Rancang Bangun Aplikasi COOL: REST API Untuk Learning Management System

Muchammad Fikri Afrizzi[1)], Raden Budiarto Hadiprakoso[2)], Aqwam Rosadi Kardian[3)]
(1) Rekayasa Kriptografi, Poltek Siber dan Sandi Negara, mfikri.afrizzi@student.polteksSn.ac.id
(2) Rekayasa Kriptografi, Poltek Siber dan Sandi Negara, raden.budiato@polteksSn.ac.id
(3) Sistem Informasi, STMIK Jakarta STI&K, aqwam@staff.jak-stik.ac.id

Abstrak

Seiring pesatnya perkembangan teknologi informasi, penting sekali untuk membuat aplikasi yang mampu terintegrasi dengan berbagai macam aplikasi. Perkembangan yang sangat pesat itu sejalan dengan kebijakan program "kampus merdeka" untuk menciptakan sarana pendidikan yang terintegrasi dengan dunia kerja. Pada penelitian ini dibangun aplikasi API (Application Programming Interface) untuk Learning Management System (LMS) yang diberi nama COOL (Collaborative Online Learning). Aplikasi tersebut dibangun dengan menerapkan aspek keamanan berupa JSON Web Token (JWT) dengan algoritma SHA-256 untuk dapat melakukan otentikasi terhadap pengguna ketika terjadi interaksi dengan aplikasi REST API LMS. Selain aspek keamanan, fitur-fitur yang disediakan oleh aplikasi COOL akan disesuaikan dengan kebutuhan dan proses bisnis yang berasal dari stake holder. Dalam pengembangan aplikasi COOL menggunakan metode pengembangan SCRUM dengan menggunakan framework Codeigniter. Pengujian aplikasi dilakukan dengan cara pengujian fungsional untuk menguji fitur aplikasi, pengujian keamanan dan pengujian performa. Hasil pengujian menunjukkan fungsi dan keamanan program berjalan sesuai dengan yang diharapkan. Selain itu dari hasil pengujian performa menunjukkan JWT lebih unggul ketimbang PASETO. Kesimpulan yang ditarik setelah melalui proses pengujian menunjukkan bahwa aplikasi COOL telah dibangun sesuai kebutuhan dengan proses bisnis serta telah memenuhi aspek keamanan yang dibutuhkan.

Kata kunci: API, *learning management system*, JSON *web token*, *Scrum*

Abstract

Along with the rapid development of information technology, creating applications that can be integrated with various applications is very important. This rapid development aligns with the "Kampus Merdeka" program policy to create educational facilities that are integrated with the world of work. In this research, an API (Application Programming Interface) application was developed for a Learning Management System (LMS) called COOL (Collaborative Online Learning). The COOL application is built by implementing security aspects in JSON Web Token (JWT) with the SHA-256 algorithm to authenticate users when interacting with the LMS REST API application. In addition to the security aspect, the features provided by the COOL application will be adjusted to the needs and business processes that come from stakeholders. COOL application development uses the SCRUM development method with the Codeigniter framework tools. Application testing is done through functional testing, security, and performance testing. The test results show that the function and security of the program are running as expected. The performance test results also show that JWT is superior to PASETO. The conclusions drawn after going through the testing process show that the COOL application has been built according to the needs of business processes and has fulfilled the required security aspects.

Keywords: API, *learning management system*, JSON *web token*, *Scrum*

1. PENDAHULUAN

Sejak munculnya pandemi COVID-19, pemerintah di seluruh dunia dengan cepat memulai kebijakan yang bertujuan menerapkan metode Pembelajaran Jarak Jauh (PJJ) di tingkat pendidikan dasar, menengah, dan tinggi. Pembelajaran ini ditandai dengan pemisahan fisik antara siswa dan pendidik, sangat bergantung pada berbagai sumber daya teknologi informasi untuk penyampaian pembelajaran [1]. Pendekatan PJJ yang lazim dan lugas melibatkan pemanfaatan platform komunikasi seperti WhatsApp, Telegram, dan Zoom. Meskipun demikian, metode ini penuh dengan tantangan. Masalah yang paling utama adalah tidak dapat diandalkannya koneksi internet, yang sering kali mengakibatkan gangguan selama interaksi PJJ [2]. Masalah privasi dan keamanan merupakan kendala kedua, karena media sosial dan platform obrolan konvensional sering kali gagal dalam memastikan kerahasiaan, ketersediaan, dan keamanan data pendidikan yang sensitif, termasuk soal ujian dan nilai siswa. Selain itu, tidak adanya fitur pembelajaran khusus seperti manajemen kelas menghadirkan tantangan ketiga [3].

Untuk mengatasi permasalahan multifaset ini, salah satu solusi yang layak adalah penerapan *Learning Management System* (LMS) yang dirancang untuk memfasilitasi pembuatan, distribusi, dan pengelolaan konten pembelajaran dalam bentuk elektronik [4]. Pada tahun 2020, Kementerian Pendidikan dan Kebudayaan Republik Indonesia memperkenalkan kebijakan "Merdeka Belajar", yang menganjurkan siswa untuk mengasah keterampilan,

bakat, dan minat mereka dengan melibatkan entitas eksternal, khususnya mitra industri [5]. Di era kampus mandiri yang ditandai dengan kebutuhan untuk mengembangkan alat belajar mengajar yang terintegrasi dengan sistem eksternal, salah satu pendekatan yang menjanjikan adalah dengan mengintegrasikan LMS dengan aplikasi lain, sehingga memungkinkan kolaborasi yang lancar [4].

