Jurnal E-Komtek
Vol. 6, No. 1 (2022) pp. 89-100
https://jurnal.politeknik-kebumen.ac.id/index.php/E-KOMTEK
p-ISSN : 2580-3719 e-ISSN : 2622-3066

The Implementation of REST API in Multi-Platform Software Development for Food and Beverage Learning Application

Yuda Syahidin, Randy Ramadhan
Department of Information Systems, Politeknik Piksi Ganesha, Bandung, Indonesia, 40274

yuda.syahidin@piksi.ac.id

https://doi.org/10.37339/e-komtek.v6i1.814

Published by Politeknik Piksi Ganesha Indonesia

Artikel Info
Submitted:
05-02-2022
Revised:
12-02-2022
Accepted:
15-02-2022
Online first :
30-06-2022

Abstract

This website-based food and beverage learning application is an educational application about food and drink. The problem was that information was more often accessed on smartphone devices, such as Android. The research method used was quantitative, so this research provides a concrete value for the optimal implementation of the REST API. The result obtained was that the website-based REST API application worked on multi-platform devices. With the implementation of this REST API, various devices can communicate with each other with the same source and data, so it does not cause multiple databases to store information.

Keywords: Food and beverage, Multi-platform, REST API

Abstrak

Aplikasi pembelajaran makanan dan minuman berbasis website ini merupakan aplikasi edukasi tentang makanan dan minuman. Masalah yang terjadi adalah informasi lebih sering diakses di perangkat smartphone, seperti android. Metode penelitian yang digunakan adalah metode penelitian kuantitatif, sehingga ada nilai konkrit untuk implementasi REST API yang optimal. Hasil yang diperoleh adalah aplikasi REST API berbasis website dapat dikonsumsi oleh perangkat multi perangkat. Dengan implementasi REST API ini, perangkat dapat saling berkomunikasi dengan sumber dan data yang sama, sehingga tidak menimbulkan multiple database untuk menyimpan informasi.

***Kata-kata kunci**: Makanan dan minuman, multi-platform, REST API*

1. Introduction

Information and communication technology is developing rapidly and is widely used to provide information quickly. This opportunity makes developers create multi-platform applications to address user usage, both from websites and mobile devices, such as Android [1]. That way, the information submitted will lead to a single source of information. Undoubtedly, there is a lot of information that can be accessed, one of which is information on learning about Food and Beverage arts.

In fact, information and communication technology in this field, both the science of cooking and guest service, is not yet friendly to the people of Indonesia. If anything, the information submitted is incomplete in terms of knowledge, and is difficult to develop when the website wants to run on mobile devices natively [2].

The development of an ordinary website providing Food and Beverage information still has several shortcomings, including ordinary websites that cannot be visited through native multi-platform applications such as Android. Also, manual access methods with website addresses make users need time to access them when they are in locations where the internet network is not strong, which will take longer to load data.

Based on these problems, the authors conducted research where they created a multi-platform-based Food and Beverage Learning application by applying REST API technology to give an impression and added value to the application, and titled it "The Implementation of the REST API in Multi-Platform Software Development for Food and Beverage Learning Application" [2] [3] [4].

2. Method

The research consisted of five stages, namely: (1) Problem Identification, (2) Literature Review, (3) Building REST API, (4) Blackbox REST API Testing on Postman, Web Application using ReactJS and Android Application using React Native, and (5) Scientific Article Writing [1]. Research gate is presented in Figure 1.

