Jurnal Teknik Informatika (JUTIF)
Vol. 5, No. 4, October 2024, pp. 1355-1364

DOI: https://doi.org/10.52436/1.jutif.2024.5.5.2493
p-ISSN: 2723-3863
e-ISSN: 2723-3871

IMPLEMENTATION OF REST API ARCHITECTURE FOR FEELSQUEST ONLINE COURSE FEATURE IN FEELSBOX APPLICATION USING LARAVEL FRAMEWORK

Faza Alexander Riawan*1, Dana Sulistyo Kusumo2, Nungki Selviandro3

1,2,3Software Engineering, Informatics Faculty, Universitas Telkom, Indonesia
Email: 1fazaalexander@student.telkomuniversity.ac.id , 2danakusumo@telkomuniversity.ac.id, 3nselviandro@telkomuniversity.ac.id

(Article received: July 10, 2024; Revision: August 5, 2024; published: October 25, 2024)

Abstract

Feelsbox is a digital-based startup that focuses on the importance of mental health issues and offers innovative solutions to help people maintain their mental health. FeelsBox took the initiative to develop an online course feature "FeelsQuest" with the aim of providing education and helping prevent and overcome mental health problems to the wider community, especially teenagers. The development of this feature uses the PHP programming language with the Laravel framework and implements the REST API architecture. The choice of REST API architecture is based on the concept of separation of responsibilities so that the API can be reused on different platforms. In addition, a suitable test is needed to test the REST API that has been built. Testing of the REST API that has been built is done with the API testing method which is focused on aspects of functionality and performance using Postman to ensure that the API built produces responses and behaves according to the needs of the FeelsQuest feature of the FeelsBox application. The test results show that the implementation of the REST API on the FeelsQuest feature is in accordance with the functional requirements and successfully applies the concept of separation of concerns and meets the non-functional needs of the FeelsQuest feature related to the response time of each API, which is under 3 seconds.

Keywords: backend, functional testing method, laravel, load testing method, online course, REST API

1. INTRODUCTION

The development of information and communication technology causes humans to continue to innovate to meet their needs. Information technology is a human tool for processing activities that produce information. In addition, information technology also plays a role in solving a problem, encouraging creativity, increasing effectiveness and efficiency in human activities. [1]. One of the innovations from the development of information technology is online courses. E-learning, or can be called an online course, is an online-based learning or training that is not limited by place and time, and utilizes information technology to gain access to learning and teaching.[2].

FeelsBox is a digital-based startup that focuses on the importance of mental health issues and offers innovative solutions to help people maintain their mental health. FeelsBox was established in 2021 under the name "Zyon" before finally rebranding in 2022 to "FeelsBox" which means feeling box. Currently, FeelsBox is developing a new feature called "FeelsQuest" as an effort to educate the wider community, especially teenagers on the importance of mental health. FeelsQuest is a web-based online course feature, but in its development, an architecture is needed that can be used on different platforms. Therefore, an architecture with an approach that applies the principle of separating client and server responsibilities or concerns is needed so that the business logic developed can be reused on various platforms, namely web and mobile platforms.

To solve these problems, an architecture is needed that is able to handle the integration and access needs of various platforms (multiplatform). An architecture that is interdependent between one system component and another will certainly not be effective to be applied in this case. This is because if there is an error or change that leads to a mismatch, the developer must make changes to all system components [3].

Web Service is a set of standards and programming methods for sharing data between different software applications. Web Service provides interoperability between various applications running on different platforms [4]. Web Service acts as a service function provider unit in a system into a standardized software component, can be used and updated dynamically and independently. The services provided by a web service are in the form of information through the transfer of Javascript Object Notation (JSON) or Extensible Markup Language (XML) type data between systems in a network through the Hypertext Transfer - Transfer Protocol (HTTP) protocol, thus enabling interaction between systems with the support of interoperability offered [5].

