Pengembangan Layanan Autentikasi dan Manajemen Akses Menggunakan Pendekatan *Waterfall* untuk Integrasi Aplikasi Fakultas berbasis *Microservice* (Studi Kasus: Fakultas Rekayasa Industri)

1st Rizki Al Fatwa
Fakultas Rekayasa Industri
Universitas Telkom
Bandung,Indonesia
rizkialfatwa@student.telkomuniversity.ac.id

2nd Ekky Novriza Alam, S.Kom., M.T..
Fakultas Rekayasa Industri
Universitas Telkom
Bandung, Indonesia
ekkynovrizalam@telkomuniversity.ac.id

3rd Nur Ichsan Utama, S.T., M.T., Ph.D.,
Fakultas Rekayasa Industri
Universitas Telkom
Bandung, Indonesia
nichsan@telkomuniversity.ac.id

Abstrak— Penelitian ini bertujuan mengembangkan layanan autentikasi dan manajemen akses berbasis microservices untuk mengatasi masalah redundansi data master user dan ketidakseimbangan manajemen akses di Fakultas Rekayasa Industri (FRI) Telkom University. Sistem ini menggunakan RESTful API dengan protokol HTTP, terbagi menjadi API public untuk aplikasi klien dan API private untuk pengelolaan melalui dashboard. Keamanan akses dijaga dengan Bearer token. Pengembangan dilakukan dengan metode Waterfall. Pengujian menunjukkan bahwa semua fitur utama berfungsi dengan baik, namun pengujian keamanan menggunakan OWASP ZAP mengidentifikasi beberapa kerentanan serius seperti Cloud Metadata Exposure dan SQL Injection, yang memerlukan mitigasi lebih lanjut melalui validasi input dan konfigurasi server yang lebih aman.

Kata kunci— autentikasi, manajemen akses, microservices, redundansi data, metode waterfall

I. PENDAHULUAN

Dalam era digital yang semakin berkembang. Kebutuhan informasi dalam kegiatan perkuliahan menjadi semakin cepat. Teknologi informasi semakin dibutuhkan dalam meningkatkan efisiensi kegiatan perkuliahan sehingga pengolahan informasi berlangsung dengan cepat dan akurat. Oleh karena itu banyak institusi pendidikan tinggi yang telah beralih menggunakan sistem informasi untuk mendukung manajemen administrasi perkuliahan mereka.

Telkom University merupakan kampus swasta yang berlokasi di kota Bandung. Telkom University memiliki tujuh fakultas salah satu diantaranya yaitu Fakultas Rekayasa Industri (FRI). FRI memiliki 3 jurusan yaitu Teknik Industri, Sistem Informasi dan Teknik Logistik. Untuk mendukung kegiatan perkuliahan seperti administrasi, FRI telah mengembangkan berbagai macam aplikasi yang digunakan untuk membantu mempercepat berbagai proses administrasi. Berdasarkan wawancara kepada pihak pengelola aplikasi-aplikasi yang terdapat pada lingkungan FRI dapat diketahui bahwa setiap aplikasi yang ada pada FRI terhubung dengan Single Sign On (SSO) Telkom sebagai sistem autentikasi dan setiap aplikasi tersebut memiliki data master user dan manajemen aksesnya masing-masing. Namun terdapat redudansi data master user pada aplikasi-aplikasi yang dikelola oleh FRI yang menyebabkan beberapa masalah yang diuraikan pada Tabel 1.

TABEL 1 URAIAN MASALAH

1	Ketika ada *user* baru yang ditambahkan ke sistem atau aplikasi, data *master user* tersebut perlu disinkronkan ke berbagai aplikasi yang digunakan oleh FRI.
2	Ketidaksesuaian sistem keamanan di setiap aplikasi sehingga menjadi sulit untuk mengelola dan memantau akses pengguna secara efektif.
3	Pengelolaan akses yang tidak seragam pada aplikasi menimbulkan kesulitan dalam mengontrol siapa saja yang memiliki akses.
4	Pengelolaan data yang tidak efektif pada setiap aplikasi yang ada apabila terjadi perubahan struktur data pada *Single Sign On* Telkom.

Masalah serupa juga terjadi pada penelitian yang dilakukan oleh [1] sehingga solusi yang diusulkan berupa pengembangan sistem *single sign on* agar akses pada aplikasi yang ada menjadi terpusat dalam satu sistem. Penelitian serupa juga dilakukan oleh [2] dimana pengguna harus mengelola banyak identitas untuk berbagai layanan sehingga mengembangkan sistem *single-sign on* dan akses manajemen untuk mengatasi masalah tersebut.

Berdasarkan penelitian yang dilakukan oleh [3] metode *Waterfall* dapat meningkatkan keberhasilan proyek dengan menyediakan pendekatan yang terstruktur dan linier. Berdasarkan penelitian yang dilakukan oleh [4] model *Waterfall* memberikan struktur dan dokumentasi yang baik, serta komitmen yang jelas terhadap tujuan proyek. Hal ini membuatnya cocok untuk proyek-proyek dengan persyaratan yang stabil dan tidak terlalu membutuhkan fleksibilitas.

Berdasarkan permasalahan dan penelitian yang telah dilakukan sebelumnya, peneliti bertujuan untuk mengembangkan layanan autentikasi dan manajemen akses berbasis *microservices* yang akan menjadi penghubung sekaligus berperan sebagai data master user antara aplikasi-aplikasi yang ada di lingkungan FRI. Diharapkan, penggunaan arsitektur *microservices* ini akan memberikan fleksibilitas dalam penyesuaian dan skalabilitas sistem tanpa mengganggu keseluruhan.

Microservices juga mempermudah pengelolaan data master user secara lebih terstruktur, mengurangi risiko redundansi, serta memastikan konsistensi data di seluruh aplikasi yang ada pada lingkungan FRI. Pemeliharaan dan pembaruan sistem juga menjadi lebih mudah karena perubahan dapat dilakukan tanpa memengaruhi keseluruhan aplikasi yang ada di lingkungan FRI. Dengan manfaat-manfaat ini, diharapkan FRI dapat mengatasi masalah redundansi data, meningkatkan integrasi antar aplikasi, dan menyediakan layanan yang lebih responsif serta andal bagi penggunanya.

