitektur Sistem Informasi Ekstrakurikuler (SIXKUL) dirancang dengan pendekatan multi-layer yang terstruktur untuk memastikan sistem memiliki skalabilitas, maintainability, dan performa yang optimal. Arsitektur SIXKUL dibangun berdasarkan prinsip separation of concerns, di mana setiap layer memiliki tanggung jawab spesifik dan berkomunikasi melalui interface yang terdefinisi dengan jelas. Pemilihan arsitektur ini mempertimbangkan kebutuhan fungsional dan non-fungsional dari sistem, serta karakteristik teknologi yang digunakan dalam tech stack.

1. Arsitektur Sistem Secara Keseluruhan

SIXKUL mengadopsi **arsitektur Client-Server** dengan model **Three-Tier Architecture** yang terdiri dari Presentation Layer, Application Logic Layer, dan Data Layer. Arsitektur ini dipilih karena memberikan pemisahan yang jelas antara user interface, business logic, dan data management, sehingga memudahkan pengembangan, testing, dan maintenance sistem. Dalam implementasinya, SIXKUL menggunakan pendekatan **Monolithic Architecture** dengan Next.js sebagai full-stack framework yang mengintegrasikan frontend dan backend dalam satu codebase, namun tetap mempertahankan separation of concerns melalui struktur folder dan modularisasi kode yang ketat.

Pada level tertinggi, arsitektur SIXKUL dapat dibagi menjadi dua komponen utama: **Client-Side** yang berjalan di browser pengguna dan **Server-Side** yang berjalan di server hosting. Client-Side bertanggung jawab untuk rendering user interface, handling user interactions, dan mengirimkan HTTP requests ke server. Server-Side bertanggung jawab untuk processing business logic, authentication/authorization, database operations, dan mengirimkan HTTP responses kembali ke client. Komunikasi antara client dan server dilakukan melalui RESTful API dengan format data JSON, memastikan interoperability dan standarisasi protokol komunikasi.

2. Technology Stack dan Peran dalam Arsitektur

Technology stack yang dipilih untuk SIXKUL dirancang untuk mendukung arsitektur three-tier dengan seamless integration antar komponen. Setiap teknologi memiliki peran spesifik dalam layer-layer arsitektur:

a. Next.js sebagai Core Framework

Next.js berperan sebagai backbone dari seluruh arsitektur SIXKUL, menyediakan framework full-stack yang mengintegrasikan frontend dan backend. Pada sisi

frontend, Next.js menyediakan fitur Server-Side Rendering (SSR) dan Static Site Generation (SSG) untuk optimasi performa dan SEO. Pada sisi backend, Next.js menyediakan API Routes yang memungkinkan pembuatan RESTful API endpoints tanpa perlu server terpisah. Next.js juga menyediakan file-based routing system yang memetakan struktur folder ke URL routes secara otomatis, serta built-in optimization features seperti automatic code splitting, image optimization, dan font optimization.

b. React.js untuk Presentation Layer

React.js digunakan secara eksklusif untuk membangun Presentation Layer, yaitu seluruh user interface yang berinteraksi langsung dengan pengguna. React memungkinkan pembangunan komponen UI yang reusable dan reactive, di mana perubahan state secara otomatis memicu re-rendering komponen yang terpengaruh. Dalam arsitektur SIXKUL, React components diorganisir dalam hierarki mulai dari atomic components (button, input, card) hingga page-level components (dashboard, profile, ekstrakurikuler list). React Router yang terintegrasi dengan Next.js menangani client-side navigation untuk memberikan Single Page Application (SPA) experience.

c. TypeScript untuk Type Safety di Seluruh Stack

TypeScript berperan sebagai superset language dari JavaScript yang menyediakan static type checking di seluruh arsitektur, dari frontend hingga backend. Pada Presentation Layer, TypeScript memastikan props dan state dari React components memiliki tipe yang konsisten. Pada Application Logic Layer, TypeScript memastikan API request/response contracts sesuai dengan spesifikasi. Pada Data Layer, TypeScript types yang auto-generated oleh Prisma memastikan database queries type-safe. Integrasi TypeScript menciptakan end-to-end type safety yang mengurangi runtime errors dan meningkatkan developer experience melalui autocomplete dan intelligent code suggestions.

d. Tailwind CSS untuk Styling System

Tailwind CSS menyediakan utility-first styling system yang digunakan di Presentation Layer untuk styling komponen UI. Dalam arsitektur SIXKUL, Tailwind diintegrasikan melalui PostCSS dan dikonfigurasi untuk purging unused CSS di production build, sehingga ukuran CSS bundle minimal. Tailwind juga dikustomisasi melalui tailwind.config.js untuk mendefinisikan design tokens seperti color palette, spacing scale, typography scale, dan breakpoints yang konsisten di seluruh aplikasi. Dengan Tailwind, styling dapat dilakukan co-located dengan component logic, meningkatkan developer velocity dan maintainability.

e. Prisma ORM untuk Data Access Layer

Prisma ORM berperan sebagai Data Access Layer yang menghubungkan Application Logic Layer dengan database. Prisma menyediakan abstraksi high-level untuk database operations melalui Prisma Client, yang merupakan auto-generated dan type-safe database client. Prisma Schema mendefinisikan data models, relationships, dan constraints dalam sintaks deklaratif, dan dari schema ini Prisma dapat menggenerate migration files untuk versioned database schema changes. Prisma juga menyediakan Prisma Studio sebagai GUI tool untuk inspecting dan manipulating database data selama development. Dalam arsitektur SIXKUL, semua database queries dilakukan melalui Prisma Client, memastikan consistency, type safety, dan protection terhadap SQL injection attacks.