Representational State Transfer (REST) is a web service architecture that allows various systems to communicate, send or receive data in a very simple way. REST is one type of web service architecture that applies the concept of transferring between states, where the way the REST architecture works is by navigating through links or HTTP endpoints [1]. Application Programming Interface (API) is an interface that can be used to integrate data and connect a system that runs on different platforms. In addition, APIs can also speed up the development process by providing functions separately so that developers do not need to create similar features. In the context of implementing REST API, there are many programming languages and frameworks that can be used to implement REST API architecture [6].

One of the popular and widely used programming languages and frameworks is the Hypertext Preprocessor (PHP) programming language and the Laravel framework. Laravel framework is one of the PHP-based frameworks that is open source and uses the concept of separation of concerns model - view - controller (MVC) [7]. MVC is a method of creating applications by separating data (model), view, and data processing bridge (controller) [8]. Therefore, the implementation of the REST API in this research uses the PHP programming language and the Laravel framework.

Functional Testing is part of the Black box Testing method, which in the process aims to find out whether each API functions and responds as expected without knowing how the program code is implemented [9]. Pengujian fungsionalitas pada API sistem FeelsQuest Functionality testing on the FeelsQuest system API is carried out to analyze and ensure that each API that is built responds and functions as expected.

Load Testing is a testing method used to evaluate the performance and durability of a system under high workload conditions or at virtual maximum loads [10]. Load testing on the FeelsQuest system API is carried out to analyze and ensure that each API built has good performance and durability in handling high workload conditions.

As in the research conducted by Iman Nurjaman, Fandy Setyo Utomo, and Nandang Hermanto with a study entitled "Penerapan REST API Laravel sebagai Fondasi *Back-end* Aplikasi G-MOOC 4D", discussing the development of the G-MOOC 4D website-based online course application by applying the REST API architecture and the Laravel framework. From this previous research, the author found that the application of the REST API to the online course system produces a number of access points (API endpoints) that can be used by the front-end team to be able to interact and access data from the back-end system efficiently and safely. In addition, the tests carried out in the study tested the functionality of each API by accessing the access point of each API with a response that successfully met the expected results. This previous research has similarities with this research where this research will help the system for the FeelsQuest feature by implementing the REST API architecture and the Laravel framework, and conducting functionality testing on the API that has been built [11].

Research from Romi Choirudin and Ahmat Adil with the research title "Implementasi REST API Web Service Dalam Membangun Aplikasi *Multiplatform* Untuk Usaha Jasa" discusses the challenges of building a carpentry service application with a multiplatform system with the implementation of the REST API as a solution. In this previous research, the author found that the REST API architecture that had been implemented successfully achieved the goal of being used in multiplatform applications. This previous research has similarities with this research, where this research has the same problem, namely the need for an architecture that can be used on various platforms (multiplatform) [12].

Just like the research from Mohammad Akmaluddin Novianto, and Sirojul Munir with the research title "Analisis dan Implementasi RESTful API Guna Pengembangan Sistem Informasi Akademik Pada Perguruan Tinggi" discusses how to answer challenges in developing systems and applications that were previously available. One of these challenges includes how to build a system that can be reused in the future with the ability to run on a multiplatform and programming language. The solution applied in this previous research was to apply the concept of REST architecture as a bridge for data exchange without regard to differences in platforms or programming languages. This previous research also applied a functionality-based testing method that was carried out using the Postman application. In this previous research, the author found that the system had been successfully built by applying the concept of REST architecture and API functionality testing showed that all tested scenarios ran well and successfully produced responses as expected. This previous research has similarities with this research, where this research has the same problem, namely the need for an architecture that can be used as a solution to multiplatform problems and implements testing that focuses on the API functionality aspect to test the suitability of the API that has been developed [13].

Research from Denis Akbar, Freza Riana, and Fitrah Satrya with research entitled “Pembuatan *Web Service* Pada Aplikasi SIJAB Dengan Metode REST” also focuses on cross-platform problems. This previous research focused on creating a web service architecture, REST API, as a solution to the problem. In addition, the previous research also conducted a feasibility test of the system that had been built by testing the functionality of the API that had been built. The previous research has similarities with this research in the aspects of problems, solutions, and testing carried out [14].