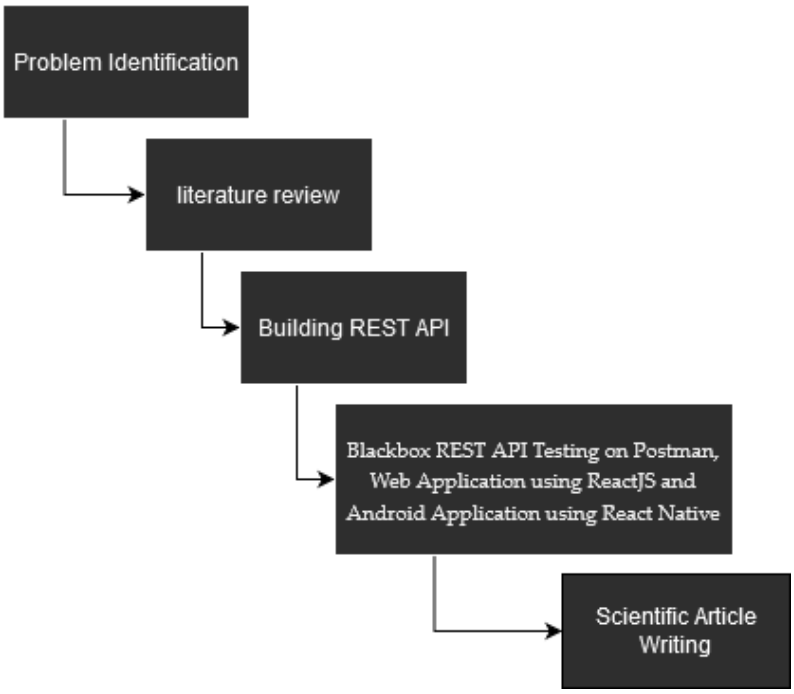

Figure 1. Research Stages

In the first stage, problem identification aimed to find the main problem, namely inflexible data consumption when the application is multi-platform based. The problem was solved by separating the backend and frontend by creating a bridge in the form of a REST API so that it can be consumed by multi-platforms. REST API consumption concept is presented in **Figure 2**.

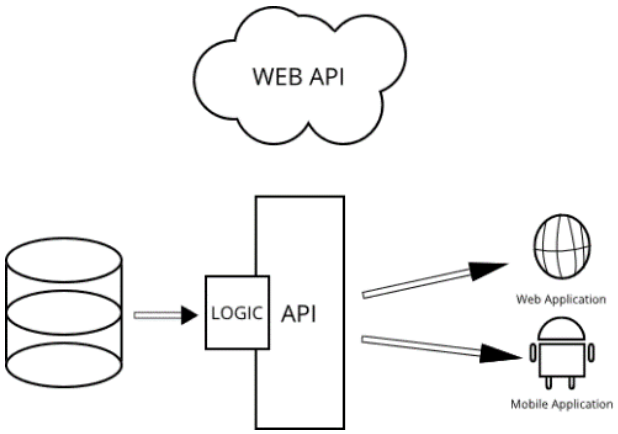

Figure 2. REST API Consumption Concept

The second stage was a literature review, carried out by collecting references from scientific journals, books, or other sources. The library review is useful in designing REST APIs using Gin-Gonic and consuming REST APIs on each platform.

The third stage is to build a REST API. In the REST API program, to make it easier when developing, the author separates each development by taking the MVC (Model, View,

Controller) concept so that it is more focused on one part [3]. MVC concep is presented in **Figure 3**.

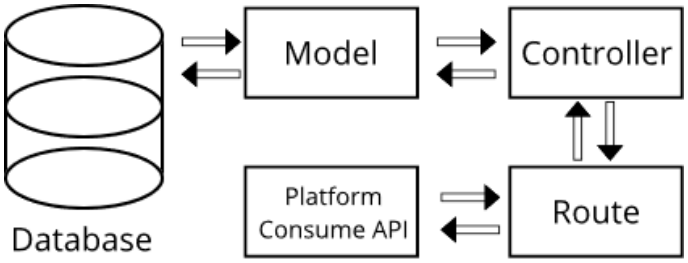

Figure 3. MVC Concept

The fourth stage is Black box testing, testing is carried out on each REST API access endpoint, and testing whether the program on the platform that consumes it can run properly. Black box Testing is testing software in terms of functional specifications without testing the design and program code **[5] [6] [7]**. Black box test is presented in **Figure 4**.

Figure 4. Blackbox Test

The fifth and final stage is the writing of scientific articles, this stage is the methodological and systematic presentation of all research, as in this research article. Another goal is publication so that everyone knows about the research and it can be implemented and helps make additions or re-discussions related to this research.

3. Results and Discussion

In the implementation of the REST API, the development of multi-platform Food and Beverage learning software carried out through a series of Entity-Relationship Diagrams (ERD) **[5]**, Context Diagram Analysis, Data Flow Diagrams (DFD) **[8]**, REST API Design, and Blackbox testing is elaborated as follows **[9] [10]**.

a. The Implementation of System Analysis

The most perfect approach is to knowing the flow of data sent and received (because the REST API uses data as its object), then the data-based analysis approach or Data Flow Diagram is the closest [11].

