

BAB 1 PENDAHULUAN

1.1. Latar Belakang

Universitas Telkom merupakan perguruan tinggi yang berfokus pada Teknologi yang berlokasi di Bandung, Indonesia [1]. Salah satu visi Universitas Telkom yaitu menjadi universitas yang unggul dan berfokus pada pengembangan ilmu pengetahuan teknologi. Dalam upaya mencapai visi ini, mahasiswa maupun mahasiswi diharapkan tidak hanya berprestasi pada bidang akademik, namun juga memiliki sikap integritas dalam akademik. Tetapi pelanggaran demi pelanggaran pada lingkungan universitas masih sering mendapati perhatian yang kurang [2]. Pada banyak kasus terutama di lingkungan universitas, beberapa mahasiswa dengan status cumlaude masih memiliki catatan pelanggaran yang tidak teridentifikasi [3]. Kondisi ini dapat menciptakan situasi yang tidak adil terutama pada proses evaluasi akademik, hal ini terjadi karena pengakuan atas prestasi akademik tidak selalu mencerminkan integritas akademik yang sebenarnya. Dengan menyediakan sistem yang dapat mengelola aduan, keluhan ataupun protes dapat mencegah terjadinya masalah yang serupa [4].

Dalam pengembangan *website*, *backend* adalah bagian penting pada sebuah *website* dan *backend* sendiri memiliki fungsi dalam hal pengelolaan data, penyediaan API, pengolahan *server-side* dan logika fungsi dari *website* itu sendiri. *Backend* adalah suatu program yang beroperasi pada sisi server (*server-side*) memiliki tugas yang berhubungan langsung dengan basis data, sehingga interaksi tidak secara langsung terjadi kepada pengguna [5]. Kemudian dalam melakukan interaksi layanan *client* yang berbeda maka teknologi yang digunakan adalah *Application Programming Interface* (API). API adalah perangkat yang dapat mengintegrasikan pertukaran data pada berbagai aplikasi dalam waktu yang sama. Sehingga fitur yang sudah ada tidak akan dibuat kembali karena sudah tersedia pada API yang diakses [6]. Sedangkan *Representational State Transfer* (REST) merupakan arsitektur yang menggunakan HTTP untuk melakukan pertukaran data [7]. Bahasa

pemrograman yang cukup populer dan banyak digunakan saat ini yaitu *javascript* [8]. *Javascript* sendiri menggunakan sebuah platform yang bernama Node.JS dan memiliki *framework* Express.JS [9].

Penelitian ini bertujuan untuk pengembangan *backend* sebuah sistem pendataan pelanggaran akademik dengan mengimplementasikan REST API dalam pengintegrasian data, sehingga data yang diinputkan bisa mendeteksi pelanggaran akademik yang ada. Penelitian ini menghasilkan sistem informasi yang dapat berintegritas dengan sistem yang telah ada sebelumnya.

1.2. Perumusan Masalah

Perumusan masalah pada penelitian ini sebagai berikut:

- a) Bagaimana rancangan arsitektur REST API pada *website* yang akan dikembangkan?
- b) Bagaimana implementasi REST API yang akan dilakukan terhadap aplikasi yang akan dikembangkan?
- c) Bagaimana pengujian API pada *website* yang telah dikembangkan?

1.3. Tujuan

Mengacu pada rumusan masalah yang telah dijelaskan, beberapa tujuan dari penelitian ini sebagai berikut:

- a) Merancang arsitektur REST API pada *website* yang efektif dan efisien guna mendukung fungsionalitas *website* yang akan dikembangkan.
- b) Mengimplementasikan REST API sesuai dengan rancangan yang telah dibuat, sehingga memungkinkan komunikasi data antara server dan *website* dapat berjalan dengan lancar.
- c) Melakukan pengujian terhadap API yang telah dikembangkan, guna memastikan kualitas, keandalan, dan kinerja *website* sesuai dengan kebutuhan yang telah ditentukan.

1.4. Ruang Lingkup

Beberapa ruang lingkup dari penelitian ini sebagai berikut:

- a) Penelitian ini terbatas hanya pada pengembangan *backend* dan *database* serta aspek performa dari website.
- b) Bahasa pemrograman yang digunakan yaitu menggunakan *Javascript* dan *enviromtent* Node.JS
- c) Web framework yang digunakan untuk mengakomodasi Bahasan pemrograman Node.JS yaitu Express.JS.
- d) Pengujian dilakukan menggunakan aplikasi postman.
- e) Penelitian ini hanya berfokus pada penerapan REST API.
- f) Penelitian ini berfokus pada pengembangan website pendataan pelanggaran akademik mahasiswa pada fakultas informatika Universitas Telkom.

1.5. Metode Penelitian

Metode pada penelitian ini menggunakan node.js dan express.js sebagai framework, lalu menggunakan SDLC (Software Development Life Cycle) dengan pendekatan metode agile untuk pengembangan perangkat lunak dengan tahap sebagai berikut:

- a) Studi literatur. Dibutuhkan sebagai pendukung maupun teori yang melandasi penelitian ini.
- b) Analisis kebutuhan. Melakukan analisa terkait apa kebutuhan dan tujuan dari pengguna.
- c) Perancangan sistem. melakukan perancangan sistem yang akan di bangun.
- d) Impelementasi. Melakukan implementasi dari rancangan yang sudah dibuat.
- e) Pengujian sistem. menguji kembali apakah kinerja dan kualitas sistem yang sudah dikembangkan sudah sesuai dengan kebutuhan yang telah ditentukan.

1.6. Rencana Kegiatan

Rencana kegiatan yang akan dilakukan selama penelitian yaitu:

a) Studi Literatur

Mempelajari sumber informasi yang digunakan baik sebagai acuan maupun metode yang dibutuhkan sesuai dengan topik yang diambil.

b) Pengumpulan Data

Sebelum menentukan kebutuhan aplikasi, dilakukan pengumpulan data dan menentukan kebutuhan dari pengguna berupa wawancara.

c) Analisis Kebutuhan Aplikasi

Mengidentifikasi kebutuhan dan tujuan dari *website* yang akan dikembangkan sebagai acuan untuk sistem yang akan dikembangkan.

d) Pengembangan *Website*

Setelah melakukan perancangan kebutuhan maka pengerjaan dapat dimulai, seperti memilih *environment* yang akan digunakan, perangkat lunak yang dibutuhkan dan lain sebagainya.

e) Pengujian Sistem

Setelah sistem dikembangkan, selanjutnya yaitu melakukan testing terhadap sistem. Ini bertujuan untuk mengetahui terkait performa dan fungsionalitas dari sistem.

1.7. Jadwal Kegiatan

Tabel 1. 1 Rencana Kegiatan

Kegiatan	Bulan & Tahun					
	Feb 2025	Mar 2025	Apr 2025	Mai 2025	Jun 2025	jul 2025
Studi Literatur						
Pengumpulan Data						
Analisis Kebutuhan Aplikasi						
Perancangan REST API						
Implementasi						
Pengujian						

BAB 2 KAJIAN PUSTAKA

2.1. Penelitian Terdahulu

Adapun beberapa penelitian terdahulu yang menjadi landasan dalam penelitian ini. Referensi beberapa penelitian sebelumnya yang menjadi acuan disajikan dalam tabel 2.1.

Tabel 2. 1 Penelitian Terdahulu

No	Peneliti	Judul Jurnal	Ide Pokok	Relasi dengan Penelitian
1	Mubariz, A., Nur, D., Tungadi, E., Utomo, M. N. Y.	Perancangan <i>Back-End Server</i> Menggunakan Arsitektur Rest dan Platform Node.JS (Studi Kasus: Sistem Pendaftaran Ujian Masuk Politeknik Negeri Ujung Pandang)	Menggunakan Arsitektur Rest dan Platform Node.JS dalam perancangan Back-End.	Penelitian menjelaskan bagaimana penggunaan metode Arsitektur Rest dan Platform Node.JS dalam perancangan Back-End dan menggunakan <i>black box</i> untuk pengujian. Dimana arsitektur REST dan platform Node.JS saya gunakan dalam penelitian ini dan untuk pengujiannya juga menggunakan <i>black box testing</i> .
2	Prasetyawan, D dan Rahmanto, D. R.	Pengembangan Sistem Seleksi Proposal Penelitian Berbasis Web Service	Mengembangkan <i>web service</i> menggunakan REST API dan pengujian API menggunakan postman	Pada penelitian ini menggunakan REST API untuk pengembangan <i>web service</i> dengan pengujian API

No	Peneliti	Judul Jurnal	Ide Pokok	Relasi dengan Penelitian
		Menggunakan REST API		menggunakan Postman Dimana REST API berhubungan dengan penelitian saya dan pengujian API yang saya gunakan juga dengan postman.
3	Nasution	Implementasi Mongo Db, Express Js, React Js Dan Node Js (Mern) Pada Pengembangan Aplikasi Formulir, Kuis, Dan Survei Online	Pengembangan <i>frontend</i> menerapkan SPA (<i>Single Page Application</i>) sementara <i>backend</i> dengan implementasi Express.JS dan database menggunakan MongoDB serta untuk <i>supporting</i> ketiga teknologi itu adalah Node.JS.	Penelitian ini menggunakan Express.JS dan Node.JS juga Postman untuk pengujian API. Dimana penelitian saya juga mengimplementasikan Express.JS dan Node.JS serta menggunakan Postman untuk pengujian API.
4	Sutara, B. dan Gunawan, S.	Analisis Perbandingan Performa REST API Antara Framework Express.Js Dengan Hapi.Js Menggunakan Apache Jmeter	Perbandingan performa REST API yang dibangun dengan antara dua framework yaitu Express.JS dan Hapi.JS	Penelitian ini berhubungan dengan penelitian saya karena membangun REST API menggunakan framework Express.JS.

No	Peneliti	Judul Jurnal	Ide Pokok	Relasi dengan Penelitian
5	Riady, A. M. N., Paniran, P., dan Suksmadana, I. M. B.	Perancangan <i>Backend</i> API Berbasis REST-API pada Aplikasi Rekomendasi Resep Makanan	Merancang <i>backend</i> dengan REST API menggunakan Node.JS Express dan pengujian dengan <i>black box testing</i> .	Penelitian ini berhubungan dengan penelitian saya karena mengembangkan <i>backend</i> berbasis REST API menggunakan Node.JS Express serta menggunakan <i>black box testing</i> untuk pengujian.
6	Rusadi, R. F.	Pengembangan <i>Back-End</i> Berbasis REST API Pada Aplikasi Kado Buket	Mengembangkan <i>web service</i> dengan arsitektur REST dan API dengan Node.JS Express dan <i>database</i> sebagai <i>endpoints</i> serta menggunakan <i>black box testing</i> untuk pengujian.	Penelitian ini berhubungan dengan penelitian saya karena mengembangkan <i>backend</i> berbasis REST API menggunakan Node.JS Express serta menggunakan <i>black box testing</i> untuk pengujian.
7.	Khairi, A., Pratama, W. K. T., Iawan, J., Efendi, A. T. A. U., dkk.	Sistem Informasi Berbasis Web Pada Pelanggaran Santri Di Pondok Pesantren Nurul Jadid	Pengembangan website pelanggaran santri di pondok pesantren dengan menggunakan metode waterfall	Penelitian ini berhubungan dengan penelitian saya karena terdapat kemiripan yang mendekati dari segi topik yaitu pelanggaran akademik para santri di pesantren nurul jadid

2.2. Node.JS

Node.JS atau yang disebut juga sebagai *runtime environment* merupakan sistem perangkat lunak yang didesain untuk mengembangkan aplikasi berbasis website. Aplikasi ini ditulis dalam campuran bahasa JavaScript dan bahasa C++. Tidak seperti kebanyakan bahasa JavaScript yang dijalankan pada web browser, Node.JS dijalankan sebagai aplikasi server. Node.JS mampu berjalan di server karena mempunyai dukungan dari V8 Engine yang dibuat oleh Google dan beberapa modul bawaan yang terintegrasi seperti modul HTTP, modul filesystem, modul keamanan dan beberapa modul penting lainnya [10].

2.3. REST API

REST (*Representational State Transfer*) atau RESTfull merupakan sebuah arsitektur *web service* yang paling populer saat ini karena ringan dan memiliki kemampuan beradaptasi ke berbagai aplikasi web. REST merupakan arsitektur yang digunakan untuk merancang *service* yang dapat digunakan oleh berbagai platform untuk mendukung interoperabilitas. REST API dapat juga disebut API (*Application Program Interface*) karena bertugas sebagai jembatan antar aplikasi untuk saling berkomunikasi. REST API dapat dipanggil dengan sebuah alamat yang dikenal dengan URI (*Uniform Resource Identifier*). API ini umumnya dapat diakses melalui HTTP (*Hypertext Transfer Protocol*) dengan menyertakan standar yang telah ditentukan seperti GET, POST, PUT, dan DELETE [11]. REST API (*Representational State Transfer Application Programming Interface*) adalah sebuah arsitektur perangkat lunak yang menggunakan beberapa aturan dalam pembuatan *website*. REST API didasari prinsip-prinsip arsitektur REST seperti kesederhanaan, skalabilitas, dan sifat tanpa status (*stateless*). REST API dirancang untuk menjembatani komunikasi antara sebuah sistem kepada sistem lainnya melalui internet. API memungkinkan pengembang atau *developer* untuk mengakses dan merancang *service* menggunakan metode HTTP standar seperti GET, POST, PUT, dan DELETE. REST API juga dapat digunakan dengan bahasa

pemrograman apa pun yang mendukung protokol HTTP. Penggunaan REST API sangat berguna dalam mendukung pengembangan sistem pihak ketiga dan untuk melakukan akses data maupun *service* yang tersedia. API menyediakan *service* yang terdefinisi dengan baik untuk mengakses dan merancang sebuah sistem, sehingga memungkinkan *developer* untuk mengakses dan mengembangkan *website* yang memanfaatkan data atau *service* dari organisasi lain. hal ini mendorong kolaborasi dan memungkinkan terciptanya *website* yang inovatif dengan menggabungkan kemampuan dari berbagai *service* yang ada [12].

2.4. Database

Database adalah kumpulan informasi yang disimpan secara sistematis di dalam komputer sehingga dapat dikendalikan oleh program komputer untuk mengambil informasi dari *database*. istilah “basis data” berasal dari ilmu komputer. Artikel ini adalah tentang database komputer, meskipun pentingnya kemudian diperluas untuk memasukkan hal-hal selain elektronik. Catatan seperti *database* ada sebelum revolusi industri dalam bentuk buku, kuitansi, dan kumpulan data bisnis [13].

2.5. Express.JS

Express.JS merupakan sebuah *framework* berbasis website untuk Node.JS yang ditulis menggunakan bahasa pemrograman JavaScript. *Framework* ini bersifat *open source* dan dibuat oleh TJ Holowaychuk pada tahun 2010. Express.JS merupakan sebuah *framework backend* yang dimana artinya framework ini bertanggungjawab untuk mengatur fungsionalitas dari *website*, contohnya seperti pengelolaan *routing* dan *session*, HTTP *request*, *error handling*, serta pertukaran data di *server*. *Framework* ini merupakan *framework* yang tangguh dan efisien serta memungkinkan *developer* untuk dengan cepat membangun aplikasi berbasis *website* dan API yang kuat dan dapat diandalkan [14].

2.6. Black Box Testing

Black box Testing merupakan sebuah pengujian kualitas *software* yang bertitik pada fungsionalitas *software*. Pengujian ini bertujuan untuk menemukan fungsi yang tidak benar, kesalahan *interface*, kesalahan pada struktur data, kesalahan *performance software* maupun kesalahan inisialisasi dan terminasi [15]. Dalam pengujian *black-box* biasanya menggunakan *machine learning* dari *black-box* itu sendiri untuk menentukan apakah ratusan fungsi bekerja dengan baik atau tidak, namun jika pengujian gagal atau mengalami masalah maka penting untuk *developer* memahami alasan dari masalah tersebut [16]. Pengujian *black-box* pada umumnya mendeteksi kesalahan dengan baik karena pengujian dilakukan hanya menggunakan spesifikasi dari API, namun alat atau tools ini masih memiliki keterbatasan yang cukup besar dalam beberapa jenis pengujian seperti validasi respon, alur dalam beberapa *endpoint*, pengujian latency, dan identifikasi keamanan karena tools ini hanya akan mendeteksi kode tanpa melakukan validasi yang benar dalam beberapa kondisi [17].

2.7. Postman




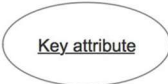
Postman merupakan aplikasi atau alat berupa plugin untuk *browser chrome*, dimana aplikasi ini berfungsi sebagai alat uji coba untuk REST API yang telah dibuat. Fungsi utama dari Postman sebagai GUI API *caller* namun sekarang Postman juga menyediakan berbagai macam fitur lain seperti *Sharing Collection API for Documentation, Testing API, Realtime Collaboration Team*, dan lainnya [10].

2.8. Boundary Value Analysis (BVA)

Boundary Value Analysis (BVA) adalah metode pengujian nilai batas dari masukan sistem dengan tujuan menentukan jumlah maksimum dan minimum digit yang diuji. Prinsip dasar dari *Boundary Value Analysis* meliputi identifikasi kesalahan input yang mungkin terjadi.

2.9. ERD (*Entity Relationship Diagram*)

ERD (*Entity Relationship Diagram*) adalah sebuah diagram yang bertujuan untuk merepresentasikan struktur hubungan antar entitas dengan menggambarkan bagaimana data akan disusun dan dihubungkan satu sama lain.