In addition, research from Ahmad Pulodi, Yulianawati, and Iqbal Suwandi with the title "Implementasi *Web Service* Restful Dengan Autentikasi JSON *Web Token* Berbasis Web dan Android", the author found that the main problem in the study is that researchers need a solution to support interoperability and can facilitate access from different devices or cross platforms for online course systems. The solution applied in this research is to implement a REST web service architecture to answer these problems. In addition, functionality testing is also carried out to ensure that the API built is in accordance with the needs of the system. This previous research has similarities with this research in terms of problems, solutions, testing, and types of systems developed, namely online course systems [15].

The formulation of the problem in this study is how the application of REST API architecture, from design to implementation, and how the results of REST API testing on the FeelsQuest feature backend system of the FeelsBox application using API functional testing and API load testing methods. The purpose of this research is to produce the process of designing and implementing the REST API, as well as testing the REST API that has been built to determine its success and compatibility with the needs of the FeelsQuest feature of the FeelsBox application. The results of this research are expected to provide insights and recommendations related to the application of the REST API architecture so that it can be applied to similar case studies.

2. RESEARCH METHODS

The research method used for this research is based and described on the flowchart in Figure 1. As can be seen in figure 1, the stages of this research are generally divided into four stages, which is API design stage, API implementation stage, API testing stage, and finally report writing.

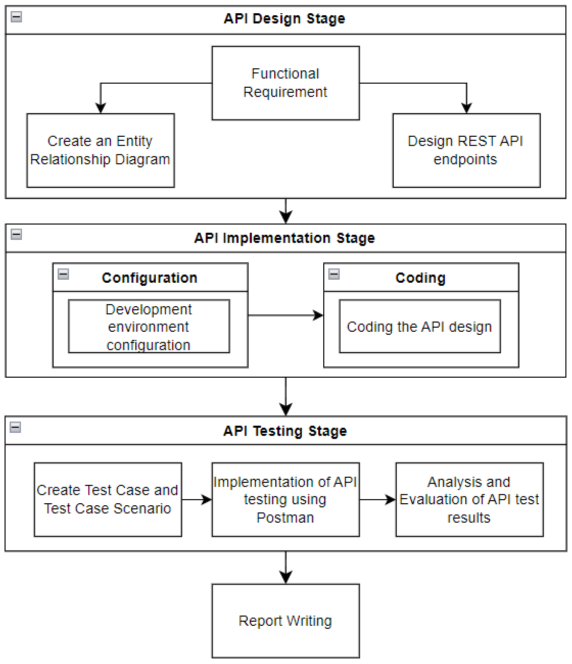
API Design Stage
Functional Requirement
Create an Entity Relationship Diagram
Design REST API endpoints
API Implementation Stage
Configuration
Development environment configuration
Coding
Coding the API design
API Testing Stage
Create Test Case and Test Case Scenario
Implementation of API testing using Postman
Analysis and Evaluation of API test results
Report Writing

Figure 1: Flow of Research Methods

The first stage in this research is the API design process with reference to the functional requirements of the FeelsQuest system. API design is done by designing the endpoint of each API to determine the access point contract for each API along with the request and response data. In addition, Entity Relationship Diagram (ERD) design is done to create a system model based on needs. ERD is made in the form of a conceptual database that describes the concept or database schema that is more specifically related to the need for data to state its relationship and boundaries [16].

The second stage in this research is the API implementation stage. The implementation stage is divided into two processes, including the development environment configuration process and the API design coding process. The development environment configuration process is carried out to ensure that the tools used to develop the API (development environment) are ready for use. This configuration process broadly includes but is not limited to the installation and configuration of programming languages and frameworks used, database configuration, and dependency configuration. After the configuration process is complete, the next process is the API design coding process.