II. KAJIAN TEORI

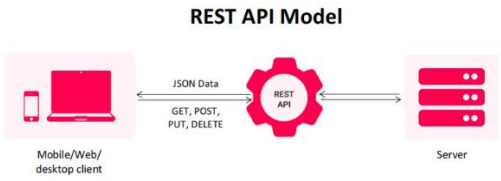
A. Manajemen Akses

Manajemen akses adalah proses pengelolaan hak dan izin pengguna dalam mengakses sumber daya sistem informasi, seperti data, aplikasi, dan layanan. Proses ini mencakup identifikasi, autentikasi, dan otorisasi pengguna, serta pemantauan aktivitas mereka untuk memastikan bahwa akses yang diberikan sesuai dengan kebijakan keamanan yang telah ditetapkan [5]

B. *Application Programming Interface (API)*

API adalah seperangkat aturan dan protokol yang memungkinkan satu perangkat lunak untuk berkomunikasi dengan perangkat lunak lainnya. Ini dapat diibaratkan sebagai sebuah kontrak antara dua pihak: satu pihak meminta data atau layanan, sementara pihak lainnya memberikan data atau layanan tersebut sesuai dengan permintaan.

Representational State Transfer API (REST API) adalah salah satu bentuk API yang paling umum digunakan saat ini. REST API bekerja dengan menggunakan protokol HTTP untuk melakukan operasi CRUD (*create, read, update, delete*) pada sumber daya yang disediakan oleh server. Setiap sumber daya diwakili oleh URL, dan operasi terhadap sumber daya tersebut dilakukan dengan menggunakan metode HTTP seperti GET, POST, PUT, dan DELETE.



GAMBAR 1 CARA KERJA REST API [8]

C. *Keamanan API*

API Security merujuk pada praktik dan teknologi yang digunakan untuk melindungi *Application Programming Interfaces* (API) dari ancaman dan serangan. Dengan meningkatnya penggunaan API dalam aplikasi web, *mobile*, dan IoT, risiko keamanan seperti akses tidak sah, eksposur data berlebihan, dan serangan injeksi menjadi lebih umum. Oleh karena itu, penting untuk menerapkan langkah-langkah keamanan seperti autentikasi yang kuat, kontrol akses, dan pemantauan untuk melindungi API dari potensi ancaman.[9]. Salah satu cara untuk mengamankan API adalah dengan menerapkan autentikasi dan otorisasi pada server API. Otentikasi merupakan proses untuk memverifikasi identitas pengguna sebelum mereka dapat mengakses sistem atau data. Sedangkan otorisasi merupakan hak akses pengguna terhadap data tertentu, seperti mengizinkan atau menolak akses ke fitur tertentu dalam aplikasi. Tanpa otorisasi, pengguna yang terautentikasi tidak dapat melakukan tindakan yang tidak diizinkan [10]

Salah satu cara untuk menerapkan sistem otentikasi dan otorisasi adalah dengan menggunakan *JSON Web Tokens* (JWT) pada server.

1) *JSON Web Tokens (JWT)*
   JSON Web Token (JWT) adalah standar terbuka (RFC 7519) yang digunakan untuk berbagi informasi secara aman antara pihak-pihak melalui objek JSON. JWT terdiri dari tiga bagian: header, payload, dan signature, yang di-*encode* dalam format *Base64* [9]. JWT sering digunakan dalam otentikasi dan otorisasi, memungkinkan aplikasi untuk memverifikasi identitas pengguna dan memberikan akses ke sumber daya tertentu tanpa perlu menyimpan sesi di server. JWT memiliki 3 struktur utama yaitu [9]:
2) *Refresh JSON Web Token (JWT)*
   Refresh JWT adalah proses untuk mendapatkan token baru setelah token asli (*access* token) kedaluwarsa, tanpa memerlukan pengguna untuk melakukan otentikasi ulang [9].

D. *Microservices*

Microservices adalah sebuah pendekatan arsitektur perangkat lunak yang memecah sistem monolitik besar menjadi layanan-layanan kecil dan independen yang dapat dikembangkan, dikelola, dan dioperasikan secara terpisah. Setiap layanan biasanya menjalankan fungsionalitas spesifik dan berkomunikasi dengan layanan lain melalui antarmuka yang ringan, seperti API berbasis HTTP. ,Pengembangan Independen dan Skalabilitas: Setiap *microservice* bisa dikembangkan, diuji, dan dideploy secara independen, yang memungkinkan skalabilitas tinggi dan fleksibilitas dalam pengembangan. Misalnya, pada aplikasi marketplace berbasis web, arsitektur *microservices* memungkinkan pengelolaan berbagai layanan seperti pembayaran, pencarian produk, atau autentikasi pengguna secara terpisah, sehingga tidak saling bergantung satu sama lain [11].

E. *Unified Model Language (UML)*

Unified Modeling Language (UML) merupakan bahasa pemodelan standar yang digunakan dalam rekayasa perangkat lunak untuk memvisualisasikan desain dan struktur sistem perangkat lunak. UML menyediakan cara untuk membuat cetak biru dari sebuah sistem, termasuk strukturnya, perilakunya, dan interaksinya. Tujuan utama UML adalah membantu pengembang dalam menentukan, memvisualisasikan, membangun, dan mendokumentasikan artefak sistem perangkat lunak[12].

F. *Waterfall Methodology*

Metodologi *Waterfall* adalah salah satu pendekatan paling klasik dalam pengembangan perangkat lunak yang mengikuti proses berurutan. Setiap fase dalam metodologi ini dimulai hanya setelah fase sebelumnya selesai, sehingga memberikan struktur yang jelas dan terdefinisi dengan baik. Metodologi ini sering kali diibaratkan sebagai "air terjun" karena langkah-langkahnya yang mengalir turun tanpa adanya umpan balik ke langkah sebelumnya [4]

Metodologi *Waterfall* terdiri dari beberapa tahapan berurutan yang mencakup:

a. *Requirement Analysis*,

b. *Design*,

c. *Implementation*,.

d. *Testing*,

e. *Deployment*,

f. *Maintance*,

GAMBAR 2 ALUR *WATERFALL* [14]

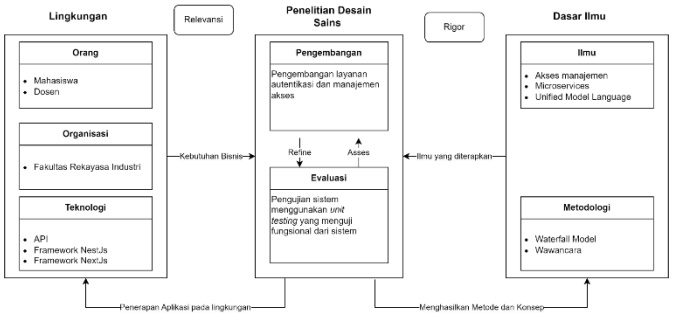
G. *Software Testing*

Pengujian perangkat lunak didefinisikan sebagai proses sistematis untuk mengevaluasi dan memverifikasi bahwa aplikasi perangkat lunak atau sistem berfungsi seperti yang diharapkan. Proses ini melibatkan eksekusi perangkat lunak untuk mengidentifikasi perbedaan antara hasil aktual dan hasil yang diharapkan serta memastikan bahwa perangkat lunak tersebut bebas dari cacat. Pengujian sangat penting dalam memastikan kualitas, keandalan, dan kinerja perangkat lunak, dan dapat dilakukan secara manual atau otomatis [15].