3. Arsitektur Layer: Three-Tier Architecture

Arsitektur SIXKUL diimplementasikan dengan Three-Tier Architecture yang terdiri dari tiga layer utama:

a. Presentation Layer (Frontend)

Presentation Layer bertanggung jawab untuk menampilkan informasi kepada pengguna dan menerima input dari pengguna. Layer ini diimplementasikan menggunakan React.js dengan Next.js sebagai framework, dan di-styling menggunakan Tailwind CSS. Presentation Layer terdiri dari pages, components, hooks, dan utilities yang secara eksklusif berjalan di client-side browser. Pages dalam Next.js (seperti /pages/index.tsx, /pages/ekstrakurikuler/[id].tsx) merepresentasikan routes dalam aplikasi, sementara components merepresentasikan reusable UI pieces. React Hooks digunakan untuk state management, side effects handling, dan custom business logic yang terkait dengan UI.

Presentation Layer berkomunikasi dengan Application Logic Layer melalui HTTP requests menggunakan Fetch API atau library seperti Axios. Setiap request ke backend melalui RESTful API endpoints yang didefinisikan di Application Logic Layer. Presentation Layer tidak memiliki akses langsung ke database dan tidak mengandung business logic yang kompleks; semua business logic ditempatkan di Application Logic Layer untuk mempertahankan separation of concerns.

b. Application Logic Layer (Backend API)

Application Logic Layer bertanggung jawab untuk processing business logic, validation, authentication, authorization, dan orchestration dari data operations. Layer ini diimplementasikan menggunakan Next.js API Routes yang berada di folder /pages/api/. Setiap file dalam folder ini merepresentasikan satu API endpoint dengan HTTP methods (GET, POST, PUT, DELETE) yang dapat di-handle.

Dalam Application Logic Layer, request dari Presentation Layer diterima, divalidasi, dan diproses. Business rules diterapkan di layer ini, misalnya: validasi apakah siswa sudah terdaftar di ekstrakurikuler tertentu sebelum melakukan enrollment, checking apakah user memiliki permission untuk mengakses resource tertentu, atau aggregating data dari multiple database tables sebelum dikirim ke client. Application Logic Layer juga bertanggung jawab untuk error handling, logging, dan returning appropriate HTTP status codes dan response messages.

Application Logic Layer berkomunikasi dengan Data Layer melalui Prisma Client untuk melakukan database operations. Setiap operation seperti membaca data ekstrakurikuler, membuat pendaftaran baru, update attendance, atau delete notification dilakukan melalui Prisma Client API yang type-safe. Layer ini tidak pernah menggunakan raw SQL queries, memastikan security dan maintainability.

c. Data Layer (Database)

Data Layer bertanggung jawab untuk persistent storage dari semua data dalam sistem SIXKUL. Layer ini terdiri dari relational database (misalnya PostgreSQL atau MySQL) yang menyimpan data dalam bentuk tables dengan relationships yang terdefinisi. Schema database didefinisikan dalam Prisma Schema file (prisma/schema.prisma), yang merupakan single source of truth untuk struktur data.

Data Layer menyimpan entities seperti User (dengan role Student, Pembina, Admin), Ekstrakurikuler, Enrollment (pendaftaran siswa ke ekstrakurikuler), Attendance (kehadiran siswa dalam kegiatan), Notification (notifikasi sistem), dan Schedule (jadwal kegiatan). Setiap entity memiliki fields dengan data types yang spesifik, constraints (unique, required, default values), dan relationships (one-to-many, many-to-many) dengan entities lain.

Prisma ORM mengelola connection pooling, query optimization, dan transaction management, sehingga Application Logic Layer tidak perlu menangani low-level database operations secara manual. Prisma Migrate digunakan untuk managing schema changes secara version-controlled, memastikan database schema di development, staging, dan production environments selalu synchronized.

4. Desain API Architecture

API architecture dalam SIXKUL mengikuti prinsip RESTful API dengan resource-based URL structure dan HTTP methods yang semantik. Setiap resource dalam sistem direpresentasikan sebagai endpoint dengan CRUD operations (Create, Read, Update, Delete) yang terdefinisi dengan jelas.