Application Programming Interface (API), sebuah komponen perangkat lunak yang memungkinkan pengembang untuk menghubungkan dan berbagi data antara beberapa aplikasi, muncul sebagai alat penting dalam proses integrasi ini [6]. Makalah ini mengusulkan penggunaan metode arsitektur *Representational State Transfer* (REST), yang terkenal karena efektivitasnya dalam menciptakan dan mengelola sistem terdistribusi. Melalui pengembangan aplikasi REST API untuk LMS, tujuannya adalah memungkinkan integrasi komprehensif dengan berbagai sistem informasi yang ada, selaras dengan visi strategis lembaga dalam membina lingkungan pendidikan yang konsisten dengan konsep kampus merdeka.

REST API yang diterapkan pada LMS memungkinkan proses pembuatan, pengambilan, pembaruan, dan penghapusan data. Model REST API dan dipilih karena konfigurasi latensi rendahnya, yang kondusif untuk integrasi data yang efisien [7]. REST juga unggul dalam waktu respons dan ukuran data jika dibandingkan dengan model API lainnya [8]. Mengingat data yang terlibat mencakup informasi sensitif seperti alamat email dan kata sandi, langkah-langkah keamanan yang kuat sangat penting. Oleh karena itu, penelitian ini menerapkan standar keamanan REST API JSON Web Token (JWT) untuk melindungi data penting.

Secara metodologis, penelitian ini menggunakan pendekatan kualitatif, menggunakan kerangka studi kasus yang ditandai dengan eksplorasi mendalam pada tingkat individu, kelompok, atau organisasi untuk mendapatkan wawasan komprehensif terhadap tantangan yang ada [9]. Implementasi praktis penelitian ini dilakukan di kampus XYZ, dengan metodologi pengembangan perangkat lunak *Scrum*. Metodologi *Scrum* dikenal dengan kemampuan adaptasinya dalam mengatasi permasalahan yang kompleks dan dinamis, merupakan pilihan yang tepat untuk memfasilitasi pengembangan perangkat lunak selama penelitian ini, memastikan keselarasan dengan kebutuhan pengguna dan pemangku kepentingan.

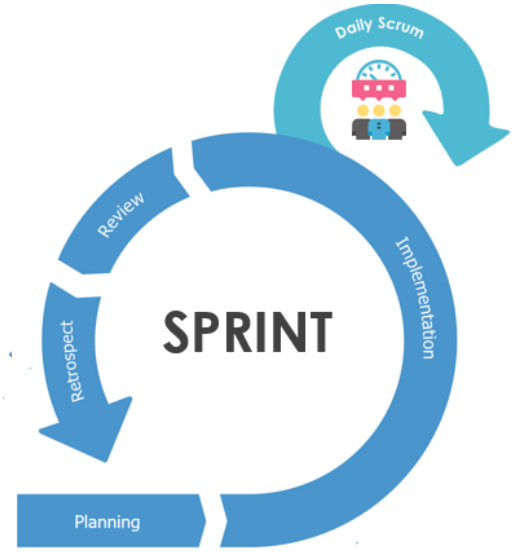
Tujuan dari penelitian ini adalah untuk membuat konsep dan membangun aplikasi COOL (*Colaborative Online Learning*) yang disesuaikan untuk LMS Moodle. LMS tersebut dipilih karena sifat *open-source* dan penggunaannya yang luas, sesuai dengan laporan Open Learning [9]. Penelitian ini berupaya mengevaluasi fungsionalitas, keamanan, dan kinerja dalam konteks aplikasi COOL. Manfaat yang diharapkan dari penelitian ini mencakup potensi penerapan COOL untuk mengintegrasikan beragam sistem informasi secara lancar, sehingga mendukung kebijakan kampus merdeka. Selain itu, penelitian ini juga dapat menjadi referensi untuk pengembangan aplikasi yang menggunakan metodologi *Scrum*.

2. LANDASAN TEORI

2.1 *Scrum*

Scrum merupakan salah satu bentuk kerangka kerja tangkas (*agile*) yang paling populer digunakan [11]. *Scrum* pertama kali dipublikasikan oleh Schwaber dan Sutherland pada tahun 1995. Pencetus *Scrum* mendefinisikannya sebagai kerangka kerja yang simpel untuk dipahami namun sulit dikuasai. Kebanyakan orang dapat menjalankan *Scrum* hanya dalam beberapa jam. Tetapi mendapatkan hasil yang optimal dengan *Scrum* membutuhkan pengalaman dan latihan.

Pada metodologi *Scrum* memiliki beberapa *event*. *Event* ini layaknya sebuah acara yang dihadiri atau dilakukan masing-masing peran untuk menghasilkan artefak. Terdapat 5 *event* yang dikenal sebagai *daily scrum*, *Sprint plan*, *Sprint*, *Sprint review*, *Sprint retrospective* seperti yang digambarkan pada gambar 1 [12].