III. METODE

A. *Model Konseptual*

Model konseptual dapat diartikan sebagai kerangka kerja yang menggambarkan hubungan antara *variabel* atau konstruk yang ingin diteliti dalam suatu penelitian. Model ini digunakan untuk mengarahkan penyusunan pertanyaan penelitian, merumuskan hipotesis, serta menentukan metode analisis yang akan digunakan. Model konseptual pada penelitian ini digambarkan pada **Kesalahan! Sumber referensi tidak ditemukan.**dibawah.

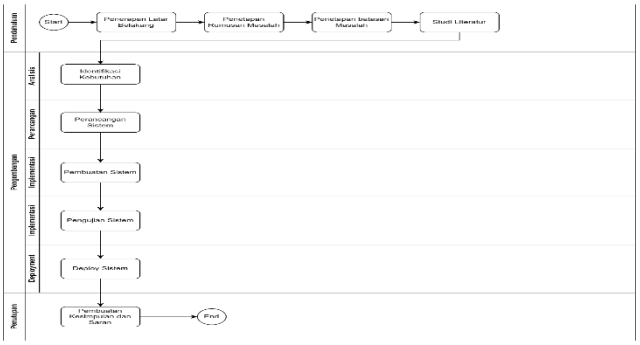


GAMBAR 3 MODEL KONSEPTUAL

Model konseptual pada gambar 3 menampilkan kerangka kerja untuk Penelitian Desain Sains dalam konteks pengembangan layanan autentikasi dan manajemen akses. Diagram ini dibagi menjadi tiga bagian utama: Lingkungan, Penelitian Desain Sains, dan Dasar Ilmu, masing-masing menggambarkan komponen penting yang mendukung proses penelitian dan pengembangan sistem ini.

B. *Sistematika Penelitian*

Pada Gambar 4 diatas merupakan gambaran dari alur sistematika penelitian. Sistematika penelitian menggunakan model *Waterfall*.



GAMBAR 4 SISTEMATIKA PENELITIAN

1) Tahap Pendahuluan
2) Tahap Pengembangan
3) Tahap Penutup

IV. HASIL DAN PEMBAHASAN

A. *Analisa Kebutuhan (Requirement Analysis)*

Berikut ini adalah hasil analisis kebutuhan yang diuraikan dalam Tabel 2 Analisis GAP dengan menggunakan pendekatan analisis GAP.

TABEL 2 ANALISIS GAP

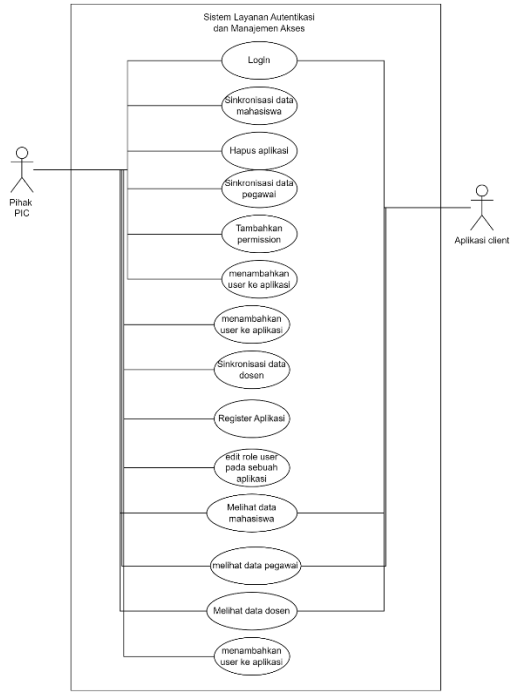
Aspek	Keadaan eksisting	Kondisi yang diharapkan	GAP	Solusi yang diusulkan
Data master user	Data master user terpisah pada setiap aplikasi. Setiap aplikasi menyimpan dan mengelola data user secara mandiri.	Data master user terpusat, hanya ada satu sumber kebenaran untuk seluruh aplikasi.	Data master user terpisah vs. Data master user terpusat	Implementasi sistem manajemen data master user terpusat untuk mengurangi redundansi dan meningkatkan konsistensi data.
Sistem keamanan	Standar keamanan berbeda-beda di setiap aplikasi, sehingga sulit untuk mengelola dan memantau akses pengguna.	Standar keamanan seragam di seluruh aplikasi untuk pengelolaan dan pemantauan akses yang lebih efektif.	Standar keamanan beragam vs. Standar keamanan seragam	Penyeragaman standar keamanan di seluruh aplikasi agar lebih mudah dikelola dan dipantau.
Pengelolaan Akses	Pengelolaan akses tidak seragam di setiap aplikasi,	Pengelolaan akses yang terintegrasi dan seragam di	Pengelolaan akses tidak seragam vs.	Pengembangan sistem manajemen akses terintegrasi

Aspek	Keadaan eksisting	Kondisi yang diharapkan	GAP	Solusi yang diusulkan
	menyulitkan kontrol dan pengawasan akses pengguna.	seluruh aplikasi, memudahkan kontrol dan monitoring akses pengguna.	Pengelolaan akses terintegrasi	untuk memudahkan kontrol dan pengawasan akses pengguna.
Pengelolaan Data	Pengelolaan data tidak efektif, bergantung pada struktur data yang berbeda-beda. Perubahan pada SSO Telkom mempengaruhi setiap aplikasi secara manual.	Struktur data yang fleksibel dan dapat disesuaikan dengan perubahan tanpa mempengaruhi operasi sehari-hari aplikasi.	Pengelolaan data tidak efektif vs. Struktur data yang fleksibel	Penyesuaian struktur data menjadi lebih fleksibel untuk mengakomodasi perubahan dari SSO Telkom tanpa mempengaruhi aplikasi yang ada.

