Multitek Indonesia: Jurnal Ilmiah
Volume: 19 No. 1, 2025, Hal. 1 - 16
ISSN: 1907-6223 (print)
ISSN: 2579-3497 (Online)
http://journal.umpo.ac.id/index.php/multitek

IMPLEMENTASI AUTHENTICATION & AUTHORIZATION BERBASIS JWT PADA SISTEM PENGELOLAAN PERKULIAHAN MENGGUNAKAN ALGORITMA HMAC

Erik Rio Setiawan[1], Adi Fajaryanto Cobantoro[2], Mohammad Bhanu Setyawan[3]

[1) ,2),3)] Teknik Informatika Universitas Muhammadiyah Ponorogo.
Jl. Budi Utomo No 10 Ponorogo
Email : erikriosetiawan15@gmail.com[1], adifajaryanto@umpo.ac.id[2],
mohammad.setyawan@gmail.com[3]

Dikimkan: 16 Februari 2022 Direvisi: 1 Juli 2025 Diterima: 9 Juli 2025

Abstrak

Keamanan dan kerahasiaan data menjadi hal yang sangat penting karena data tersebut bisa digunakan oleh orang yang tidak bertanggung jawab untuk berbuat kejahatan. Hal tersebut tentunya akan berakibat fatal bagi pemilik data apabila terjadi penyalahgunaan data. Fakultas Teknik, merupakan salah satu fakultas yang ada di Universitas Muhammadiyah Ponorogo yang menerapkan sistem pengelolaan perkuliahan berupa praktikum dan PKN (Praktek Kerja Nyata) berbasis digital. Namun, implementasi keamanan data akademik yang disimpan pada sistem tersebut dirasa masih kurang karena hanya menerapkan basic authentication. Maka dari itu, demi meningkatkan keamanan pada data akademik, maka juga ditekankan penggunaan algoritma HMAC (Hash-Based Message Authentication Code) untuk meningkatkan keamanan sistem dari sisi authentication dan authorization. Hasil dari penelitian ini dapat meningkatkan keamanan sistem pengelolaan perkuliahan, sehingga nantinya bisa dihasilkan perangkat lunak yang teruji baik dari sisi fungsionalitas maupun keamanan sistem.

Kata Kunci: Algoritma HMAC, Keamanan Sistem, REST API, JWT, Authentication, Authorization

Abstract

The security and confidentiality of data is very important because the data can be used by people who are not responsible for committing crimes. This of course will be fatal for the data owner in the event of data misuse. Faculty of Engineering, is one of the faculties at Muhammadiyah University of Ponorogo that implements a lecture management system in the form of practicum and digital-based internship. However, the implementation of the security of academic data stored on the system is still lacking because it only applies basic authentication. Therefore, in order to improve the security of academic data, it is also emphasized the use of the HMAC (Hash-Based Message Authentication Code) algorithm to improve system security in terms of authentication and authorization. The results of this research can improve the security of the lecture management system, so that later it can produce software that is tested both in terms of functionality and system security.

Keywords: HMAC Algorithm, System Security, REST API, JWT, Authentication, Authorization

PENDAHULUAN

Pada era digital saat ini, data telah menjadi aset krusial yang beririsan dengan hampir semua aspek kehidupan manusia[1-7]. Setiap kali pengguna mendaftar ke suatu situs atau menelusuri informasi melalui mesin pencari, jejak data senantiasa tercipta. Pelanggaran data—terutama di lingkungan akademik—dapat menurunkan reputasi institusi pendidikan secara signifikan [8-10].

Dalam pengembangan sistem informasi, keamanan menjadi perhatian utama. Dua pilar penting di dalamnya ialah authentication (autentikasi) dan authorization (otorisasi) [11-12]. Mekanisme yang lemah membuka peluang serangan umum seperti *Man-in-the-Middle* (MITM) attack [5][13-15]. Demi meningkatkan perlindungan, banyak studi merekomendasikan penerapan Hash-based Message Authentication Code (HMAC) pada lapisan autentikasi dan otorisasi [16-19].

Fakultas Teknik Universitas Muhammadiyah Ponorogo kini mengelola proses Praktikum dan PKN melalui platform digital. Namun sistem tersebut masih memakai Basic Authentication, yaitu skema HTTP di mana identitas pengguna (user:password) dienkode ke dalam *base64* lalu dikirim di header *Authorization* [7][20-21]. Karena *base64* mudah didekode tanpa lapisan proteksi tambahan, Basic Auth dinilai kurang aman untuk data akademik yang sensitif.

Sebagai solusi, komunitas keamanan merekomendasikan JSON Web Token (JWT)—standar terbuka (RFC 7519) untuk *token-based authentication* dan *information exchange* [22-25]. JWT ringkas, mandiri, serta dapat ditandatangani secara digital menggunakan HMAC, RSA, maupun ECDSA. Kombinasi JWT + HMAC memperkuat integritas token sekaligus mempertahankan kinerja tinggi untuk beban < 10 k rps [26,27].

Berdasarkan permasalahan di atas, penelitian ini mengusung judul "Implementasi Authentication & Authorization Berbasis JWT pada Sistem Pengelolaan Perkuliahan menggunakan Algoritma HMAC." Tujuannya adalah merancang, mengimplementasikan, dan menguji modul keamanan berbasis JWT-HMAC pada platform Fakultas Teknik—sehingga perangkat lunak yang dihasilkan terverifikasi baik secara fungsional maupun dari aspek keamanan.

Authentication adalah proses verifikasi identitas digital subjek yang berkomunikasi dengan sistem. Pengujian skema autentikasi berarti memahami alur verifikasinya dan mencari celah untuk melewatinya. Authorization merupakan proses memvalidasi aksi apa saja yang diizinkan bagi subjek setelah ter-autentikasi—misalnya, API messaging yang hanya mengizinkan pengguna membaca pesan miliknya sendiri. Struktur JWT terdiri atas header, payload, dan signature yang dipisahkan tanda titik (.). JWT dimanfaatkan pada dua skenario utama, Authorization: token dilampirkan pada setiap permintaan setelah proses login berhasil; server memverifikasi signature sebelum memberikan akses. Information exchange: karena ditandatangani secara digital, pesan dijamin autentik dan tidak diubah pihak lain selama transmisi.

Akanksha et al. menerapkan JWT-HMAC pada sistem dan mencatat rata-rata verifikasi < 3 ms [28]. Kim & Park membandingkan JWT-RSA dan JWT-HMAC di arsitektur mikro-layanan, merekomendasikan HMAC untuk beban sedang [29]. Xu et al. menyatakan bahwa JWTKey adalah alat analisis statik berbasis program slicing yang secara otomatis menelusuri source code aplikasi untuk mencari 15 pola salah-guna kriptografi seputar pengelolaan kunci JWT—mulai dari hard-coded secret, panjang kunci HMAC yang lemah, pemilihan algoritme yang keliru, hingga kegagalan mem-verifikasi tanda tangan [30].