Gambar 1. Gambaran umum proses pada *Scrum*

Berikut merupakan penjelasan tahap-tahap pada metode *Scrum* seperti yang diilustrasikan pada gambar 1 [13]:

Sprint planning, Tugas yang akan dikerjakan dalam *Sprint*, direncanakan terlebih dahulu saat *Sprint planning*. Tahap ini dilaksanakan oleh seluruh anggota tim *Scrum* secara kolaboratif. *Scrum master* memastikan kegiatan ini dilaksanakan dan peserta memahami tujuannya. *Sprint planning* menjawab pertanyaan: apa yang akan dihantarkan ke dalam *increment* dari *Sprint* serta bagaimana caranya.

Daily Scrum, pada dasarnya sama seperti *stand meeting* pada XP. *Daily Scrum* adalah acara untuk *Development Team* di mana secara opsional *product*

owner dan atau *Scrum master* mengikuti acara ini. *Scrum master* bertugas memastikan *Development Team* telah menjalankan pertemuan tersebut, akan tetapi *Development Team* yang bertanggung jawab penuh terhadap *daily Scrum*.

Sprint, Inti dari sebuah proyek *Scrum* adalah *Sprint* yang merupakan proses inkremental yang iteratif. Batasan *Sprint* maksimal adalah 30 hari; meskipun beberapa orang lebih suka *Sprint* pendek satu, dua, atau tiga minggu. Hal ini dikarenakan jika durasi *Sprint* terlalu lama maka akibatnya definisi dari permasalahan dapat berubah, kompleksitas dapat meningkat, dan pada akhirnya risiko juga dapat meningkat. Hasil dari setiap *Sprint* adalah perangkat lunak yang sepenuhnya teruji dan disetujui.

Sprint Review, biasanya diselenggarakan di akhir *Sprint* untuk menginspeksi inkremen. Tim pengembang mempresentasikan inkremen produk kepada *product owner*, kemudian *product owner* memastikan *Sprint goal* telah terpenuhi. *Product owner* dapat mengundang pemegang kepentingan atau pelanggan untuk berkolaborasi untuk meninjau apa yang sudah diselesaikan di *Sprint*. Berdasarkan hasil tinjauan tersebut diberikan umpan balik untuk perubahan terhadap *product backlog*..

Sprint retrospective, dilakukan oleh *Scrum master* dan tim pengembang di mana mereka membahas *Sprint* yang terakhir. Di sini mereka membahas tiga pertanyaan besar tentang: (1) Apa yang berjalan dengan baik dan bagaimana bisa mewujudkannya kembali? (2) Apa yang berjalan buruk dan bagaimana untuk mengatasinya di masa depan? (3) Bagaimana dapat lebih baik lagi pada *Sprint* berikutnya? Di sini merupakan *Event* bagi tim *Scrum* untuk menginspeksi dirinya sendiri dan membuat perencanaan mengenai peningkatan yang akan dilakukan di *Sprint* berikutnya.

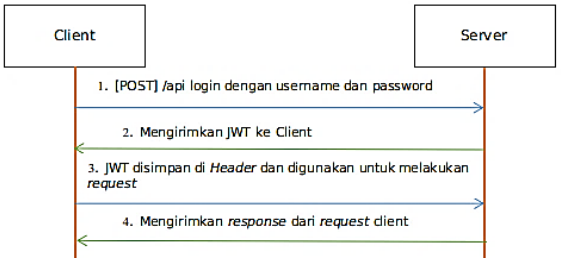
2.2 REST API

API merupakan sebuah perangkat lunak yang bertujuan untuk menyediakan layanan mengakses data dalam berbagai macam jenis aplikasi. Dengan menggunakan API maka aplikasi lain dapat mengakses data atau layanan tanpa harus mengimplementasikan objek yang terdapat pada kode sumber. Hampir semua perangkat lunak modern saat ini telah menggunakan API [14].

Dalam pengembangan aplikasi API, dapat berguna untuk menyederhanakan pemrograman dengan menyediakan abstraksi tingkat tinggi untuk dapat di implementasikan. *Representational State Transfer* (REST) adalah suatu istilah yang diciptakan oleh Roy Fielding. Istilah REST mengacu pada gaya arsitektur yang menjelaskan enam batasan arsitektur yaitu *Uniform Interface*, *Stateless*, *Cacheable*, *Client-Server*, dan *Layered System*, *Code on Demand* [15]. REST memiliki standardisasi dalam pemakaian yaitu mengenai URL dan HTTP *verbs*.

2.3 JSON Web Token

JWT adalah untaian *string* karakter (token) yang terdiri dari tiga bagian yakni *header*, *payload* dan *signature.* Token ini digunakan untuk proses otentikasi dan pertukaran informasi. Token terdiri dari dua jenis yaitu token pembawa dan token pemegang kunci. Berdasarkan tujuannya, token dibagi menjadi dua skema yaitu token identitas dan token akses [16]. Sistematika penggunaan JWT sama seperti penggunaan *password*, ketika *client* berhasil melakukan *login* maka *server* akan memberikan sebuah Token. Token tersebut akan disimpan oleh *client* pada *local storage* atau *cookies browser* dan bila *client* ingin mengakses halaman-halaman tertentu maka harus menyertakan token tersebut.