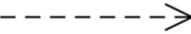

Simbol	Nama Simbol	Keterangan
Entities		
	Entity	Suatu entity digambarkan sebagai sebuah persegi panjang memiliki nama entity tersebut.
	Weak Entity	Suatu entity yang tidak dapat diidentifikasi melalui at dengan sendirinya. Keberadaan <i>weak entity</i> bergantung l entity lain yang disebut <i>owner entity</i> .
	Associative Entity	Entity yang digunakan pada many-to-many relationship (antar banyak).
Attributes		
	Attribute	Dalam notasi Chen, Sebuah atribut digambarkan sebago oval yang memuat nama atribut tersebut.
	Key attribute	Suatu atribut yang mengidentifikasi suatu entity dengan spesifik atau unik. Nama dalam Key Attribute selalu di-u
	Multivalued attribute	Attribute yang dapat memuat lebih dari satu nilai (Multiv Multivalued Attribute digambarkan dengan dua oval.
	Derived attribute	Suatu attribute di mana nilainya dihitung atau berdasar c lain. Derived attribute mungkin atau tidak dapat disimpa database. Attribute ini digambarkan dengan oval putus-

Gambar 2. 1 Simbol ERD

Pada gambar 2.1 disajikan gambar simbol ERD yang menunjukkan fungsi dari masing-masing simbol yang digunakan pada ERD, tiap bentuk simbol memiliki keterangan untuk menggambarkan bagaimana data terhubung satu sama lain.

2.10. Class Diagram

Class diagram sendiri adalah sebuah diagram yang merepresentasikan relasi antar objek pada sistem dengan menggunakan model class sebagai objek. Gambar 3.4 menjelaskan simbol *class diagram* yang akan digunakan.

Simbol	Keterangan
<p><i>Interface</i></p> 	Digunakan sebagai sarana untuk memberikan identitas atau menyatakan ide dalam proses pengembangan program.
Simbol	Keterangan
<p><i>Class</i></p> 	Merupakan struktur dasar yang mendefinisikan sistem.
<p><i>Generalization</i></p> 	Menggambarkan relasi antara kelas-kelas, baik yang bersifat umum maupun spesifik.
<p><i>Association</i></p> 	Berfungsi untuk penghubung di antara kelas tersebut di dalam sistem.
<p><i>Directed Association</i></p> 	Menyatukan koneksi antara hubungan kelas dan kelas lainnya.
<p><i>Dependency</i></p> 	Menunjukkan saling ketergantungan antar kelas.
<p><i>Aggregation</i></p> 	Memperlihatkan interaksi antar kelas bersama seluruh aspek dalam sistem.

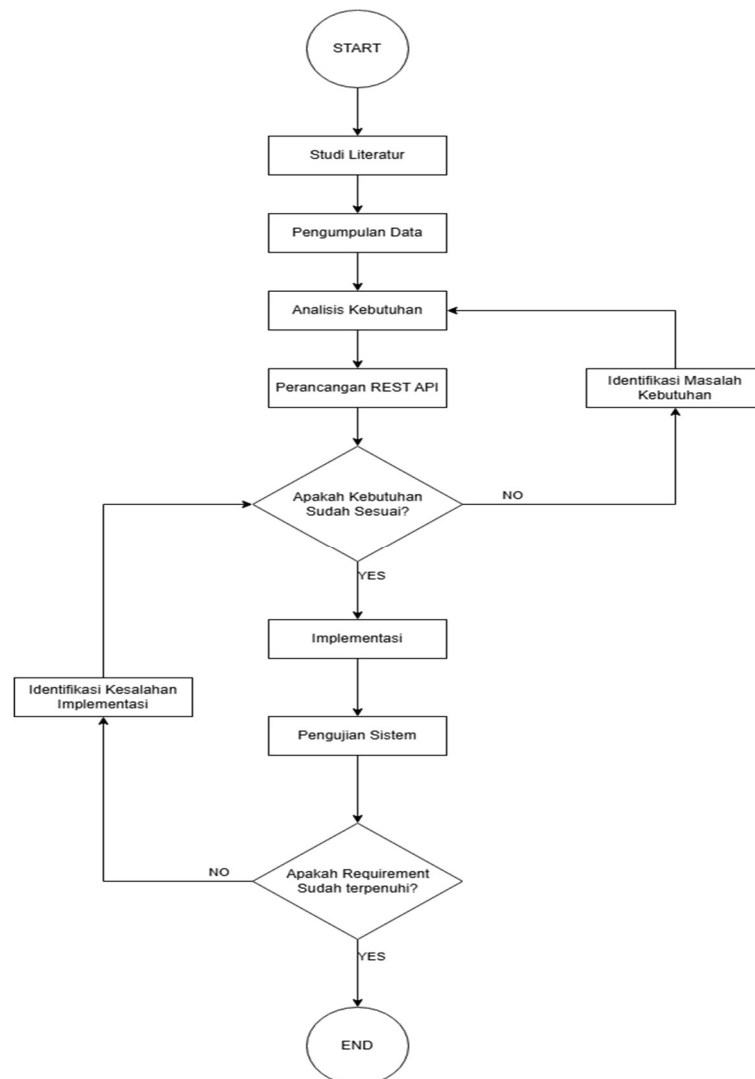
Gambar 2. 2 Simbol *Class Diagram*

Pada gambar 3.4 disajikan gambar dengan simbol yang menjelaskan fungsi dari simbol-simbol tersebut, symbol ini memiliki fungsi untuk

memberikan Gambaran terkait dengan relasi antar objek dengan menggunakan *class* sebagai model.

BAB 3 PERANCANGAN SISTEM

Penelitian ini menggunakan metode yang berdasarkan pada siklus hidup pengembangan perangkat lunak (*Software Development Life Cycle*) dengan pendekatan model *agile* yang memiliki tahap studi literatur, analisis kebutuhan, perancangan sistem, implementasi, pengujian, dan evaluasi [18]. SDLC dibutuhkan agar kualitas dari perangkat lunak bisa meningkat secara keseluruhan. Tahap penelitian bisa dilihat pada Gambar 3.1.



Gambar 3. 1 Tahapan penelitian

Dari Gambar 3.1 bisa disimpulkan bahwa alur penelitian ini dimulai dari tahap studi literatur sebagai pendukung maupun teori yang melandasi

penelitian ini, lalu analisis kebutuhan guna mengetahui apa kebutuhan dan tujuan dari pengguna, selanjutnya melakukan perancangan sistem yang akan di bangun, lalu melakukan implementasi dari rancangan yang sudah dibuat tadi, selanjutnya pengujian untuk menguji kembali apakah kinerja dan kualitas sistem yang sudah dikembangkan sudah sesuai dengan kebutuhan yang telah ditentukan, dan yang terakhir adalah hasil, pada gambar disajikan bahwa jika tahap pengujian sudah sesuai maka tahap selanjutnya adalah hasil, namun jika pada tahap tertentu belum sesuai maka perubahan bisa langsung dilakukan pada tahap tersebut.

3.1. Studi Literatur

Studi literatur dilakukan untuk melandasi teori yang dapat mendukung berjalannya penelitian ini dengan cara mengumpulkan referensi yang relevan seperti jurnal, artikel maupun buku. Studi literatur juga menjadi sumber pendukung dalam pengembangan backend dan database sesuai dengan tujuan yang diinginkan [11].

3.2. Analisis Kebutuhan

Aplikasi Pendataan Pelanggaran Akademik memiliki tujuan untuk membantu mengelola laporan pelanggaran dari mahasiswa. Analisis kebutuhan bertujuan untuk memenuhi kebutuhan yang ingin dicapai sesuai dengan *requirement*. Beberapa hal yang dianalisa yaitu:

1. Memahami kebutuhan aplikasi

Dalam proses memahami kebutuhan *website*, maka dilakukan diskusi kepada *stakeholder*. Ini dilakukan agar dapat mengetahui dan mengumpulkan data yang bisa digunakan untuk menentukan spesifikasi dari kebutuhan *website*.

2. Membuat spesifikasi kebutuhan (*requirement*)

Pada tahap penentuan spesifikasi kebutuhan *website* dibagi menjadi dua, yaitu *functional requirement* dan *non-functional*

requirement. Pada bagian *backend*, data yang dibutuhkan hanya *functional requirement* sebagai acuan untuk pengembangan.

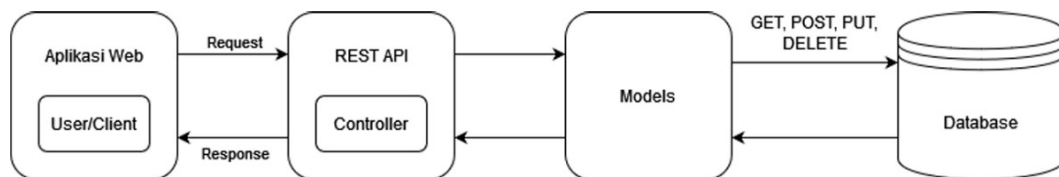
3. Pembuatan rancangan arsitektur sistem

Setelah mendapatkan kebutuhan dari pengguna, selanjutnya, dilakukan perancangan arsitektur website yang akan dibangun. Rancangan yang dibuat yaitu menggunakan *design pattern* MVC (*Model-View Controller*) dan struktur *class diagram*.

MVC merupakan arsitektur yang secara otomatis dapat mengelola kode dan membantu programmer dalam mengembangkan *website* dengan baik [19].

MVC merupakan sebuah design pattern yang digunakan untuk mengatur dan menyusun kode program. MVC digunakan untuk memodelkan pola berjalannya sistem. MVC membagi sistem kedalam tiga bagian [14], yaitu:

- *Model* adalah objek yang mewakili bagian akses data pada *database*
- *View* sebagai bentuk dari visualisasi model *interface* untuk pengguna
- *Controller* merupakan bagian penghubung dari *view* dan model dalam melakukan proses komunikasi data.



Gambar 3. 2 Model MVC pada REST API

4. Menyiapkan lingkungan pengembangan

Tahap ini *developer* menyiapkan kebutuhan untuk kegiatan pengembangan aplikasi. Hal yang dibutuhkan seperti *environment* yang akan digunakan, *framework*, *software*, dan lain sebagainya.

5. Memulai pengerjaan *project*

Setelah analisis-analisis sebelumnya selesai dilakukan, maka proses pengerjaan *project* dapat dikerjakan sesuai dengan spesifikasi teknis yang telah dibangun.

3.3. Perancangan REST API

Dalam melakukan perancangan REST API dilakukan berdasarkan konsep arsitektur MVC dan class diagram yang sudah dibuat sebelumnya. *Class* diagram berguna untuk memodelkan bagaimana REST API berjalan. *Class* diagram berfungsi untuk mendeskripsikan objek-objek yang diperlukan dalam aplikasi beserta data yang direpresentasikan oleh setiap objek, serta menjelaskan hubungan dan interaksi antar class selama proses *website* berjalan.

3.4. Implementasi

Setelah REST API dirancang, dilakukan implementasi REST API menggunakan Bahasa pemrograman Node.JS dengan menggunakan *framework* Express.JS. sesuai dengan *class diagram* yang telah dibuat.

Dalam penggunaan *framework* Exspress.js, dilakukan beberapa tahap yaitu:

- a) Instalasi Node.JS untuk menjalankan JavaScript di server lalu gunakan perintah “npm init” untuk menginisialisasi agar *project* memiliki metadata dan depedensi.
- b) Instalasi Express.JS untuk membuat kerangka dasar REST API.
- c) Jalankan *server* dengan perintah “node server.js” untuk menjalankan *server* secara langsung.

3.5. Pengujian Sistem

Pengujian sistem pada penelitian ini dilakukan dengan menggunakan metode *black box testing*. Pengujian ini dapat dilakukan tanpa harus mengetahui detail dari internal sistem. Pengujian dilakukan dengan menggunakan aplikasi postman. Tujuan dari pengujian adalah memastikan kembali REST API sudah berjalan sesuai dengan keinginan, pengujian ini

menggunakan teknik BVA karena teknik ini sangat efektif untuk mengidentifikasi cacat pada nilai ekstrem, yaitu pada nilai yang mendekati atau tepat di batas maksimum dan minimum dari rentang input, sehingga memastikan aplikasi berfungsi dengan baik dalam kondisi ekstrem. Langkah-langkah dalam melakukan pengujian ini melibatkan penetapan nilai batas atas dan bawah untuk setiap kolom melalui prosedur yang telah ditentukan, serta merancang kasus uji yang akan digunakan dalam pengujian [12]. Pengujian ini mencakup persiapan skenario uji untuk kondisi batas, pengujian respons aplikasi, dan evaluasi hasil untuk memastikan aplikasi bekerja dengan baik dalam berbagai kondisi input. Hal ini memungkinkan pengujian yang menyeluruh dan dapat diandalkan, membantu mendeteksi dan memperbaiki potensi masalah sejak tahap awal pengembangan.

Fokus utama dari pengujian ini terletak pada fitur penambahan kasus yang merupakan komponen penting pada aplikasi ini. Metode ini diharapkan bisa secara efektif menguji dan memastikan bahwa API sudah berjalan sesuai dengan tujuannya, Adapun tahapan dalam *black box testing* meliputi:

- A. Membuat test case pengujian berbagai fungsi pada API
- B. Mengembangkan *test case* yang sesuai dengan kebutuhan dan permintaan pengguna.
- C. Mengidentifikasi kesalahan dan bug berdasarkan fungsi dari API yang sudah dikembangkan sebelumnya.

Dalam melakukan pengujian dengan menggunakan Teknik boundary value analysis memiliki beberapa langkah pengujian yang dirancang untuk memastikan kembali aplikasi sudah diuji secara menyeluruh dan akurat [21]. Dalam proses ini dimulai dari mengidentifikasi masalah yang akan diuji untuk menentukan bagian mana yang membutuhkan perhatian khusus terkait batas nilai yang akan diuji. Setelah teridentifikasi, langkah berikutnya adalah mendefinisikan *test case*, di mana setiap elemen yang akan diuji didefinisikan secara rinci untuk memastikan cakupan pengujian yang lengkap.

Selanjutnya, elemen-elemen *test case* yang telah didefinisikan dimasukkan ke dalam aplikasi, dan pengujian dilakukan menggunakan teknik *Boundary Value Analysis* untuk memastikan aplikasi berfungsi dengan benar di sekitar batas nilai yang ditentukan. Tahap terakhir dari langkah ini adalah melakukan dokumentasi dari hasil pengujian dan menarik Kesimpulan, dokumentasi ini mencatat temuan selama pengujian dan memberikan dasar bagi rekomendasi perbaikan. Kesimpulan yang ditarik membantu mengarahkan langkah-langkah berikutnya dalam pengembangan aplikasi. Alur penelitian bisa dilihat seperti gambar 3.5 berikut.



Gambar 3. 3 Alur Pengujian

3.6. Analisis Penggunaan Sistem

Berdasarkan hasil wawancara dan analisis kebutuhan, sistem memiliki tiga kategori pengguna utama dengan peran dan hak akses yang berbeda:

3.6.1. Staff Akademik

Staf Akademik merupakan pengguna sistem yang berperan dalam memantau dan melaporkan data pelanggaran akademik mahasiswa. Hak akses yang diberikan bersifat read-only, sehingga hanya dapat melihat data tanpa dapat menambah, mengubah, atau menghapus informasi. Alur kerja dimulai dari login, melihat daftar pelanggaran, hingga menghasilkan laporan yang hanya dapat dibaca. Staf Akademik membutuhkan akses untuk melihat daftar dan riwayat pelanggaran mahasiswa guna memahami kondisi akademik serta memberikan konseling yang tepat, tanpa mengganggu integritas data yang ada.

3.6.2. Staff Kemahasiswaan

Staf Kemahasiswaan adalah pengguna dengan akses penuh untuk mengelola data pelanggaran dan pembinaan mahasiswa. Mereka dapat melihat, menambah, mengubah, dan menghapus data sesuai kebutuhan. Alur kerja mencakup input pelanggaran, pembaruan status, pengeditan informasi, dan penghapusan jika diperlukan. Peran ini memungkinkan Staf Kemahasiswaan mendokumentasikan setiap kasus, memperbaiki detail dan status pelanggaran, serta menjaga akurasi data demi mendukung proses pembinaan yang tepat.

3.6.3. Staff Wakil Dekan

Wakil Dekan berperan sebagai supervisor sistem dengan tanggung jawab dalam pengambilan keputusan strategis. Pengguna ini memiliki akses penuh termasuk manajemen akun dan pengaturan hak akses. Alur kerjanya mencakup pemantauan dashboard, pengelolaan pengguna, persetujuan keputusan penting, serta pembuatan kebijakan. Wakil Dekan perlu mengelola

akun pengguna untuk memastikan kontrol akses yang tepat, memantau keseluruhan kasus melalui dashboard komprehensif, mengakses seluruh fitur sistem untuk supervisi menyeluruh, serta memberikan persetujuan atas keputusan penting sebagai bentuk kontrol kualitas.

3.7. Perancangan API Endpoint

Perancangan endpoint API bertujuan untuk menyediakan antarmuka komunikasi antara klien dan server. Dalam sistem pendataan pelanggaran akademik mahasiswa ini, API dirancang menggunakan arsitektur RESTful, sehingga setiap resource dapat diakses melalui metode HTTP yang sesuai seperti GET, POST, PUT, dan DELETE. Berikut adalah perancangan endpoint yang digunakan dalam sistem:

3.7.1. Authentication Endpoint

Endpoint autentikasi digunakan untuk menangani proses otentikasi pengguna, termasuk login, logout, dan mendapatkan informasi pengguna yang sedang login. Endpoint ini memastikan bahwa hanya pengguna yang sah yang dapat mengakses sistem. Tabel 3.1 menampilkan daftar endpoint yang digunakan untuk proses autentikasi.