Gap penelitian tersebut ialah (1) belum ada studi di lingkungan akademik yang memadukan JWT-HMAC dengan pengujian *unit*, *integration*, dan *end-to-end* terstruktur, serta (2) belum ada evaluasi *usability* bagi pengguna non-teknis. Riset ini bertujuan menutup kedua celah tersebut.

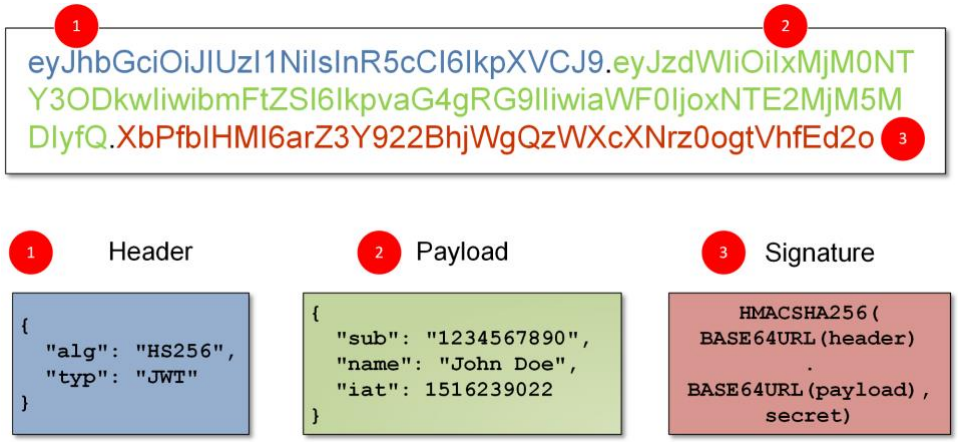
JSON Web Token (*JWT*) adalah suatu standar terbuka (*RFC 7519*) yang mendefinisikan cara ringkas dan mandiri untuk mentransmisikan informasi antar pihak secara aman dengan format objek *JSON* (*JavaScript Object Notation*). Informasi ini dapat diverifikasi dan terpercaya karena ditandatangani secara digital. *JWT* dapat ditandatangani menggunakan kunci privat maupun kunci privat dan publik menggunakan algoritma *HMAC*, *RSA*, atau *ECDS* [31].

JWT dapat digunakan sebagai *authorization* dan pertukaran informasi [32].

1. *Authorization*, yakni skenario *JWT* yang umum digunakan. Setelah pengguna *login*, untuk setiap *request* ke *server* akan menyertakan *JWT*, sehingga pengguna dapat diizinkan mengakses rute, layanan, maupun sumber yang diizinkan menggunakan token tersebut.

2. *Information exchange* (pertukaran informasi), dimana *JWT* bisa dijadikan pilihan yang tepat untuk mentransmisikan informasi antar pihak. Karena *JWT* dapat ditandatangani (sebagai contoh, menggunakan pasangan kunci publik/privat), dapat dipastikan pesan yang dikirimkan original dari pengirim, dan tidak dimodifikasi oleh pihak yang tidak bertanggung jawab.

JWT terdiri dari tiga bagian yang dipisahkan oleh *dot* (.), yakni *header*, *payload*, dan *signature*.



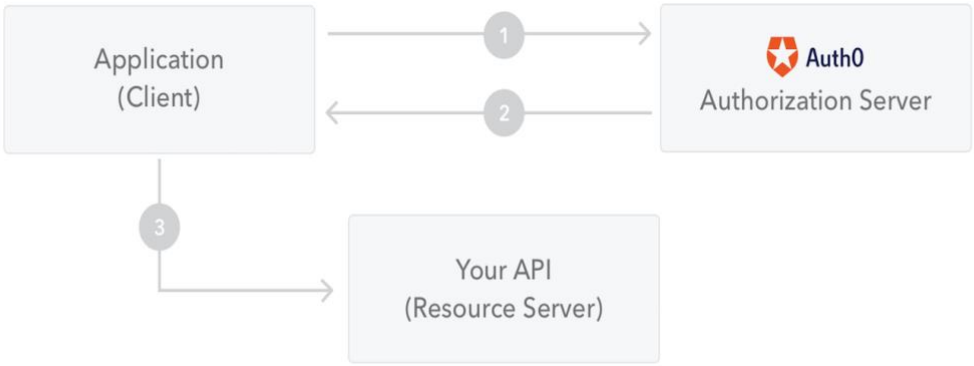
Gambar 1. Struktur *JWT*

1. *Header*
   Header secara khusus terdiri dari 2 bagian, yaitu tipe tokennya (*JWT*) dan *signing algorithm* (algoritma penandatanganan) yang digunakan, seperti *HMAC SHA265* maupun *RSA*. Sebagai contoh:
   ```
   {
     "alg": "HS256",
     "typ": "JWT"
   }
   ```
   Kemudian, *JSON* ini di-encode menggunakan *Base64Url* untuk membentuk bagian pertama dari *JWT*.
2. *Payload*
   Bagian kedua dari token *JWT* adalah *payload*, yang berisikan *claims*. *Claims* merupakan pernyataan mengenai entitas (khususnya pengguna) dan data tambahan. Contoh *payload* sebagai berikut:
   ```
   {
     "sub": "1234567890",
     "name": "John Doe",
     "admin": true
   }
   ```
   Kemudian, *payload* ini di-*encode* menggunakan *Base64Url* untuk membentuk bagian kedua dari *JWT*.
3. *Signature*
   Untuk membuat bagian *signature* (tanda tangan), akan dibutuhkan *header* yang telah di-*encoding*, *payload* yang telah di-*encoding*, sebuah kunci rahasia, algoritma yang telah ditentukan pada *header*, dan menandatanganinya menggunakan algoritma tersebut. Sebagai contoh, jika kita ingin menggunakan algoritma *HMAC SHA256*, tanda tangan akan dibuat dengan cara berikut.
   ```
   HMACSHA256(
     base64UrlEncode(header) + "." +
     base64UrlEncode(payload),
     secret)
   ```
   Signature digunakan untuk memverifikasi bahwa data tidak diubah selama proses transmisi.

Output dari proses *signing* (penandatanganan) ialah *Base64-URL string* yang dipisahkan oleh tanda *dot* (.) serta dapat dengan mudah dimasukkan pada protokol *HTTP* atau yang lain. Struktur akhir dari *JWT* yang siap ditransmisikan sebagai berikut.