The third stage in this research is the API testing stage. Testing in this study was carried out using functional testing and load testing methods. Functional testing is done by testing each API endpoint and then analyzing whether the response generated by each API is as expected. In addition, API performance testing is carried out using the load testing method through three different scenarios, each for 10 minutes.

Table 1 describes test scenarios with 20, 40, and 60 Virtual Users (VU) using peak load testing. This test aims to evaluate API performance under conditions that resemble actual use, with the expectation that the average response time does not exceed 3 seconds as a performance benchmark. Through this test, an overview of the performance of each API in responding to simultaneous and continuous access loads can be obtained, providing insight into the system's ability to handle real usage.

Table 1. Load Test Scenario

Scenario	Virtual Users (VU)	Load Type	Test Duration	Expected Results
Scenario 1	20	Peak	10 min	<= 3 sec
Scenario 2	40	Peak	10 min	<= 3 sec
Scenario 3	60	Peak	10 min	<= 3 sec

The final stage after all stages from design to API testing have been completed is to compile a report related to the results that have been obtained.

Broadly speaking, the implementation of the REST API architecture on the FeelsQuest feature applies a systematic approach consisting of three main stages, including the API design phase, API implementation, and API testing. The end result of this stage is to produce an Application Programming Interface (API) that has been tested and in accordance with the needs of the FeelsQuest feature system. The API design phase is carried out based on functional requirements including the creation of API endpoint designs to the Entity Relationship Diagram (ERD) design.

Table 2 presents a sample API endpoint design for the FeelsQuest feature system. This design includes several important elements, including the functional requirement ID, the HTTP method used, the endpoint and its request and response, as well as a brief description of the function or purpose of each endpoint.

3. RESULT AND DISCUSSION

Table 2 Sample API endpoint design

No	FR ID	Method	Endpoint	Request	Response	Description
1	FR01	GET	/api/courses	page	Status code, message, data, page, limit, total page	Get all course data
2	FR07	POST	/api/user/ postFeedback/ {courseId}	Course id, questions	Status code, message	Create new feedback data or feedback from a user for a course
3	FR11	PUT	/api/admin/ updateCourse/ {slug}	Course id	Status code, message	Change or edit existing course data
4	FR18	DELETE	/api/admin/ deleteFeedback Question/ {questionId}	Question id	Status code, message	Delete feedback question data or feedback for a course

Entity Relationship Diagram (ERD) is a high-level conceptual model of a database to describe a system and its boundaries. Figure 2 presents the ERD of the FeelsQuest feature where there are nine entities that are related to each other. FeelsQuest ERD has several main entities (database tables), including users, courses, course details, course contents, course ratings, course feedbacks, course feedback questions, enrolled courses, and transactions.