Contoh endpoint struktur:

- GET /api/ekstrakurikuler - Mengambil list semua ekstrakurikuler

- GET /api/ekstrakurikuler/[id] - Mengambil detail ekstrakurikuler spesifik
- POST /api/ekstrakurikuler - Membuat ekstrakurikuler baru (admin only)
- PUT /api/ekstrakurikuler/[id] - Update ekstrakurikuler (pembina/admin)
- DELETE /api/ekstrakurikuler/[id] - Hapus ekstrakurikuler (admin only)
- POST /api/enrollment - Daftar ke ekstrakurikuler (student)
- GET /api/attendance?ekstrakurikuler_id=X - Ambil data absensi
- POST /api/attendance - Submit absensi (pembina)
- GET /api/notifications - Ambil notifikasi user
-

Setiap endpoint dilengkapi dengan authentication dan authorization middleware yang memvalidasi JWT token dari client dan memeriksa apakah user memiliki permission untuk mengakses resource tersebut. Response dari API menggunakan standard JSON format dengan struktur konsisten: { success: boolean, data: any, error?: string }.

5. Authentication dan Authorization Architecture

SIXKUL mengimplementasikan authentication menggunakan **JWT (JSON Web Token)** based authentication. Ketika user login dengan credentials (email dan password), server memvalidasi credentials terhadap database, dan jika valid, menggenerate JWT token yang berisi user information (id, role, email) dan mengirimkannya ke client. Client menyimpan token di browser storage (httpOnly cookie atau localStorage) dan menyertakan token di setiap subsequent request melalui Authorization header.

Authorization diimplementasikan menggunakan **Role-Based Access Control (RBAC)**, di mana setiap user memiliki role (Student, Pembina, atau Admin) dan setiap endpoint/resource memiliki permission requirements. Middleware di Application Logic Layer melakukan checking apakah user role memenuhi permission requirements sebelum mengizinkan akses ke resource. Sebagai contoh, endpoint untuk membuat ekstrakurikuler baru hanya dapat diakses oleh Admin, sementara endpoint untuk submit absensi hanya dapat diakses oleh Pembina.

6. Database Schema Design

Database schema SIXKUL dirancang dengan normalisasi yang tepat untuk menghindari data redundancy dan memastikan data integrity. Schema terdiri dari beberapa main entities dengan relationships yang terdefinisi:

Entities dan relationships:

- **User** (id, email, password_hash, name, role, created_at) - Central entity untuk semua user
- **Student** (user_id, class, nis) - Extends User dengan student-specific data

- **Pembina** (user_id, phone, nip) - Extends User dengan pembina-specific data
- **Admin** (user_id, nip) - Extends User dengan admin-specific data
- **Ekstrakurikuler** (id, name, description, category, pembina_id, created_at)
- **Enrollment** (id, student_id, ekstrakurikuler_id, enrolled_at, status) - Many-to-many relationship antara Student dan Ekstrakurikuler
- **Attendance** (id, enrollment_id, date, status, notes) - Tracking kehadiran per enrollment
- **Schedule** (id, ekstrakurikuler_id, day, start_time, end_time, location)
- **Notification** (id, user_id, title, message, is_read, created_at)

Relationships didefinisikan dengan foreign keys dan Prisma relations untuk memastikan referential integrity. Indexes ditambahkan pada fields yang sering di-query (seperti user.email, ekstrakurikuler.category) untuk optimasi query performance.

7. Deployment Architecture

Deployment architecture SIXKUL dirancang untuk production environment dengan high availability dan scalability. Aplikasi Next.js di-deploy ke platform seperti **Vercel** (recommended platform untuk Next.js) atau alternatif seperti **Netlify, AWS Amplify, atau Railway**. Platform-platform ini menyediakan:

- **Automatic deployment** dari Git repository (continuous deployment)
- **Edge network/CDN** untuk serving static assets dengan latency rendah
- **Serverless functions** untuk API routes yang auto-scale berdasarkan traffic
- **Environment variables management** untuk storing secrets (database URL, JWT secret)
- **Preview deployments** untuk setiap pull request, memudahkan testing sebelum merge ke production

Database di-host di managed database service seperti **Supabase, PlanetScale, Railway PostgreSQL, atau AWS RDS**, yang menyediakan automated backups, monitoring, dan scaling capabilities. Connection antara Next.js application dan database dilakukan melalui connection string yang disimpan sebagai environment variable, dengan connection pooling yang dikelola oleh Prisma untuk efficient database connection management.

Domain management dan DNS configuration dilakukan untuk mapping custom domain ke deployed application. SSL/TLS certificates di-provision secara otomatis oleh deployment platform untuk memastikan semua komunikasi client-server encrypted (HTTPS).