Gambar 2. Struktur JWT yang Telah Di-encode

Di saat proses *authentication*, ketika pengguna berhasil *login* menggunakan kredensialnya, sebuah *JSON Web Token* akan didapatkan dari *back-end*. Karena token tersebut bersifat kredensial, token harus disimpan dengan aman. Ketika pengguna ingin mengakses rute atau *resource* yang di proteksi, pengguna (aplikasi *client* / *front-end*) harus mengirim *JWT*, khususnya pada *header Authorization* menggunakan skema *Bearer*. Kemudian, *back-end server* akan melakukan validasi *JWT*. Apabila proses validasi berhasil, pengguna akan diizinkan untuk mengakses *resource* yang diproteksi [8].



Gambar 3. Proses Authentication dan Authorization dengan JWT

Hash-based Message Authentication Code (*HMAC*) adalah algoritma untuk menghitung *tag authentication* dari suatu pesan dan kunci rahasia. Pengguna dengan kunci rahasia yang sama akan dapat menghitung *tag* yang sama dari pesan yang sama, akan tetapi setiap perubahan pesan akan menghasilkan *tag* yang berbeda. Penyerang tanpa akses kunci rahasia tidak akan bisa meng-*generate tag* yang benar dari suatu pesan. Algoritma *HMAC* bekerja seperti fungsi *hash* kriptografi pada umumnya, akan tetapi menggabungkan kunci rahasia yang hanya diketahui oleh *server API*-nya [7].

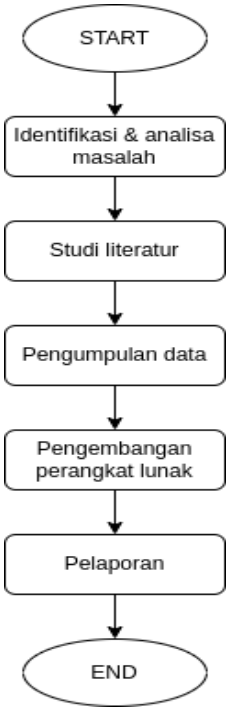
O*utput* yang dihasilkan dari fungsi / Algoritma *HMAC* adalah *tag authentication* singkat yang dapat ditambahkan ke token. Penyerang tanpa akses kunci rahasia tidak akan bisa memanipulasi data token, karena *tag* akan berubah setiap kali data pada token diubah. Dengan demikian, dapat dilakukan pencegahan terhadap pemalsuan token [7].

Protokol komunikasi yang aman seperti *IPSec*, *SSH*, dan *TLS* menggunakan algoritma *HMAC* sebagai hash function. *HMAC* menggunakan fungsi *hash* untuk meng-*generate tag* dari suatu data. Algoritma *HMAC* sendiri memiliki beberapa jenis, yakni *HMAC SHA-256* dan *HMAC SHA-512* [9].

METODE PENELITIAN

Tahap Penelitian

Tahapan yang ditempuh dari penelitian ini akan dijabarkan pada *flowchart* berikut:



Gambar 4. *Flowchart* Tahap Penelitian

Identifikasi dan Analisa Masalah

Pada tahap ini, peneliti melakukan identifikasi dan analisa terhadap permasalahan yang ada. Penelitian berlokasi di Fakultas Teknik Universitas Muhammadiyah Ponorogo yang beralamat di Jalan Budi Utomo No. 10, Ronowijayan, Kec. Siman, Kabupaten Ponorogo, Jawa Timur 63471. Subjek pada penelitian ini adalah dosen dan mahasiswa di Fakultas Teknik Universitas Muhammadiyah Ponorogo. Selanjutnya, dilakukan analisa masalah yang terjadi, berupa kerentanan terhadap sistem authentication dan authorization pada sistem pendukung perkuliahan di Fakultas Teknik Universitas Muhammadiyah Ponorogo.

Studi Literatur

Di dalam melakukan pengumpulan data sebagai sarana pendukung pada penelitian ini, digunakan studi literatur dari berbagai studi pustaka, baik berupa buku, jurnal, dan website resmi terkait permasalahan dan metode yang akan digunakan untuk menyelesaikan permasalah tersebut.

Pengumpulan Data

Pada tahap ini, dilakukan pengumpulan data dengan cara berkoordinasi bersama LPSI sebagai penyedia data akademik berupa data mata kuliah, mahasiswa, dosen, prodi, dan beberapa data penting lainnya

Pada tahap ini, dilakukan pengumpulan data dengan cara berkoordinasi bersama LPSI sebagai penyedia data akademik berupa data mata kuliah, mahasiswa, dosen, prodi, dan beberapa data penting lainnya.

Pengembangan Perangkat Lunak

Dalam melakukan pengembangan perangkat lunak, digunakan model *prototyping*.

1. *Authentication* & *Authorization* Berbasis *JWT* dengan Algoritma *HMAC*
   a. Struktur *JWT*

Berikut ini merupakan struktur token *JWT* pada sistem pengelolaan perkuliahan Fakultas Teknik Universitas Muhammadiyah Ponorogo.


```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ey
JuaW0iOjE4NTMyOTY4LCJrb2RlRnAiOiI1MyIsI
nJvbGUiOiJzdHVkZW50IiwiaWF0IjoxNjMzMzI0
NjgyfQ.ONqZs25zXOiEoDu7QzWjmNKJyOaJwfPa
gClfqadWOc8
```


Gambar 5. *Encoded JWT* Token

Dari gambar diatas, dapat dicermati bahwa token *JWT* tersebut terdiri dari *header, payload,* dan *signature* yang masing-masing dipisahkan menggunakan tanda titik.

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

PAYLOAD: DATA

```
{
  "nim": 18532968,
  "kodeFp": "53",
  "role": "student",
  "iat": 1633324682
}
```


Gambar 6. *Decoded JWT* Token (Mahasiswa)

Pada *user* dengan *role* mahasiswa, token *JWT* yang telah di *decode* berisikan *field alg* (algoritma), *type* (tipe token), nim, kodeFp, *role*, dan *iat* (*issued at*).

HEADER: ALGORITHM & TOKEN TYPE

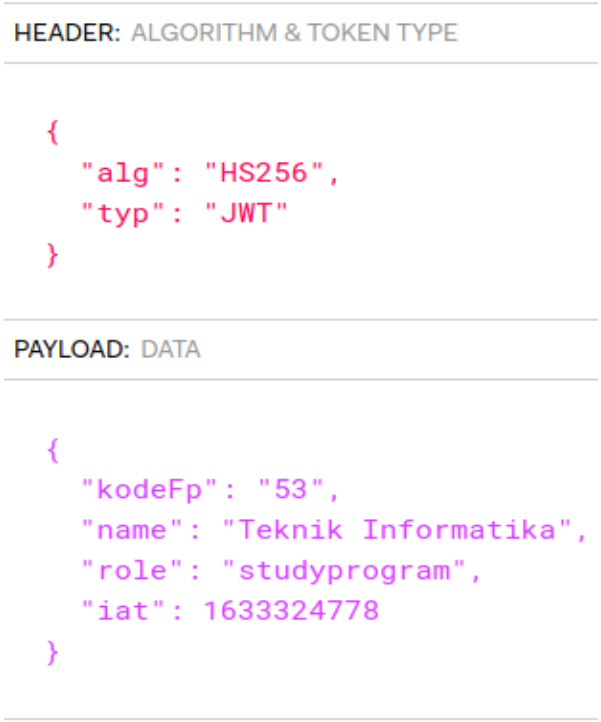
```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

PAYLOAD: DATA

