Design and Implementation of Lightweight, Scalable, and Secure REST APIs for Seamless Integration with Hospital ICT Infrastructure

Raghavendra Ganiga[1,2], Youngwoo Oh[1], and Wooyeol Choi[1,*]

[1]Chosun University, [2]Manipal Academy of Higher Education (MAHE)

raghavendra_ganiga@chosun.ac.kr, ywo@chosun.kr, wyc@chosun.ac.kr

Abstract

Electronic health records (EHRs) systems are generally perceived as difficult to install and maintain. In addition to this, it also carries a cost of educating the hospital personnel on using these systems. The approach we have presented in this paper presents an EHR system as a collection of APIs that are lightweight, scalable, secure, and fast. Since these APIs have been developed in accordance with REST, which is largely compatible with most other APIs available today, it can be integrated with third-party applications as well and can reduce the need to have proprietary systems and applications which will be consuming these APIs. Due to this, we can reduce the costs of educating hospital staff. In addition to this, we have created our APIs to be secure and to prevent unauthorized data access. We have also used dependencies with no vulnerabilities to prevent any security breaches in the future.

I. Introduction

Electronic health record (EHR) systems have revolutionized the way healthcare is managed, but their adoption has been challenging [1][2]. One of the biggest hurdles is the cost associated with training hospital staff on the new system, resulting in delays and decreased efficiency. The exchange of health data between different devices and software programs can be a complex and cumbersome process, often requiring manual intervention. However, open and standardized application programming interfaces (APIs) provide a solution to this problem by allowing seamless data exchange between different applications. In various industries, APIs have already been successfully implemented to consolidate data from multiple sources into a single app or website. In the healthcare industry, APIs have the potential to revolutionize the way patient data is accessed and shared [3].

APIs can unlock electronic health record data, providing patients with efficient access to their health information and giving healthcare providers a comprehensive view of each patient's medical history. This can enable patients to make informed decisions and support drug and therapy research. REST APIs provide a flexible and standardized approach for easy communication between healthcare applications while ensuring security and privacy. This paper describes the design and implementation of secure REST APIs for seamless integration with hospital ICT infrastructure using the AWS cloud platform [4][5]. The system's architecture includes multiple layers of security to protect patient data, and benefits include improved interoperability, reduced development time and costs, and enhanced security[6]. The implementation provides a robust and efficient solution for integrating healthcare systems and exchanging patient information in a secure and scalable manner. Additionally, the paper presents a scalable EHR system using microservices, cloud services, distributed caching, and multiple layers of security to prevent unauthorized data access [7][8].

II. Proposed Method

The healthcare system's APIs are designed using Spring Boot technology, with the API Gateway serving as the client interface that grants requested services to clients. All service requests are routed through the gateway, and user information is only exposed to the endpoint of the client, which reduces the entry points for security attacks. When using APIs in cloud services, requests are sent after the client contacts the gateway, abstracting the level of security and validation in the gateway to ensure secure communication between the client and server [9][10].

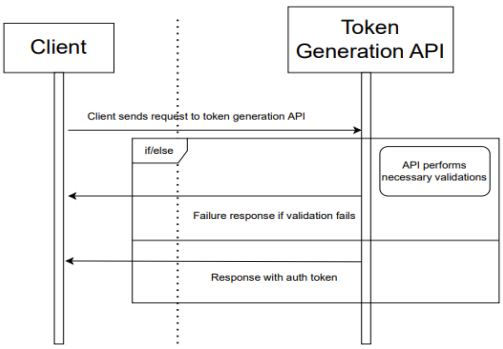

Figure 1. Token generation APIs.

The token generation APIs process flow as shown in Figure 1. To interact with the API Gateway, hospitals must obtain an authorization token by calling the token generation service, valid for 30 minutes, before generating their credentials using the client credentials service. These credentials are unique to the hospital, encrypted, and stored securely in MongoDB, with access prohibited after

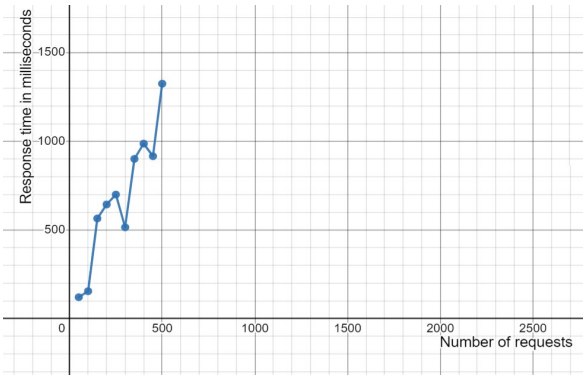

Figure 2. Average response time in milliseconds for the medical record services.

generation. When the token is generated, it is stored in MongoDB along with its timestamp and scope, with UUID format for uniqueness, and signed using a secret key for added security. This process ensures that only authorized clients access the APIs and that their actions are limited by their assigned scope, with secure storage and encryption ensuring the safety of credentials and tokens.