8. Integrasi Antar Layer dalam Arsitektur

Integrasi antar layer dalam arsitektur SIXKUL dirancang dengan interface contracts yang jelas dan type-safe berkat TypeScript. Data flow dalam sistem mengikuti pattern berikut:

1. **User interaction** di Presentation Layer (misalnya, submit form pendaftaran ekstrakurikuler)
2. **Client-side validation** dilakukan terlebih dahulu untuk immediate feedback
3. **HTTP request** dikirim ke Application Logic Layer (POST /api/enrollment)
4. **Server-side validation** dan business logic processing di Application Logic Layer
5. **Database operation** melalui Prisma Client di Data Layer (insert enrollment record)
6. **HTTP response** dikirim kembali ke Presentation Layer dengan status dan data
7. **UI update** di Presentation Layer berdasarkan response (menampilkan success message atau error)

Setiap stage dalam flow ini memiliki error handling yang robust, memastikan bahwa errors ditangani dengan graceful dan user mendapatkan feedback yang jelas. TypeScript types memastikan bahwa data structure yang dikirim dan diterima di setiap stage konsisten dan validated.

Dengan arsitektur yang terstruktur dan terintegrasi ini, SIXKUL dapat dikembangkan secara modular, di-test secara independent per layer, dan di-maintain dengan effort yang minimal. Arsitektur juga memungkinkan future enhancements seperti penambahan features baru, integration dengan external services, atau migration ke microservices architecture jika diperlukan seiring dengan pertumbuhan sistem.

1.6. Perancangan Sequence Diagram

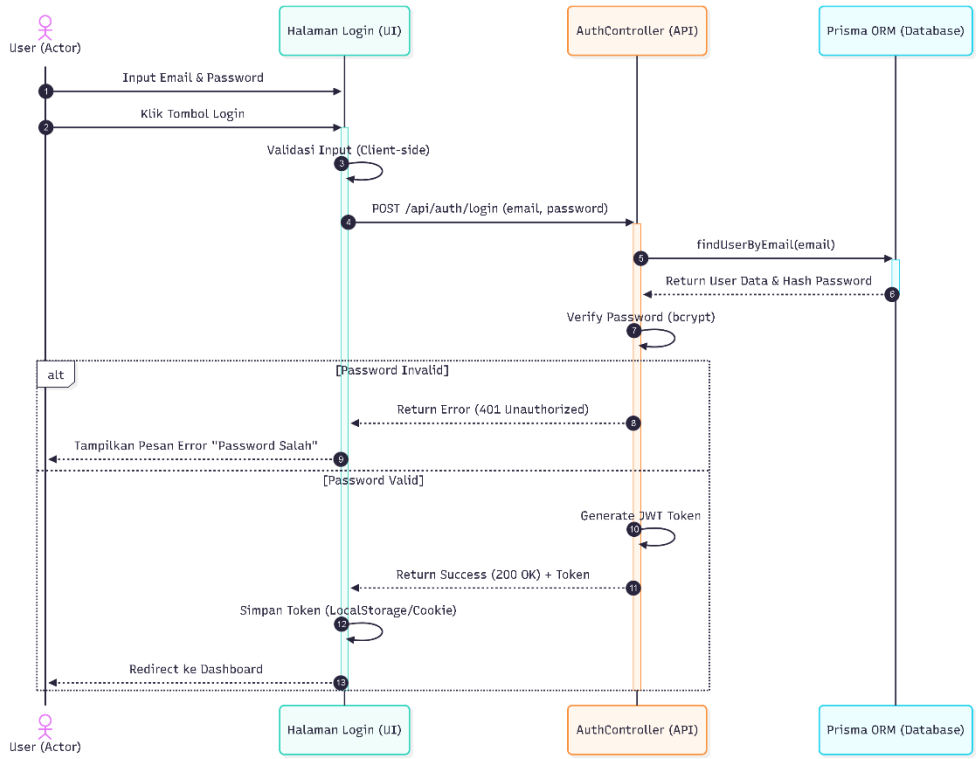
Perancangan Sequence Diagram digunakan untuk menggambarkan interaksi antara objek-objek dalam sistem SIXKUL berdasarkan urutan waktu. Diagram ini memperlihatkan bagaimana pesan (message) dikirim antar objek untuk menjalankan fungsionalitas tertentu sesuai dengan Use Case yang telah didefinisikan. Berikut adalah perancangan sequence diagram untuk proses-proses utama dalam sistem SIXKUL.

1. Sequence Diagram: Login & Autentikasi User

Diagram ini menggambarkan alur proses login user (Siswa, Pembina, atau Admin) ke dalam sistem informasi manajemen. Proses ini melibatkan validasi kredensial dan pembuatan sesi yang aman dengan menggunakan teknologi JSON Web Token (JWT).

Alur Proses:

1. User memasukkan email dan password pada halaman login.
2. Sistem memvalidasi format input.
3. Frontend mengirim request login ke Backend API.
4. Backend memverifikasi kredensial user ke Database melalui Prisma.
5. Jika valid, Database mengembalikan data user.
6. Backend membuat (generate) JWT Token.
7. Sistem mengembalikan respon sukses beserta Token ke Frontend.
8. User diarahkan ke Dashboard utama sesuai role masing-masing.