```
{
  "nik": "1984092420130913",
  "kodeFp": "53",
  "role": "lecturer",
  "iat": 1633324726
}
```


Gambar 7. *Decoded JWT* Token (Dosen)

Pada *user* dengan *role lecturer*, token *JWT* yang telah di *decode* berisikan *field alg* (algoritma), *type* (tipe token), nik, kodeFp, *role*, dan *iat* (*issued at*).

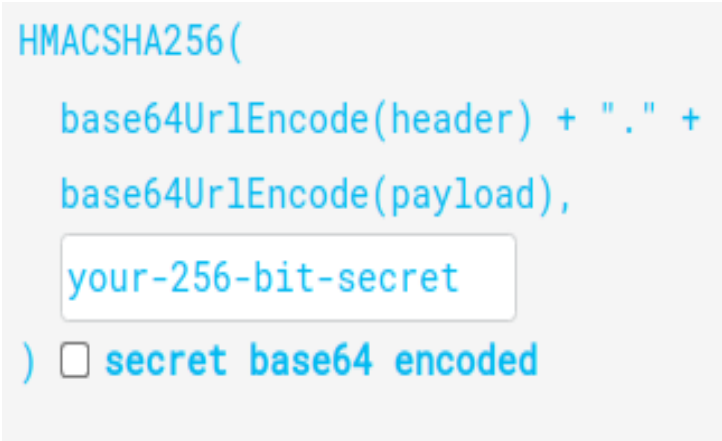


Gambar 8. *Decoded JWT* Token (Program Studi)

Pada *user* dengan *role studyprogram*, token *JWT* yang telah di *decode* berisikan *field alg* (algoritma), *type* (tipe token), kodeFp, name, *role*, dan *iat* (*issued at*).

b. Proses Pembuatan *JWT* dengan Algoritma *HMAC*

Pembuatan token *JWT* melalui beberapa proses. Pertama-tama, buat terlebih dahulu *header* yang berisi data algoritma yang dipakai dan tipe tokennya dalam format *JSON*. Lalu *encode header* tersebut menggunakan algoritma *Base64Url.* Kemudian definisikan *payload* yang akan digunakan dengan format *JSON* dan lakukan *encode* payload menggunakan algoritma *Base64Url*. Terakhir, *generate signature* menggunakan algoritma *HMACSHA256* dengan rumus sebagai berikut:



Gambar 9. Proses *Generate Signature*

Dari gambar diatas, algoritma *HMAC* memerlukan 2 parameter, yakni *header* dan *payload* yang dipisahkan dengan tanda titik serta sebuah *private key.* Dengan demikian, akan dihasilkan *signature* untuk menjaga validitas suatu token *JWT*.

```
const jwt = require("jsonwebtoken");
const config = require("config");
const Joi = require("joi");

class Student {
  constructor(nim, kodeFp, password) {
    this.nim = nim;
    this.kodeFp = kodeFp;
    this.password = password;
  }

  generateAuthToken() {
    const token = jwt.sign(
      { nim: this.nim, kodeFp: this.kodeFp, role: "student" },
      config.get("jwtPrivateKey"),
      { algorithm: 'HS256' }
    );
    return token;
  }

  static validate(student) {
    const schema = Joi.object({
      studentId: Joi.string().length(8).required(),
      password: Joi.string().min(5).max(50).required(),
    });
    return schema.validate(student);
  }
}

module.exports = Student;
```

Gambar 10. Kode Program Pembuatan *JWT* Token.

Dari kode program diatas, dapat disimpulkan bahwa di dalam pembuatan token *JWT*, kita membutuhkan *header* berupa algoritma yang akan digunakan untuk pembuatan *signature*, *payload*, dan *private key*.

c. Proses *Validasi JWT* dengan Algoritma *HMAC*

Untuk mengecek kevalidan token *JWT*, *header* dan *payload* yang dikirim aplikasi *client* akan di *hash* menggunakan algoritma *HMAC* dan *private key* yang sama ketika melakukan *signature*. Apabila *cipher text* hasil dari *hash* sama dengan *signature* yang dikirim oleh *client,* maka token tersebut valid. Namun apabila berbeda, maka token tersebut tidak valid.

```go
package middleware

import (
    "fmt"
    "github.com/gofiber/fiber/v2"
    "github.com/golang-jwt/jwt"
    "os"
    "strconv"
)

// tokenHandler is a middleware function to handle the JWT token
func tokenHandler(c *fiber.Ctx) error {
    tokenString := string(c.Request().Header.Peek("Auth-Token"))
    if tokenString == "" {
        return c.Status(fiber.StatusUnauthorized).JSON(fiber.Map{
            "status":  "error",
            "message": "No token provided.",
            "data":    nil,
        })
    }

    token, err := jwt.Parse(tokenString, func(token *jwt.Token) (interface{}, error) {
        if _, ok := token.Method.(*jwt.SigningMethodHMAC); !ok {
            return nil, fmt.Errorf("unexpected signing method: %v", token.Header["alg"])
        }
        jwtPrivateKey := os.Getenv("JWT_PRIVATE_KEY")
        return []byte(jwtPrivateKey), nil
    })

    if err != nil {
        return c.Status(fiber.StatusUnauthorized).JSON(fiber.Map{
            "status":  "error",
            "message": "Invalid token.",
            "data":    nil,
        })
    }

    if claims, ok := token.Claims.(jwt.MapClaims); ok && token.Valid {
        role := claims["role"].(string)
        if role == "student" {
            studentId := int(claims["nim"].(float64))
            c.Request().Header.Set("Role", role)
            c.Request().Header.Set("Id", strconv.Itoa(studentId))
            c.Request().Header.Set("Study-Program-Code", claims["kodeFp"].(string))
        } else if role == "lecturer" {
            c.Request().Header.Set("Role", role)
            c.Request().Header.Set("Id", claims["nik"].(string))
            c.Request().Header.Set("Study-Program-Code", claims["kodeFp"].(string))
        } else if role == "studyprogram" {
            c.Request().Header.Set("Role", role)
            c.Request().Header.Set("Id", claims["kodeFp"].(string))
            c.Request().Header.Set("Study-Program-Code", claims["kodeFp"].(string))
        } else {
            return c.Status(fiber.StatusUnauthorized).JSON(fiber.Map{
                "status":  "error",
                "message": "Role unknown.",
                "data":    nil,
            })
        }
        return c.Next()
    } else {
        return c.Status(fiber.StatusUnauthorized).JSON(fiber.Map{
            "status":  "error",
            "message": "Invalid token.",
            "data":    nil,
        })
    }
}
```

Gambar 11. Kode Program Validasi *JWT* Token.

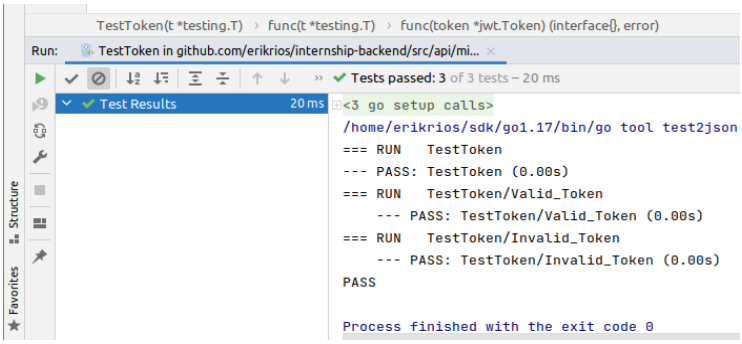
Unit Testing