Table 3.1 Daftar Endpoint Authentication

No	Endpoint	Method	Deskripsi
1	/api/auth/login	POST	Proses login pengguna dengan mencocokkan kredensial (username & password)
2	/api/auth/logout	POST	Mengakhiri sesi pengguna yang sedang aktif
3	/api/auth/me	GET	Mendapatkan informasi pengguna berdasarkan token yang dikirim

3.7.2. User Management Endpoint

Endpoint ini digunakan untuk pengelolaan data pengguna, yang mencakup pengambilan daftar pengguna, melihat detail pengguna berdasarkan ID, serta melakukan pembaruan data profil pengguna. Tabel 3.2 berikut menyajikan daftar endpoint yang digunakan dalam pengelolaan pengguna.

Table 3.2 Daftar Endpoint User Management

No	Endpoint	Method	Deskripsi
1	/api/users	GET	Mendapatkan daftar seluruh pengguna
2	/api/users/:id	GET	Mendapatkan detail informasi pengguna berdasarkan ID
3	/api/users/:id	PUT	Memperbarui data profil pengguna berdasarkan ID
4	/api/users/add	POST	Mendaftarkan staff sebagai pengguna website
5	/api/users/:id	DELETE	Menghapus data staff sebagai pengguna

3.7.3. Pelanggaran Management Endpoint

Endpoint pelanggaran digunakan untuk mengelola data pelanggaran akademik mahasiswa. Data yang dapat diakses melalui endpoint ini meliputi daftar pelanggaran, detail berdasarkan ID, hingga pencarian berdasarkan NIM mahasiswa. Tabel 3.3 menampilkan endpoint yang digunakan dalam pengelolaan pelanggaran.

Table 3.3 Daftar Endpoint Pelanggaran Management

No	Endpoint	Method	Deskripsi
1	/api/violations	GET	Mendapatkan daftar seluruh pelanggaran

No	Endpoint	Method	Deskripsi
2	/api/violations	POST	Menambahkan data pelanggaran baru
3	/api/violations/:id	GET	Mendapatkan detail pelanggaran berdasarkan ID
4	/api/violations/:id	PUT	Memperbarui data pelanggaran berdasarkan ID
5	/api/violations/:id	DELETE	Menghapus data pelanggaran berdasarkan ID
6	/api/violations/student/:nim	GET	Mendapatkan daftar pelanggaran berdasarkan NIM mahasiswa

3.7.4. Pelaporan Endpoint

Endpoint pelaporan menyediakan informasi dalam bentuk ringkasan dan tren pelanggaran yang terjadi, serta fitur untuk mengeksport data ke dalam bentuk dokumen yang dapat diunduh. Tabel 3.4 berikut adalah daftar endpoint yang digunakan untuk pelaporan.

Table 3.4 Daftar Endpoint Reporting

No	Endpoint	Method	Deskripsi
1	/api/reports/summary	GET	Menampilkan ringkasan pelanggaran yang telah terjadi
2	/api/reports/trends	GET	Menampilkan tren data pelanggaran berdasarkan waktu tertentu
3	/api/reports/export	POST	Mengeksport data laporan pelanggaran ke dalam format tertentu seperti PDF

BAB 4 IMPLEMENTASI DAN PENGUJIAN

Pada bab ini akan dibahas mengenai sistem pendataan pelanggaran akademik mulai dari tahap pembuatan hingga tahap implementasi ke bentuk aplikasi. Setelah sistem berjalan maka akan dilakukan tahap evaluasi dengan menggunakan Teknik pengujian black box testing agar aplikasi sudah sesuai dengan keinginan.

4.1. Studi Literatur

Studi Literatur merupakan pengumpulan data dengan cara mengumpulkan berbagai referensi atau literatur berdasarkan hasil penelitian yang berkaitan dengan pengembangan aplikasi ini. Literatur yang dikumpulkan digunakan sebagai landasan untuk merancang data, studi kasus, penggunaan metode pengembangan, dan penggunaan metode evaluasi.

4.2. Pengumpulan Data

Pengumpulan data dilakukan untuk mengetahui kebutuhan dan permasalahan yang dihadapi pengumpulan data dilakukan dengan wawancara dari calon pengguna. Hal ini digunakan untuk merumuskan kebutuhan sistem, baik fungsional maupun non-fungsional, yang menjadi dasar dalam perancangan dan pengembangan aplikasi, Adapun hasil dari wawancara yang telah dilakukan dapat dilihat pada Table 4.1 berikut.

Tabel 4.1 Hasil Wawancara

Jabatan	Masalah yang Dihadapi	Kebutuhan / Harapan	Fitur yang Diinginkan
Staf Akademik	- Data pelanggaran masih	- Sistem pencatatan digital	- Input pelanggaran Riwayat per mahasiswa

Jabatan	Masalah yang Dihadapi	Kebutuhan / Harapan	Fitur yang Diinginkan
	manual (Excel) - Sulit mencari riwayat pelanggaran mahasiswa	- Data mudah dicari dan terdokumentasi	- Export laporan pelanggaran
Staf Kemahasiswaan	- Tidak tahu riwayat pelanggaran mahasiswa - Sulit dalam proses pembinaan	- Akses cepat ke data pelanggaran - Dokumentasi pembinaan lebih lengkap dan rapi	- Akses data pelanggaran - Notifikasi pelanggaran aktif - Kronologi penanganan
Wakil Dekan	- Data pelanggaran tidak terintegrasi - Sulit melihat tren atau membuat laporan	- Data real-time untuk kebijakan dan evaluasi	- Grafik tren pelanggaran - Dashboard monitoring - Validasi dan manajemen pelanggaran

4.3. Analisis Kebutuhan

Analisis kebutuhan melibatkan serangkaian proses untuk menganalisis kebutuhan pengguna dengan mengidentifikasi berbagai elemen penting. Proses ini meliputi identifikasi aktor, *functional requirement*, *constraints*, *use case diagram*, *activity diagram* dan *class diagram*.

4.3.1. Identifikasi Aktor

Pada sub-bab ini bertujuan untuk mengidentifikasi pihak yang akan menggunakan sistem yang sedang dikembangkan. Dalam pengembangan sistem ini, aktor atau pengguna yang terlibat adalah staff yang akan berperan sebagai wakil dekan, akademik, dan kemahasiswaan yang mengelola Web Pendataan Pelanggaran Akademik.

4.3.2. Functional Requirement

Functional Requirement merupakan layanan yang harus disediakan oleh sistem agar dapat memenuhi kebutuhan pengguna. Dalam penelitian ini, analisis terhadap functional requirement dijelaskan pada Tabel 4.2 berikut.

Tabel 4.2 Functional Requirement

No	Functional Requirement
FR01	Sistem memiliki fungsi login dari user class wakil dekan, akademik maupun kemahasiswaan
FR02	Sistem memiliki fungsi untuk mengelola data pelanggaran baik dengan menambahkan pelanggaran, mengubah dan menghapus data pelanggaran
FR03	Sistem memiliki fungsi untuk membantu staff dalam mengelola pendataan pelanggaran sehingga dapat melakukan pelacakan status kasus yang ada
FR04	Sistem memiliki fungsi untuk membantu staff dalam menambahkan kasus pelanggaran mahasiswa, baik dengan melakukan input nama, nim, kelas, program studi, dan input kasus
FR05	Sistem memiliki fungsi untuk membantu staff dalam menambahkan informasi kedalam kasus pelanggaran yang ada, baik dengan menambahkan informasi dari kasus yang sudah ada, menambah informasi jadwal sidang, dan pemindahan kasus dari tahap investigasi ke tahap sidang

No	Functional Requirement
FR06	Sistem memiliki fungsi untuk membantu staff dalam melakukan pembukaan sidang, menambahkan notulensi sidang, membuat draft Keputusan sidang, membuat hasil Keputusan sidang dan melakukan penutupan kasus
FR07	Sistem memiliki fungsi untuk membantu staff dalam melacak status kasus, melakukan update status kasus, memberikan notifikasi terkait perubahan kasus, dan memberhentikan pelacakan kasus

4.3.3. Constraints

Constraints adalah batasan atau pembatasan yang harus diperhitungkan dalam pengembangan dan implementasi sistem. Batasan ini dapat berasal dari keterbatasan teknologi, regulasi, anggaran, atau lingkungan operasional yang memengaruhi cara sistem dirancang dan diimplementasikan. Dalam penelitian ini, analisis terhadap constraints dijelaskan pada Tabel 4.3 berikut.

Tabel 4.3 Constraints

No	Batasan Sistem
C01	Backend dikembangkan hanya menggunakan Node.js dan framework Express.js. Tidak diperkenankan framework lain.
C02	Penggunaan database dibatasi hanya pada sistem berbasis SQL (MySQL/MariaDB) sesuai dengan rancangan awal.
C03	Sistem dikembangkan menggunakan pendekatan Agile dan tidak mengikuti metode SDLC lain seperti Waterfall.
C04	Pengembangan sistem dibatasi hingga maksimal Juli 2025 sesuai jadwal rencana kegiatan.
C05	Sistem dibatasi hanya untuk 3 tipe pengguna: Staf Akademik, Staf Kemahasiswaan, dan Wakil Dekan.

4.3.4. Use Case Diagram

Use Case Diagram merupakan representasi visual dalam bentuk diagram yang menggunakan notasi UML (Unified Modeling Language) untuk menggambarkan interaksi antara aktor atau pengguna dengan sistem yang dikembangkan. Tujuan dari use case diagram adalah untuk memberikan gambaran awal mengenai *functional requirement* yang harus dipenuhi oleh sistem.

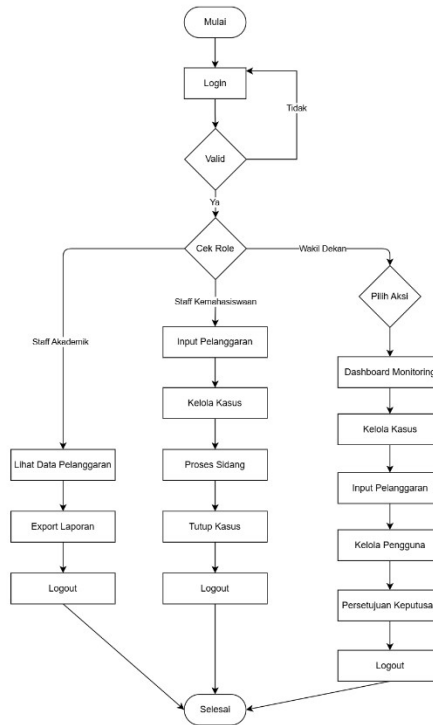


Gambar 4.1 Use Case Diagram Sistem SiPPAK

Berdasarkan Gambar 4.1, dalam konteks sistem informasi ini terdapat tiga aktor utama yang berinteraksi dengan sistem, yaitu Akademik, Kemahasiswaan, dan Wakil Dekan. Setiap aktor memiliki hak akses dan fungsi yang berbeda sesuai dengan perannya dalam sistem. Aktor Akademik dapat mengakses menu profil, edit profil, lihat status case, dan buat case. Aktor Kemahasiswaan memiliki akses ke fungsi buat case, edit case information, close case, tutup sidang etik, dan buka sidang etik. Sedangkan Wakil Dekan dapat mengakses dashboard monitoring, edit user, akses menu admin, add new user, dan edit case log.

4.3.5. Activity Diagram

Activity Diagram berfungsi untuk menggambarkan alur interaksi antara aktor dengan sistem secara detail dan berurutan. Diagram ini menunjukkan langkah-langkah aktivitas yang dilakukan dalam setiap proses bisnis, termasuk kondisi percabangan dan pengambilan keputusan yang terjadi selama interaksi berlangsung.

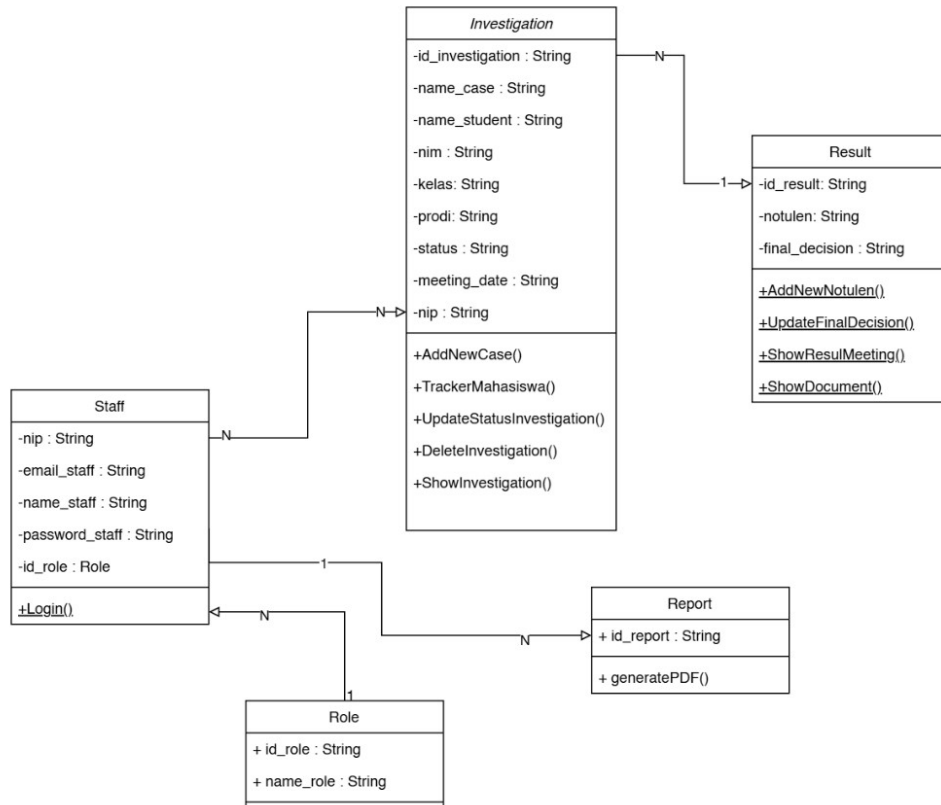


Gambar 4.2 Activity Diagram Sistem SiPPAK

Sebagaimana ditunjukkan pada Gambar 4.2, *activity diagram* sistem dimulai dengan proses login dimana sistem akan memvalidasi kredensial pengguna. Jika login tidak valid, sistem akan kembali ke halaman login. Setelah berhasil login, sistem akan melakukan pengecekan role pengguna untuk menentukan hak akses yang sesuai. Untuk Staff Kemahasiswaan, alur dilanjutkan dengan input pelanggaran, kelola kasus, proses sidang, hingga tutup kasus. Staff Akademik dapat mengakses fitur lihat data pelanggaran dan export laporan. Sedangkan Wakil Dekan memiliki akses ke dashboard monitoring, kelola kasus, input pelanggaran, kelola pengguna, dan persetujuan keputusan.

4.3.6. Class Diagram

Class Diagram digunakan untuk menggambarkan struktur statis sistem, menunjukkan kelas-kelas yang ada dalam sistem beserta atribut, metode, dan hubungan antar kelas. *Class Diagram* ini dirancang berdasarkan analisis dari *use case scenario* dan *activity diagram* yang telah dibuat sebelumnya.



Gambar 4.3 Class Diagram Sistem SiPPAK

Berdasarkan Gambar 4.3, diagram ini memberikan pandangan yang lebih mendalam tentang struktur internal sistem, menjelaskan bagaimana berbagai komponen sistem saling terkait dan berinteraksi satu sama lain. *Class diagram* mencakup entitas-entitas utama seperti Staff, Investigation, Result, Role, dan Report, beserta dengan atribut dan method yang dimiliki oleh masing-masing kelas.

Kelas Staff memiliki atribut seperti nip, kode_dosen, email_dosen, name_dosen, password_dosen, dan id_role, serta method Login(). Kelas Investigation berisi atribut lengkap untuk pengelolaan kasus investigasi

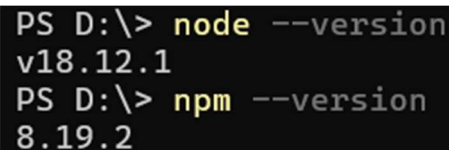
seperti `id_investigation`, `name_case`, `name_student`, `nim`, `kelas`, `prodi`, `status`, `meeting_date`, dan `nip`, dengan berbagai method seperti `AddNewCase()`, `TrackerMahasiswa()`, `UpdateStatusInvestigation()`, `DeleteInvestigation()`, dan `ShowInvestigation()`. Kelas `Result` memiliki atribut `id_result`, `notulen`, `final_decision` dan method seperti `AddNewNotulen()`, `UpdateFinalDecision()`, `ShowResultMeeting()`, dan `ShowDocument()`. Kelas `Role` dengan atribut `id_role` dan `name_role`, serta kelas `Report` dengan atribut `id_report` dan method `generatePDF()`.

4.4. Persiapan Environment Backend

Sebelum melakukan pengembangan *backend website* dengan `Express.js`, dilakukan beberapa instalasi dan konfigurasi lingkungan pengembangan perangkat. Berikut adalah tahapan yang dilakukan:

4.4.1. Instalasi Node.js dan Express.js

Memastikan bahwa `Node.js` sudah terinstall di komputer. `Express.js` telah dipilih sebagai framework yang digunakan untuk pengembangan. Selain itu, beberapa alat pendukung seperti `npm` digunakan untuk mendukung instalasi dan konfigurasi `Express.js`. Gambar 4.4 menunjukkan bahwa `Node.js` dan `Express.js` sudah terinstall dengan benar.