B. *Perancangan*

1) Use Case Diagram

Use case diagram dapat menggambarkan aktor dan fungsional atau *use case* yang terdapat pada layanan autentikasi dan manajemen akses secara umum. Pada penelitian ini peneliti menggunakan *use case* diagram untuk melakukan validasi kesesuaian *microservice* yang akan dibentuk. Berikut merupakan penjabaran dari use case diagram layanan autentikasi dan manajemen akses yang telah terbentuk berdasarkan kebutuhan pengguna.



GAMBAR 5 *USE CASE DIAGRAM*

Use case scenario dapat dilihat pada Tabel 3 *Use case scenario* berikut.

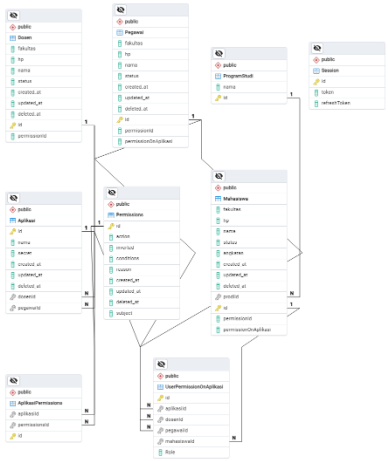
TABEL 3 *USE CASE SCENARIO*

Use case	Aktor	Prekondisi	Deskripsi langkah	Postkondisi
Login	Pihak PIC, Aplikasi pengguna	Pengguna memiliki akun terdaftar	1. Masukkan username dan password 2. Sistem memvalidasi 3. Akses diizinkan atau ditolak	Pengguna masuk ke sistem atau menerima kesalahan
Sinkronisasi data mahasiswa	Pihak PIC	Akses ke database mahasiswa	1. Pihak PIC memulai sinkronisasi data. 2. Sistem mengambil data terbaru. 3. Sistem menyimpan data yang diperbarui	Data mahasiswa diperbarui dalam sistem.
Register aplikasi	Pihak PIC	Pihak PIC memiliki akses ke pengaturan sistem	1. Pihak PIC mengisi form registrasi aplikasi 2. Sistem menyimpan data aplikasi	Aplikasi baru terdaftar dalam sistem.
Tambahkan permission	Pihak PIC	Aplikasi terdaftar di sistem	1. Pihak PIC memilih aplikasi dan permission. 2. Sistem menyimpan perubahan permission	Permission untuk aplikasi diperbarui
Menambahkan *user* ke aplikasi	Pihak PIC	*User* dan aplikasi terdaftar di sistem	1. Pihak PIC menambahkan *user* ke aplikasi 2. sistem menyimpan perubahan *role*	*User* mendapatkan akses ke aplikasi
Edit role user pada aplikasi	Pihak PIC	*User* sudah memiliki *role* di aplikasi	1. Pihak PIC memilih *user* dan merubah *role*-nya 2. Sistem menyimpan perubahan *role.*	*Role user* pada aplikasi diperbarui

Use case	Aktor	Prekondisi	Deskripsi langkah	Postkondisi
Melihat data mahasiswa/dose/pegawai	Pihak PIC, penggunaa terautorisasi	Data mahasiswa tersedia di sistem	1. pengguna memilih untuk melihat data mahasiswa 2. sistem manampilkan data yang relevan	Pengguna melihat data mahasiswa

2) Entity Relational Diagram

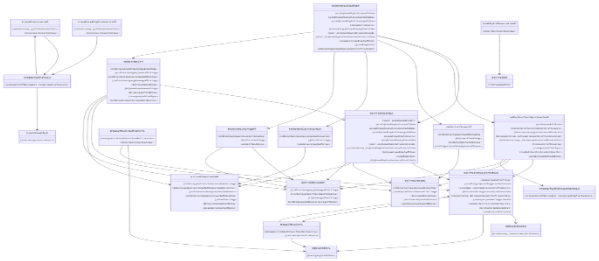
Entity Relational Diagram digunakan untuk menjelaskan skema pada sebuah database. *Entity Relational Diagram* dirancang berdasarkan kebutuhan yang sesuai dengan penjabaran pada *use case diagram* yang sudah dibuat. Gambar 6 merupakan rancangan *Entity Relational Diagram* pada sistem. Gambar 6 memperlihatkan data-data yang diperlukan untuk mendukung pengembangan layanan autentikasi dan manajemen akses beserta hubungan antara data-data tersebut.



GAMBAR 6 *ENTITY RELATIONAL DIAGRAM*

1) Class Diagram

Pada Gambar 7 merupakan *class diagram* pada sistem layanan auatentikasi dan manajemen akses.

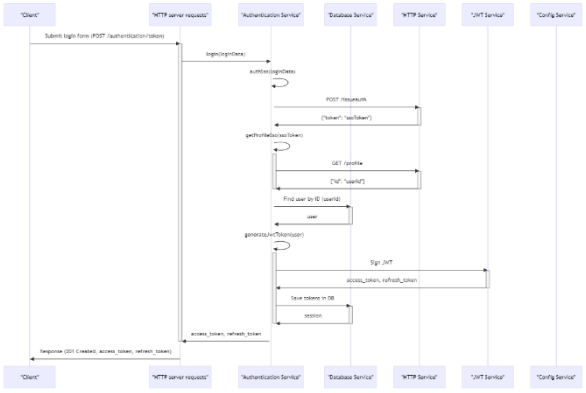


GAMBAR 7 *CLASS DIAGRAM*

2) Sequence Diagram

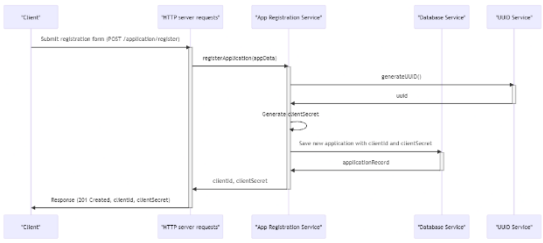
Sequence diagram digunakan untuk memetakan alur interaksi antar objek dalam sebuah sistem, dengan menampilkan urutan pesan atau aksi yang terjadi seiring waktu. Diagram ini membantu menggambarkan bagaimana berbagai komponen atau entitas dalam sistem berkomunikasi satu sama lain dalam konteks sebuah skenario tertentu. *Sequence diagram* berfokus pada waktu dan urutan pesan yang dikirimkan antar objek, dan sangat bermanfaat untuk memahami detail implementasi, terutama pada tahap desain sistem.