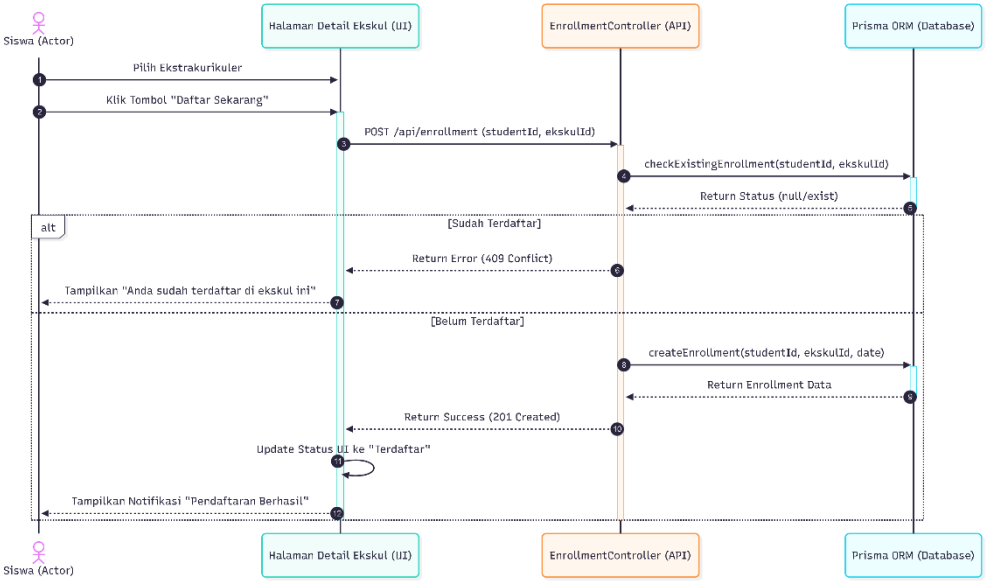
2. Sequence Diagram: Pendaftaran Ekstrakurikuler (Siswa)

Diagram ini menggambarkan proses inti di mana seorang siswa mendaftar untuk bergabung ke dalam sebuah ekstrakurikuler.

Alur Proses:

1. Siswa memilih menu "Daftar Ekstrakurikuler" dan memilih salah satu kegiatan.
2. Frontend mengirim request pendaftaran ke Backend.

3. Backend memvalidasi status siswa (apakah sudah terdaftar sebelumnya atau kuota penuh).
4. Backend menyimpan data pendaftaran baru ke tabel Enrollment di Database.
5. Database mengonfirmasi penyimpanan data.
6. Sistem memberikan notifikasi sukses kepada siswa dan mengupdate status tampilan menjadi "Terdaftar".

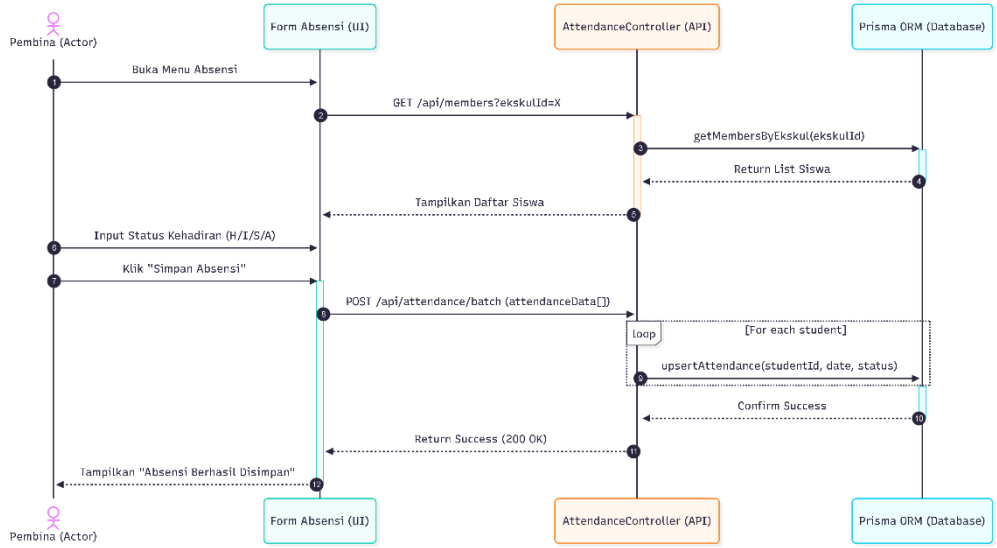


3. Sequence Diagram: Input Absensi Kegiatan (Pembina)

Diagram ini menggambarkan aktivitas Pembina dalam melakukan pencatatan kehadiran siswa pada saat kegiatan ekstrakurikuler berlangsung.

Alur Proses:

1. Pembina membuka halaman absensi dan memilih tanggal kegiatan.
2. Sistem menampilkan daftar siswa yang terdaftar di ekstrakurikuler tersebut.
3. Pembina menandai status kehadiran (Hadir/Izin/Sakit/Alpha) untuk setiap siswa.
4. Pembina menyimpan data absensi.
5. Backend memproses data bulk insert/update ke Database.
6. Sistem mengonfirmasi bahwa data absensi telah berhasil disimpan.