1) Entity Relationship Diagram (ERD)

The first stage was designing the database. This stage was carried out to find out what data were needed in discussing the implementation of the REST API. ERD design is presented in **Figure 5**.

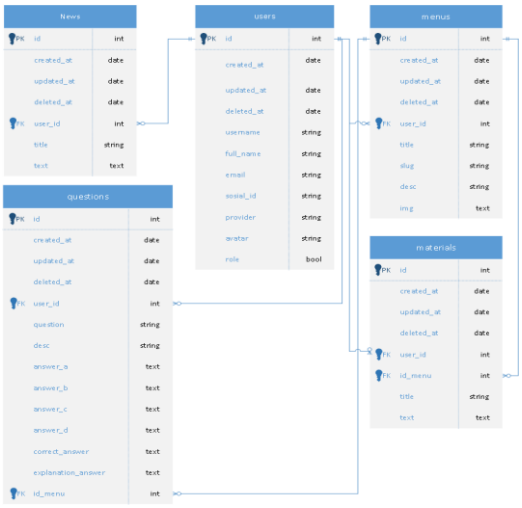

Figure 5. ERD Design

In the REST API development, several relation tables were needed. To be precise, as shown by Figure 5, there were five tables, namely User's table, News table, Menus table, Materials table, and Questions table. User's table contains information for ordinary users and admins. The difference comes from the role of each user. Then, News table serves to accommodate information in the form of news about Food and Beverage arts. Third, Menus table is in charge of storing the title and description of the material, and the Materials table stores the materials according to the title and description in the Menus table. Finally, Questions table has the function to store questions about the materials that has been studied.

2) Context Diagrams

This section analyzes what data flows are happening from the system to the platform. They are shown by **Figure 6**.

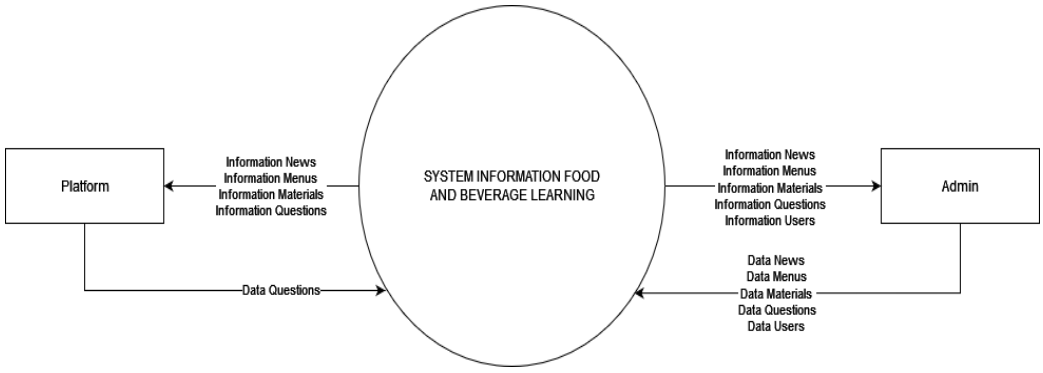

Figure 6. Context Diagram Analysis

There are two entities in the analysis: the Platform that consumes the REST API, and the Admin that functions as the data manager.

3) Data Flow Diagrams (DFD)

Two levels of Data Flow diagrams (DFD) are presented in this section, of which goal is to deeper identify the data flows in each part.

a) Level 1

Level 1 elaboration has 3 processes, namely: (1) Auth Process, (2) Information Access, and (3) Data Processing. The detail is shown by **Figure 7**.