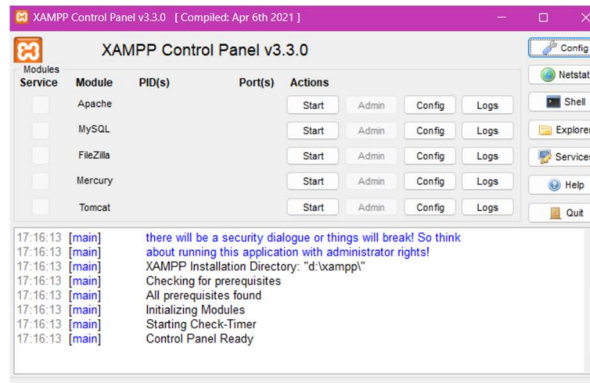


```
PS D:\> node --version
v18.12.1
PS D:\> npm --version
8.19.2
```

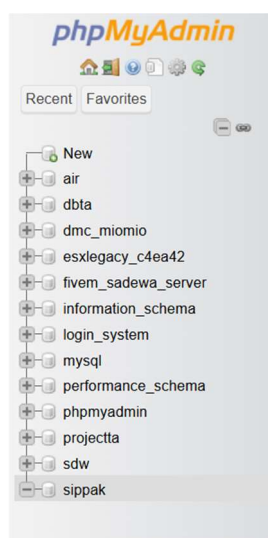
Gambar 4.4 Memastikan instalasi `Node.js` dan `Express.js`

4.4.2. Instalasi MySQL

Pastikan bahwa `MySQL` (Database Management System) seperti `XAMPP`, `MySQL Workbench`, atau standalone `MySQL` server telah terpasang dan berfungsi dengan baik.



Gambar 4.5 Tampilan XAMPP



Gambar 4.6 Berhasil membuat database sippak

Gambar 4.6 menampilkan bahwa XAMPP berjalan dengan baik, selanjutnya tekan tombol "Admin" pada Modul "MySQL" lalu buat database dengan nama yang diinginkan, dalam implementasi peneliti memberi nama "sippak" seperti pada Gambar 4.6.

4.4.3. Membuat proyek Express.js

Membuat proyek Express.js baru menggunakan editor kode seperti Visual Studio Code dengan membuat folder proyek baru dan instal dependensi yang diperlukan menggunakan npm.

```
PS D:\> mkdir SIPPak

Directory: D:\

Mode                LastWriteTime         Length Name
----                -
d-----          5/26/2025   4:08 PM             SIPPak

PS D:\> cd SIPPak
PS D:\SIPPak> npm init -y
Wrote to D:\SIPPak\package.json:

{
  "name": "sippak",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \\\"Error: no test specified\\\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}

PS D:\SIPPak> npm install express
added 66 packages, and audited 67 packages in 2s

14 packages are looking for funding
  run 'npm fund' for details

found 0 vulnerabilities
PS D:\SIPPak> npm install dotenv cors mysql2
added 14 packages, and audited 81 packages in 1s

16 packages are looking for funding
  run 'npm fund' for details
PS D:\SIPPak> npm install express
added 66 packages, and audited 67 packages in 2s

14 packages are looking for funding
  run 'npm fund' for details

found 0 vulnerabilities
PS D:\SIPPak> npm install dotenv cors mysql2
added 14 packages, and audited 81 packages in 1s

16 packages are looking for funding
  run 'npm fund' for details

found 0 vulnerabilities
PS D:\SIPPak> npm install --save-dev nodemon
added 27 packages, and audited 108 packages in 2s

28 packages are looking for funding
  run 'npm fund' for details

found 0 vulnerabilities
```

Gambar 4.7 Hasil pembuatan proyek baru Express.js

Pada Gambar 4.7 telah dilakukan pembuatan proyek Express.js baru dan langkah selanjutnya adalah instalasi dependencies yang diperlukan menggunakan "npm install" hingga proses instalasi selesai.

4.4.4. Konfigurasi file .env

```
.env
1 DB_HOST=localhost
2 DB_USER=root
3 DB_PASSWORD=
4 DB_NAME=sippak
5 PORT=3001
6 SECRET_KEY=v_WJT7ueWhef3hesaDg7kDdh5hMR08JYJFHD2NI2Cjkd90F1P70UdCx_H0LPzLW3DUYeuI1MA38zwPP4x8D3wg
7 |
```

Gambar 4.8 Konfigurasi file .env

Lakukan konfigurasi seperti Gambar 4.8 dengan database yang sudah dibuat sebelumnya terutama untuk nama database "sippak", untuk konfigurasi lainnya biarkan saja apabila tidak melakukan perubahan pada DBMS.

4.5. Perancangan REST API

Perancangan REST API pada sistem pendataan pelanggaran akademik mahasiswa ini dilakukan berdasarkan konsep arsitektur Model-View-Controller (MVC) dan class diagram yang telah dirancang sebelumnya. REST API berfungsi sebagai jembatan komunikasi antara client dan server, memungkinkan pertukaran data dalam format JSON melalui protokol HTTP.

4.5.1. Arsitektur REST API

Arsitektur REST API yang diterapkan menggunakan pola MVC yang membagi sistem menjadi tiga komponen utama:

a. Model

Model merepresentasikan struktur data dan logika bisnis aplikasi. Dalam sistem ini, model utama yang digunakan antara lain:

- User Model: Mengelola data pengguna (staff akademik, kemahasiswaan, wakil dekan)
- Violation Model: Mengelola data pelanggaran akademik mahasiswa
- Investigation Model: Mengelola proses investigasi pelanggaran
- Report Model: Mengelola data laporan dan ringkasan

b. View

View dalam konteks REST API berupa representasi data dalam format JSON yang dikembalikan sebagai response kepada client. Setiap endpoint menghasilkan response JSON dengan struktur yang konsisten, mencakup status code, message, dan data.

c. Controller

Controller bertindak sebagai penghubung antara Model dan View, menangani request dari client, memproses data melalui model, dan mengembalikan response yang sesuai. Controller utama dalam sistem ini meliputi:

- AuthController: Menangani proses autentikasi
- UserController: Mengelola operasi CRUD pengguna
- ViolationController: Mengelola operasi pelanggaran

- InvestigationController: Mengelola proses investigasi

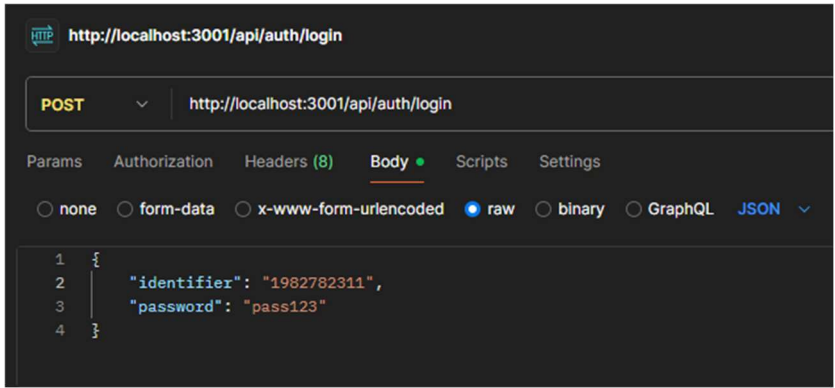
4.5.2. Desain Endpoint API


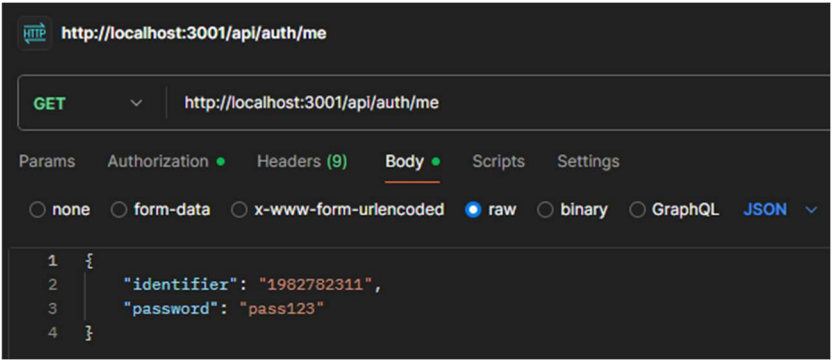

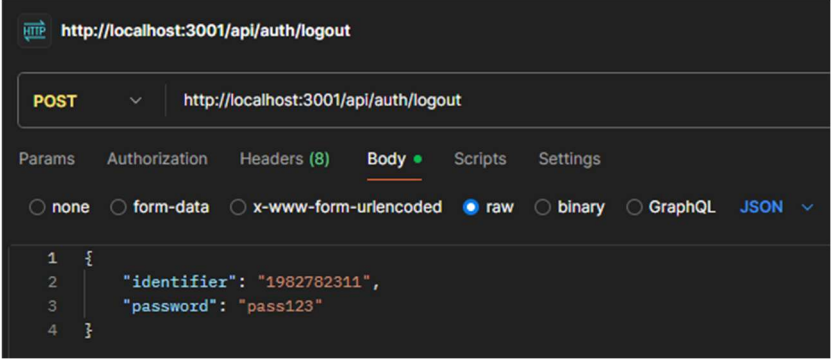

Endpoint autentikasi dirancang untuk menangani proses login, logout, dan verifikasi pengguna dengan implementasi JSON Web Token (JWT) untuk keamanan.

4.5.3. Authentication Endpoints

Fitur *Authentication Endpoints* pada website sistem pendataan pelanggaran akademik telah berhasil memenuhi Functional Requirement (FR01), yaitu sistem memiliki fungsi login dari user class wakil dekan, akademik maupun kemahasiswaan. Dengan fitur ini, pengguna memiliki kemampuan untuk melakukan autentikasi melalui endpoint `/api/auth/login` dengan menggunakan identifier dan password, mendapatkan informasi profil pengguna melalui endpoint `/api/auth/me`, serta melakukan logout melalui endpoint `/api/auth/logout`. Sistem memberikan respons berupa token autentikasi dan informasi user yang mencakup id, email, nama, dan role pengguna. Tabel 4.4 menampilkan hasil implementasi pengembangan fitur Authentication Endpoints pada website sistem pendataan pelanggaran akademik.

Tabel 4.4 Authentication Endpoints

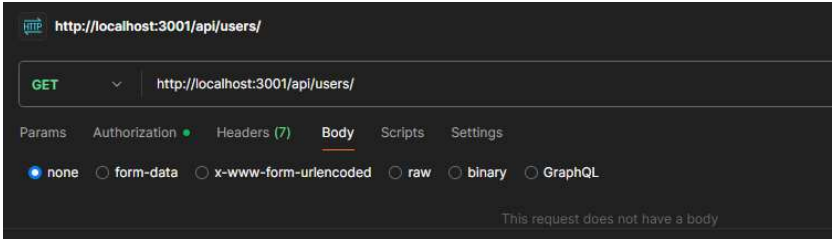
FR-ID	Implementasi
FR01	<p>Request Login</p>  <pre> 1 { 2 "identifier": "1982782311", 3 "password": "pass123" 4 } </pre>

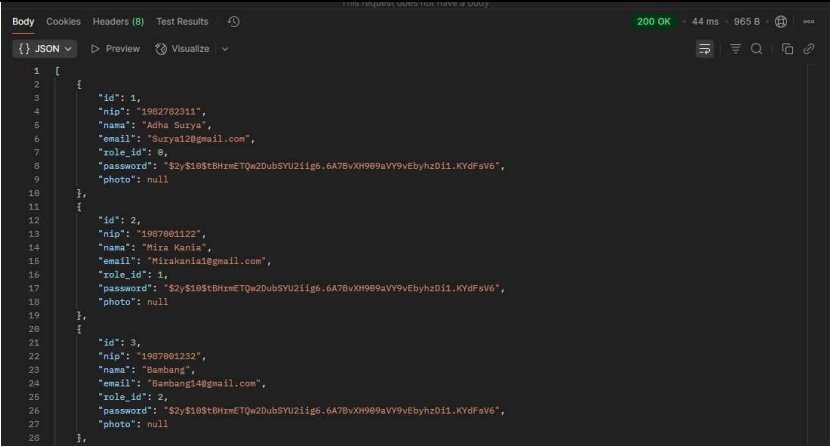
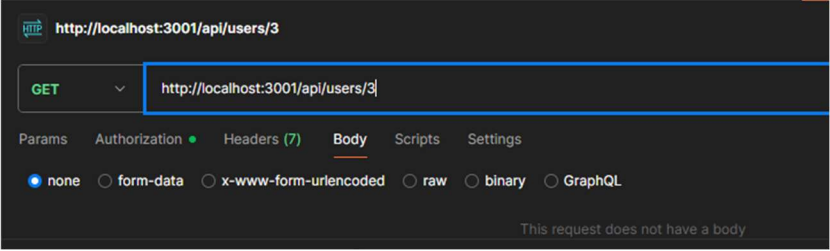

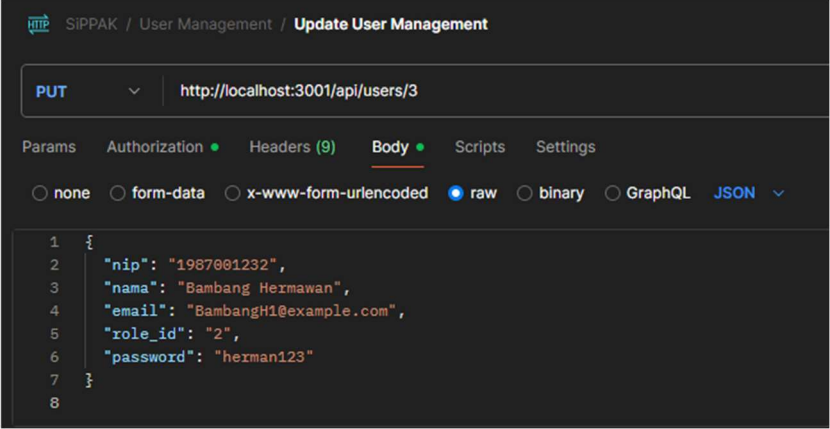
FR-ID	Implementasi
	<p data-bbox="788 293 1007 327">Response Login</p>  <pre data-bbox="507 376 1123 591"> 1 { 2 "message": "Login berhasil", 3 "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZiI6Im90OTg2MzE1IiwiaWF0Ij01OTg3OTYyYQdGdYVWlslmNvbSI6Im1hdCI6MTc0ODI3MDA6MSw1ZjoxNzQ4Mjg4MDQ6fQ.mjNTMyOdectmhkx48kbtZfCeEOTZL8R1K-XD8Qk9zFc", 4 "user": { 5 "id": 1, 6 "nip": "1982782311", 7 "name": "Adha Surya", 8 "email": "Surya12@gmail.com", 9 "role": 0 10 } 11 }</pre> <p data-bbox="667 622 1126 656">Request Get Info User with Token</p>  <pre data-bbox="523 927 863 1016"> 1 { 2 "identifier": "1982782311", 3 "password": "pass123" 4 }</pre> <p data-bbox="655 1055 1137 1088">Response Get Info User with Token</p>  <pre data-bbox="507 1173 836 1285"> 1 { 2 "message": "Informasi pengguna berhasil didapatkan", 3 "user": { 4 "id": 1, 5 "email": "Surya12@gmail.com", 6 "iat": 1748271845, 7 "exp": 1748289845 8 } 9 }</pre> <p data-bbox="788 1323 1007 1357">Request Logout</p>  <pre data-bbox="523 1628 863 1718"> 1 { 2 "identifier": "1982782311", 3 "password": "pass123" 4 }</pre> <p data-bbox="775 1756 1019 1789">Response Logout</p>  <pre data-bbox="507 1874 708 1912"> 1 { 2 "message": "Logout berhasil" 3 }</pre>

4.5.2.1. User Management Endpoints

Fitur *User Management Endpoints* pada sistem pendataan pelanggaran akademik telah memenuhi Functional Requirement (FR02), yaitu memberikan fungsi untuk mengelola data pengguna secara menyeluruh. Administrator dapat melakukan operasi CRUD (Create, Read, Update, Delete) melalui beberapa endpoint, seperti menambahkan pengguna baru melalui endpoint `/api/users/add` dengan data lengkap (nip, email, password, role_id, nama, no_telp, fakultas, jurusan, dan photo), mengambil seluruh data pengguna melalui endpoint `/api/users/`, mengambil data pengguna tertentu berdasarkan ID melalui endpoint `/api/users/{id}`, memperbarui data pengguna menggunakan method PUT pada endpoint `/api/users/{id}`, serta menghapus pengguna melalui method DELETE pada endpoint yang sama. Setiap operasi memberikan respons yang sesuai, termasuk konfirmasi atas proses registrasi, pembaruan, dan penghapusan data. Hasil dari implementasi fitur ini ditampilkan pada Tabel 4.5.