Gambar 2. Skema penggunaan JWT pada REST API

Gambar 2 merupakan alur skema dari penggunaan JWT pada REST API. Dari gambar tersebut menjelaskan terkait skema JWT yang mana *client* akan mengirimkan otentikasi terlebih dahulu berupa *username* dan *password.* Setelah dilakukan otentikasi oleh *server* dan bernilai benar maka *server* akan mengirimkan token JWT yang telah di-*generate* kepada *client.* Setelah *client* menerima token JWT, token tersebut akan selalu disisipkan pada *header* setiap kali *client* melakukan *request.* Pada setiap *request* yang dikirimkan oleh *client*, *server* akan memeriksa token JWT yang tersimpan dalam *header* dan selanjutnya jika token sesuai maka *server* akan memberikan respons sesuai dengan *request* yang diminta oleh *client.*

Terdapat beberapa penelitian terkait dengan makalah ini. Peneliti [17] melakukan penelitian terkait keamanan RESTful Web Service menggunakan JSON Web Token. Pada penelitian ini dilakukan optimasi keamanan JWT menggunakan algoritma HMAC SHA-512 dibandingkan dengan HMAC SHA-256. Dari penelitian ini membuktikan bahwa algoritma SHA-512 memiliki keunggulan dalam hal waktu dan besar ukuran token yang dibangkitkan.

Salman Ahmed dan Qamar Mahmood melakukan penelitian terkait "*An authentication based scheme for applications using JSON web token*". Pada penelitian ini dilakukan pengembangan aplikasi dengan memanfaatkan arsitektur REST dan autentikasi token JWT, sehingga setiap permintaan klien nilai *timestamp* yang benar-benar acak untuk meningkatkan keaslian klien pada *server* [18].

Selain JWT terdapat alternatif lainnya yakni PASETO (Platform-Agnostic Security Tokens) yang

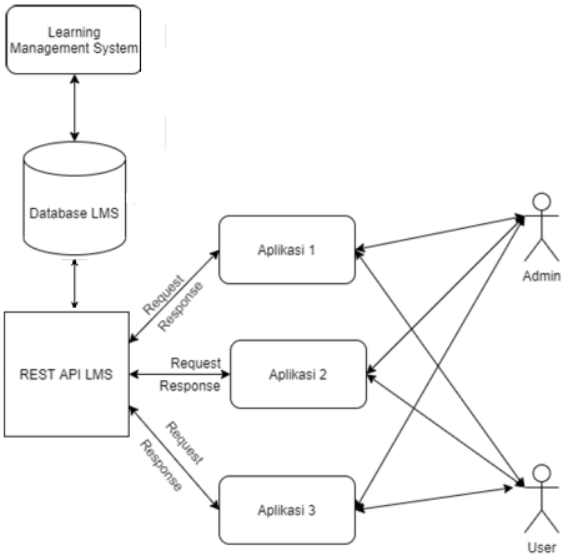
merupakan standar untuk membuat dan memvalidasi token keamanan menggunakan *encoding* JSON. Dibandingkan dengan JWT, PASETO memiliki pendekatan keamanan yang lebih ketat, selalu menyiratkan enkripsi dan tanda tangan atau hanya tanda tangan saja dalam setiap token. PASETO juga menggunakan algoritma enkripsi standar (XChaCha20-Poly1305) [19]. JWT memerlukan penentuan algoritma secara eksplisit dalam token. Dengan pendekatan keamanan yang jelas dan struktur yang lebih sederhana, PASETO dianggap lebih aman dan lebih mudah digunakan daripada JWT, yang memiliki beberapa versi dengan perbedaan dalam struktur dan masalah keamanan [20].

3. METODE PENELITIAN

Penelitian ini menggunakan *Learning Management System* (LMS) milik kampus XYZ sebagai obyek penelitian. Untuk mencapai hal tersebut maka diperlukan pengembangan pada LMS berupa mengintegrasikan LMS dengan berbagai macam sistem informasi atau aplikasi yang berasal dari luar sistem internal. Penelitian ini merupakan jenis penelitian kualitatif dengan menggunakan terminologi studi kasus. Analisis yang dilakukan dengan berfokus terhadap permasalahan pengembangan LMS saat ini untuk dapat mencapai rencana strategis dalam menciptakan lingkungan *smart campus* serta mendukung terciptanya program kampus merdeka. Dilihat dari tujuan penelitian, penelitian ini tergolong penelitian studi kasus.

Metodologi penelitian menggunakan metode pengembangan perangkat lunak *Scrum* karena cocok untuk mengembangkan aplikasi COOL dengan berbagai teknik. Penelitian ini menganut skema desain dan siklus pengembangan perangkat lunak *Scrum*. Berikut ini garis besar tahapan-tahapan utama dalam kerangka *Scrum* yang diterapkan dalam penelitian ini:

1. *Backlog* Produk
   Tahap awal melibatkan penciptaan *product backlog* yang dinamis dan terus berkembang. *Backlog* ini berfungsi sebagai daftar terurut yang mencakup semua fitur dan persyaratan yang diketahui untuk aplikasi COOL pada akhirnya. Sifat dinamis ini memungkinkan penyempurnaan berkelanjutan, memastikan keselarasan aplikasi dengan kebutuhan dan harapan yang terus berkembang dari pemilik produk, yang bertanggung jawab atas pengembangan produk. *Product backlog* mencakup fitur fungsional dan non-fungsional yang penting untuk pengembangan aplikasi COOL.
2. *Sprint*
   Sprint merupakan inti dari pendekatan pengembangan perangkat lunak *Scrum*. Setiap s*print* ditandai dengan jangka waktu tetap di mana serangkaian tugas dilakukan dengan tujuan mencapai penyelesaian tugas. Tugas-tugas yang diselesaikan ini berkontribusi pada pengembangan aplikasi COOL yang sedang berlangsung. Yang penting, durasi setiap *sprint* tetap konsisten sepanjang siklus pengembangan perangkat lunak. Setiap *sprint* mewakili proyek dalam kerangka yang lebih besar, menampilkan serangkaian tujuan yang memandu desain, perencanaan, dan pelaksanaan aktivitas pengembangan perangkat lunak secara fleksibel.
   Selain membahas mengenai metode pengembangan yang digunakan, desain penelitian juga akan membahas mengenai gambaran sistem yang menjelaskan hubungan antar entitas dengan aplikasi yang akan dibuat. Gambar 3 merupakan gambaran secara umum dari hubungan antara entitas dengan sistem.



Gambar 3. Skema Penggunaan JWT pada REST API

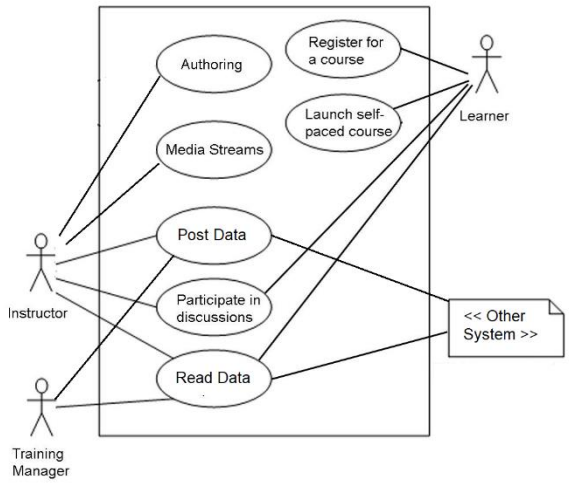
Dari gambar 3 dapat dipahami bahwa LMS saat ini telah memiliki basis data LMS yang terdapat pada *server*. Dari basis data tersebut akan dihubungkan dengan REST API LMS yang akan dibuat pada penelitian ini. Setelah REST API LMS dapat terhubung dengan basis data LMS maka REST API LMS akan dapat menyediakan fitur *request* dan *response* untuk aplikasi yang ingin melakukan pengiriman dan penerimaan data dengan basis data. Dengan berjalannya proses pengiriman dan penerimaan data maka aplikasi akan dapat menjalankan perintah *input output* dari pengguna dan admin sesuai dengan kebutuhan data yang diperlukan.

Proses *Scrum* dimulai dengan pembuatan *Scrum board*. *Scrum board* ini merupakan hasil analisis kebutuhan telah diolah menjadi bagian-bagian dalam *backlog* yang akan dibagi menjadi 3 *sprint*. Pada tahapan *sprint* pertama akan membahas mengenai desain dari aplikasi COOL yang akan dibuat. Adapun pada tahap kedua adalah implementasi dan pada tahap ketiga adalah pengujian dari aplikasi.

Pada tahap perancangan *use case diagram* digunakan untuk menangkap kebutuhan fungsional dari aplikasi. *Use case* merupakan teknik dari desain

pengembangan perangkat lunak yang menjelaskan mengenai interaksi yang terjadi antara aktor dengan sistem. Pada penelitian ini *use case* akan dibuat berdasarkan hasil analisis kebutuhan fungsional yang telah dilakukan pada tahapan sebelumnya. Gambar 4 merupakan *use case* dari aplikasi REST API LMS.

Gambar 4 menjelaskan bahwa terdapat tiga aktor yaitu siswa (*learners*), pengajar (*instructor*), dan pengelola (*training manager*) yang akan berinteraksi dengan aplikasi REST API LMS. Ketiga aktor pada *use case* masing-masing aktor akan mendapatkan hak akses terhadap fungsi-fungsi tertentu yang disesuaikan dengan kebutuhan dan proses bisnis yang ada. Untuk masing-masing fungsi pada *use case* memerlukan aktor untuk login terlebih dahulu. Sebagai pengaman login, diterapkan JWT *(JSON web Token*).



Gambar 4. *Use Case Diagram*

Pada tahapan selanjutnya dilakukan serangkaian pengujian untuk mengetahui apakah aplikasi dapat berjalan sesuai dengan yang diharapkan. Pengujian yang dilakukan meliputi fungsionalitas dan keamanan aplikasi sebagai berikut:

a. *Functional testing*
   Melalui pengujian ini dapat diketahui bagaimana fungsi-fungsi yang terdapat dalam aplikasi saling bersinergi dan dapat berjalan dengan baik. Pada *function testing* ini akan dilakukan pengujian terhadap *method post, get* dan *delete*.
b. *Security testing*
   Pembahasan *security tesing* pada makalah ini berfokus pada penggunaan JWT, serangkaian pengujian yang akan dilakukan sebagai berikut:
   1) Melakukan akses antar halaman melalui URL pada browser tanpa melakukan *login.*
   2) Melakukan akses ke dalam suatu halaman dengan menggunakan URL yang memiliki hak akses yang berbeda.
   3) Melakukan uji coba *login* untuk melihat apakah token berhasil dibangkitkan.
c. *Performance testing*
   Melalui pengujian ini diukur waktu yang diperlukan untuk menghasilkan token. Pada pengujian akan dilakukan perbandingan antara JWT token dan PASETO. Masing-masing akan dihasilkan sebanyak 30 token untuk melihat perbandingan waktu rata-rata..