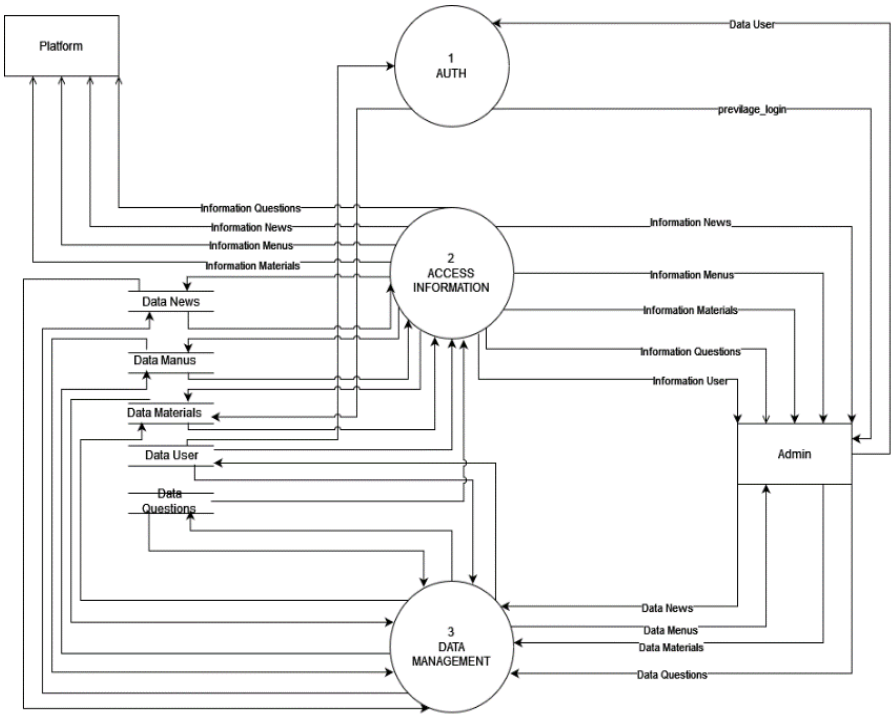

Figure 7. Data Flow Diagram Analysis Level 1

b) Level 2 Auth

At Level 2 Auth, the Admin performs authentication when processing data, as shown by **Figure 8**.

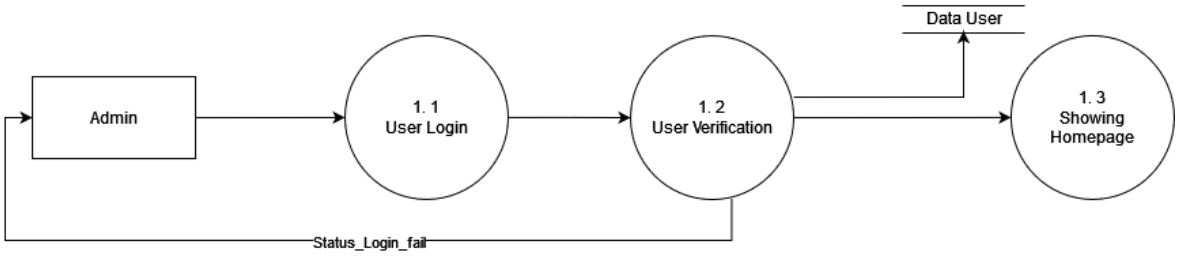

Figure 8. Data Flow Diagram Analysis Level 2 Auth

c) Level 2 Information Access

The Information Access process has the aim of displaying all data that can be accessed by the Platform and the Admin. Data flow diagram analysis level 2 information access is presented in **Figure 9**.

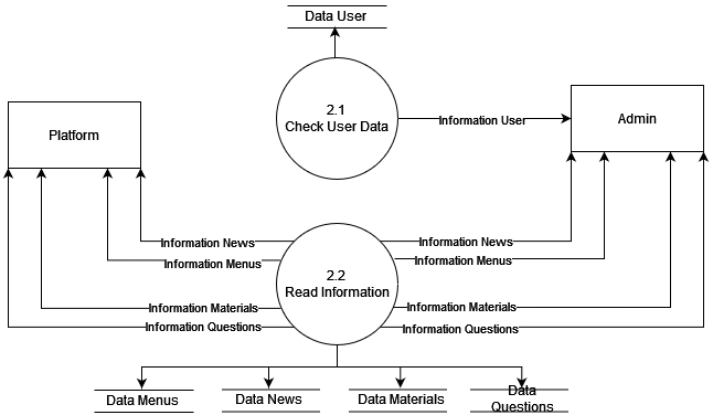

Figure 9. Data Flow Diagram Analysis Level 2 Information Access

d) Level 2 Data Processing

The last process in the Data Flow Diagram (DFD) analysis is to input, delete, and update which is fully carried out by the admin. Data flow diagram analysis level 2 data processing is presented in **Figure 10**.