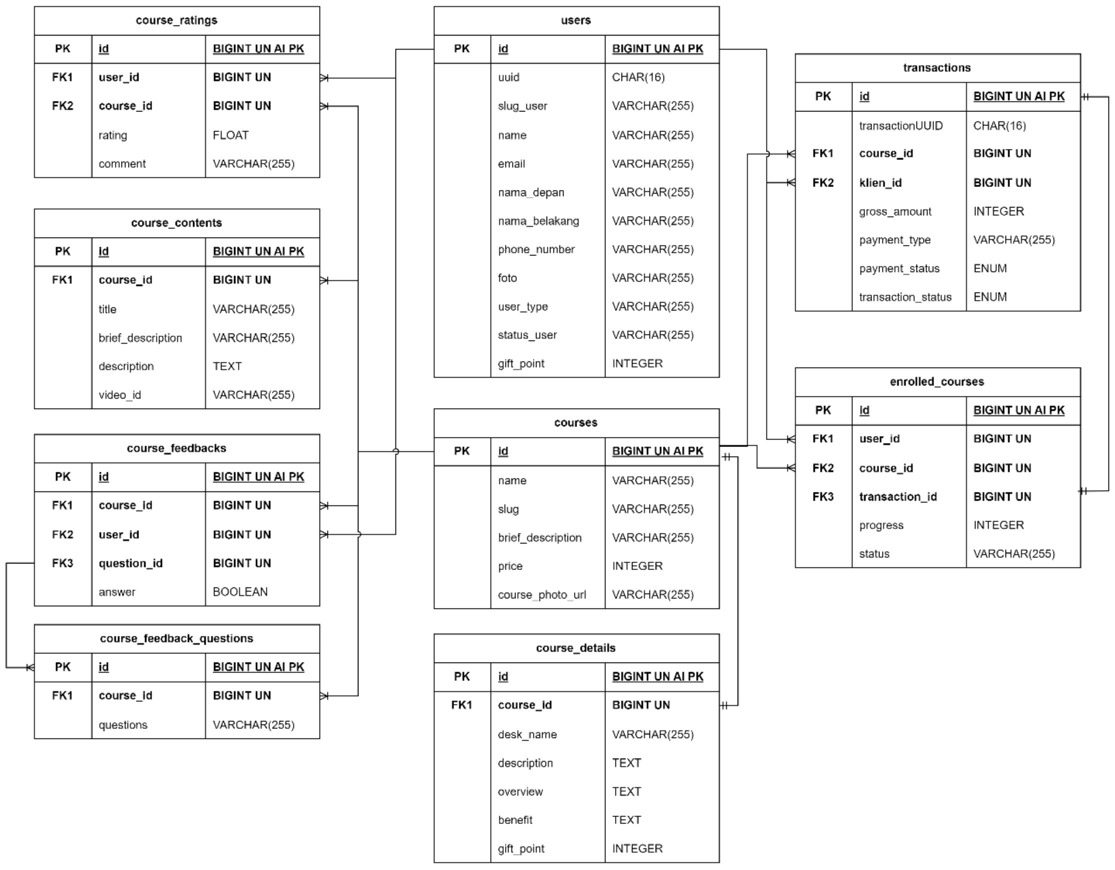

Figure 2: Entity Relationship Diagram

The next stage after the API design stage is the API implementation stage. The API implementation stage begins with configuring the development environment. Configuration begins by ensuring that the PHP programming language and Laravel framework are installed on the environment or device that will be used for API development. Supporting tools such as Composer are also needed to ensure the smooth configuration process of PHP and the Laravel framework. Composer is a dependency manager for the PHP programming language and is used to manage PHP-based software dependencies and libraries that are needed for the API development [17].

Before doing the coding process, several things need to be prepared in addition to configuring the programming language and framework. This includes the installation and configuration of the Database Management System (DBMS) as a database to store data from the system being developed.

The next API implementation step after the configuration process is complete is to implement the API design in the form of code. This process involves several main components in the Laravel framework, such as models to represent entities or data tables that have been designed in ERD, migration to define the structure and relationships between data tables in the database, seeders to fill in dummy data or initial data for testing purposes, controllers to handle business logic and its interaction with the database, and routes to define access points or endpoints

.

Program Code 1. Model

```
<?php

namespace App\Models\ (…);

use (…);

class Course extends Model
{
        use HasFactory;
        protected $hidden = [‘created_at];
        protected $fillable = [‘name’];
        public function detail()
        {
           return
           $this->hasOne(CourseDetail::class);
        }
}
```

Program Code 2. Migration

```
<?php

use (…);

```

```
return new class extends Migration
{
    public function up(): void
    {
    Schema::create('course_details', function (Blueprint $table) {
            $table->id();
            $table->foreignId("course_id")->constrained("courses");
            $table->text("description");
          });
    }

    public function down(): void
    {
        Schema::dropIfExists('courses');
     }
};
```

Program codes 1 and 2 are examples of the application of model and migration program codes for course entities in the FeelsQuest system. These components serve to define the structure and relationships for each entity in the database. The model (program code 1) defines the course entity along with attributes or data that can be filled or hidden. In addition, defining relationships between entities related to the course entity is also set in the model component, for example a one-to-one relationship with the course details entity. Meanwhile, migration (program code 2) serves to define the structure of each entity or data table in the database. The structure in the database consists of columns and their data types as well as relationships between entities using foreign keys.