Tabel 4.5 *User Management Endpoints*

FR-ID	Implementasi
FR-02	<div><div>Request</div><div>Response</div></div>

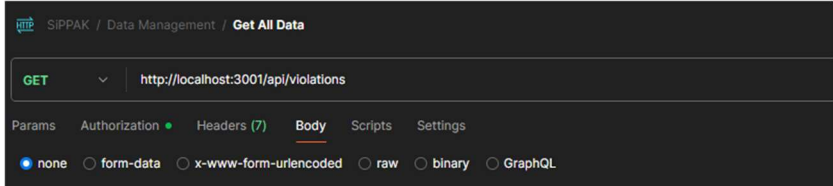
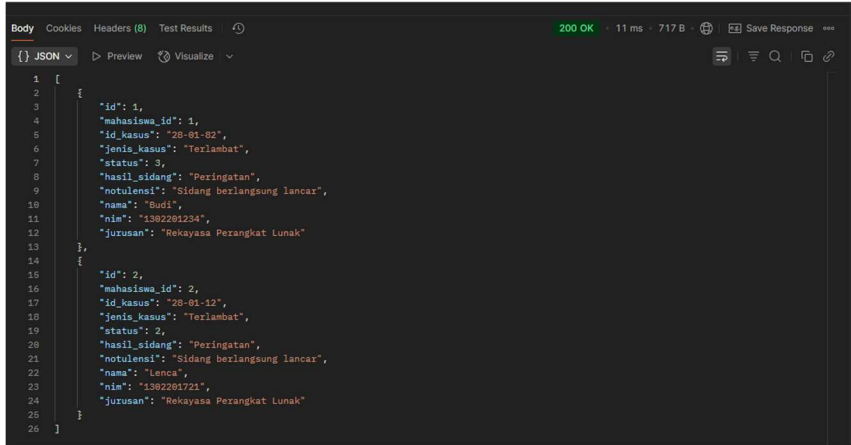
FR-ID	Implementasi
	<div data-bbox="483 286 1316 730">  <pre> 1 [2 { 3 "id": 1, 4 "nip": "1982782311", 5 "nama": "Adha Surya", 6 "email": "Surya12@gmail.com", 7 "role_id": 0, 8 "password": "\$2y\$10\$tBHzmETQw2DubSYU2iig6.6A7BvXH989aVY9vEbyhzD11.KYdFv6", 9 "photo": null 10 }, 11 { 12 "id": 2, 13 "nip": "1987881122", 14 "nama": "Mira Kania", 15 "email": "Mirakania1@gmail.com", 16 "role_id": 1, 17 "password": "\$2y\$10\$tBHzmETQw2DubSYU2iig6.6A7BvXH989aVY9vEbyhzD11.KYdFv6", 18 "photo": null 19 }, 20 { 21 "id": 3, 22 "nip": "1987881232", 23 "nama": "Bambang", 24 "email": "Bambang14@gmail.com", 25 "role_id": 2, 26 "password": "\$2y\$10\$tBHzmETQw2DubSYU2iig6.6A7BvXH989aVY9vEbyhzD11.KYdFv6", 27 "photo": null 28 } 29] </pre> </div> <div data-bbox="842 752 959 786">Request</div> <div data-bbox="483 808 1316 1057">  </div> <div data-bbox="842 1079 975 1113">Responses</div> <div data-bbox="483 1135 1316 1321">  </div> <div data-bbox="842 1344 959 1377">Request</div> <div data-bbox="483 1400 1316 1827">  <pre> 1 { 2 "nip": "1987881232", 3 "nama": "Bambang Hermawan", 4 "email": "BambangH1@example.com", 5 "role_id": "2", 6 "password": "herman123" 7 } 8 </pre> </div> <div data-bbox="842 1850 968 1883">Response</div>

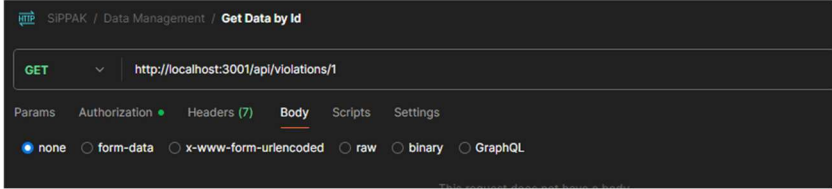
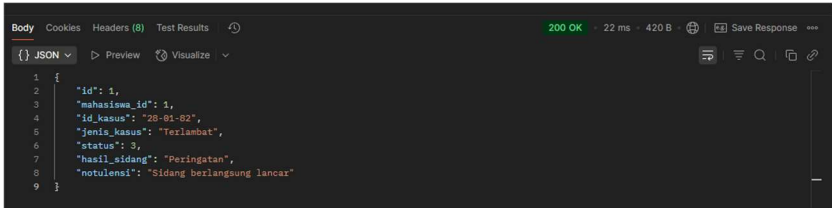
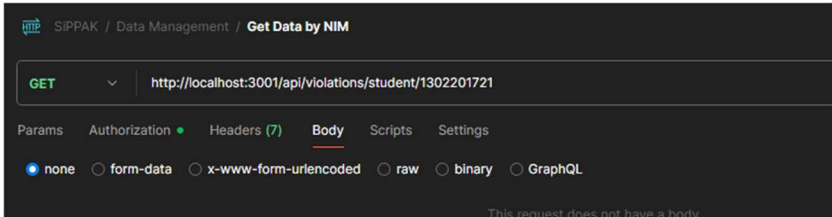

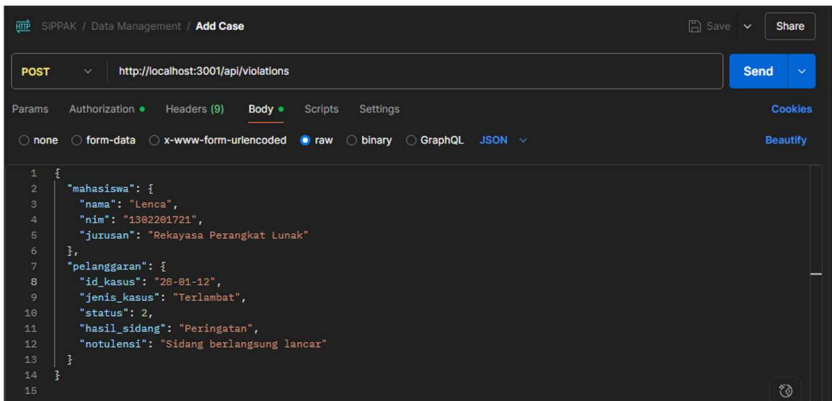
FR-ID	Implementasi																																												
	<div><div><div><div><div><div></div><div>Body</div><div>Cookies</div><div>Headers (8)</div><div>Test Results</div><div></div></div><div></div><div><div>200 OK</div><div>517 ms</div><div>485 B</div><div></div><div>Save Response</div><div></div></div></div><div><div><div><div></div><div>JSON</div><div></div></div><div>Preview</div><div>Visualize</div><div></div></div><div><div>1</div><div>{</div><div>2</div><div> "message": "User berhasil diupdate",</div><div>3</div><div> "updatedUser": {</div><div>4</div><div> "nip": "1987891232",</div><div>5</div><div> "nama": "Bambang Hermawan",</div><div>6</div><div> "email": "BambangH@example.com",</div><div>7</div><div> "role_id": "2",</div><div>8</div><div> "password": "\$2b\$10\$GcTBeRv12fR3d65zB4xNee5f6aLAJMDtHL4xvPq/36DmI3H1auC2i"</div><div>9</div><div> }</div><div>10</div><div>}</div></div></div></div></div><div>Request</div><div><div><div>SIPPAK / User Management / Add User</div><div>Save</div><div>Share</div></div><div><div>POST</div><div>http://localhost:3001/api/users/add</div><div>Send</div></div><div><div>Params</div><div>Authorization</div><div>Headers (9)</div><div>Body</div><div>Scripts</div><div>Settings</div><div>Cookies</div></div><div><div>none</div><div>form-data</div><div>x-www-form-urlencoded</div><div>raw</div><div>binary</div><div>GraphQL</div></div><div><table><tr><th>Key</th><th>Value</th><th>Description</th><th>Bulk Edit</th></tr><tr><td><input checked="" type="checkbox"/> nip</td><td>1972736711</td><td></td><td></td></tr><tr><td><input checked="" type="checkbox"/> email</td><td>LuqmanHer1@gmail.com</td><td></td><td></td></tr><tr><td><input checked="" type="checkbox"/> password</td><td>Luqman123</td><td></td><td></td></tr><tr><td><input checked="" type="checkbox"/> role_id</td><td>2</td><td></td><td></td></tr><tr><td><input checked="" type="checkbox"/> nama</td><td>Luqman Hermawan</td><td></td><td></td></tr><tr><td><input checked="" type="checkbox"/> no_telp</td><td>+6282361562561</td><td></td><td></td></tr><tr><td><input checked="" type="checkbox"/> fakultas</td><td>Informatika</td><td></td><td></td></tr><tr><td><input checked="" type="checkbox"/> jurusan</td><td>rekayasa perangkat lunak</td><td></td><td></td></tr><tr><td><input checked="" type="checkbox"/> photo</td><td>d6de39e98bda807afe73d0865806710.jpg</td><td></td><td></td></tr><tr><td>Key</td><td>Value</td><td>Description</td><td></td></tr></table></div></div><div>Response</div><div><div><div>Body</div><div>Cookies</div><div>Headers (8)</div><div>Test Results</div><div></div></div><div></div><div><div>201 Created</div><div>383 ms</div><div>609 B</div><div></div><div>Save Response</div><div></div></div></div><div><div><div><div></div><div>JSON</div><div></div></div><div>Preview</div><div>Visualize</div><div></div></div><div><div>1</div><div>{</div><div>2</div><div> "message": "User registered successfully",</div><div>3</div><div> "register": {</div><div>4</div><div> "nip": "1972736711",</div><div>5</div><div> "email": "LuqmanHer1@gmail.com",</div><div>6</div><div> "password": "\$2b\$10\$116P8tCWMDU11HA8Leut/EApfkaE3Q10yVWak8ckmb/.Q.nck6",</div><div>7</div><div> "role_id": "2",</div><div>8</div><div> "nama": "Luqman Hermawan",</div><div>9</div><div> "no_telp": "+6282361562561",</div><div>10</div><div> "fakultas": "Informatika",</div><div>11</div><div> "jurusan": "rekayasa perangkat lunak",</div><div>12</div><div> "photo": "1748282567593.jpg"</div><div>13</div><div> }</div><div>14</div><div>}</div></div></div></div> <div>Request</div> <div><div><div>SIPPAK / User Management / Delete User</div><div></div></div><div><div>DELETE</div><div>http://localhost:3001/api/users/5</div></div><div><div>Params</div><div>Authorization</div><div>Headers (7)</div><div>Body</div><div>Scripts</div><div>Settings</div></div><div><div>none</div><div>form-data</div><div>x-www-form-urlencoded</div><div>raw</div><div>binary</div><div>GraphQL</div></div><div>This request does not have a body</div></div> <div>Response</div> <div><div><div>Body</div><div>Cookies</div><div>Headers (8)</div><div>Test Results</div><div></div></div><div></div><div><div>200 OK</div><div>41 ms</div><div>319 B</div><div></div><div>Save Response</div><div></div></div></div> <div><div><div><div></div><div>JSON</div><div></div></div><div>Preview</div><div>Visualize</div><div></div></div><div><div>1</div><div>{</div><div>2</div><div> "message": "User berhasil dihapus",</div><div>3</div><div> "deleteUser": "5"</div><div>4</div><div>}</div></div></div>	Key	Value	Description	Bulk Edit	<input checked="" type="checkbox"/> nip	1972736711			<input checked="" type="checkbox"/> email	LuqmanHer1@gmail.com			<input checked="" type="checkbox"/> password	Luqman123			<input checked="" type="checkbox"/> role_id	2			<input checked="" type="checkbox"/> nama	Luqman Hermawan			<input checked="" type="checkbox"/> no_telp	+6282361562561			<input checked="" type="checkbox"/> fakultas	Informatika			<input checked="" type="checkbox"/> jurusan	rekayasa perangkat lunak			<input checked="" type="checkbox"/> photo	d6de39e98bda807afe73d0865806710.jpg			Key	Value	Description	
Key	Value	Description	Bulk Edit																																										
<input checked="" type="checkbox"/> nip	1972736711																																												
<input checked="" type="checkbox"/> email	LuqmanHer1@gmail.com																																												
<input checked="" type="checkbox"/> password	Luqman123																																												
<input checked="" type="checkbox"/> role_id	2																																												
<input checked="" type="checkbox"/> nama	Luqman Hermawan																																												
<input checked="" type="checkbox"/> no_telp	+6282361562561																																												
<input checked="" type="checkbox"/> fakultas	Informatika																																												
<input checked="" type="checkbox"/> jurusan	rekayasa perangkat lunak																																												
<input checked="" type="checkbox"/> photo	d6de39e98bda807afe73d0865806710.jpg																																												
Key	Value	Description																																											

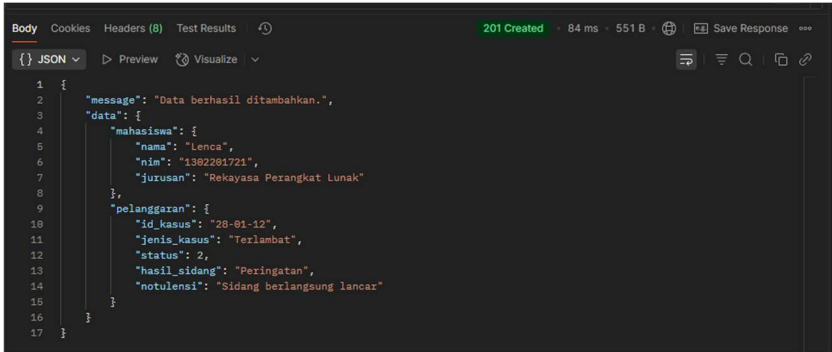
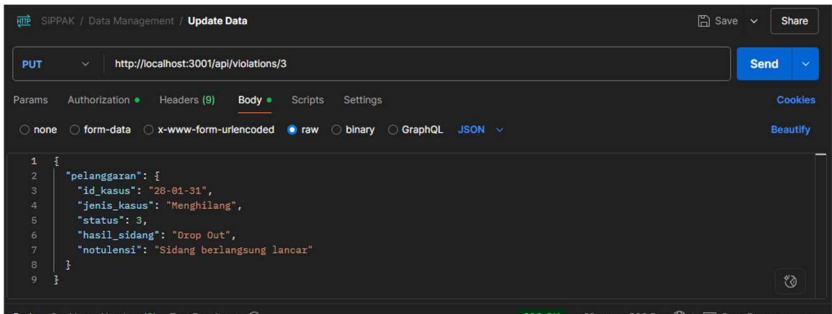
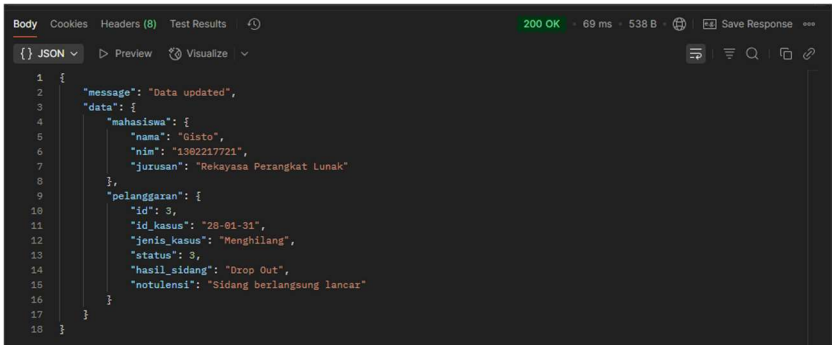
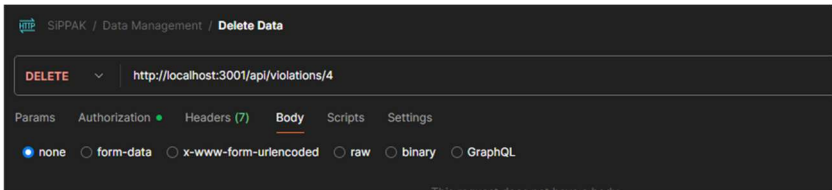
4.5.2.2. Violation Management Endpoints

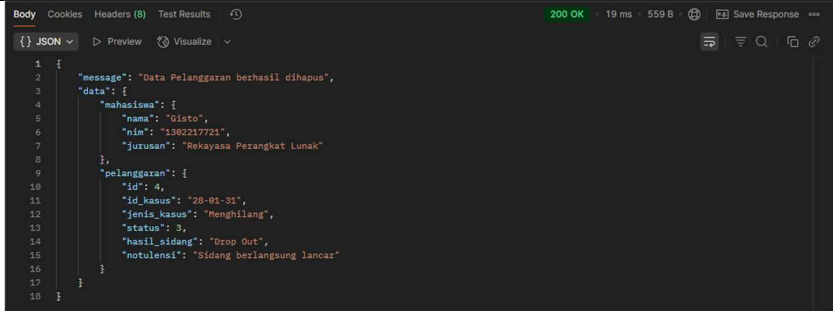
Fitur *Violation Management Endpoints* pada sistem pendataan pelanggaran akademik telah berhasil memenuhi Functional Requirement (FR03 hingga FR07). Fitur ini memungkinkan staff untuk mengelola data pelanggaran secara menyeluruh, seperti menambahkan kasus pelanggaran mahasiswa, memperbarui informasi kasus, membuka sidang dan menyusun draft keputusan, melacak status kasus, serta menerima notifikasi atas perubahan data. Operasi CRUD terhadap data pelanggaran dapat dilakukan melalui berbagai endpoint, antara lain: menambahkan data melalui endpoint `/api/violations` dengan method POST, mengambil seluruh data atau data spesifik berdasarkan ID atau NIM mahasiswa dengan method GET, memperbarui data dengan method PUT, dan menghapus data dengan method DELETE. Setiap operasi disertai respons yang sesuai, termasuk konfirmasi keberhasilan serta detail informasi mahasiswa dan pelanggaran yang terkait. Hasil implementasi fitur ini ditampilkan pada Tabel 4.6.