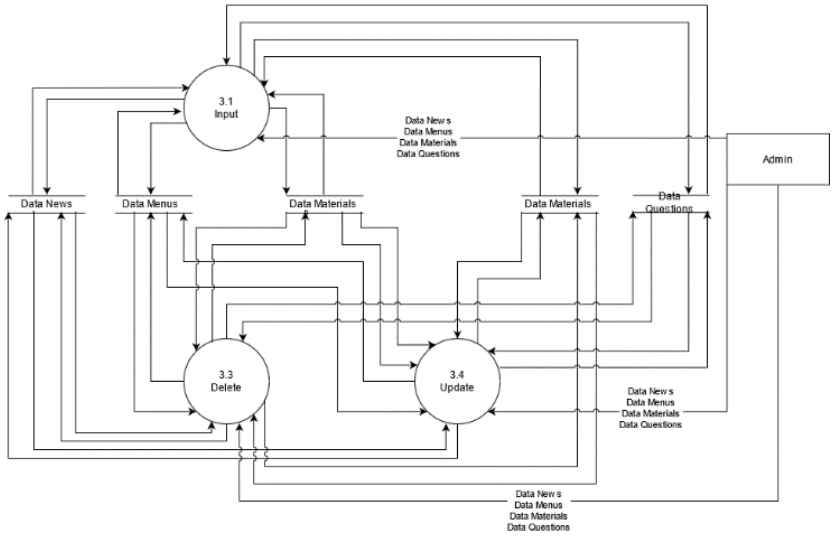

Figure 10. Data Flow Diagram Analysis Level 2 Data Processing

e) REST API Design

Based on the Data Flow Diagram, the REST API design performed the following stages: (1) defining the struct, (2) making connections and migrating the database, (3) creating a route to process data, and (4) determining the access endpoint that would be used by the platform.

(1) Defining the Struct

The structure comes from the table and its relations as a blueprint for migrating the database and schema for the CRUD process for each data.

(2) Creating Database Connections and Migration

In this section, we need two libraries: Gorm library with the Open method to open a connection from the Go language to the database and MySQL with the Open method. When connects to a MySQL database, the migration to the struct is automatically performed.

(3) Creating Routes

To make it easier to develop, routes are separated according to the struct that has been created. The route itself contains methods for manipulating the database, starting from getting data or sending data. Flowchart response data algorithm is presented in **Figure 11**.

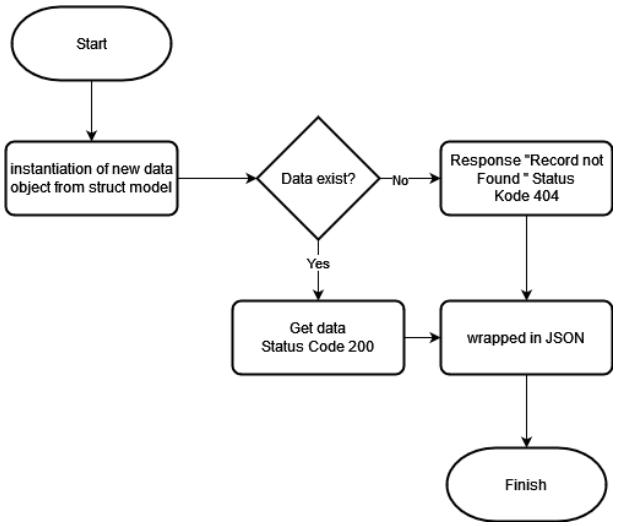

Figure 11. Flowchart Response Data Algorithm

At this stage, the output of the REST API is created, starting with creating a new object from the struct model. Automatically, the initialization of a new object will request data from the database, and return data in the form of an object array.

The request at the time of initialization exists or not also affects the result. If not, it will return a "record not found" response with HTTP 404 code status. If the data exist in the database, it will return response data and HTTP 200 status code. Response data is wrapped in JSON form so that it is easy to consume by the platform.

In the method that has access to add, delete, and process data, a middleware was added, to perform authentication: (1) Is the admin logged in? and (2) Does the admin have the right to manipulate the data? This middleware is Boolean; if the conditions are met, the next method will be executed, and while they are not, it will provide HTTP 403 code status information.