a) Login



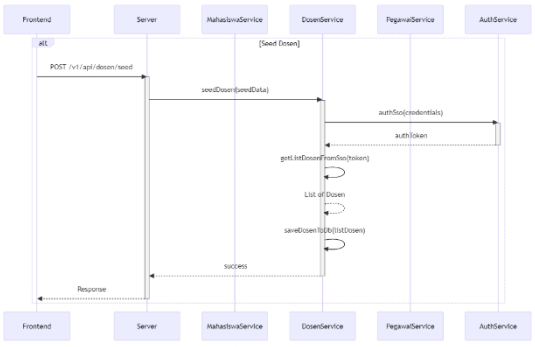
GAMBAR 8 LOGIN

b) Register aplikasi



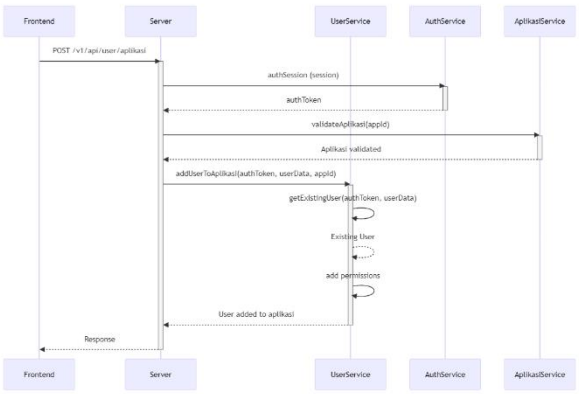
GAMBAR 9 REGISTER APLIKASI

c) Sinkronisai data mahasiswa,dosen,pegawai



GAMBAR 10 SINKRONISASI DATA MAHASISWA

d) Memberikan akses aplikasi ke pengguna



GAMBAR 11 MEMBERIKAN AKSES APLIKASI KE PANGGUNA

C. *Implementasi*

Setelah proses perancangan, selanjutnya masuk ke tahap pembentukan layanan autentikasi dan manajemen akses. Pembentukan sistem akan dibagi menjadi 2 tahapan, yaitu *back-end* dan *front-end.*

1) Implementasi Layanan API

Layanan autentikasi dan manajemen akses menggunakan *RestfulAPI* melalui *protocol HTTP* sebagai *API gateway*. *API* yang dibentuk terdapat empat jenis method yaitu *GET*, *POST*, *PUT*, dan *DELETE*. Pembentukan *API* menggunakan *framework* NestJs. *API* dibagi menjadi 2 kategori

public dan *private*, *public* digunakan oleh aplikasi klien yang akan menggunakan layanan dan *private* digunakan untuk *front-end* dashboard sistem. berikut daftar *API* yang telah dibuat.

TABEL 4 DAFTAR *API PRIVATE*

No	API	Deskripsi	Method	Header	Route
1	Auth token	Mendapatkan token untuk	POST	-	/authentication/token
2	Refresh token	Mendapatkan token baru	POST	-	/authentication/refresh-token
3	Seed staff dosen	Sinkronisasi data pegawai dan dosen	POST	-	/authentication/seed
4	Create permission	Menambahkan permission	POST	Bearer token	/authentication/permission
5	Get list permission	Mendapatkan data permission	GET	Bearer token	/authentication/permission
6	Delete permission	Menghapus permission	DELETE	Bearer token	/authentication/permission/{permissionId}
7	My profile	Mendapatkan data profile login	GET	Bearer token	/authentication/my-profile
8	Edit user permission	Mengubah permission user	PUT	Bearer token	/authentication/edit-user-permission/{type}
9	Delete user permission	Menghapus permission pada user	DELETE	Bearer token	/authentication/delete-user-permission/{type}
10	Get list dosen	Mendapatkan daftar dosen	GET	Bearer token	/dosen
11	Seed dosen	Sinkronisasi data dosen	POST	Bearer token	/seed/dosen
12	Get one dosen	Mendapatkan data dosen	GET	Bearer token	/dosen/{id}
13	Get list mahasiswa	Mendapatkan daftar mahasiswa	GET	Bearer token	/mahasiswa
14	Seed mahasiswa	Sinkornisasi data mahasiswa	POST	Bearer token	/mahasiswa/seed
15	Get list prodi	Daftar program studi	Get	-	/mahasiswa/prodi
16	Get list pegawai	Daftar pegawai	GET	Bearer token	/pegawai
17	Seed staff	Sinkronisasi data pegawai	POST	Bearer token	/pegawai/seed
18	Get one pegawai	Data pegawai	GET	Bearer token	/pegawai/{id}
19	List aplikasi	Daftar aplikasi	GET	Bearer token	/aplikasi
20	Create aplikasi	Mendaftarkan aplikasi	POST	Bearer token	/aplikasi
21	Get my aplikasi	Melihat daftar aplikasi yang user punya	GET	Bearer token	/aplikasi/my-aplikasi
22	Delete my aplikasi	Menghapus aplikasi yang dimiliki user	DELETE	Bearer token	/aplikasi/my-aplikasi/{aplikasiId}
23	Add dosen to aplikasi	Memberikan hak akses dosen ke aplikasi	POST	Bearer token	/aplikasi/add-user/dosen
24	Add pegaw	Memberikan hak	POST	Bear er	/aplikasi/add-user/pegawai

No	API	Deskripsi	Method	Header	Route
	ai to aplikasi	akses pegawai ke aplikasi		token	
25	Add mahasiswa to aplikasi	Memberikan hak akses mahasiswa ke aplikasi	POST	Bearer token	/aplikasi/add-user/mahasiswa
26	Delete user on aplikasi	Menghapus hak akses user ke palikasi	DELETE	Bearer token	/aplikasi/delete/user-permission
27	Edit user pemission on aplikasi	Mengubah role user pada aplikasi tertentu	PUT	Bearer token	/aplikasi/user-permission
28	List user permission on aplikasi	Daftar user pada sebuah aplikasi	GET	Bearer token	/aplikasi/user-permission/{aplikasiId}

TABEL 5 DAFTAR API *PUBLIC*

No	API	Deskripsi	Method	Header	Route
1	Login	Akses token untuk aplikasi client	POST	-	/v1/login
2	Refresh token	Akses token baru	POST	-	/v1/refresh-token
3	Get all mahasiswa	Daftar mahasiswa	GET	Bearer token	/v1/get-all-mahasiswa
4	Get all dosen	Daftar dosen	GET	Bearer token	/v1/get-all-dosen
5	Get all pegawai	Daftar pegawai	GET	Bearer token	/v1/get-all-pegawai
6	Get one mahasiswa	Data mahasiswa	GET	Bearer token	/v1/get-one-mahasiswa
7	Get one dosen	Data mahasiswa	GET	Bearer token	/v1/get-one-dosen
8	Get one pegawai	Data pegawai	GET	Bearer token	/v1/get-one-pegawai
9	My-profile	Data user login	GET	Bearer token	/v1/me

2) Functional Testing

Pengujian unit testing bertujuan untuk mengukur fungsionalitas layanan autentikasi dan manajemen akses dengan melakukan konsumsi *API* dan memverifikasi apakah *body* dan *header* sesuai dengan ekspektasi yang diharapkan. Seperti yang tertera pada Tabel 4 Daftar *API private*, Test Class menunjukkan fungsionalitas yang akan diuji. *Test Case* adalah kondisi yang dijalankan sebagai input terhadap suatu fungsionalitas. Hasil pengujian ini dinyatakan valid jika sesuai dengan ekspektasi, atau tidak valid jika tidak sesuai dengan ekspektasi yang diharapkan.