Tabel 4.6 *Violation Management Endpoints*

FR-ID	Implentasi
FR03	<p>Request</p>  <p>Response</p> 

FR-ID	Implentasi
FR04	<p>Request</p>  <p>Response</p> 
FR05	<p>Request</p>  <p>Response</p> 
FR06	<p>Request</p> 

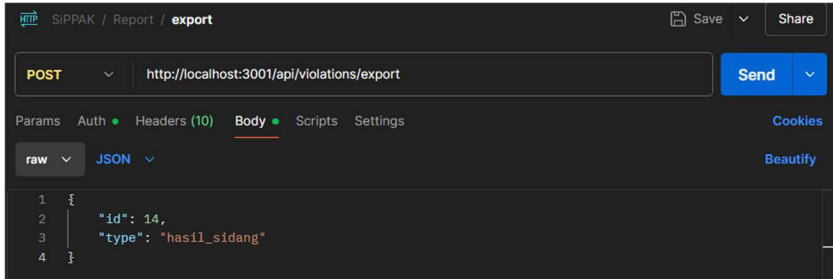
FR-ID	<p>Implentasi</p> <p>Response</p>  <pre> 1 { 2 "message": "Data berhasil ditambahkan.", 3 "data": { 4 "mahasiswa": { 5 "nama": "Lenca", 6 "nim": "1382281721", 7 "jurusan": "Rekayasa Perangkat Lunak" 8 }, 9 "pelanggaran": { 10 "id_kasus": "28-01-12", 11 "jenis_kasus": "Terlambat", 12 "status": 2, 13 "hasil_sidang": "Peringatan", 14 "notulensi": "Sidang berlangsung lancar" 15 } 16 } 17 } </pre>
FR07	<p>Request</p>  <pre> 1 { 2 "pelanggaran": { 3 "id_kasus": "28-01-31", 4 "jenis_kasus": "Menghilang", 5 "status": 3, 6 "hasil_sidang": "Drop Out", 7 "notulensi": "Sidang berlangsung lancar" 8 } 9 } </pre> <p>Response</p>  <pre> 1 { 2 "message": "Data updated", 3 "data": { 4 "mahasiswa": { 5 "nama": "Gisto", 6 "nim": "1382217721", 7 "jurusan": "Rekayasa Perangkat Lunak" 8 }, 9 "pelanggaran": { 10 "id": 3, 11 "id_kasus": "28-01-31", 12 "jenis_kasus": "Menghilang", 13 "status": 3, 14 "hasil_sidang": "Drop Out", 15 "notulensi": "Sidang berlangsung lancar" 16 } 17 } 18 } </pre> <p>Request</p>  <pre> DELETE http://localhost:3001/api/violations/4 </pre> <p>Response</p>

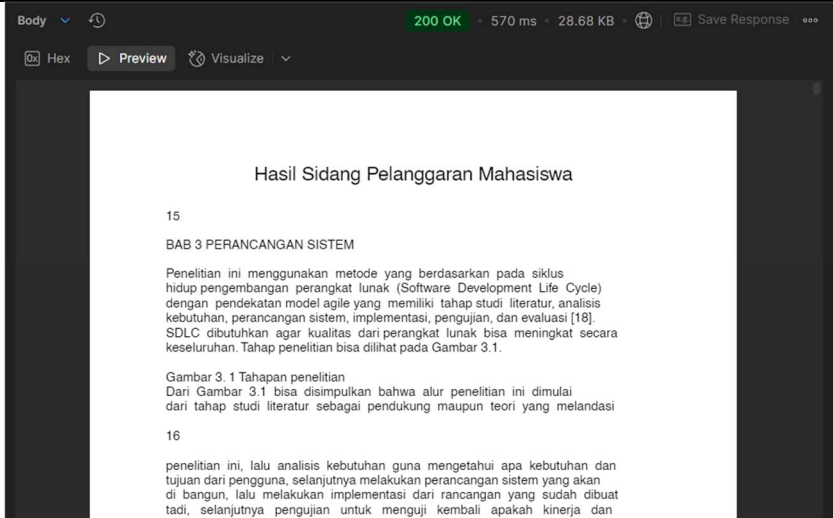
FR-ID	Implentasi
	 <pre> 1 { 2 "message": "Data Pelanggaran berhasil dihapus", 3 "data": { 4 "mahasiswa": { 5 "nama": "Gisto", 6 "nim": "1302217721", 7 "jurusan": "Rekayasa Perangkat Lunak" 8 }, 9 "pelanggaran": { 10 "id": 4, 11 "id_kasus": "28-01-31", 12 "jenis_kasus": "Menghilang", 13 "status": 3, 14 "hasil_sidang": "Drop Out", 15 "notulensi": "Sidang berlangsung lancar" 16 } 17 } 18 } </pre>

4.5.2.3. Reporting Endpoints

Fitur *Reporting Endpoints* pada website sistem pendataan pelanggaran akademik telah berhasil memenuhi sebagian dari Functional Requirement (FR), yaitu FR-06 yang berkaitan dengan fungsi sistem untuk membantu staff dalam membuat hasil Keputusan sidang dan melakukan penutupan kasus. Reporting Endpoints berfungsi untuk export hasil sidang dan notulensi, sehingga staff memiliki kemampuan untuk mengekspor data sidang dan dokumentasi notulensi dalam format yang dapat digunakan untuk pelaporan dan arsip. Tabel 4.7 menampilkan hasil implementasi pengembangan fitur Reporting Endpoints pada website sistem pendataan pelanggaran akademik.

Tabel 4.7 *Reporting Endpoints*

FR-ID	Implementasi
FR01	<div>Request</div>  <pre> 1 { 2 "id": 14, 3 "type": "hasil_sidang" 4 } </pre> <div>Response</div>

FR-ID	Implementasi
	

4.6. Implementasi

Proses implementasi dari sistem yang telah dirancang sebelumnya dilakukan melalui beberapa tahap. Tahap ini bertujuan untuk merealisasikan rancangan sistem ke dalam bentuk aplikasi yang dapat dijalankan sesuai dengan kebutuhan. Implementasi dilakukan menggunakan Node.js dengan framework Express.js dan database MySQL/MariaDB menggunakan XAMPP sesuai dengan arsitektur MVC yang telah ditetapkan.

4.6.1. Implementasi Database *Schema*

Koneksi database dikonfigurasi melalui file config/db.js yang terhubung ke MySQL menggunakan environment variables dari file .env untuk menjaga keamanan kredensial. Implementasinya ditunjukkan pada Gambar 4.9.

```

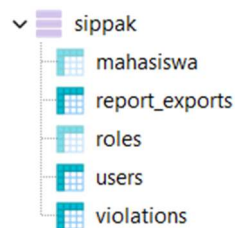
config > JS db.js > ...
1  const mysql = require('mysql2');
2
3  const db = mysql.createConnection({
4    host: process.env.DB_HOST,
5    user: process.env.DB_USER,
6    password: process.env.DB_PASSWORD,
7    database: process.env.DB_NAME
8  });
9
10 module.exports = db;
11

```

Gambar 4.9 Implementasi konfigurasi database

Setelah konfigurasi database, dilakukan implementasi struktur database yang terdiri dari lima tabel utama: users, roles, mahasiswa,

violations, dan report_exports, yang disesuaikan dengan kebutuhan sistem pendataan pelanggaran akademik. Masing-masing tabel memiliki fungsi spesifik, seperti penyimpanan data pengguna, peran, mahasiswa yang terlibat, detail pelanggaran, hingga riwayat ekspor laporan. Struktur lengkap ditampilkan pada Gambar 4.10.



Gambar 4.10 Struktur database

4.6.2. Implementasi Model Layer

Model layer diimplementasikan dengan menggunakan pendekatan MVC (Model-View-Controller). Setiap model merepresentasikan entitas dalam database dan menyediakan fungsi-fungsi untuk operasi CRUD (Create, Read, Update, Delete).

Model authModel.js bertanggung jawab untuk menangani proses autentikasi pengguna, seperti yang ditunjukkan pada Gambar 4.11.

```

models > JS authModel.js > ...
1  const db = require('../config/db');
2
3  exports.findByIdentifier = (identifier, callback) => {
4    const query = `
5      SELECT users.*, roles.name AS role_name
6      FROM users
7      JOIN roles ON users.role_id = roles.id
8      WHERE users.nip = ? OR users.email = ?
9      LIMIT 1
10   `;
11    db.query(query, [identifier, identifier], callback);
12  };
13

```

Gambar 4.11 Model authModel.js

Model userModel.js mengelola operasi yang berkaitan dengan data pengguna seperti registrasi, update profil, dan manajemen pengguna, implementasinya dapat dilihat pada Gambar 4.12 dan Gambar 4.13.

```

model > # userModel.js > @createUser > @createUser
1 const db = require('../config/db');
2
3 exports.getUserByEmail = (email, callback) => {
4   const query = `
5     SELECT users.*, roles.name AS role
6     FROM users
7     JOIN roles ON users.role_id = roles.id
8     WHERE users.email = ?
9   `;
10  db.query(query, [email], callback);
11};
12
13 exports.createUser = (data, callback) => {
14  const query = `
15    INSERT INTO users (nip, email, password, role_id, nama, no_telp, fakultas, jurusan, photo)
16    VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)
17  `;
18  const values = [
19    data.nip,
20    data.email,
21    data.password,
22    data.role_id,
23    data.nama,
24    data.no_telp,
25    data.fakultas,
26    data.jurusan,
27    data.photo,
28  ];
29  db.query(query, values, callback);
30};
31
32 exports.getAllUsers = (callback) => {
33  const query = `
34    SELECT users.id, users.nip, users.nama, users.email,
35           users.password, users.photo, roles.name AS role
36    FROM users
37    JOIN roles ON users.role_id = roles.id
38  `;
39  db.query(query, callback);
40};

```

Gambar 4.12 Model userModel.js

```

model > # userModel.js > ...
41
42 exports.getUserById = (id, callback) => {
43  db.query(
44    `
45      SELECT
46        users.id, users.nip, users.nama, users.email, users.role_id,
47        users.password, users.photo, users.no_telp, users.fakultas, users.jurusan,
48        roles.name AS role
49      FROM users
50      JOIN roles ON users.role_id = roles.id
51      WHERE users.id = ?
52    `,
53    [id],
54    callback
55  );
56};
57
58 exports.updateUser = (id, data, callback) => {
59  const query = `
60    UPDATE users SET
61      nip = ?, nama = ?, email = ?, no_telp = ?, fakultas = ?, jurusan = ?
62    WHERE id = ?
63  `;
64  const values = [
65    data.nip, data.nama, data.email,
66    data.no_telp, data.fakultas, data.jurusan,
67    id
68  ];
69  db.query(query, values, callback);
70};
71
72 exports.updateUserWithoutPhoto = (id, data, callback) => {
73  const query = `UPDATE users SET nip = ?, nama = ?, email = ?, role_id = ? password = ? WHERE id = ?`;
74  db.query(query, [data.nip, data.nama, data.email, data.role, data.password, id], callback);
75};
76
77 exports.deleteUser = (id, callback) => {
78  db.query(`DELETE FROM users WHERE id = ?`, [id], callback);
79};
80
81 exports.updateUserPhoto = (id, photo, callback) => {
82  const query = `UPDATE users SET photo = ? WHERE id = ?`;
83  db.query(query, [photo, id], callback);
84};
85
86 exports.customQuery = (query, values, callback) => {
87  db.query(query, values, callback);
88};

```

Gambar 4.13 Model userModel.js

Model violationModel.js merupakan model utama yang menangani seluruh operasi pelanggaran akademik mulai dari pembuatan kasus, update status, hingga penutupan kasus, seperti yang ditampilkan pada Gambar 4.14, Gambar 4.15 dan Gambar 4.16.

```

models > violationModel.js > update > update
1 const db = require('../config/db');
2
3 exports.getAll = cb => {
4   db.query(
5     SELECT v.*, m.nama, m.nim, m.jurusan, m.semester
6     FROM violations v JOIN mahasiswa m ON v.mahasiswa_id = m.id
7     , cb);
8   };
9
10 exports.getById = (id, cb) => {
11   db.query(
12     SELECT v.*, m.nama, m.nim, m.jurusan, m.semester
13     FROM violations v
14     JOIN mahasiswa m ON v.mahasiswa_id = m.id
15     WHERE v.id = ?
16     , [id], cb);
17   };
18
19 exports.getByNIM = (nim, cb) => {
20   db.query(
21     SELECT v.*, m.nama, m.nim, m.jurusan, m.semester
22     FROM violations v
23     JOIN mahasiswa m ON v.mahasiswa_id = m.id
24     WHERE m.nim = ?
25     , [nim], cb);
26   };
27
28 exports.create = (mahasiswa, pelanggaran, cb) => {
29   db.query(
30     SELECT id FROM mahasiswa WHERE nim = ?,
31     [mahasiswa.nim],
32     (err, result) => {
33       if (err) return cb(err);
34       if (result.length > 0) {
35         insertViolation(result[0].id);
36       } else {
37         db.query(
38           INSERT INTO mahasiswa (nama, nim, jurusan, semester) VALUES (?, ?, ?, ?),
39           [mahasiswa.nama, mahasiswa.nim, mahasiswa.jurusan, mahasiswa.semester],
40           (err, result) => {
41             if (err) return cb(err);
42             insertViolation(result.insertId);
43           }
44         );
45       }
46     }
47   );
48 }

```

Gambar 4.14 Model violationModel.js

```

models > violationModel.js > update > update
49 function insertViolation(mahasiswaId) {
50   db.query(
51     INSERT INTO violations
52     (mahasiswa_id, id_kasus, jenis_kasus, status, hasil_sidang, notulensi,
53     foto, deskripsi, status_approval, type)
54     VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?),
55     [
56       mahasiswaId,
57       pelanggaran.id_kasus,
58       pelanggaran.jenis_kasus,
59       pelanggaran.status,
60       pelanggaran.hasil_sidang,
61       pelanggaran.notulensi,
62       pelanggaran.foto || null,
63       pelanggaran.deskripsi || null,
64       pelanggaran.status_approval || 'Pending',
65       pelanggaran.type || 'New',
66     ],
67     (err, res) => {
68       if (err) return cb(err);
69       cb(null, res);
70     }
71   );
72 }
73
74 };
75
76 exports.update = (id, data, cb) => {
77   const allowedFields = [
78     "id_kasus", "jenis_kasus", "status", "hasil_sidang", "notulensi",
79     "foto", "deskripsi", "status_approval", "type"
80   ];
81
82   const updates = [];
83   const values = [];
84
85   for (const key of allowedFields) {
86     if (data[key] !== undefined) {
87       updates.push(`${key} = ?`);
88       values.push(data[key]);
89     }
90   }
91
92   if (updates.length === 0) {
93     return cb(null, { message: "Tidak ada field yang diupdate." });
94   }
95
96   values.push(id);
97
98   const sql = `UPDATE violations SET ${updates.join(', ')} WHERE id = ?`;
99   db.query(sql, values, cb);
100 }

```

Gambar 4.15 Lanjutan Model violationModel.js

```

models > violationModel.js > update > update
102 exports.delete = (id, cb) => {
103   db.query('DELETE FROM violations WHERE id = ?', [id], cb);
104 };
105
106 exports.getViolationsByDate = async (start, end) => {
107   let query = 'SELECT * FROM violations';
108   let params = [];
109
110   if (start && end) {
111     query += ' WHERE DATE(created_at) BETWEEN ? AND ?';
112     params.push(start, end);
113   }
114
115   query += ' ORDER BY created_at DESC LIMIT 100';
116
117   db.query(query, params);
118 };
119

```

Gambar 4.16 Lanjutan Model violationModel.js

Model `reportExportModel.js` mengelola proses export laporan dan tracking file yang telah di-generate, implementasinya dapat dilihat pada Gambar 4.17.

```
1 const db = require('../config/db');
2
3 exports.saveExportLog = async (filename) => {
4   const [result] = await db.promise().query(
5     'INSERT INTO report_exports (filename, created_at) VALUES (?, NOW())',
6     [filename]
7   );
8   return result.insertId;
9 };
10
```

Gambar 4.17 Model `reportExportModel.js`

4.6.3. Implementasi Controller Layer

Controller layer berfungsi sebagai penghubung antara model dan route, menangani logika bisnis aplikasi. Implementasi controller dilakukan dengan membuat beberapa file controller yang sesuai dengan fungsi masing-masing.