Flowchart of Middleware Access Algorithm is presented in **Figure 12**.

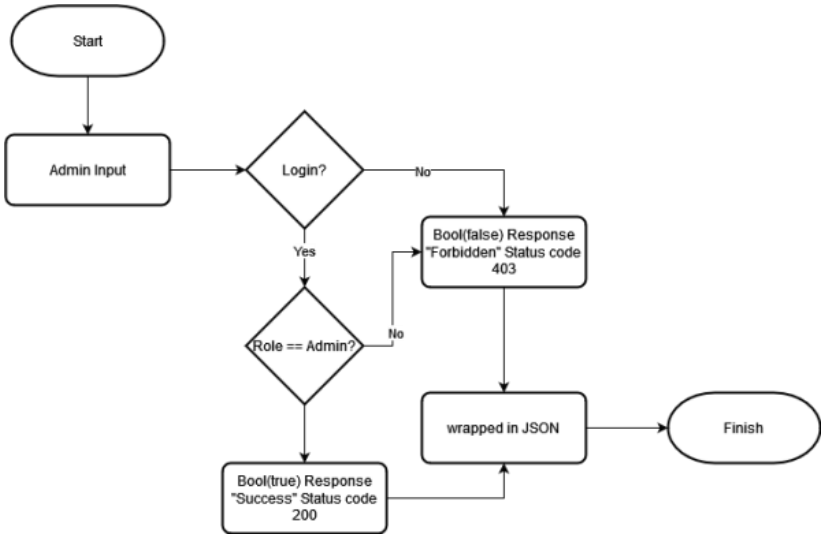

Figure 12. Flowchart of Middleware Access Algorithm

(4) Defining Endpoint Access

Access endpoints were wrapped in a router group ("/API/v1/"). "/API" indicates that it will access the REST API. Meanwhile, "/v1" refers to the version of the REST API that had been designed. Therefore, when the next REST API is developed, it does not break the structure of the previous REST API but creates a new router with the name "v2". Lizt router endpoint with method is presented in **Table 1**.

Table 1. List Router Endpoint with Method

Endpoint Routers	HTTP's Method	Method Call
/	GET	getWelcome()
/visitor	GET	getWelcomeVisitor()
/visitor/news	GET	getNews()
/visitor/menus	GET	getMenus()
/visitor/materials	GET	getMaterials()
/visitor/learn/:slug	GET	getMaterial()
/visitor/quiz/:slug	GET	getlearnQuiz()
/admin	GET	getWelcomeAdmin()
/admin/news/	POST	postNew ()
/admin/news/update/:id	PUT	updateNews()
/admin/news/delete/:id	DELETE	deleteNews()
/admin/menus/	POST	postMenu()
/admin/menus/update/:id	PUT	updateMenu()
/admin/menus/delete/:id	DELETE	deleteMenu()
/admin/materials/	POST	postMaterial()
/admin/materials/update/:id	PUT	updateMaterial()
/admin/materials/delete/:id	DELETE	deleteMaterial()
/admin/question/	POST	postQuestion()
/admin/question /update/:id	PUT	updateQuestion()
/admin/question/delete/:id	DELETE	deleteQuestion()

b. Presenting the Result

A REST API testing was carried out to find errors in accessing a REST API and resolve responses that did not exist (404 Not Found) or (403 Forbidden). The test used was Blackbox Testing, so it = focused on the input in the form of request endpoints and responses from the REST API. Various tests were performed to give the expected results.

These tests were grouped according to the platform used. Among them are (1) Web Application Platform with React JS and (2) Mobile Android Application Platform with React Native. These two tests focused on testing the /visitor/news and /visitor/learn/:slug endpoint router samples and testing the admin middleware for the POST news method.

1) Web Application Platform with React JS

As discussed earlier, the first test was based on a web application using React JS. This test was carried out on the GET and POST methods, assuming they can overcome the most frequently used handles. It seemed to take longer time because the REST API carries a lot of data objects and loads the web application using React JS. Below are the results of the tests. Black box web application react is test results is presented **Table 2**.