1.7. Perancangan Class Diagram

Class Diagram menggambarkan struktur statis dari sistem SIXKUL dengan menunjukkan kelas-kelas (classes) yang ada, atribut, metode (methods), serta hubungan (relationships) antar kelas tersebut. Perancangan ini menjadi blueprint dalam implementasi kode program, khususnya dalam pendefinisian model data pada Prisma ORM dan Type Interfaces pada TypeScript.

Berdasarkan analisis kebutuhan dan tech stack yang digunakan, struktur Class Diagram SIXKUL dirancang untuk mendukung konsep Object-Oriented (melalui TypeScript classes/interfaces) dan Relational Data Model (melalui Prisma Schema).

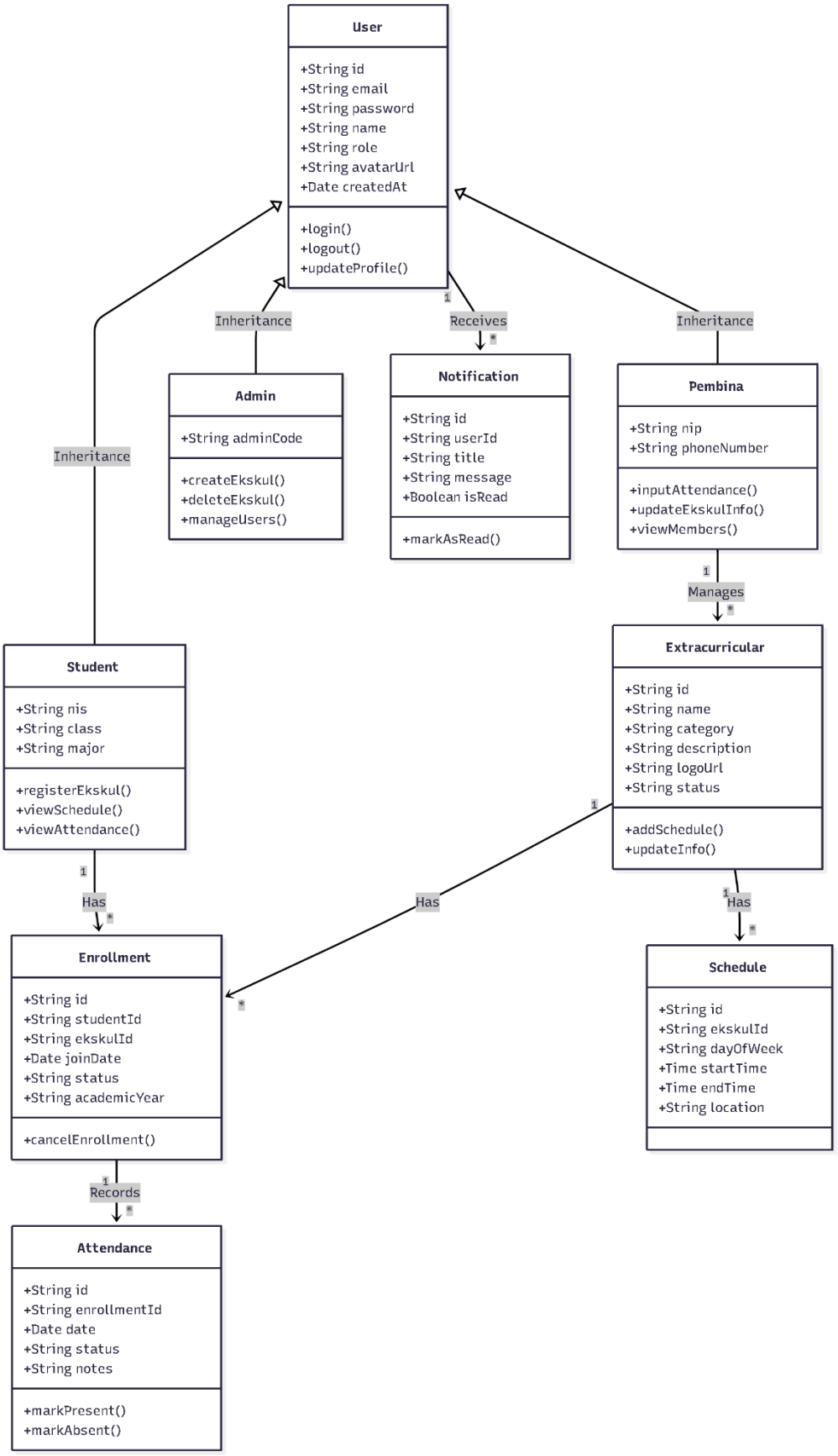
1. Struktur Kelas Utama

Sistem terdiri dari beberapa kelas entitas utama:

1. **User**: Kelas induk (parent class) yang merepresentasikan pengguna sistem secara umum.
2. **Student (Siswa)**: Turunan dari User, memiliki atribut khusus akademik.
3. **Pembina**: Turunan dari User, memiliki atribut kepegawaian.
4. **Admin**: Turunan dari User, memiliki hak akses penuh.
5. **Extracurricular**: Merepresentasikan kegiatan ekstrakurikuler.
6. **Enrollment**: Kelas asosiasi yang menghubungkan Siswa dan Ekstrakurikuler.
7. **Attendance**: Merepresentasikan data kehadiran.
8. **Schedule**: Merepresentasikan jadwal kegiatan.