`authController.js` menangani proses autentikasi pengguna termasuk login, logout, dan validasi token JWT, seperti yang ditunjukkan pada Gambar 4.18.

```
controllers / authController.js
1 const bcrypt = require('bcryptjs');
2 const jwt = require('jsonwebtoken');
3 const User = require('../models/authModel');
4 require('dotenv').config();
5
6 exports.login = (req, res) => {
7   const { identifier, password } = req.body;
8   const identifierStr = String(identifier);
9   console.log('with parameters:', [identifierStr, password]);
10
11   User.findbyIdentifier(identifierStr, async (err, results) => {
12     if (err) return res.status(400).json({ message: 'Database error', error: err });
13     if (results.length === 0) return res.status(404).json({ message: 'User not found' });
14
15     const user = results[0];
16     const match = await bcrypt.compare(password, user.password);
17     if (!match) return res.status(401).json({ message: 'Invalid credentials' });
18
19     const token = jwt.sign({
20       id: user.id,
21       email: user.email,
22       role: user.role,
23     }, process.env.SECRET_KEY, { expiresIn: '3h' });
24
25     res.status(200).json({
26       message: 'Login berhasil',
27       token,
28       user: {
29         id: user.id,
30         nim: user.nim,
31         nip: user.nip,
32         nama: user.nama,
33         email: user.email,
34         role: user.role_name,
35         photo: user.photo
36       }
37     });
38   });
39 };
40
41 exports.logout = (req, res) => {
42   res.status(200).json({ message: 'Logout berhasil' });
43 };
44
45 exports.getMe = (req, res) => {
46   const user = req.user;
47   if (!user) return res.status(401).json({ message: 'Unauthorized' });
48
49   res.status(200).json({
50     message: 'Informasi pengguna berhasil didapatkan',
51     user
52   });
53 };
54
```

Gambar 4.18 `authController.js`

`userController.js` mengelola operasi CRUD pengguna, termasuk registrasi pengguna baru, update profil, dan manajemen role,

implementasinya dapat dilihat pada Gambar 4.19, Gambar 4.20, dan Gambar 4.21.

```

const bcrypt = require('bcrypt');
const User = require('../models/userModel');
const fs = require('fs');
const path = require('path');

exports.getUsers = (req, res) => {
  User.getAllUsers((err, results) => {
    if (err) return res.status(500).json({ message: 'Gagal mengambil data', error: err });
    res.json(results);
  });
};

exports.getuser = (req, res) => {
  const id = req.params.id;
  User.getUserById(id, (err, results) => {
    if (err) return res.status(500).json({ message: 'Gagal mengambil data', error: err });
    if (results.length === 0) return res.status(404).json({ message: 'User tidak ditemukan' });
    res.json(results[0]);
  });
};

exports.register = (req, res) => {
  const { nip, email, password, role_id, nama, no_telp, fakultas, jurusan } = req.body;
  const photo = req.file ? req.file.filename : null;
  bcrypt.hash(password, 10, (err, hashedPassword) => {
    if (err) return res.status(500).json({ error: 'Hashing error' });
    const user = {
      nip,
      email,
      password: hashedPassword,
      role_id,
      nama,
      no_telp: no_telp || '',
      fakultas: fakultas || 'Informatika',
      jurusan: jurusan || 'Kecayaan Perangmat Lunak',
      photo,
    };
    User.createUser(user, (err, result) => {
      if (err) return res.status(500).json({ error: err.message });
      res.status(201).json({
        message: 'User registered successfully',
        register: user,
      });
    });
  });
};

exports.updateUser = (req, res) => {
  const id = req.params.id;
  const { nip, nama, email, no_telp, fakultas, jurusan } = req.body;
  const userData = { nip, nama, email, no_telp, fakultas, jurusan };
  User.updateUser(id, userData, (err) => {
    if (err) return res.status(500).json({ message: 'Gagal update user', error: err });
    res.json({ message: 'User berhasil diupdate' });
  });
};

```

Gambar 4.19 userController.js

```

exports.updateUserPhoto = (req, res) => {
  const id = req.params.id;
  const photo = req.file ? req.file.filename : null;
  if (!photo) return res.status(400).json({ message: 'Foto tidak ditemukan' });
  // email foto lama
  const db = require('../config/db');
  db.query("SELECT photo FROM users WHERE id = ?", [id], (err, results) => {
    if (err) return res.status(500).json({ message: 'Gagal email user', error: err });
    // hapus foto lama jika ada
    if (results.length > 0 && results[0].photo) {
      const fs = require('fs');
      const path = require('path');
      const oldPath = path.join(__dirname, '..', 'uploads', 'profile', results[0].photo);
      fs.unlink(oldPath, (err) => {
        if (err) console.warn('Gagal hapus foto lama', err.message);
      });
    }
    // update via model
    User.updateUserPhoto(id, photo, (err) => {
      if (err) return res.status(500).json({ message: 'Gagal update foto', error: err });
      res.json({ message: 'Foto berhasil diperbarui', updatedPhoto: photo });
    });
  });
};

exports.addNewManagement = async (req, res) => {
  let { nip, nama, email, password, role_id, fakultas, jurusan, no_telp } = req.body;
  try {
    const hashedPassword = await bcrypt.hash(password, 10);
    const userData = {
      nip,
      nama,
      email,
      password: hashedPassword,
      role_id,
      fakultas: fakultas || '',
      jurusan: jurusan || '',
      no_telp: no_telp || '',
    };
    User.createUser(userData, (err, result) => {
      if (err) return res.status(500).json({ message: 'Gagal menambah user', error: err });
      res.status(201).json({
        message: 'User berhasil ditambahkan',
        newuser: userData,
      });
    });
  } catch (error) {
    return res.status(500).json({ message: 'Gagal hashing password', error });
  }
};

```

Gambar 4. 20 Lanjutan userController.js

```

120 export.updateUserManagement = async (req, res) => {
121     const id = req.params.id;
122     let nlp, name, email, password, role_id, fakultas, jurusan, n_m_telp = req.body;
123
124     const userData = { nlp, name, email, role_id };
125
126     if (fakultas) userData.fakultas = fakultas;
127     if (jurusan) userData.jurusan = jurusan;
128     if (no_telp) userData.no_telp = no_telp;
129
130     if (password) {
131         try {
132             const hashedPassword = await bcrypt.hash(password, 10);
133             userData.password = hashedPassword;
134         } catch (error) {}
135         return res.status(500).json({ message: 'Gagal mengisi password', error });
136     }
137 }
138
139 const fields = Object.keys(userData);
140 const values = Object.values(userData);
141 const setClause = fields.map(field => `${field} = ?`).join(', ');
142 const query = `UPDATE users SET ${setClause} WHERE id = ?`;
143
144 valent.push(id);
145
146 User.customQuery(query, values, err) => {
147     if (err) return res.status(500).json({ message: 'gagal update user', error: err });
148     res.json({
149         message: 'User berhasil diupdate',
150         updatedUser: userData
151     });
152 }
153
154 exports.deleteUser = (req, res) => {
155     const id = req.params.id;
156     User.deleteUser(id, err) => {
157         if (err) return res.status(500).json({ message: 'gagal menghapus user', error: err });
158         res.json({
159             message: 'User berhasil dihapus',
160             deleteUser: id
161         });
162     }
163 }

```

Gambar 4.21 Lanjutan userController.js

violationController.js merupakan controller utama yang menangani seluruh proses pelanggaran akademik dari tahap investigasi hingga sidang, seperti yang ditampilkan pada Gambar 4.22, Gambar 4.23, Gambar 4.24, Gambar 4.25, dan Gambar 4.26.

[illegible]

Gambar 4.22 violationController.js

Gambar 4.23 Lanjutan violationController.js

Gambar 4.24 Lanjutan violationController.js

Gambar 4.25 Lanjutan violationController.js

```

268     doc.text('File tidak ditemukan');
269   }
270   doc.end();
271   stream.on('finish', async () => {
272     console.log('[EXPORT] PDF finished. Checking file availability...');
273     try {
274       await fs.promises.access(filePath, fs.constants.F_OK);
275       await reportReportModel.saveExporting(filename);
276       res.setHeader('Content-Type', 'application/pdf');
277       res.setHeader('Content-Disposition', `attachment; filename="${filename}"`);
278       res.sendFile(filePath, {
279         root: exportsDir,
280         headers: {
281           'Content-Type': 'application/pdf',
282           'Content-Disposition': `attachment; filename="${filename}"`
283         }, (err) => {
284           if (err) {
285             console.error('[EXPORT] Gagal mengirim file PDF', err);
286             return res.status(500).json({ message: 'Gagal mengirim file hasil export' });
287           }
288         });
289       } catch (fileErr) {
290         console.error('[EXPORT] File tidak ditemukan setelah selesai ditulis', fileErr);
291         res.status(500).json({ message: 'File export PDF tidak ditemukan' });
292       }
293     }
294   });
295   stream.on('error', (err) => {
296     console.error('[EXPORT STREAM ERROR]', err);
297     res.status(500).json({ message: 'Gagal menulis file PDF' });
298   });
299 } catch (err) {
300   console.error('[EXPORT ERROR]', err);
301   res.status(500).json({ message: 'Gagal mengirim hasil' });
302 }
303 }

```

Gambar 4.26 violationController.js

reportController.js menangani proses export laporan dalam format PDF dan tracking file export, implementasinya dapat dilihat pada Gambar 4.27.

```

1  const moment = require('moment');
2  const PDFDocument = require('pdfkit');
3  const fs = require('fs');
4  const path = require('path');
5  const stream = require('stream');
6  const violationModel = require('../models/violationModel');
7  const reportReportModel = require('../models/reportReportModel');
8
9  exports.exportReport = async (req, res) => {
10    try {
11      const { start_date, end_date } = req.body;
12
13      const violations = await violationModel.getViolationsByDate(start_date, end_date);
14
15      const doc = new PDFDocument({ margin: 50 });
16      const filename = `report-${moment().format('YYYY-MM-DD')}.pdf`;
17      const filePath = path.join(__dirname, '..', 'exports', filename);
18      const writeStream = fs.createWriteStream(filePath);
19
20      doc.pipe(writeStream);
21
22      const imagePath = path.join(__dirname, '..', 'assets', 'tailor-logo.png');
23      if (fs.existsSync(imagePath)) {
24        doc.image(imagePath, { fit: [100, 100], align: 'center' });
25      }
26
27      doc.savePage();
28      doc.fontSize(20).text('Laporan Pelanggaran Mahasiswa', { align: 'center' });
29      doc.savePage();
30
31      doc.fontSize(12).text('No', 50, doc.y, { continued: true });
32      doc.text('Nilai', 80, doc.y, { continued: true });
33      doc.text('Detail', 150, doc.y, { continued: true });
34      doc.text('Deskripsi', 250, doc.y, { continued: true });
35      doc.text('Tanggal', 450, doc.y);
36
37      doc.moveDown(0.5);
38      doc.moveTo(50, doc.y).lineTo(150, doc.y).stroke();
39      doc.savePage();
40
41      violations.forEach((v, i) => {
42        doc.text(`${i + 1}`, 50, doc.y, { continued: true });
43        doc.text(`${v.nilai}`, 80, doc.y, { continued: true });
44        doc.text(`${v.detail}`, 150, doc.y, { continued: true });
45        doc.text(`${v.description}`, 250, doc.y, { continued: true });
46        doc.text(`${v.created_at}`, 450, doc.y);
47      });
48
49      doc.end();
50
51      writeStream.on('finish', async () => {
52        await reportReportModel.saveExporting(filename);
53        res.download(filePath);
54      });
55    } catch (err) {
56      console.error(err);
57      res.status(500).json({ message: 'Gagal mengirim laporan' });
58    }
59  }

```

Gambar 4.27 reportController.js

4.6.4. Implementasi Middleware

Middleware diimplementasikan untuk menangani berbagai keperluan seperti autentikasi, otorisasi, validasi, dan upload file.

authMiddleware.js bertanggung jawab untuk memverifikasi token JWT dan memastikan pengguna terautentikasi sebelum mengakses endpoint yang memerlukan autentikasi, seperti yang ditunjukkan pada Gambar 4.28.

```

module.exports = {
  1 const jwt = require('jsonwebtoken');
  2 require('dotenv').config();
  3
  4 exports.verifyToken = (req, res, next) => {
  5   const authHeader = req.headers['authorization'];
  6
  7   if (!authHeader || !authHeader.startsWith('Bearer ')) {
  8     return res.status(403).json({ message: 'Token diperlukan dalam format Bearer token' });
  9   }
  10
  11   const token = authHeader.split(' ')[1];
  12
  13   jwt.verify(token, process.env.SECRET_KEY, (err, decoded) => {
  14     if (err) return res.status(401).json({ message: 'Token tidak valid' });
  15     req.user = decoded;
  16     next();
  17   });
  18 };
}

```

Gambar 4.28 authMiddleware.js

uploadMiddleware.js menangani proses upload file untuk foto profil, dokumentasi pelanggaran, dan file pendukung lainnya, implementasinya dapat dilihat pada Gambar 4.29.

```

module.exports = {
  1 const multer = require('multer');
  2 const path = require('path');
  3 const fs = require('fs');
  4
  5 const storage = multer.diskStorage({
  6   destination: (req, file, cb) => {
  7     const tempPath = 'uploads/temp';
  8     fs.mkdirSync(tempPath, { recursive: true });
  9     cb(null, tempPath);
  10   },
  11   filename: (req, file, cb) => {
  12     const timestamp = Date.now();
  13     const ext = path.extname(file.originalname);
  14     cb(null, `${timestamp}${ext}`);
  15   }
  16 });
  17
  18 const allowedMimeTypes = ['image/png', 'image/jpeg', 'image/jpg'];
  19
  20 const fileFilter = (req, file, cb) => {
  21   if (req.query.type === 'photo') {
  22     if (allowedMimeTypes.includes(file.mimetype)) {
  23       cb(null, true);
  24     } else {
  25       cb(new Error('Format gambar tidak didukung. Hanya PNG, JPG, JPEG yang diizinkan.'), false);
  26     }
  27   } else {
  28     const allowedMimeTypes = [
  29       'application/pdf',
  30       'application/msword',
  31       'application/vnd.openxmlformats-officedocument.wordprocessingml.document',
  32       'text/plain'
  33     ];
  34     if (allowedMimeTypes.includes(file.mimetype)) {
  35       cb(null, true);
  36     } else {
  37       cb(new Error('Format file tidak didukung.'), false);
  38     }
  39   }
  40 };
  41
  42 const upload = multer({ storage, fileFilter });
  43 module.exports = upload;
}

```

Gambar 4.29 uploadMiddleware.js

uploadProfileMiddleware.js khusus menangani upload foto profil pengguna dengan validasi ukuran dan format file yang sesuai, seperti yang ditampilkan pada Gambar 4.30.

```

module.exports = {
  1 const multer = require('multer');
  2 const path = require('path');
  3 const fs = require('fs');
  4
  5 const storage = multer.diskStorage({
  6   destination: (req, file, cb) => {
  7     const profilePath = 'uploads/profile';
  8     fs.mkdirSync(profilePath, { recursive: true });
  9     cb(null, profilePath);
  10   },
  11   filename: (req, file, cb) => {
  12     const timestamp = Date.now();
  13     const ext = path.extname(file.originalname);
  14     cb(null, `${timestamp}${ext}`);
  15   }
  16 });
  17
  18 const allowedMimeTypes = ['image/png', 'image/jpeg', 'image/jpg'];
  19
  20 const fileFilter = (req, file, cb) => {
  21   if (allowedMimeTypes.includes(file.mimetype)) {
  22     cb(null, true);
  23   } else {
  24     cb(new Error('Format gambar tidak didukung. Hanya PNG, JPG, JPEG yang diizinkan.'), false);
  25   }
  26 };
  27
  28 const uploadProfile = multer({ storage, fileFilter });
  29 module.exports = uploadProfile;
}

```

Gambar 4.30 uploadProfileMiddleware.js

4.6.5. Implementasi Route Layer

Route layer mengatur endpoint API yang dapat diakses oleh client. Implementasi routing dilakukan dengan membuat file route yang terpisah untuk setiap fungsi utama sistem.

authRoutes.js mendefinisikan endpoint untuk proses autentikasi seperti login, logout, dan get profile, seperti yang ditunjukkan pada Gambar 4.31.

```
routes > JS authRoutes.js > _
1  const express = require('express');
2  const router = express.Router();
3  const authController = require('../controllers/authController');
4  const { verifyToken } = require('../middleware/authMiddleware');
5
6  router.post('/login', authController.login);
7  router.post('/logout', authController.logout);
8  router.get('/me', verifyToken, authController.getMe);
9
10 module.exports = router;
11
```

Gambar 4.31 authRoutes.js

userRoutes.js berisi endpoint untuk manajemen pengguna termasuk CRUD user dan upload foto profil, implementasinya dapat dilihat pada Gambar 4.32.

```
routes > JS userRoutes.js > _
1  const express = require('express');
2  const router = express.Router();
3  const userController = require('../controllers/userController');
4  const { verifyToken } = require('../middleware/authMiddleware');
5  const uploadProfile = require('../middleware/uploadProfileMiddleware');
6
7  router.get('/', verifyToken, userController.getAllUsers);
8  router.get('/:id', verifyToken, userController.getUser);
9  router.post('/add', verifyToken, userController.addUserManagement);
10 router.put('/:id', verifyToken, userController.updateUserManagement);
11 router.post('/register', verifyToken, uploadProfile.single('photo'), userController.register);
12 router.delete('/:id', verifyToken, userController.deleteUser);
13 router.put('/profile/:id', verifyToken, userController.updateUser);
14 router.put('/profile/:id/photo', verifyToken, uploadProfile.single('photo'), userController.updateUserPhoto);
15
16 module.exports = router;
17
```

Gambar 4.32 userRoutes.js

violationRoutes.js merupakan route utama yang menangani seluruh endpoint pelanggaran akademik, seperti yang ditampilkan pada Gambar 4.33.

```
routes > JS violationRoutes.js > _
1  const express = require('express');
2  const router = express.Router();
3  const upload = require('../middleware/uploadMiddleware');
4  const { verifyToken } = require('../middleware/authMiddleware');
5  const controller = require('../controllers/violationController');
6
7  router.post('/upload', verifyToken, upload.single('file'), (req, res) => {
8    if (!req.file) {
9      return res.status(400).json({ error: 'Gagal upload file' });
10    }
11    res.json({
12      message: 'Upload sementara berhasil',
13      file: {
14        name: req.file.filename,
15        tempPath: req.file.path,
16      },
17    });
18  });
19
20 router.get('/violations', verifyToken, controller.getAll);
21
22 router.post(
23   '/violations',
24   verifyToken,
25   upload.fields([
26     { name: 'hasil_sidang_path', maxCount: 1 },
27     { name: 'notulensi_path', maxCount: 1 },
28     { name: 'photo_path', maxCount: 1 }
29   ]),
30   controller.createWithUpload
31 );
32
33 router.get('/violations/:id', verifyToken, controller.getById);
34 router.get('/violations/student/:nie', verifyToken, controller.getByNIM);
35
36 router.put(
37   '/violations/:id',
38   verifyToken,
39   upload.fields([
40     { name: 'hasil_sidang_path', maxCount: 1 },
41     { name: 'notulensi_path', maxCount: 1 },
42     { name: 'photo_path', maxCount: 1 }
43   ]),
44   controller.update
45 );
46
47 router.put('/violations/:id/status', verifyToken, controller.updateStatusApproval);
48
49 router.delete('/violations/:id', verifyToken, controller.delete);
50
51 router.post('/violations/export', verifyToken, controller.exportReport);
52
53 module.exports = router;
54
```