Table 2. Black Box Web Application React JS Test Results

No.	*Router Endpoint*	*Test Case*	Time	Expected Output	Testing Result
1.	/visitor/news	Consume endpoint	7.5 s	*GET data*	Success *Get data*
2.	/visitor/ learn/ : slug	Consume *endpoint* with *slug* exist in table *materials*	*0.337 s*	*GET data*	Success *Get data*
3.	/visitor/ learn/ : slug	Consume endpoint with a slug isn't exist in the database	*2.56 s*	*GET data*	Success Handle 404 *Not Found*
4.	/admin/news/	Store Data without Authentication	*2.15 s*	*POST data*	Success Handle 403 *Forbidden*
5.	/admin/news/	Store Data with Authentication	*3.50 s*	*POST data*	Success *POST data*

2) Platform Mobile Android Application with React Native.

The second test was carried out on a mobile device application using React Native. The same as before, it was based on getting and POST access and handles that are carried out so that they did not cause errors. However, compared to the process on website devices, it seemed that the speed of accessing data objects in this test was faster.

It could be coming from the hardware used. Black box mobile application react native result is presented in **Table 3**.

Table 3. Black Box Mobile Application React Native Test Result

No.	*Router Endpoint*	*Test Case*	Time	Expected Output	Testing Result
1.	/visitor/news	Consume *endpoint*	5.3 s	*GET data*	Success *Get data*
2.	/visitor/ learn/ : slug	Consume *endpoint* with *slug* existing table *materials*	*0.2 s*	*GET data*	Success *Get data*
3.	/visitor/ learn/ : slug	Consume endpoint with slug without existing in the database	*2 s*	*GET data*	Success Handle 404 *Not Found*
4.	/admin/news/	Consume Data without Authentication	*1.5 s*	*POST data*	Success Handle 403 *Forbidden*
5.	/admin/news/	Consume Data with Authentication	*3.55 s*	*POST data*	Success Handle *POST data*

4. Conclusion

In implementing the REST API in developing a multi-platform software integrated with one data source, REST API technology is essential to manage request and response data from the platform with Web Services. The REST API is accessed through the endpoint router using the HTTP protocol. However, it is vital to know that endpoint routers can be easily stolen, and so can their data. Therefore, Authentication is required to prevent sensitive requests. The use of Authentication is assisted by middleware to confirm the admin who can perform data management on the REST API. The multi-platform Food and Beverage learning application can run well on devices such as websites and android mobiles. However, the internet network affects the speed of data retrieval on the platform.

Reaferences

[1] Yuda Syahidin and Randy Ramadhan, "REST API Architecture Design on Multi-Platform Device Development", E-Komtek, vol. 5, no. 2, pp. 178-189, Dec. 2021.

[2] T. Kristanto, R. K. Hapsari, V. S. Nita, and S. Maimunah, "Rancang Bangun Aplikasi E-Learning Berbasis Multiplatform untuk Mata Pelajaran Bahasa Indonesia dengan Menggunakan Pendekatan Technology Acceptance Model (TAM)," *J. Tek. Inform. dan Sist. Inf.*, vol. 1, no. 3, 2015, doi: 10.28932/jutisi.v1i3.408.

[3] J. Herlian, "Perancangan Sistem Mobile POS (Point of Sale) Dengan Menggunakan Restful Web Services," 2015.

[4] I. Binantro, "Tinjauan Metode Pengembangan Perangkat Lunak Multimedia Yang Sesuai Untuk Mahasiswa Tugas Akhir," 2015.

[5] M. Rosa, AS & Shalahuddin, "Rekayasa Perangkat Lunak Terstruktur Dan Berorientasi Objek. Penerbit Informatika, Bandung.," p. 100, 2014.

[6] Q. Mahmoud, "Middleware for communications," 2004.

[7] S. . Peyrott, *The JWT Handbook*, Version 0. 2017.

[8] A. Kristanto, "Perancangan Sistem Informasi Dan Aplikasinya," 2008.

[9] M. A. K. Perdana, "Pengembangan Rest Api Layanan Penyimpanan Menggunakan Metode Rapid Application Development (Studi Kasus."

[10] M. Salahudin and A. S. Rosa, "Rekayasa Perangkat Lunak," *Bandung: Pustaka Setia*, 2014.

[11] Y. K. Kurniawan and H. K. Yetli Oslan, "Implementasi Rest - Api Untuk Portal Akademik UKDW Berbasis Android," 2013.