TABEL 6 HASIL PENGUJIAN PADA *DASHBOARD*

No	Tes Class	Tes case	Hasil yang diharapkan	Hasil yang ditemukan	Status
1	Masuk ke dashboard	Pengguna dapat masuk ke dashboard	Pengguna berhasil masuk ke dashboard	Sesuai	Berhasil
2	Menampilkan data mahasiswa	Pengguna dapat melihat data mahasiswa	Data mahasiswa tampil dengan benar	Sesuai	Berhasil
3	Menampilkan data dosen	Pengguna dapat melihat data dosen	Data dosen tampil dengan benar	Sesuai	Berhasil
4	Menampilkan data pegawai	Pengguna dapat melihat data pegawai	Data pegawai tampil dengan benar	Sesuai	Berhasil
5	Mensinkronisasikan data mahasiswa	Pengguna dapat mensinkronisasikan data mahasiswa	Data mahasiswa berhasil disinkronisasikan	Sesuai	Berhasil
6	Mensinkronisasikan data dosen	Pengguna dapat mensinkronisasikan data dosen	Data dosen berhasil disinkronisasikan	Sesuai	Berhasil
7	Mensinkronisasikan data pegawai	Pengguna dapat mensinkronisasikan data pegawai	Data pegawai berhasil disinkronisasikan	Sesuai	Berhasil
8	Menampilkan daftar aplikasi	Pengguna dapat melihat daftar aplikasi	Daftar aplikasi tampil dengan benar	Sesuai	Berhasil
10	Menampilkan daftar user yang ada pada aplikasi	Pengguna dapat melihat daftar user pada aplikasi	Daftar user tampil dengan benar	Sesuai	Berhasil
11	Menampilkan halaman setting	Pengguna dapat melihat halaman setting	Halaman setting tampil dengan benar	Sesuai	Berhasil
12	Mendaftarkan aplikasi baru	Pengguna dapat mendaftarkan aplikasi	Aplikasi baru berhasil didaftarkan	Sesuai	Berhasil
13	Menambahkan permission	Pengguna dapat	Permission	Sesuai	Berhasil

		menambahkan permission	berhasil ditambahkan		
14	Menampilkan daftar aplikasi user	Pengguna dapat melihat aplikasi yang miliknya	Daftar aplikasi user tampil dengan benar	Sesuai	Berhasil
15	Menghapus aplikasi yang terdaftar	Pengguna dapat menghapus aplikasi miliknya	Aplikasi yang terdaftar berhasil dihapus	Sesuai	Berhasil
16	Menambahkan mahasiswa ke aplikasi	Pengguna dapat menambahkan mahasiswa ke aplikasi	Mahasiswa berhasil ditambahkan ke aplikasi	Sesuai	Berhasil
17	Menambahkan dosen ke aplikasi	Pengguna dapat menambahkan dosen ke aplikasi	Dosen berhasil ditambahkan ke aplikasi	Sesuai	Berhasil
18	Menambahkan pegawai ke aplikasi	Pengguna dapat menambahkan pegawai ke aplikasi	Pegawai berhasil ditambahkan ke aplikasi	Sesuai	Berhasil
19	Menghapus mahasiswa/dosen/pegawai dari aplikasi	Pengguna dapat menghapus mahasiswa/dosen/pegawai dari aplikasi miliknya	Mahasiswa/ dosen/ pegawai berhasil dihapus dari aplikasi	Sesuai	Berhasil
20	Mengubah role mahasiswa/dosen/pegawai pada aplikasi	Pengguna dapat mengubah mahasiswa/dosen/pegawai	Role mahasiswa/dosen/pegawai berhasil diubah pada aplikasi	Sesuai	Berhasil

TABEL 7 HASIL PENGUJIAN API UNTUK APLIKASI CLIENT

No	Tes Class	Tes case	Hasil yang diharapkan	Hasil yang ditemukan	Status
1	Mendapatkan token akses dan refresh token	Pengguna mendapatkan token akses dan refresh token	Token akses dan dan refresh token berhasil didapatkan	Sesuai	Berhasil
3	Mendapatkan token akses dan refresh token yang baru	Pengguna mendapatkan token akses dan refresh token yang baru	Token akses dan dan refresh token yang baru berhasil didapatkan	Sesuai	Berhasil
4	Mendapatkan daftar mahasiswa	Pengguna mendapatkan data daftar mahasiswa	Data daftar mahasiswa berhasil didapatkan	Sesuai	Berhasil
5	Mendapatkan daftar pegawai	Pengguna mendapatkan data daftar pegawai	Data daftar pegawai berhasil didapatkan	Sesuai	Berhasil
6	Mendapatkan daftar dosen	Pengguna mendapatkan data daftar dosen	Data daftar dosen berhasil didapatkan	Sesuai	Berhasil
7	Mendapatkan data mahasiswa	Pengguna mendapatkan data mahasiswa	Data daftar mahasiswa berhasil didapatkan	Sesuai	Berhasil
8	Mendapatkan data pegawai	Pengguna mendapatkan data pegawai	Data pegawai berhasil didapatkan	Sesuai	Berhasil
9	Mendapatkan data dosen	Pengguna mendapatkan data dosen	Data dosen berhasil didapatkan	Sesuai	Berhasil
10	Mendapatkan data user yang login	Pengguna mendapatkan data user yang sedang login	Data user yang sedang login berhasil didapatkan	Sesuai	Berhasil

3) Pengujian Keamanan

Pengujian keamanan bertujuan untuk memastikan sistem aman dari serangan cyber. Pada tahap ini peneliti melakukan uji keamanan menggunakan tools Bernama OWASP ZAP. ZAP merupakan sebuah aplikasi yang digunakan untuk menemukan kerentanan pada suatu aplikasi web. ZAP dapat digunakan untuk menguji server,jaringan,perangkat dan endpoints. Berikut merupakan hasil pengujian menggunakan teknik brute force pada fitur login.