Gambar 4.33 violationRoutes.js

reports.js menangani endpoint untuk export laporan dan download file hasil sidang, implementasinya dapat dilihat pada Gambar 4.34.

```

routes > # reports.js > -
1 const express = require('express');
2 const router = express.Router();
3 const controller = require('../controllers/reportsController');
4
5 router.post('/export', controller.exportReport);
6
7 module.exports = router;
8

```

Gambar 4.34 reports.js

4.6.6. Implementasi Helper Functions

Helper functions diimplementasikan untuk mendukung operasi-operasi yang sering digunakan dalam sistem. fileHelper.js berisi fungsi-fungsi untuk menangani operasi file seperti upload, delete, dan validasi format file, seperti yang ditunjukkan pada Gambar 4.35. Helper ini memastikan pengelolaan file yang aman dan efisien dalam sistem.

```

1 const fs = require('fs');
2 const path = require('path');
3
4 function moveFileToFinalLocation(tempPath, type) {
5     const folderMap = {
6         'hasil_sidang': 'hasil_sidang',
7         'notulen_sidang': 'notulen_sidang',
8         'photo': 'photo'
9     };
10
11     const folderName = folderMap[type];
12     if (!folderName) {
13         console.warn('X Jenis folder tidak dikenali: $[type]');
14         return null;
15     }
16
17     // Jika file sudah ada di folder final, skip
18     if (tempPath.includes('data_pelanggaran')) {
19         return path.basename(tempPath);
20     }
21
22     if (fs.existsSync(tempPath)) {
23         console.warn('File tidak ditemukan: $[tempPath]');
24         return null;
25     }
26
27     const filename = path.basename(tempPath);
28     const finalFolder = path.join('uploads', 'data_pelanggaran', folderName);
29     const finalPath = path.join(finalFolder, filename);
30
31     fs.mkdirSync(finalFolder, { recursive: true });
32
33     try {
34         fs.renameSync(tempPath, finalPath);
35         return filename;
36     } catch (err) {
37         console.error('X Gagal memindahkan file: $[tempPath] -> $[finalPath]');
38         console.error(err);
39         return null;
40     }
41 }
42
43 function deleteFile(relativePath) {
44     const absolutePath = path.join(__dirname, '..', relativePath);
45     if (!fs.existsSync(absolutePath)) {
46         fs.unlinkSync(absolutePath);
47     }
48 }
49
50 module.exports = {
51     moveFileToFinalLocation,
52     deleteFile
53 };
54

```

Gambar 4.35 Helper.js

4.6.7. Implementasi Server Configuration

Konfigurasi server utama dilakukan melalui file server.js yang menginisialisasi aplikasi Express.js, middleware, dan routing. File ini juga mengatur port listening dan konfigurasi CORS untuk memungkinkan komunikasi dengan client frontend. Implementasi server configuration dapat dilihat pada Gambar 4.36.


```

1 require('dotenv').config();
2 const express = require('express');
3 const cors = require('cors');
4 const path = require('path');
5 const fs = require('fs');
6 const bodyParser = require('body-parser');
7 const authRoutes = require('./routes/authRoutes');
8 const userRoutes = require('./routes/userRoutes');
9 const violationRoutes = require('./routes/violationRoutes');
10 const reportRoutes = require('./routes/reportRoutes');
11 const db = require('./config/db');
12
13 const app = express();
14 const PORT = process.env.PORT || 3001;
15
16 const uploadDir = path.join(__dirname, 'uploads');
17 if (!fs.existsSync(uploadDir)) {
18   fs.mkdirSync(uploadDir);
19   console.log('Folder uploads berhasil dibuat');
20 }
21
22 app.use(cors());
23 app.use(express.json());
24
25 app.use(bodyParser.json());
26 app.use('/uploads', express.static(path.join(__dirname, 'uploads')));
27 app.use('/api/auth', authRoutes);
28 app.use('/api/users', userRoutes);
29 app.use('/api/report', express.static(path.join(__dirname, 'exports')));
30 app.use('/api', violationRoutes);
31
32 db.connect((err) => {
33   if (err) {
34     console.error('Database connection failed:', err);
35     return;
36   }
37   console.log('Connected to MySQL');
38 })
39 app.listen(PORT, () => {
40   console.log('Server running on port ${PORT}');
41 });
42

```

Gambar 4.36 server.js

4.7. Pengujian Sistem

Tahap pengujian sistem merupakan proses evaluasi untuk memastikan bahwa sistem yang telah diimplementasikan dapat berjalan sesuai dengan spesifikasi dan kebutuhan yang telah ditetapkan. Pengujian dilakukan dengan menggunakan metode black box testing untuk menguji fungsionalitas sistem dari perspektif pengguna.

4.7.1. Black box Testing

Black box testing merupakan metode pengujian yang fokus pada fungsionalitas sistem tanpa mempertimbangkan struktur internal kode. Pengujian ini dilakukan untuk memverifikasi bahwa setiap input yang diberikan menghasilkan output yang sesuai dengan ekspektasi. Dalam konteks sistem SiPPAK, black box testing dilakukan terhadap seluruh endpoint API yang telah diimplementasikan.

Pengujian black box dilakukan dengan menguji setiap functional requirement yang telah didefinisikan sebelumnya. Setiap test case dirancang untuk menguji skenario normal maupun skenario error yang mungkin terjadi dalam penggunaan sistem. Pengujian dilakukan menggunakan tools seperti Postman atau Thunder Client untuk menguji endpoint API secara langsung.

4.7.2. Rancangan Test Case

Rancangan test case dibuat berdasarkan functional requirement (FR01-FR07) yang telah didefinisikan sebelumnya. Setiap test case mencakup kondisi input, langkah-langkah pengujian, dan expected output yang diharapkan. Test case dirancang untuk mencakup berbagai skenario penggunaan sistem mulai dari skenario normal hingga skenario edge case.

Test case yang dirancang meliputi pengujian autentikasi pengguna, manajemen pengguna, pengelolaan pelanggaran akademik, proses sidang, dan export laporan. Setiap test case juga mencakup pengujian validasi input, error handling, dan response format yang konsisten.

4.7.3. Hasil Pengujian Black Box Testing

Hasil pengujian black box testing menunjukkan tingkat keberhasilan implementasi sistem dalam memenuhi functional requirement yang telah ditetapkan. Pengujian dilakukan terhadap seluruh endpoint API dengan berbagai skenario input dan kondisi.

Pada Tabel 4.8, ditampilkan hasil pengujian black box testing untuk fitur autentikasi dan manajemen pengguna. Pengujian mencakup proses login, logout, registrasi pengguna, update profil, dan manajemen role pengguna.

Tabel 4.8 Hasil Pengujian Autentikasi dan Manajemen Pengguna

Test Case ID	Nama Test Case	Input	Expected Output	Actual Output	Status
TC01	Login Valid	Email dan password benar	Token JWT dan data user	Token JWT dan data user	Pass
TC02	Login Invalid	Email atau password salah	Error message	Error message	Pass
TC03	Logout	Token valid	Success message	Success message	Pass

Test Case ID	Nama Test Case	Input	Expected Output	Actual Output	Status
TC04	Get Profile	Token valid	Data profil user	Data profil user	Pass
TC05	Register User	Data user lengkap	Success message	Success message	Pass
TC06	Update Profile	Data update valid	Success message	Success message	Pass
TC07	Delete User	ID user valid	Success message	Success message	Pass

Pada Tabel 4.9, ditampilkan hasil pengujian black box testing untuk fitur pengelolaan pelanggaran akademik. Pengujian mencakup pembuatan kasus, update status, proses sidang, dan penutupan kasus.

Tabel 4.9 Hasil Pengujian Pengelolaan Pelanggaran Akademik

Test Case ID	Nama Test Case	Input	Expected Output	Actual Output	Status
TC08	Create Violation	Data pelanggaran lengkap	Success message	Success message	Pass
TC09	Get All Violations	-	List semua pelanggaran	List semua pelanggaran	Pass
TC10	Get Violation by ID	ID violation valid	Data violation	Data violation	Pass
TC11	Update Violation	Data update valid	Success message	Success message	Pass
TC12	Delete Violation	ID violation valid	Success message	Success message	Pass
TC13	Update Status	Status dan ID valid	Success message	Success message	Pass

Test Case ID	Nama Test Case	Input	Expected Output	Actual Output	Status
TC07	Delete User	ID user valid	Success message	Success message	Pass

Pada Tabel 4.10, ditampilkan hasil pengujian black box testing untuk fitur export laporan dan monitoring sistem. Pengujian mencakup export PDF, download file, dan tracking export.

Tabel 4.10 Hasil Pengujian Export Laporan dan Monitoring

Test Case ID	Nama Test Case	Input	Expected Output	Actual Output	Status
TC15	Export Hasil Sidang	ID violation valid	File PDF	File PDF	Pass
TC16	Export Notulensi	ID violation valid	File PDF	File PDF	Pass
TC17	Download File	Filename valid	File download	File download	Pass
TC18	Get Export History	-	List export history	List export history	Pass
TC19	Invalid File Download	Filename invalid	Error message	Error message	Pass

4.7.4. Pengujian Error Handling

Pengujian error handling dilakukan untuk memastikan sistem dapat menangani berbagai kondisi error dengan baik. Pengujian mencakup validasi input, penanganan database error, dan response error yang konsisten.

Pada Tabel 4.11, ditampilkan hasil pengujian error handling untuk berbagai skenario error yang mungkin terjadi dalam sistem.

Tabel 4.11 Hasil Pengujian Error Handling

Test Case ID	Nama Test Case	Input	Expected Output	Actual Output	Status
TC20	Invalid Token	Token tidak valid	401 Unauthorized	401 Unauthorized	Pass
TC21	Missing Required Field	Data tidak lengkap	400 Bad Request	400 Bad Request	Pass
TC22	Duplicate Entry	Data sudah ada	409 Conflict	409 Conflict	Pass
TC23	Resource Not Found	ID tidak ditemukan	404 Not Found	404 Not Found	Pass
TC24	File Upload Error	File tidak valid	400 Bad Request	400 Bad Request	Pass
TC25	Database Connection Error	Database down	500 Internal Server Error	500 Internal Server Error	Pass

4.7.5. Pengujian Performance

Pengujian performance dilakukan untuk mengevaluasi kinerja sistem dalam menangani request dan response. Pengujian mencakup response time, throughput, dan resource utilization.

Hasil pengujian performance menunjukkan bahwa sistem mampu menangani request dengan response time yang baik. Rata-rata response time untuk endpoint API berkisar antara 100-500ms tergantung kompleksitas operasi yang dilakukan.

4.8. Pembahasan

4.8.1. Analisis Black box Testing

Berdasarkan hasil pengujian blackbox testing yang telah dilakukan, dapat disimpulkan bahwa sistem SiPPAK telah berhasil memenuhi seluruh functional requirement yang telah ditetapkan. Dari 25 test case yang dijalankan, semua menunjukkan status "Pass" yang mengindikasikan bahwa sistem berfungsi sesuai dengan ekspektasi. Pengujian autentikasi menunjukkan bahwa sistem mampu menangani proses login dan logout dengan validasi yang tepat, termasuk penanganan kesalahan untuk kredensial yang tidak valid. Fitur manajemen pengguna juga berfungsi dengan baik, memungkinkan operasi CRUD yang lengkap untuk pengelolaan data pengguna dengan berbagai role.

Pengujian pengelolaan pelanggaran akademik menunjukkan hasil yang memuaskan, di mana sistem dapat menangani seluruh siklus pengelolaan kasus mulai dari pembuatan kasus baru, update status, hingga penutupan kasus. Fitur export laporan dan monitoring juga berhasil diimplementasikan dengan baik, memungkinkan staff untuk mengekspor dokumentasi dalam format PDF dan melacak riwayat export. Pengujian error handling menunjukkan bahwa sistem memiliki mekanisme penanganan error yang robust, dengan response code yang sesuai untuk berbagai kondisi error yang mungkin terjadi.

4.8.2. Evaluasi Pemenuhan Functional Requirements

Evaluasi terhadap pemenuhan functional requirements menunjukkan bahwa sistem SiPPAK telah berhasil mengimplementasikan seluruh fitur yang dibutuhkan. FR01 terkait sistem login telah terpenuhi dengan implementasi autentikasi berbasis JWT yang memungkinkan akses sesuai dengan role pengguna. FR02 mengenai pengelolaan data pelanggaran telah diimplementasikan melalui API endpoints yang lengkap dengan operasi CRUD yang komprehensif. FR03 hingga FR07 yang berkaitan dengan pengelolaan kasus pelanggaran, mulai dari pembuatan kasus, penambahan informasi,

pembukaan sidang, hingga pelacakan status, telah berhasil diimplementasikan dengan baik.

Sistem juga telah memenuhi kebutuhan yang diidentifikasi dari hasil wawancara dengan stakeholder. Staf akademik dapat melakukan pencatatan digital dan export laporan pelanggaran, staf kemahasiswaan memiliki akses ke data pelanggaran dan dapat mengelola kronologi penanganan, sedangkan wakil dekan dapat mengakses dashboard monitoring dan validasi pelanggaran. Integrasi data yang diharapkan telah tercapai melalui implementasi database yang terpusat dan API yang terstruktur.

4.8.3. Analisis Kinerja Sistem

Analisis kinerja sistem menunjukkan bahwa SiPPAK memiliki performa yang baik dalam menangani request dan response. Response time rata-rata berkisar antara 100-500ms untuk berbagai endpoint, yang masih dalam batas toleransi yang dapat diterima untuk aplikasi web enterprise. Endpoint yang melibatkan operasi database sederhana seperti login dan get profile menunjukkan response time yang lebih cepat (100-200ms), sementara operasi yang lebih kompleks seperti export PDF dan query data yang melibatkan multiple table memerlukan waktu yang lebih lama (300-500ms).

Sistem juga menunjukkan stabilitas yang baik dalam menangani concurrent request, meskipun pengujian dilakukan dalam lingkungan development dengan beban yang relatif ringan. Penggunaan arsitektur MVC dan implementasi middleware yang tepat berkontribusi pada kinerja sistem yang optimal. Database MySQL yang digunakan juga menunjukkan performa yang stabil dalam menangani operasi CRUD yang dilakukan oleh sistem.

4.8.4. Analisis Keamanan Sistem

Implementasi keamanan sistem telah mempertimbangkan berbagai aspek penting dalam pengembangan aplikasi web. Penggunaan JSON Web Token (JWT) untuk autentikasi memberikan tingkat keamanan yang memadai dengan expiration time yang dapat dikonfigurasi. Middleware autentikasi yang diimplementasikan memastikan bahwa hanya pengguna yang terautentikasi

yang dapat mengakses endpoint yang memerlukan otorisasi. Sistem juga mengimplementasikan role-based access control yang membatasi akses pengguna sesuai dengan peran mereka dalam organisasi.

Validasi input yang diimplementasikan di level controller membantu mencegah berbagai jenis serangan seperti SQL injection dan cross-site scripting. Penggunaan environment variables untuk menyimpan kredensial database dan secret key JWT juga meningkatkan keamanan sistem. File upload middleware yang diimplementasikan memiliki validasi format dan ukuran file yang membantu mencegah upload file yang berbahaya ke sistem.

4.8.5. Keterbatasan dan Saran Pengembangan

Meskipun sistem telah berhasil memenuhi functional requirements yang ditetapkan, terdapat beberapa keterbatasan yang perlu diperhatikan untuk pengembangan selanjutnya. Sistem saat ini hanya mendukung tiga role pengguna yang mungkin perlu diperluas untuk mengakomodasi struktur organisasi yang lebih kompleks. Fitur notifikasi real-time belum diimplementasikan, yang dapat meningkatkan responsivitas sistem dalam menangani perubahan status kasus.

Untuk pengembangan selanjutnya, disarankan untuk mengimplementasikan fitur dashboard yang lebih interaktif dengan visualisasi data yang lebih menarik. Integrasi dengan sistem akademik yang sudah ada juga dapat meningkatkan efisiensi dalam pengelolaan data mahasiswa. Implementasi sistem backup dan recovery yang lebih robust juga diperlukan untuk menjamin ketersediaan data dalam kondisi darurat. Penambahan fitur audit trail yang lebih detail dapat membantu dalam pelacakan perubahan data dan meningkatkan akuntabilitas sistem.

4.8.6. Implikasi Terhadap Organisasi

Implementasi sistem SiPPAK diharapkan dapat memberikan dampak positif yang signifikan terhadap pengelolaan pelanggaran akademik di institusi pendidikan. Sistem digital yang terintegrasi dapat meningkatkan efisiensi dalam pencatatan dan pelacakan kasus pelanggaran, mengurangi

waktu yang diperlukan untuk mengakses informasi, dan meningkatkan akurasi data. Otomatisasi proses yang diimplementasikan dapat mengurangi beban kerja staff dan memungkinkan fokus yang lebih baik pada aspek pembinaan mahasiswa.

Dari perspektif manajemen, sistem ini menyediakan data yang lebih terstruktur dan mudah dianalisis untuk pengambilan keputusan kebijakan. Dashboard monitoring yang tersedia dapat membantu pimpinan dalam mengidentifikasi tren pelanggaran dan mengambil tindakan preventif yang tepat. Dokumentasi yang lebih baik dan terstruktur juga dapat meningkatkan transparansi dan akuntabilitas dalam penanganan kasus pelanggaran akademik.