4. HASIL DAN PEMBAHASAN

4.1. *Functional Testing*

Functionality testing merupakan pengujian yang terdiri dari pengujian terhadap setiap *request method* yang ada di setiap fungsi pada aplikasi REST API LMS. Tujuan dari *functionality testing* ini yaitu untuk menguji setiap fungsi dapat berjalan dengan baik sesuai dengan yang diharapkan. Berikut adalah penjelasan dari hasil *functionality testing* pada setiap fungsi REST API LMS.

Pengujian *method get* dilakukan pada setiap fungsi yang menyediakan fitur *request method get* pada aplikasi REST API. Teknis pengujian ini yaitu mengirimkan *request method get* dari aplikasi pihak ketiga yaitu Postman kepada aplikasi COOL yang selanjutnya akan diteruskan pada basis data LMS. Setelah *request method get* dikirimkan maka akan diamati apakah hasil yang diharapkan sesuai atau tidak dengan hasil yang terima pada pengujian. Berikut adalah contoh pengujian *request method get* yang ada pada salah satu fungsi REST API LMS.

Pengujian *request method get* dilakukan dengan cara aplikasi pihak ketiga yaitu postman mengirimkan *request method get* kepada aplikasi REST API LMS. Setelah *request* dikirimkan maka dilakukan observasi terhadap *feedback* yang dikirimkan oleh aplikasi REST API LMS. Hasil pengujian dinyatakan bahwa *request method get* secara keseluruhan telah berjalan dengan baik dan sesuai dengan yang diharapkan.

Pengujian *method post* dilakukan pada setiap fungsi yang menyediakan fitur *request method post* pada aplikasi REST API LMS. Teknis pengujian ini yaitu mengirimkan *request method post* dari aplikasi pihak ketiga yaitu Postman kepada aplikasi COOL yang selanjutnya akan diteruskan pada *database* LMS. Setelah *request method post* dikirimkan maka akan diamati apakah hasil yang diharapkan sesuai atau tidak dengan hasil yang diterima pada pengujian. Berikut adalah contoh pengujian *request method post* yang ada pada salah satu fungsi REST API LMS.

Pengujian *request method post* dilakukan dengan cara aplikasi pihak ketiga yaitu Postman mengirimkan *request method get* kepada aplikasi REST API LMS. Setelah *request* dikirimkan maka dilakukan observasi terhadap *feedback* yang dikirimkan oleh aplikasi REST API LMS. Hasil pengujian dinyatakan bahwa *request method post* secara keseluruhan telah berjalan dengan baik dan sesuai dengan yang diharapkan.

Pengujian *method put* dilakukan pada setiap fungsi yang menyediakan fitur *request method put* pada aplikasi REST API. Teknis pengujian ini yaitu mengirimkan *request method put* dari aplikasi pihak ketiga yaitu Postman kepada aplikasi COOL yang

selanjutnya akan diteruskan pada *database* LMS. Setelah *request method put* dikirimkan maka akan diamati apakah hasil yang diharapkan sesuai atau tidak dengan hasil yang diterima pada pengujian. Berikut adalah contoh pengujian *request method put* yang ada pada salah satu fungsi REST API LMS.

Pengujian *request method put* dilakukan dengan cara aplikasi pihak ketiga yaitu Postman mengirimkan *request method put* kepada aplikasi REST API LMS. Setelah *request* dikirimkan maka dilakukan observasi terhadap *feedback* yang dikirimkan oleh aplikasi REST API LMS. Hasil pengujian dinyatakan bahwa *request method put* secara keseluruhan telah berjalan dengan baik dan sesuai dengan yang diharapkan.

Tabel 1. Hasil *function testing*

Method	Fungsi	Hasil Pengujian
GET	Mengambil data	Sesuai dengan fungsi yang diharapkan
POST	Menambahkan data	Sesuai dengan fungsi yang diharapkan
PUT	Mengedit data	Sesuai dengan fungsi yang diharapkan

4.2. *Security Testing*

Security testing dilakukan dengan sejumlah *request* pada data yang telah ditentukan. Menggunakan media pengujian aplikasi Postman untuk melihat secara rinci kebutuhan aplikasi terhadap token ketika akan melakukan suatu *request*. Berikut merupakan kondisi yang digunakan untuk menguji HTTP *request* yang digunakan:

Tabel 2. Hasil *security testing*

Skenario	Hasil yang diharapkan	Hasil Pengujian
Jalankan request HTTP GET terhadap data berdasarkan *role* yang terdapat pada token	Notifikasi *error* akan ditampilkan jika *user* tidak memiliki token, token yang digunakan telah dimodifikasi, atau memiliki nilai yang tidak valid	Seperti yang Diharapkan
Menguji data baru dengan mengirimkan permintaan HTTP POST	Jika pengguna tidak memiliki izin, penambahan data baru akan gagal, dan kesalahan akan ditampilkan	Seperti yang Diharapkan
Menguji permintaan HTTP PUT untuk pengeditan data	Hanya peran admin yang dapat mengedit data apa pun di aplikasi COOL	Seperti yang Diharapkan

a. Melakukan pengujian pada HTTP *request* GET pada data berdasarkan *role* terdapat pada token. *Test case* ini dilakukan untuk memastikan bahwa sistem hanya akan memberikan akses terhadap suatu halaman maupun fitur sesuai dengan *role* yang didapatkan dari token yang telah dihasilkan. Ketika pengguna tidak memiliki token, atau token yang digunakan telah dimodifikasi atau bernilai tidak valid maka akan ditampilkan notifikasi *error*.
b. Melakukan pengujian pada HTTP *request* POST untuk menambahkan data baru. *Test case* ini dilakukan untuk memastikan sistem hanya akan memberikan akses kepada masing-masing *role* pengguna. Di mana *role* pengguna hanya diberikan hak akses untuk melakukan penambahan data hanya di beberapa halaman saja, sedangkan admin melakukan penambahan data yang mempengaruhi konten utama dari aplikasi. Diperlukan data pengguna yang sedang *login* untuk memastikan apakah pengguna tersebut diberikan akses untuk melakukan penambahan data di halaman tertentu. Berdasarkan hal tersebut didapatkan hasil bahwa ketika melakukan *create* data pada suatu *form* yang hanya dapat diakses oleh admin namun menggunakan *token user*, maka penambahan data akan gagal dan menampilkan *error*.
c. Melakukan pengujian pada HTTP *request* PUT untuk menyunting data. *Test case* ini memiliki bentuk yang sama *seperti request* POST, di mana hanya *role* admin yang diberikan hak akses untuk melakukan penyuntingan data apa pun pada aplikasi REST API LMS. Pada aplikasi COOL yang di uji pada Postman menunjukkan bahwa ketika token yang digunakan adalah milik *user*, maka yang sudah mengidentifikasi *role* dari pengguna akan menampilkan *error*.

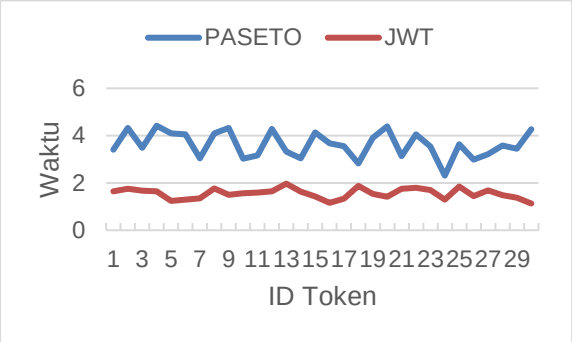
4.3. *Performance Testing*

Pengujian berikutnya berfokus pada pengujian performa dari token. Pengujian ini dilakukan dengan mengamati parameter waktu pembuatan token. Pengujian dilakukan menggunakan 30 sampel untuk token. Sebagai bahan perbandingan akan dilakukan uji perbandingan performa antara token JWT dengan PASETO.

Parameter waktu pembuatan token diukur menggunakan fungsi *performance.now()* untuk mendapatkan hasil yang akurat dalam mili detik (md). Gambar 5 menunjukkan bahwa rata-rata waktu pembuatan token untuk PASETO lebih lama dibandingkan dengan waktu pembuatan token untuk JWT. Rata-rata waktu pembuatan token untuk PASETO sekitar 3,05 md sedangkan untuk JWT sekitar 1,58 md. Dari data yang diuji, terlihat bahwa pembuatan token menggunakan PASETO memerlukan waktu yang lebih lama dibandingkan dengan JWT. Perbedaan rata-rata sekitar 1,47 md menunjukkan bahwa JWT memiliki performa yang lebih baik dalam hal waktu pembuatan token.

Waktu pembuatan token PASETO lebih lama daripada JWT karena PASETO menggunakan algoritma enkripsi yang lebih kuat, struktur *encoding*

yang lebih jelas, serta selalu menyiratkan enkripsi dan tanda tangan untuk memastikan keamanan dan integritas data dalam token. Meskipun lebih lambat, keuntungan ini menghadirkan tingkat keamanan yang lebih tinggi dan kejelasan struktur yang lebih baik daripada JWT. Kecepatan dan keamanan perlu dipertimbangkan saat memilih format token yang sesuai untuk kebutuhan aplikasi.



Gambar 5. Hasil pengujian performa JWT dan PASETO

5. KESIMPULAN

Kesimpulan yang dapat diambil dari penelitian ini bahwa pengujian fungsionalitas menunjukkan aplikasi REST API LMS berjalan dengan baik, sesuai dengan permintaan yang diharapkan, termasuk pengambilan data (GET), penambahan data (POST), dan penyuntingan data (PUT). Pengujian keamanan menunjukkan bahwa mekanisme autentikasi menggunakan JWT berhasil melindungi akses yang sesuai berdasarkan peran (*role*) pengguna, dan menolak akses jika token tidak valid. Pengujian performa menunjukkan bahwa JWT memiliki waktu pembuatan token yang lebih cepat dibandingkan PASETO. Untuk penelitian lebih lanjut pada potensi penggunaan PASETO atau alternatif lain dalam situasi khusus yang membutuhkan tingkat keamanan yang lebih tinggi.