The client sends the authentication token obtained from the token generation service as part of the request header when creating a medical record. The gateway validates the scope of this token, which determines whether the user has permission to create the record, preventing unauthorized access. If the gateway validation succeeds, the request is forwarded to the medical records API. It performs a series of validations to ensure the credentials are from the same hospital as the one mentioned in the request body and prevents a user in one hospital from trying to create a record for another hospital.

The medical records API checks for the presence of any malicious content and, if the data is clean, uploads it to an S3 bucket. The URL of the object is then stored in the database, resulting in faster response times. Asynchronous API processes are used to avoid making the user wait for heavyweight operations such as uploading to S3 and creating a MongoDB entry to complete. The job status API allows hospitals to track the progress of the operation they initiated using the job ID generated during the creation process, checking the status of the job at any time. If the job is still in progress, the API returns a status message such as "In progress" or "Pending," while a successful completion returns a message such as "Created" or "Completed." In the case of an error, the API returns an error message indicating what went wrong.

Once the record is created, it can be retrieved using the retrieval API. The scope provided in the token ensures that Hospital A personnel only access Hospital A data, and only users with appropriate permissions can access the data. Records can be retrieved by patient name or all records available for that hospital. If the record contains multimedia data, an S3 URL pointing to the data is provided to avoid multimedia processing overhead, allowing for faster retrieval.

The graph shown in Figure 2 is the average response time in milliseconds for the medical record service create API, tested using Apache Jmeter with an increasing number of requests. The initial test involved 50 requests, with a delay of 1 second each, resulting in an average response time of around 120 ms. As the number of requests increased, there was a steady increase in response time. It's important to note that the API is asynchronous, meaning that the request is first accepted and sent to the processing queue, which is not the user's concern. The stress testing results were obtained using 500 requests, with the average, minimum, and maximum response times shown. The large difference between the average and max response time can be attributed to network speeds and high load at AWS load balancers.

III. Conclusion

The approach presented in this paper introduces an EHR system consisting of lightweight, scalable, secure, and fast APIs. These APIs can seamlessly integrate with various hospital services and are compatible with REST, enabling integration with third-party applications. The proposed method leverages microservices, cloud services, and distributed caching to enhance system performance. Additionally, a straightforward user authentication mechanism has been implemented to ensure data security. With its simple and universally accepted APIs, this EHR system can be easily integrated by service providers, offering a scalable solution for managing EHRs.

ACKNOWLEDGMENT

This work was supported by the Technology development Program(S3312532) funded by the Ministry of SMEs and Startups(MSS, Korea).

REFERENCES

[1] S. Upadhyay and H.-f. Hu, “A qualitative analysis of the impact of electronic health records (ehr) on healthcare quality and safety: Clinicians’ lived experiences,” Health Services Insights, vol. 15, p. 11786329211070722, 2022.

[2] A. Howarth, J. Quesada, J. Silva, et al,“ The impact of digital health interventions on health-related outcomes in the workplace: a systematic review,” Digital health, vol. 4, p.2055207618770861, 2018.

[3] C. S. Kruse, M. Mileski, A. G. Vijaykumar, et al, “Impact of electronic health records on long-term care facilities: systematic review,” JMIR medical informatics, vol. 5, no. 3, p. e7958, 2017.

[4] S. Patni, Pro RESTful APIs. Springer, 2017.

[5] F. Belqasmi, R. Glitho, and C. Fu, “Restful web services for service provisioning in next-generation networks: a survey,” IEEE Communications Magazine, vol. 49, no. 12, pp. 66–73, 2011.

[6] Hoffman and A. Podgurski, “Securing the hipaa security rule,” Journal of Internet Law, Spring, pp. 06–26, 2007

[7] A. Ven˜ckauskas, D. Kukta, ˇS. Grigali¯unas, and R. Br¯uzgien˙e, “Enhancing microservices security with token-based access control method,”Sensors, vol. 23, no. 6, p. 3363, 2023.

[8] G. F. Brian Warwick, Jimmy DeLurgio et al, “Developing healthcare applications using epic electronic health record apis.” (https://aws.amazon.com/blogs/industries/connecting-aws-chatbot-to-epic-electronic-health-record-apis-using-amazon-lambda.)

[9] P. Sbarski and S. Kroonenburg, Serverless architectures on AWS: with examples using Aws Lambda. Simon and Schuster, 2017.

[10] A. Chatterjee, M. W. Gerdes, P. Khatiwada, and A. Prinz, “Sftsdh: applying spring security framework with tsd-based oauth2 to protect microservice architecture apis,” IEEE Access, vol. 10, pp. 41 914–41 934, 2022.