Program Code 3. Seeder

```
<?php

namespace Database\Seeders;

use (…);

class CourseSeeder extends Seeder
{
    public function run(): void
    {
    $courses = [
       [
          "id" => 1,
           "name" => "Inner Child",
           "…" => "…"
          ];

          foreach ($courses as $course) {
             Course::updateOrCreate(['id' => $course['id']], $course);
          }
       }
}
```

Program code 3 is an example of implementing a seeder program code to fill in dummy data or initial data for course entities for testing purposes in the FeelsQuest feature system database. The seeder implementation is useful to facilitate the API testing process that is in the process of implementation or development. In the program code, the run() and updateOrCreate() methods are used to insert or update each course dummy data into the database using the foreach loop concept. This approach can ensure that each dummy data is inserted into the database quickly and efficiently without duplication.

Program Code 4. Controller

```
<?php

namespace App\Http\Controllers\(…);

use (…);

class CourseController extends \App\Http\Controllers\Controller
{
    public function getAllCourses(Request $request)
    {
        (…)
    }
```

Program code 4 is an example of implementing a controller to handle business logic and its interaction with the database for the course entity in the FeelsQuest system. This controller implementation serves as a bridge or intermediary between HTTP requests from users and models, responsible for receiving, retrieving or manipulating, and returning the appropriate data. In the context of the REST API implementation, each function on the controller will return a return in JSON format.

This controller allows effective management of a particular operation by implementing Create, Read, Update, and Delete (CRUD) functions, for example create to add new course data, read to retrieve available course list data, update to add new course data, and delete to delete existing course data.

Program Code 5. Route

```
<?php

use (…);

Route::middleware('auth:sanctum')->group(function() {
   Route::group(['prefix' => 'user'], function (){
       Route::post('/postFeedback/{courseId}', [CourseController::class, 'postFeedback'])->middleware('role:user');
   }
}
```

Program code 5 is an example of implementing program code to define endpoint routes for each FeelsQuest feature API. This endpoint route serves to direct HTTP requests by users to the appropriate controller function. Each endpoint can be grouped into a group and can be assigned middleware to secure and manage endpoint access. In program code 5, API routes that require authentication are grouped under the 'auth:sanctum' middleware. In addition, API routes that require authentication are further grouped based on the type of user accessing, in this case with a 'user' prefix or an 'admin' prefix. This allows for a more structured endpoint organization and separates the management of access rights based on user type.

The final stage after API design and implementation is complete is to test the API. One of the aspects tested in API testing on the FeelsQuest feature focuses on the functionality aspect which is carried out to test the suitability of the API that has been designed to be implemented against system requirements, especially functional requirements. API functionality testing will be carried out by testing each API endpoint and its suitability for functional needs as evidenced by the suitability of the response of each api with the expected results in the test case of each API. The results of functional testing of the FeelsQuest feature system are as follows.

Table 3 presents the results of the FeelsQuest feature API functionality testing. APIs that successfully produce the appropriate output will have a ‘passed’ status, which means that they pass the API functionality test and are in accordance with functional requirements, while APIs with ‘failed’ status are APIs that do not meet expectations and fail the API functionality test. The conclusion column in the table provides a final assessment of the functionality of each API based on the test results. A conclusion of ‘valid’ indicates that the API not only passed the functionality test, but also fully meet the specified requirements and is ready for further testing, such as API load testing.