```go
package middleware

import (
    "fmt"
    "github.com/golang-jwt/jwt"
    "testing"
)

func TestToken(t *testing.T) {
    tests := []struct {
        description string
        token       string
        isValid     bool
    }{
        {
            description: "Valid Token",
            token: "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9." +
                "eyJuaW0iOjE4NTMyOTY4LCJrb2RlRnAiOiI1MyIsInJvbGUiOiJzdHVkZW50IiwiaWF0IjoxNjMzMzI0NjgyfQ." +
                "ONqZs25zXOiEoDu7QzWjmNKJyOaJwfPagClfqadWOc8",
            isValid: true,
        },
        {
            description: "Invalid Token",
            token: "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9." +
                "eyJuaW0iOjE4NTHMyOTY4LCJrb2RlRnAiOiI1MyLsInJvbGUiOiJzdMVkZW50IiwiaWF0IjoxNjMzMzI0NjgyfQ." +
                "ONqZs25zXOiEoDu7QzWjmNKJyOaJwfPagClfqadWOc8",
            isValid: false,
        },
    }

    for _, test := range tests {
        t.Run(test.description, func(t *testing.T) {
            _, err := jwt.Parse(test.token, func(token *jwt.Token) (interface{}, error) {
                if _, ok := token.Method.(*jwt.SigningMethodHMAC); !ok {
                    return nil, fmt.Errorf( format: "unexpected signing method: %v", token.Header["alg"])
                }
                jwtPrivateKey := "facultySystemJwtPrivateKey"
                return []byte(jwtPrivateKey), nil
            })

            isTokenValid := err == nil
            if test.isValid != isTokenValid {
                t.Fatal(err)
            }
        })
    }
}
```

Gambar 12. Kode *Unit Testing*

Pada kode diatas, dibuat dua *test case*. *Test case* yang pertama berisi token valid, yang bernilai "*eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJuaW0iOjE4NTMyOTY4LCJrb2RlRnAiOiI1MyIsInJvbGUiOiJzdHVkZW50IiwiaWF0IjoxNjMzMzI0NjgyfQ.ONqZs25zXOiEoDu7QzWjmNKJyOaJwfPagClfqadWOc8*", dengan ekspektasi bernilai *true*, sedangkan *test case* yang kedua berisi token tidak valid, yang bernilai "*eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJuaW0iOjE4NTMyOTY4LCJrn2RlRnAiOiI1MyIsInJvbGLiOiJzdHVkZW50IiwiaWF0IjoPNjMzMzI0NjgyfQ.ONqZs25zXOiEoDu7QzWjmNKJyOaJwfPagClfqadWOc8*", dengan ekspektasi bernilai *false*. Selanjutnya dilakukan perulangan sebanyak *test case* yang ada, lalu dilakukan pengecekan antara hasil balikan nilai dari *method jwt.Parse* dengan ekspektasi yang diharapkan. Apabila *jwt.Parse* menghasilkan *error*, maka dapat dipastikan bahwa token yang dikirim

dari *client* tidak valid. Namun, sebaliknya apabila *error*nya bernilai *null*, maka dapat dipastikan bahwa token yang dikirim oleh *client* valid.



Gambar 13. Hasil *Unit Testing*

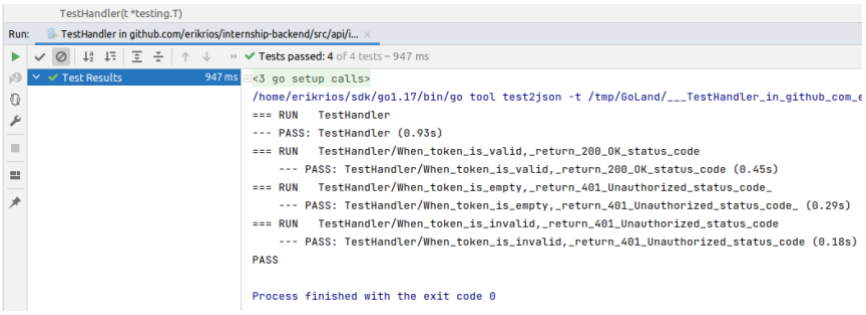
Dari kode *unit testing*, ketika dijalankan menghasilkan "*pass*", baik pada *test case* yang pertama maupun *test case* yang kedua, sehingga dapat dipastikan kode program berfungsi dengan semestinya.

Integration Testing

```go
package internships

import (
    "net/http"
    "testing"
)

func TestHandler(t *testing.T) {
    tests := []struct {
        description        string
        token              string
        expectedStatusCode int
    }{
        {
            description: "When token is valid, return 200 OK status code",
            token: "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9." +
                "eyJuaW0iOjE4NTMyOTY4LCJrb2RlRnAiOiI1MyIsInJvbGUiOiJzdHVkZW50IiwiaWF0IjoxNjMzMzI0NjgyfQ." +
                "ONqZs25zXOiEoDu7QzWjmNKJyOaJwfPagClfqadWOc8",
            expectedStatusCode: http.StatusOK,
        },
        {
            description:        "When token is empty, return 401 Unauthorized status code ",
            token:              "",
            expectedStatusCode: http.StatusUnauthorized,
        },
        {
            description: "When token is invalid, return 401 Unauthorized status code",
            token: "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9." +
                "eyJuaW0iOjE4NTMyOTY4LCJrn2RlRnAiOiI1MyIsInJvbGLiOiJzdHVkZW50IiwiaWF0IjoPNjMzMzI0NjgyfQ." +
                "ONqZs25zXOiEoDu7QzWjmNKJyOaJwfPagClfqadWOc8",
            expectedStatusCode: http.StatusUnauthorized,
        },
    }