TABEL 8 HASIL *BRUTE FORCE*

Percobaan	942838
Error	0

TABEL 9 HASIL *ACTIVE SCAN*

No	Jenis peringatan	Tingkat Resiko	Solusi
1	*Cloud Metadata Potentially Exposed*	*High*	Hindari mempercayai variabel $host di NGINX karena bisa dimanipulasi melalui header 'Host'.

No	Jenis peringatan	Tingkat Resiko	Solusi
2	*Path Traversal*	*High*	Validasi semua input, gunakan daftar izin karakter dan jalankan kode dengan hak istimewa terendah di lingkungan terisolasi.
3	*SQL Injection - SQLite*	*High*	Gunakan PreparedStatement, hindari penggabungan string dalam kueri SQL, dan gunakan pengguna database dengan hak akses minimal.
4	*.htaccess Information Leak*	*Medium*	Pastikan file .htaccess tidak dapat diakses.
5	*Absence of Anti-CSRF Tokens*	*Medium*	Gunakan perpustakaan untuk mencegah CSRF dan hasilkan nonce unik untuk setiap formulir.
6	*Content Security Policy (CSP) Header Not Set*	*Medium*	Konfigurasikan server untuk menetapkan header Content-Security-Policy.
7	*Directory Browsing*	*Medium*	Nonaktifkan penjelajahan direktori atau amankan file yang terdaftar.
8	*Missing Anti-clickjacking Header*	*Medium*	Setel header Content-Security-Policy atau X-Frame-Options untuk mencegah clickjacking.
9	*Big Redirect Detected (Potential Sensitive Information Leak)*	*Low*	Pastikan respons pengalihan tidak mengandung informasi sensitif.
10	*Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)*	*Low*	Konfigurasikan server untuk menekan header "X-Powered-By".
11	*Timestamp Disclosure – Unix*	*Low*	Verifikasi bahwa timestamp tidak dapat digunakan untuk mengeksploitasi pola sensitif. [1]
12	*X-Content-Type-Options Header Missing*	*Low*	Setel header X-Content-Type-Options ke 'nosniff' untuk mencegah MIME-sniffing. [2]

V. KESIMPULAN

A. *Kesimpulan*

Implementasi layanan autentikasi dan manajemen akses menggunakan RESTful API dengan protokol HTTP dibagi menjadi dua kategori, yaitu API public dan private. Kategori public digunakan oleh aplikasi klien yang akan memanfaatkan layanan, sedangkan kategori private digunakan untuk pengelolaan layanan melalui front-end dashboard. Penggunaan Bearer token dalam sebagian besar API menjamin keamanan akses data dan fitur sistem. Berbagai metode seperti GET, POST, PUT, dan DELETE digunakan untuk menangani kebutuhan operasional. [3] [4]

Pengujian sistem layanan autentikasi dan manajemen akses menunjukkan bahwa semua fitur utama telah berfungsi dengan baik. Hasil pengujian pada dashboard menunjukkan bahwa pengguna berhasil melakukan berbagai aktivitas seperti masuk ke dashboard, menampilkan dan mensinkronisasikan data mahasiswa, dosen, serta pegawai, mengelola aplikasi dan permission, serta melakukan pengubahan peran. Semua tes berhasil sesuai dengan hasil yang diharapkan. Selain itu, pengujian API untuk aplikasi client juga menunjukkan hasil yang memuaskan, di mana pengguna berhasil mendapatkan token akses dan refresh token, serta mengakses data mahasiswa, dosen, dan pegawai dengan benar. Dengan demikian, baik dashboard maupun API telah berfungsi sesuai spesifikasi, memastikan sistem berjalan sesuai ekspektasi tanpa kendala signifikan. Sementara itu, pengujian keamanan menggunakan OWASP ZAP mengungkap beberapa potensi kerentanan dengan tingkat risiko yang bervariasi. Pengujian brute force berhasil dilakukan tanpa adanya error dengan total percobaan sebanyak 942,838 kali. Hasil active scan mengidentifikasi beberapa risiko serius, seperti Cloud Metadata Exposure, Path Traversal, dan SQL Injection dengan tingkat risiko tinggi, yang membutuhkan mitigasi seperti validasi input yang ketat, penggunaan PreparedStatement, dan konfigurasi server yang lebih aman. Selain itu, beberapa risiko sedang dan rendah juga ditemukan, seperti Absence of Anti-CSRF Tokens dan Missing Anti-clickjacking Headers, yang memerlukan perbaikan lebih lanjut untuk meningkatkan keamanan sistem.

B. *Saran*

Disarankan agar layanan autentikasi dan manajemen akses yang telah dikembangkan terus dievaluasi dan dioptimalkan secara berkala untuk menyesuaikan dengan kebutuhan yang berkembang di lingkungan FRI. Peningkatan fitur, seperti pengelolaan log akses dan monitoring aktivitas pengguna secara real-time, dapat ditambahkan untuk meningkatkan keamanan dan transparansi.

Pengembangan fitur otomatisasi dalam layanan autentikasi dan manajemen akses, seperti otomatisasi pembaruan hak akses atau penghapusan akses bagi pengguna yang tidak aktif, dapat meningkatkan efisiensi dan mengurangi kesalahan yang terjadi akibat pengelolaan manual.

REFERENSI

M. Kondoj, T. Saroinsong, and A. Polii, “Single Sign On (SSO) System with Application of Central Authentication Service (CAS) at Manado State Polytechnic,” INSTICC, Dec. 2023, pp. 698–702. doi: 10.5220/0011863100003575.

G. S. Reddy and T. R. Konala, “EASEID-A SESSION-BASED SINGLE SIGN-ON SELF-SOVEREIGN IDENTITY AND ACCESS MANAGEMENT SYSTEM USING BLOCKCHAIN,” *Indian Journal of Computer Science and Engineering*, vol. 13, no. 4, 2022, doi: 10.21817/indjcse/2022/v13i4/221304176.

T. S. Aina, O. O. Akinte, A. J. Awelewa, and D. O. Adelakun, “International Journal of Advanced Multidisciplinary Research and Studies Critical evaluation of waterfall project management methodology: A case study of digital management conference project,” 2022. [Online]. Available: www.multiresearchjournal.com

Shamsulhuda Khan and Shubhangi Mahadik, “A Comparative Study of Agile and Waterfall Software Development Methodologies,” *International Journal of Advanced Research in Science, Communication and*

Technology, pp. 399–402, Jul. 2022, doi: 10.48175/ijarsct-5696.