2. Visualisasi Class Diagram

Berikut adalah visualisasi hubungan antar kelas dalam SIM SIXKUL:



3. Penjelasan Hubungan Antar Kelas

1. **Inheritance (Pewarisan):**
    - Kelas Student, Pembina, dan Admin merupakan turunan dari kelas User. Mereka mewarisi atribut dasar seperti id, email, password, dan name, namun memiliki atribut spesifik masing-masing (contoh: nis untuk Siswa, nip untuk Pembina).
2. **Association (Asosiasi):**
    - **Pembina manages Extracurricular**: Seorang Pembina bertanggung jawab mengelola satu atau lebih Ekstrakurikuler (One-to-Many).
    - **User receives Notification**: Setiap User dapat menerima banyak notifikasi dari sistem.
3. **Composition/Aggregation (Komposisi):**
    - **Student & Extracurricular via Enrollment**: Hubungan antara Siswa dan Ekstrakurikuler bersifat Many-to-Many. Karena perlu mencatat data tambahan seperti joinDate dan status pendaftaran, hubungan ini difasilitasi oleh kelas perantara Enrollment.
    - **Extracurricular has Schedule**: Setiap Ekstrakurikuler memiliki satu atau lebih jadwal kegiatan rutin.
    - **Enrollment records Attendance**: Data absensi (Attendance) terikat pada data pendaftaran (Enrollment) siswa tertentu, bukan langsung ke siswa, untuk memastikan absensi tercatat sesuai konteks kegiatan yang diikuti.

Perancangan Class Diagram (Diagram Kelas) ini akan diimplementasikan secara langsung menggunakan **Prisma Schema** pada tahap pengembangan backend dan **TypeScript Interfaces** pada tahap pengembangan frontend, memastikan konsistensi struktur data di seluruh sistem.

1.8. Perancangan Basis Data (Entity Relationship Diagram, ERD)

Perancangan basis data untuk Sistem Informasi Ekstrakurikuler (SIXKUL) menggunakan model relasional yang akan diimplementasikan pada database **PostgreSQL**. Struktur data dikelola menggunakan **Prisma ORM**, yang memungkinkan pendefinisian skema secara deklaratif dan type-safe. Entity Relationship Diagram (ERD) berikut menggambarkan entitas data, atribut, serta kardinalitas hubungan antar entitas dalam sistem.