    for _, test := range tests {
        t.Run(test.description, func(t *testing.T) {
            client := &http.Client{}
            req, err := http.NewRequest(http.MethodGet, url: "http://simft.umpo.ac.id/backend-internship/api/v1/internships", body: nil)
            req.Header.Set( key: "Auth-Token", test.token)
            if err != nil {
                t.Fatal(err)
            }

            resp, err := client.Do(req)
            if err != nil {
                t.Fatal(err)
            }

            if resp.StatusCode != test.expectedStatusCode {
                t.Fail()
            }
        })
    }
}
```

Gambar 14. Kode *Integration Testing*

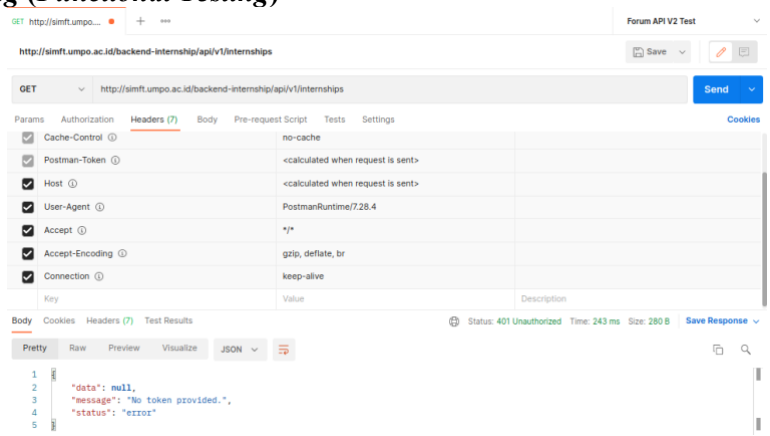
Pada kode diatas, dibuat tiga *test case*. *Test case* yang pertama berisi token *JWT* valid, yang bernilai "*eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJuaW0iOjE4NTMyOTY4LCJrb2RlRnAiOiI1MyIsInJvbGUiOiJzdHVkZW50IiwiaWF0IjoxNjMzMzI0NjgyfQ.ONqZs25zXOiEoDu7QzWjmNKJyOaJwfPagClfqadWOc8"*, dengan ekspektasi mengembalikan nilai *status code* 200, dimana *response code* tersebut mengindikasikan bahwa *request* berhasil diproses oleh *API*. *Test case* kedua berisi token kosong, dengan ekspektasi mengembalikan nilai *status code* 401. *Test case* yang terakhir, berisi token *JWT* tidak valid, yang bernilai "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJuaW0iOjE4NTMyOTY4LCJrn2RlRnAiOiI1MyIsInJvbGLiOiJzdHVkZW50IiwiaWF0IjoPNjMzMzI0NjgyfQ.ONqZs25zXOiEoDu7QzWjmNKJyOaJwfPagClfqadWOc8", dengan ekspektasi mengembalikan nilai *status code* 401, dimana *status code* tersebut mengindikasikan bahwa token yang dikirimkan tidak valid. Selanjutnya dilakukan *request* ke *API*, lalu membandingkan *response code* yang dihasilkan dengan ekspektasi *response code* yang diinginkan.



Gambar 15. Hasil *Integration Testing*

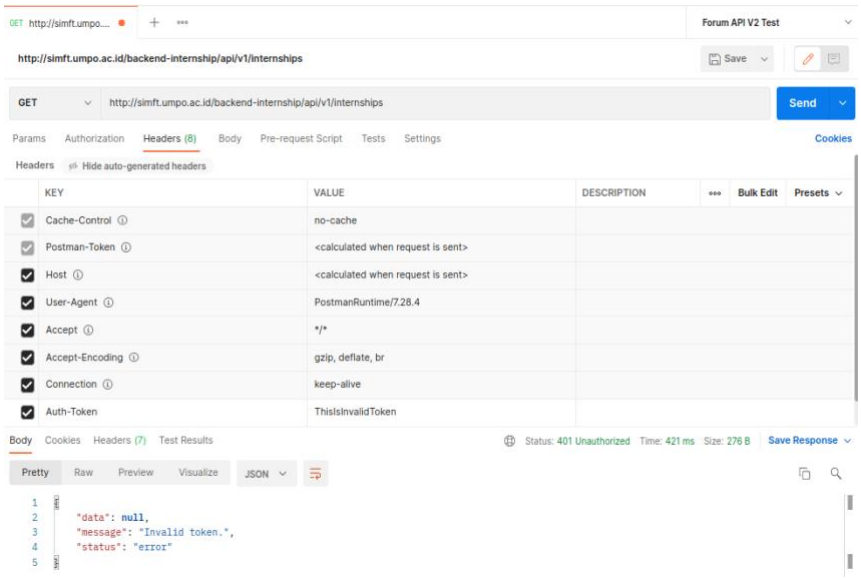
Hasil *integration testing* menunjukkan bahwa semua *test case* yang dijalankan berhasil, sehingga dapat disimpulkan bahwa dari sisi *integration test*, berjalan dengan lancar.

End-to-end Testing (*Functional Testing*)



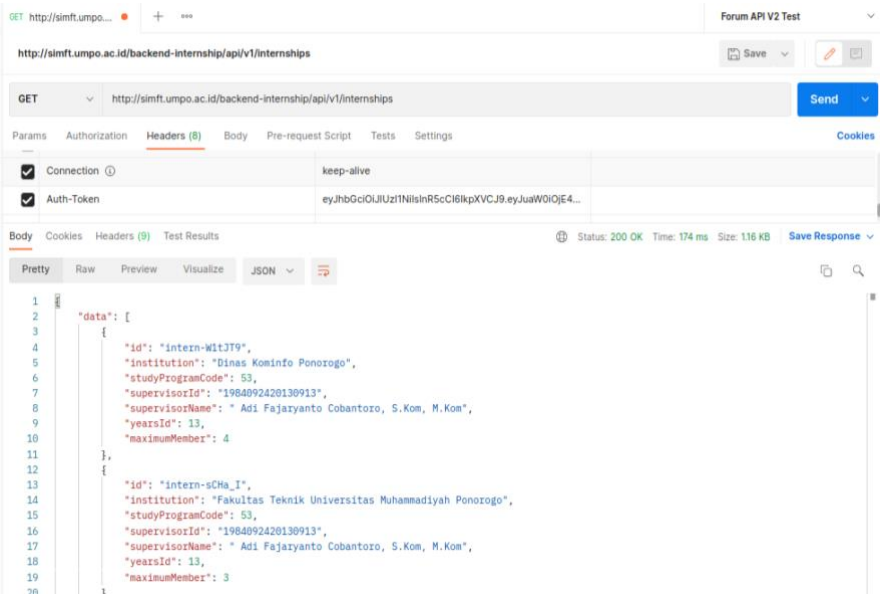
Gambar 16. *End-to-end Testing* Dengan Token Kosong

Dari percobaan menggunakan *Postman* diatas, ketika token yang dikirimkan ke *API* bernilai kosong, maka *status code* yang dihasilkan adalah 401 dengan *error message* "*No token provided*". *Status code* tersebut mengindikasikan bahwa token yang dikirimkan tidak valid.



Gambar 17. *End-to-end Testing* Dengan Token Tidak Valid

Dari hasil diatas, ketika token yang dikirimkan ke *API* bernilai tidak valid, maka *status code* yang dihasilkan adalah 401 dengan *error message* "*Invalid token*". *Status code* tersebut mengindikasikan bahwa token yang dikirimkan tidak valid.



Gambar 18. *End-to-end Testing* Dengan Token Valid

Dari hasil diatas, ketika token yang dikirimkan ke *API* adalah valid, maka *status code* yang dihasilkan adalah 200, dimana *status code* tersebut mengindikasikan bahwa *request* berhasil diproses oleh API dengan *response body* bernilai *JSON Object* yang siap diolah dan ditampilkan oleh *front-end*.

Berdasarkan hasil pengujian sistem menggunakan metode pengujian berupa unit testing, integration testing, dan end-to-end testing, dapat disimpulkan bahwa sistem berjalan dengan baik. Pengujian dilakukan untuk memastikan bahwa implementasi *authentication* dan *authorization* pada sistem pengelolaan perkuliahan menggunakan algoritma *HMAC,* dapat bekerja sesuai yang diharapkan. Dengan demikian, dapat dihasilkan perangkat lunak yang teruji baik dari sisi fungsionalitas maupun keamanan sistem.

Sebagai tindak lanjut penelitian, langkah pertama yang direncanakan adalah menerapkan protokol OAuth 2.0 beserta OpenID Connect guna menghasilkan token federatif yang dapat dipakai secara mulus di berbagai layanan internal maupun eksternal; setelah arsitektur otentikasi terdistribusi

ini stabil, sistem akan diuji melalui penetration-testing komprehensif—termasuk uji black-box dengan OWASP ZAP—untuk menyingkap kerentanan non-token, seperti konfigurasi server, injeksi, dan kelemahan kontrol sesi; temuan pengujian tersebut kemudian dijadikan dasar dalam eksperimen komparatif performa dan keamanan antara skema JWT-HMAC dan Paseto pada lingkungan microservices berbasis Kubernetes, sehingga institusi dapat menentukan mekanisme token yang paling efisien sekaligus tangguh untuk implementasi produksi skala besar.