[5] M. Kunz, A. Puchta, S. Groll, L. Fuchs, and G. Pernul, "Attribute quality management for dynamic identity and access management," *Journal of Information Security and Applications*, vol. 44, 2019, doi: 10.1016/j.jisa.2018.11.004.

[6] I. Indu, P. M. R. Anand, and V. Bhaskar, "Identity and access management in cloud environment: Mechanisms and challenges," Aug. 01, 2018, *Elsevier B.V.* doi: 10.1016/j.jestch.2018.05.010.

[7] V. C. Hu *et al.*, "NIST Special Publication 800-162 Guide to Attribute Based Access Control (ABAC) Definition and Considerations", doi: 10.6028/NIST.SP.800-162.

[8] H. Hugo, "Membuat Rest API Service Sederhana dengan PHP Native dan SQLite." [Online]. Available: https://medium.com/@hugohfh/membuat-rest-api-service-sederhana-dengan-php-native-dan-sqlite-a4892a2aae79

[9] N. Robles, N. Potes, K. Garcés, J. Luis, C. Izquierdo, and J. Cabot, "Exploratory Analysis of the Structural Evolution of public REST APIs," Apr. 2023. [Online]. Available: https://apis.guru/

[10] M. Gluhak and M. Heričko, "Comparison of Graph and REST APIs," 2023. [Online]. Available: https://foo.com/api/employees/123/projects.

[11] F. Palma, T. Olsson, A. Wingkvist, and J. Gonzalez-Huerta, "Assessing the Linguistic Quality of REST APIs for IoT Applications," May 2022, [Online]. Available: http://arxiv.org/abs/2205.06533

[12] A. Arcuri, M. Zhang, and J. Galeotti, "Advanced White-Box Heuristics for Search-Based Fuzzing of REST APIs," *ACM Transactions on Software Engineering and Methodology*, vol. 33, no. 6, Jun. 2024, doi: 10.1145/3652157.

[13] V. Atlidakis, P. Godefroid, and M. Polishchuk, "Checking Security Properties of Cloud Service REST APIs," in *Proceedings - 2020 IEEE 13th International Conference on Software Testing, Verification and Validation, ICST 2020*, Institute of Electrical and Electronics Engineers Inc., Oct. 2020, pp. 387–397. doi: 10.1109/ICST46399.2020.00046.

[14] M. Idris, I. Syarif, and I. Winarno, "Web Application Security Education Platform Based on OWASP API Security Project," *EMITTER International Journal of Engineering Technology*, pp. 246–261, Dec. 2022, doi: 10.24003/emitter.v10i2.705.

[15] P. Pant *et al.*, "Authentication and Authorization in Modern Web Apps for Data Security Using Nodejs and Role of Dark Web," in *Procedia Computer Science*, Elsevier B.V., 2022, pp. 781–790. doi: 10.1016/j.procs.2022.12.080.

[16] A. Y. Nashikhuddin, J. Karaman, and Y. Litanianda, "IMPLEMENTASI API RESTFUL DENGAN JSON WEB TOKEN (JWT) PADA APLIKASI E-COMMERCE THRIFTY SHOP UNTUK OTENTIKASI DAN OTORISASI PENGGUNA," vol. 7, no. 2, 2023, doi: 10.46880/jmika.Vol7No2.pp239-246.

[17] M. Waseem, P. Liang, M. Shahin, A. Di Salle, and G. Márquez, "Design, monitoring, and testing of microservices systems: The practitioners' perspective," *Journal of Systems and Software*, vol. 182, p. 111061, Dec. 2021, doi: 10.1016/j.jss.2021.111061.

[18] H. Aljawawdeh, M. Sabri, and L. Maghrabi, "Toward Serverless and Microservices Architecture: Literature, Methods, and Best Practices," 2023, pp. 573–584. doi: 10.1007/978-3-031-43300-9_47.

[19] H. Koç, A. M. Erdoğan, Y. Barjakly, and S. Peker, "UML Diagrams in Software Engineering Research: A Systematic Literature Review," in *The 7th International Management Information Systems Conference*, Basel Switzerland: MDPI, Mar. 2021, p. 13. doi: 10.3390/proceedings2021074013.

[20] S. Al-Fedaghi, "UML Sequence Diagram: An Alternative Model," 2021. [Online]. Available: www.thesai.org

[21] A. Alshamrani and A. Bahattab, "A Comparison Between Three SDLC Models Waterfall Model, Spiral Model, and Incremental/Iterative Model." [Online]. Available: www.IJCSI.org

[22] T. Scott, "Waterfall methodology." [Online]. Available: https://www.leadertask.com/articles

[23] S. Pargaonkar, "A Study on the Benefits and Limitations of Software Testing Principles and Techniques: Software Quality Engineering," *International Journal of Scientific and Research Publications*, vol. 13, no. 8, pp. 149–155, Aug. 2023, doi: 10.29322/IJSRP.13.08.2023.p14018.

[24] A. Golmohammadi, M. Zhang, and A. Arcuri, ."NET/C# instrumentation for search-based software testing," *Software Quality Journal*, vol. 31, no. 4, pp. 1439–1465, Dec. 2023, doi: 10.1007/s11219-023-09645-1.

[25] M. Boukhlif, M. Hanine, and N. Kharmoum, "A Decade of Intelligent Software Testing Research: A Bibliometric Analysis," *Electronics (Basel)*, vol. 12, no. 9, p. 2109, May 2023, doi: 10.3390/electronics12092109.

[26] S. Kumar, "Reviewing Software Testing Models and Optimization Techniques: An Analysis of Efficiency and Advancement Needs," *Journal of Computers, Mechanical and Management*, vol. 2, no. 1, pp. 32–46, Feb. 2023, doi: 10.57159/gadl.jcmm.2.1.23041.

[27] D. Zimmermann and A. Koziolek, "Automating GUI-based Software Testing with GPT-3," in *2023 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, IEEE, Apr. 2023, pp. 62–65. doi: 10.1109/ICSTW58534.2023.00022.

[28] N. Alshahwan, M. Harman, and A. Marginean, "Software Testing Research Challenges: An Industrial Perspective," in *2023 IEEE Conference on Software Testing, Verification and Validation (ICST)*, IEEE, Apr. 2023, pp. 1–10. doi: 10.1109/ICST57152.2023.00008.

[29] M. Aydos, Ç. Aldan, E. Coşkun, and A. Soydan, "Security testing of web applications: A systematic mapping of the literature," Oct. 01, 2022, *King Saud bin Abdulaziz University*. doi: 10.1016/j.jksuci.2021.09.018.