REFERENSI

[1] Nambiar D. “The impact of online learning during COVID-19: students’ and teachers’ perspective” The International Journal of Indian Psychology. Vol. 08, No. 2, pp 783-793, June. 2020. DOI: 10.25215/0802.094.

[2] Camargo CP, Tempski PZ, Busnardo FF, Martins M de A, Gemperli R. “Online learning and COVID-19: a meta-synthesis analysis” Clinics. 2020 doi: 10.6061/clinics/2020/e2286

[3] Siregar, N., Sahirah, R., dan Harahap, A. A. “Konsep Kampus Merdeka Belajar di Era Revolusi Industri 4.0”. Fitrah: Journal of Islamic Education, 1(1), pp 141-157, 2020.

[4] D. D. Singhal, “Understanding Student-Centered Learning and Philosophies of Teaching Practices,” International Journal of scientific research and management, vol. 5, no. 02, hal. 5123–5129, 2017, doi: 10.18535/ijsrm/v5i2.02.

[5] Sherly S, Dharma E, Sihombing HB. Merdeka belajar: kajian literatur. InUrbanGreen Conference Proceeding Library 2021 Aug 25 (pp. 183-190).

[6] E. Wittern et al., "Ohalortunities in Software Engineering Research for Web API Consumption," 2017 IEEE/ACM 1st International Workshop on API Usage and Evolution (WAPI), pp. 7-10, 2017, doi: 10.1109/WAPI.2017.1.

[7] Golmohammadi A, Zhang M, Arcuri A. Testing RESTful APIs: A Survey. ACM Transactions on Software Engineering and Methodology. 2022.

[8] R. Gunawan dan A. Rahmatulloh, “JSON Web Token (JWT) untuk Authentication pada Interoperabilitas Arsitektur berbasis RESTful Web Service,” Jurnal Edukasi dan Penelitian. Informamatika., vol. 5, no. 1, hal. 74, 2019, doi: 10.26418/jp.v5i1.27232.

[9] M. S. Rahardjo, “Studi Kasus Dalam Penelitian Kualitatif: Konsep dan Prosedurnya,” Universitas Islam Negeri Maulana Malik Ibrahim Malang, 2017.

[10] E. Indra dan A. Dwi Rizky, “Sistem Informasi Manajemen Kampus dengan Pengembangan Model Smart Campus : (Studi Kasus Di Universitas Prima Indonesia)”, JIKOMSI, vol. 3, no. 2, hal. 15-25, Sep. 2020.

[11] M. M. Yusnorizam and H. Yusof, and Mohd Satar, “The Challenges of Implementing Agile *Scrum* in Information System's Project”. Journal of Adv Research in Dynamical & Control Systems, Vol. 10, 09-Special Issue, 2018, https://www.jardcs.org /backissues/abstract.php?archiveid=5261, Available at SSRN: https://ssrn.com/ abstract=3786550

[12] Open Leaning, “Which is the best Open Source LMS, OpenEdx, Chamilo Or Moodle?”, https://www. openelearning.org/ which-is-the-best-lms-platform-openedx-chamilo-or-moodle, diakses pada 10 Juli 2021

[13] A. Rahmawati dan R. Hadiprakoso, “Rancang Bangun Aplikasi Rekapitulasi Obat dengan Menerapkan Tanda Tangan Digital”, Ultima InfoSys : Jurnal Ilmu Sistem Informasi, vol. 11, no. 2, hal. 119-124, 2020.

[14] B. A. Myers and J. Stylos, “Humancentered design can make ahallication programming

interfaces easier for developers to use," Commun. Acm, vol. 59, no. 6, 2016.

[15] V. Surwase, "REST API Modeling Languages -A Developer's Perspective," IJSTE - International Journal. Scienci and Techno. Eng., vol. 2, no. 10, hal. 634–637, 2016.

[16] S. Ahmed dan Q. Mahmood, "An authentication based scheme for ahallications using JSON web token," Proc. - 22nd International Multitopic Confonrence. INMIC 2019, hal. 1–6, 2019, doi: 10.1109/INMIC48123.2019.9022766

[17] A. Rahmatulloh, H. Sulastri, dan R. Nugroho, "Keamanan RESTful Web Service Menggunakan JSON Web Token (JWT) HMAC SHA-512," Jurnal Nasional Teknik Elektro dan Teknologi Informasi (JNTETI), vol. 7, no. 2, 2018, doi: 10.22146/jnteti.v7i2.417.

[18] M. Haekal dan Eliyani, "Token-based authentication using JSON Web Token on SIKASIR RESTful Web Service," 2016 International Confonrence Informatics Computer. ICIC 2016, hal. 175–179, 2017, doi: 10.1109/IAC.2016.7905711.

[19] N. F. Sitorus, A. Kusyanti, dan A. Bhawiyuga, "Implementasi Autentikasi Berbasis Token Menggunakan Platform-Agnostic Security Tokens (PASETO) Sebagai Mekanisme Autentikansi RESTful API", J-PTIIK, vol. 4, no. 11, hlm. 3947–3955, Okt 2020.

[20] E. Edy, F. Ferdiansyah, W. Pramusinto, and S. Waluyo, "Pengamanan Restful API menggunakan JWT untuk Aplikasi Sales Order", J. RESTI (Rekayasa Sist. Teknol. Inf.) , vol. 3, no. 2, pp. 106 - 112, Aug. 2019.