DAFTAR RUJUKAN

[1] H. Pramudya, A. F. Cobantoro, and J. Karaman, "IMPLEMENTASI ALGORITMA SELECTION SORT DALAM SISTEM ABSENSI SISWA UNTUK PENGURUTAN KEAKTIFAN BERDASARKAN KEHADIRAN," *JATI (Jurnal Mahasiswa Teknik Informatika)*, vol. 9, no. 2, pp. 2887–2895, 2025.

[2] M. F. Giandra, A. F. Cobantoro, and K. Nurfitri, "IMPLENMENTASI METODE APRIORI DALAM PENGELEMPOKAN KELAS SANTRI BARU: Studi Kasus Pondok Pensatren Salafiyah Hudatul Muna," *JI-Tech*, vol. 20, no. 2, pp. 22–30, 2024.

[3] A. Fajaryanto, I. A. Zulkarnain, F. Masykur, and Y. Litanianda, "IMPLEMENTASI INTERNET AMAN DI FASILITAS UMUM DESA NGRUPIT KABUPATEN PONOROGO MENGGUNAKAN WEB PROXY," *Jurnal Pengabdian Kepada Masyarakat Bersinergi Inovatif*, vol. 2, no. 1, pp. 206–214, 2024.

[4] N. D. Agustin, A. F. Cobantoro, M. B. Setyawan, and K. Nurfitri, "Penerapan Algoritma Linear Search Di Aplikasi Secondhand," *NERO (Networking Engineering Research Operation)*, vol. 8, no. 2, pp. 107–122, 2023.

[5] F. Masykur, A. Prasetyo, I. Abdurrozaq, A. F. Cobantoro, A. R. Yusuf, and M. B. Setyawan, "Thresholding Value for Contour Segmentation Model in the Detection of Infected Plants with Drone Acquisition.," *Ingénierie des Systèmes d'Information*, vol. 30, no. 1, 2025.

[6] Y. Litanianda, D. A. Riyanto, A. Prasetyo, A. F. Cobantoro, and I. A. Zulkarnain, "Identifikasi Performa Algoritma Fuzzy Mamdani Sebagai Kendali Proses Koagulasi pada Internet of Thing Pembuatan Tahu," *bit-Tech*, vol. 7, no. 2, pp. 608–617, 2024.

[7] M. B. Setyawan, C. W. Aditya, A. F. Cobantoro, and J. Karaman, "Prokes Warning Attendance System Dengan Kecerdasan Buatan Model Face Recognition Menggunakan Algoritma Haarcascade," *Jurnal Sistem dan Teknologi Informasi (JSTI)*, vol. 6, no. 2, 2024.

[8] Md. M. Haqqani, G. B. Regulwar, B. A. Goud, S. P. Veggalam, K. Prakash, and E. R. Kumar, "Enhancing Remote Learning By Using Competitive Coding Platform," in *Proceedings of NKCon 2024 - 3rd Edition of IEEE NKSS's Flagship International Conference: Digital Transformation: Unleashing the Power of Information*, 2024. doi: 10.1109/NKCon62728.2024.10774847.

[9] Y. Litanianda and A. Fajaryanto, "Implementasi Decision Support System Rekomendasi Beasiswa Pendidikan Menggunakan Metode Simple Additive Weighting," *Jurnal Sains Komputer dan Sistem Informasi*, vol. 2, no. 1, pp. 142–149, 2024.

[10] M. Reza, A. F. Cobantoro, and I. A. Zulkarnain, "HARDENING SERVER MENGGUNAKAN METODE PORT KNOCKING PADA SISTEM PROGRAM STUDI TEKNIK INFORMATIKA UNIVERSITAS MUHAMMADIYAH PONOROGO," *Jurnal Ilmiah Informatika Komputer*, vol. 29, no. 3, pp. 298–315, 2024.

[11] P. Varalakshmi, B. Guhan, P. Vignesh Siva, T. Dhanush, and K. Saktheeswaran, "Improvising JSON Web Token Authentication in SDN," in *2022 International Conference on Communication, Computing and Internet of Things, IC3IoT 2022 - Proceedings*, 2022. doi: 10.1109/IC3IOT53935.2022.9767873.

[12] L. Zhang, C. Zhou, and J. Wen, "APSH-JWT: an authentication protocol based on JWT with scalability and heterogeneity in edge computing," *Wireless Networks*, vol. 31, no. 3, pp. 2939 – 2953, 2025, doi: 10.1007/s11276-025-03926-2.

[13] B. M. Nema and S. J. Mohammed, "Secure Location Privacy Transmitting Information on Cellular Networks," *Iraqi Journal of Science*, vol. 63, no. 11, pp. 5004 – 5014, 2022, doi: 10.24996/ijs.2022.63.11.35.

[14] C. Sompong, A. Kaiburt, P. Penjamuk, A. Supan, T. Boonphiohat, and P. Kunmee, "Multi-Factor Authentication for Web Applications Using JWT and Face Embedding-Based Recognition," in *International Conference on Cybernetics and Innovations, ICCI 2025*, 2025. doi: 10.1109/ICCI64209.2025.10987510.

[15] A. F. Nugraha, H. Kabetta, I. K. S. Buana, and R. B. Hadiprakoso, "Performance and Security Comparison of Json Web Tokens (JWT) and Platform Agnostic Security Tokens (PASETO) on RESTful APIs," in *Proceedings - 2023 IEEE International Conference on Cryptography, Informatics, and Cybersecurity: Cryptography and Cybersecurity: Roles, Prospects, and Challenges, ICoCICs 2023*, 2023, pp. 15 – 22. doi: 10.1109/ICoCICs58778.2023.10277377.

[16] A. R. Emanuela, G. Mihaela, and T. Daniela, "Enhancing Security in Data Exchange: Mitigating Risks Solutions in Base64 Encoding and JSON Web Tokens," in *2024 16th International Symposium on Electronics and Telecommunications, ISETC 2024 - Conference Proceedings*, 2024. doi: 10.1109/ISETC63109.2024.10797302.