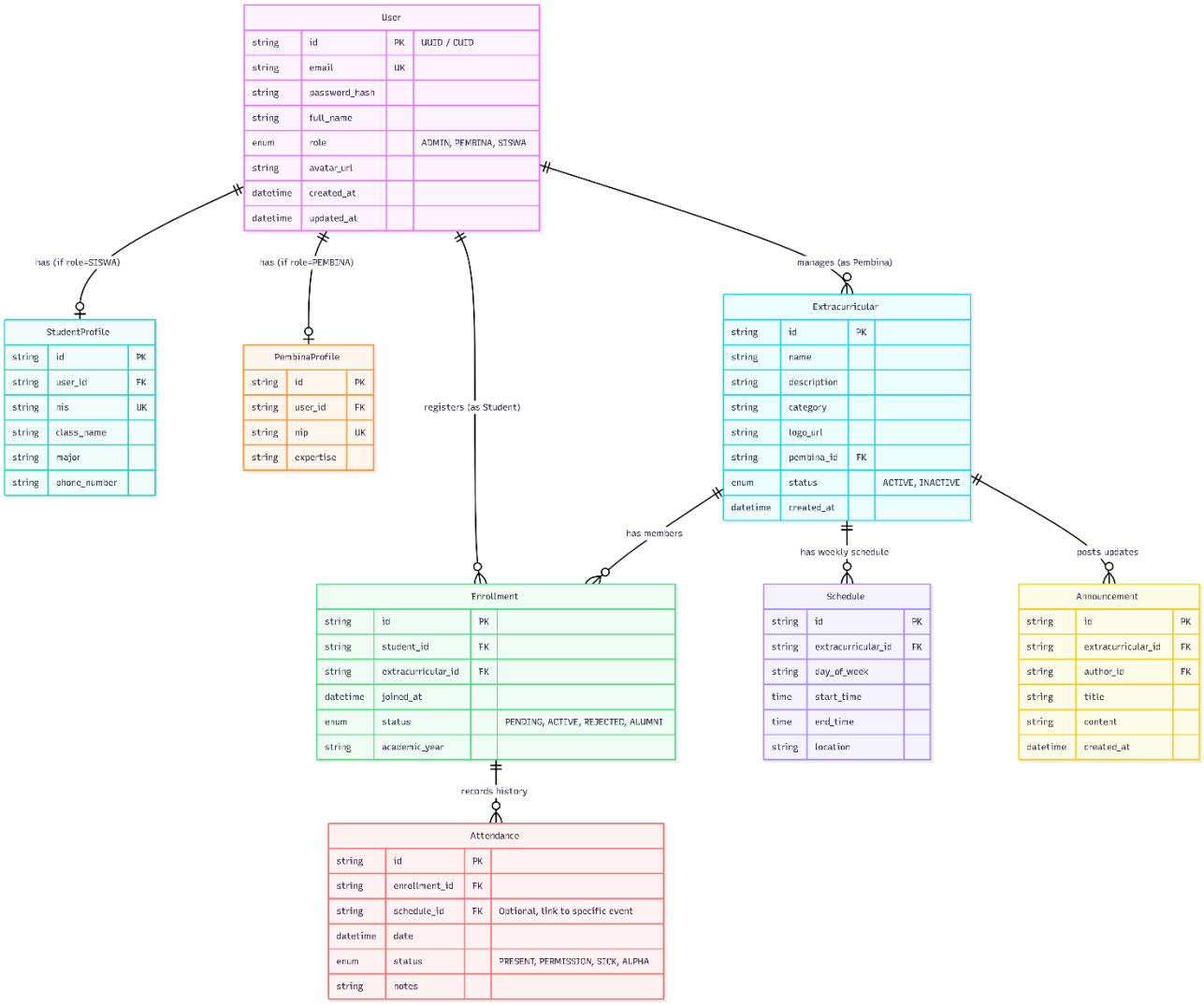
1. Komponen Entitas Utama

Berdasarkan analisis kebutuhan dan perancangan class diagram sebelumnya, terdapat entitas-entitas utama yang akan disimpan dalam basis data:

1. **Users**: Tabel sentral untuk autentikasi dan otorisasi pengguna (Siswa, Pembina, Admin).
2. **Profiles**: Tabel terpisah (atau kolom dalam User) untuk menyimpan detail spesifik profil pengguna. Dalam implementasi ini, kita menggunakan pendekatan *Single Table Inheritance* atau relasi terpisah untuk data spesifik.
3. **Extracurriculars**: Menyimpan data master kegiatan ekstrakurikuler.
4. **Enrollments**: Tabel *pivot* (junction table) yang mencatat pendaftaran siswa ke ekstrakurikuler beserta statusnya.
5. **Attendances**: Menyimpan riwayat kehadiran siswa per pertemuan.
6. **Schedules**: Menyimpan jadwal rutin kegiatan.
7. **Announcements**: Menyimpan pengumuman yang dibuat oleh Pembina/Admin untuk ekstrakurikuler tertentu.

2. Visualisasi Entity Relationship Diagram (ERD)

Berikut adalah diagram ERD yang dirancang menggunakan notasi Crow's Foot, merepresentasikan struktur tabel fisik di PostgreSQL:



3. Keterangan Relasi (Cardinality)

1. **User to Profiles (One-to-One Optional):**
    - Satu User dapat memiliki satu StudentProfile (jika role Siswa) atau satu PembinaProfile (jika role Pembina). Ini memisahkan data autentikasi (email/password) dari data profil spesifik.
2. **User (Pembina) to Extracurricular (One-to-Many):**
    - Satu User (Pembina) dapat mengelola banyak Ekstrakurikuler (User ||--o{ Extracurricular).
    - Satu Ekstrakurikuler dikelola oleh satu Pembina utama (meskipun sistem bisa dikembangkan menjadi Many-to-Many di masa depan, saat ini diasumsikan 1 pembina utama per ekskul untuk simplifikasi tanggung jawab).
3. **User (Siswa) & Extracurricular to Enrollment (Many-to-Many):**
    - Hubungan Many-to-Many antara Siswa dan Ekstrakurikuler dipecah menggunakan tabel perantara Enrollment.
    - Satu Siswa bisa mendaftar banyak Ekskul. Satu Ekskul punya banyak Siswa.
    - Tabel Enrollment menyimpan atribut tambahan seperti status (Pending/Active) dan joined_at.
4. **Extracurricular to Schedule (One-to-Many):**
    - Satu Ekstrakurikuler dapat memiliki beberapa jadwal (misal: Senin & Kamis).
5. **Enrollment to Attendance (One-to-Many):**
    - Data absensi (Attendance) terhubung ke Enrollment. Ini memastikan bahwa absen hanya bisa dilakukan oleh siswa yang benar-benar terdaftar di ekskul tersebut.

4. Implementasi Prisma Schema

Rancangan ERD di atas akan diterjemahkan langsung ke dalam file *schema.prisma* dalam proyek Next.js kami. Prisma akan secara otomatis meng-generate tabel-tabel PostgreSQL dan tipe data TypeScript berdasarkan definisi model ini, memastikan sinkronisasi sempurna antara desain database dan kode aplikasi. Penggunaan tipe data seperti UUID (cuid atau uuid di Prisma) untuk Primary Key direkomendasikan untuk keamanan dan skalabilitas sistem terdistribusi.