[17] S. Chopra, A. Singh, and A. Singh, "An Authentication Based Scheme for Mobile Applications Using THJWT," in *CEUR Workshop Proceedings*, 2022, pp. 13 – 27. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85140889509&partnerID=40&md5=00d4982e752e512239f24ad6b2967a1f

[18] S. Huda *et al.*, "A Secure Authentication for Plant Monitoring System Sensor Data Access," in *Digest of Technical Papers - IEEE International Conference on Consumer Electronics*, 2024. doi: 10.1109/ICCE59016.2024.10444465.

[19] S. Huda *et al.*, "IoT-Enabled Plant Monitoring System with Power Optimization and Secure Authentication," *Computers, Materials and Continua*, vol. 81, no. 2, pp. 3165 – 3187, 2024, doi: 10.32604/cmc.2024.058144.

[20] D. Hastbacka, M. Tran, P. Kannisto, M. Filppula, and P. Varga, "External Token-Based Authorization of Data-Driven Integrations and Service Compositions in MQTT 5," in *IECON Proceedings (Industrial Electronics Conference)*, 2023. doi: 10.1109/IECON51785.2023.10311779.

[21] M. More, R. Wagholikar, and S. Chopade, "CloudCraft: Transforming Resume Management System With Dynamic Cloud Deployment," in *14th IEEE Symposium on Computer Applications and Industrial Electronics, ISCAIE 2024*, 2024, pp. 99 – 103. doi: 10.1109/ISCAIE61308.2024.10576573.

[22] S. Ahmad, M. Arif, J. Ahmad, and S. Mehfuz, "A TOTP-based secure data storage system in the cloud environment using the JWT token approach," *International Journal of System Assurance Engineering and Management*, vol. 16, no. 4, pp. 1565 – 1578, 2025, doi: 10.1007/s13198-025-02775-8.

[23] R. Wadagave, S. Karoshi, P. Ravan, R. Santikar, and R. Deshmukh, "Web Application based Event Organisation Portal using MEAN Stack," in *International Conference on Sustainable Computing and Data Communication Systems, ICSCDS 2022 - Proceedings*, 2022, pp. 1427 – 1430. doi: 10.1109/ICSCDS53736.2022.9760955.

[24] E. S. Mansur, A. Rahmatulloh, R. N. Shofa, and I. Darmawan, "AMAN: Token-based Authentication to Improved Single Sign-On Security between Systems," in *ICADEIS 2023 - International Conference on Advancement in Data Science, E-Learning and Information Systems: Data, Intelligent Systems, and the Applications for Human Life, Proceeding*, 2023. doi: 10.1109/ICADEIS58666.2023.10270904.

[25] R. Priyadarshini *et al.*, "Novel framework based on ensemble classification and secure feature extraction for COVID-19 critical health prediction," *Eng Appl Artif Intell*, vol. 126, 2023, doi: 10.1016/j.engappai.2023.107156.

[26] S. Prinakaa, V. Bavanika, S. Sanjana, S. Srinivasan, and V. Sarasvathi, "A Real-Time Approach to Detecting API Abuses Based on Behavioral Patterns," in *Proceedings - 2024 8th International Conference on Cryptography, Security and Privacy, CSP 2024*, 2024, pp. 24 – 28. doi: 10.1109/CSP62567.2024.00012.

[27] S. Dalimunthe, J. Reza, and A. Marzuki, "THE MODEL FOR STORING TOKENS IN LOCAL STORAGE (COOKIES) USING JSON WEB TOKEN (JWT) WITH HMAC (HASH-BASED MESSAGE AUTHENTICATION CODE) IN E-LEARNING SYSTEMS," *Journal of Applied Engineering and Technological Science*, vol. 3, no. 2, pp. 149 – 155, 2022, doi: 10.37385/jaets.v3i2.662.

[28] Akanksha and A. Chaturvedi, "Comparison of Different Authentication Techniques and Steps to Implement Robust JWT Authentication," in *7th International Conference on Communication and*

Electronics Systems, ICCES 2022 - Proceedings, 2022, pp. 772 – 779. doi: 10.1109/ICCES54183.2022.9835796.

[29] Y. Il Kim, J. S. Park, and J. G. Shon, "An Efficient Management of Digital Badge Using JWT," *Lecture Notes in Electrical Engineering*, vol. 1416 LNEE, pp. 160 – 166, 2025, doi: 10.1007/978-981-96-5693-6_25.

[30] B. Xu *et al.*, "JWTKey: Automatic Cryptographic Vulnerability Detection in JWT Applications," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 14346 LNCS, pp. 263 – 282, 2024, doi: 10.1007/978-3-031-51479-1_14.

[31] M. F. Elhejazi and W. H. A. Muragaa, "Improving the Security and Reliability of SDN Controller REST APIs Using JSON Web Token (JWT) with OpenID and auth2.0," in *2024 IEEE 4th International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering, MI-STA 2024 - Proceeding*, 2024, pp. 398 – 402. doi: 10.1109/MI-STA61267.2024.10599643.

[32] N. Kaushik, P. Goel, S. Lal, and A. Kumar, "UniVibe - College Social Network," in *Proceedings - IEEE 2024 1st International Conference on Advances in Computing, Communication and Networking, ICAC2N 2024*, 2024, pp. 628 – 634. doi: 10.1109/ICAC2N63387.2024.10894818.