Table 3 Functionality test results of the FeelsQuest feature API

No	FR ID	API Functionality	TC ID	HTTP Method	Expected Results	Test Status	Conclusion
1	FR01	Get All Course	TC01-U	**GET**	Successfully returned all course data	**passed**	**valid**
2	FR02	Get Course Detail	TC02-U	**GET**	Successfully retrieved all course data	**passed**	**valid**
3	FR02A FR02B	Get All Reviews (User)	TC03-U	**GET**	Successfully retrieved all user review data for a course	**passed**	**valid**
4	FR03	Create Invoice	TC04-U	**POST**	Successfully created transaction invoices for course purchases	**passed**	**valid**
5	FR05A FR05B	Get Course Content	TC05-U	**GET**	Successfully retrieved all course content data	**passed**	**valid**
6	FR06A FR06B	Create Review	TC06-U	**POST**	Successfully added new review data to a course	**passed**	**valid**
7	FR06 FR07	Post Feedback	TC07-U	**POST**	Successfully add new feedback data to a course	**passed**	**valid**
8	FR09	Create Course	TC08-U	**POST**	Successfully add new course data	**passed**	**valid**
9	FR10A	Get All Course Registrant	TC09-U	**GET**	Successfully get data on all registrants of a course	**passed**	**valid**
10	FR10B FR10C	Get All Reviews (Admin)	TC01-A	**GET**	Successfully get all review data from users on a course	**passed**	**valid**
11	FR11	Update Course	TC02-A	**PUT**	Successfully change data on a course	**passed**	**valid**
12	FR12	Create Content	TC03-A	**POST**	Successfully add new content data to a course	**passed**	**valid**
13	FR13	Update Content	TC04-A	**PUT**	Successfully change content data on a course	**passed**	**valid**
14	FR14	Delete Content	TC05-A	**PUT**	Successfully delete content data on a course	**passed**	**valid**
15	FR15	Get All Course Contents	TC06-A	**GET**	Successfully get all content data for a course	**passed**	**valid**
16	FR16	Create Feedback Questions	TC07-A	**POST**	Successfully add new feedback question data to a course	**passed**	**valid**
17	FR17	Get All Feedback Questions	TC08-A	**GET**	Successfully get all feedback question data in a course	**passed**	**valid**
18	FR18	Delete Feedback Question	TC09-A	**PUT**	Successfully delete a feedback question	**passed**	**valid**
19	FR19	Get All Course Feedbacks	TC10-A	**GET**	Successfully get all feedback data from users in a course	**passed**	**valid**
20	FR20	Get Course Feedback Detail	TC11-A	**GET**	Successfully get detailed feedback data from users on a course	**passed**	**valid**

From the results of API functionality testing that has been carried out, each test case on the simulated API has provided results in accordance with the expected results. Therefore, it can be concluded that the FeelsQuest feature API implemented with the REST API architecture, has been designed and implemented properly, so that each API can work as expected and in accordance with the needs, especially the functional needs of the FeelsQuest feature system.

In addition, testing on the performance aspect of the FeelsQuest feature is done by testing the performance of each API in handling high workloads. This test was carried out using the Load Testing method using Postman. Load testing is performed on each FeelsQuest API endpoint with three different scenarios, each for 10 minutes. Load testing is based on the non-functional needs of the FeelsQuest system where the response time of each API must be under 3 seconds. In addition, according to research results from WebsiteBuilderExpert, the optimal response time for non-ecommerce websites is 3 seconds, and according to Cami Bird, the ideal response time for a website is less than 3 seconds [18]. Therefore, in this study, the target response time for each API is not more or equal to 3 seconds.

Table 4 presents the results of load testing on several API samples grouped by HTTP method. Each test scenario has a different number of virtual users, where in the first test scenario, the number of virtual users used is 20 VUs and will increase by 20 in the next test scenario. Min response time shows the fastest response time, max response time shows the longest response time, and avg response time shows the average response time of each API. Error Rate shows the amount of error as a percentage of the total requests given to each API. The conclusion refers to the non-functional requirements of the FeelsQuest feature, where the response time of each API must be no more or equal to three seconds, so the conclusion is divided into two categories. API is categorized as 'optimal' if the average response time of the API is not more or equal to 3 seconds. Meanwhile, an API is considered 'sub-optimal' if the average response time is more than 3 seconds. This categorization allows ease of evaluation of the FeelsQuest API performance in the face of different loads. The results of the REST API performance testing of FeelsQuest features include the following:

Table 4 Load testing results of the FeelsQuest feature API

Scenario	**Thread Group**	**Min Response Time**	**Max Response Time**	**Avg Response Time**	**Error Rate**	**Conclusion**
Scenario 1 (20 VU)	**GET**	22 ms	2,946 ms	162 ms	0,08%	**optimal**
	POST	21 ms	1,070 ms	38 ms	0,00 %	**optimal**
	POST (upload file)	341 ms	68,237 ms	3,368 ms	0,09%	**optimal**
	PUT	21 ms	1,022 ms	36 ms	0,00%	**optimal**
	PUT (upload file)	338 ms	67,998 ms	3,342 ms	0,08%	**optimal**
	DELETE	22 ms	4,227 ms	119 ms	0,08%	**optimal**
Scenario 2 (40 VU)	**GET**	22 ms	5,195 ms	203 ms	1,02%	**optimal**
	POST	20 ms	690 ms	32 ms	0,00%	**optimal**
	POST (upload file)	346 ms	121,893 ms	6,429 ms	1,92%	**sub-optimal**
	PUT	21 ms	670 ms	35 ms	0,00 %	**optimal**
	PUT (upload file)	334 ms	120,706 ms	6,317 ms	1,89%	**sub-optimal**
	DELETE	22 ms	17,358 ms	227 ms	0,47%	**optimal**
Scenario 3 (60 VU)	**GET**	23 ms	16,255 ms	363 ms	3,37%	**optimal**
	POST	21 ms	4,421 ms	45 ms	0,02 %	**optimal**
	POST (upload file)	396 ms	112,084 ms	10,183 ms	6,41%	**sub-optimal**
	PUT	21 ms	3,320 ms	50 ms	0,00%	**optimal**
	PUT (upload file)	392 ms	110,651 ms	10,009 ms	6,03%	**sub-optimal**
	DELETE	23 ms	15,175 ms	296 ms	2,27%	**optimal**

From the results of API performance testing using the Load Testing method, the results show mixed results in each scenario. Scenario 1 (20 Virtual Users) shows optimal performance for most APIs with an average response time below 1 second, except for the POST and PUT methods which require file uploads (around 3 seconds). Scenario 2 (40 VU) saw a slight drop in performance, but most APIs were still optimal, with the file upload POST and PUT methods increasing to 6 seconds (sub-optimal). Scenario 3 (60 VU) showed an increase in average response time and error rate, but most APIs remained optimized with response times below 3 seconds, except the file upload POST and PUT methods (10 seconds, suboptimal). Overall, the majority of APIs met FeelsQuest's non-functional needs with average response times below 3 seconds, even at high loads. However, the file upload POST and PUT methods require special attention as they show performance degradation as the number of virtual users increases.

4. DISCUSSIONS

From the tests that have been carried out on the FeelsQuest feature API, the author's discussion focuses on the results of API performance testing using the load testing method. Each API shows its optimal ability to handle various access load scenarios that resemble the original conditions. However, APIs that require a file upload process, such as create course and/or update course, take longer. This is due to the Laravel framework's need to process file requests before data from users enters the API.

In this study, the authors implemented a REST API architecture to overcome the problem of cross-platform use, and tested the application results in the form of an API to prove that the API meets functional and non-functional requirements. Thus, the author can compare the differences between this research and previous research that is a reference.

Some previous studies used the same solution, namely implementing REST APIs for cross-platform problems, but some only produced applications and tested the functionality of APIs without testing or measuring their performance. Therefore, to complete the reference research study, the author in this research complements previous research by testing the performance of each API using the load testing method. This test is performed using the Postman test tool on the FeelsQuest feature that has been deployed, thus providing a more comprehensive picture of API performance under different load conditions.

With the results of performance testing in this study, it is evident that testing APIs that have been built from the aspect of functionality alone is not enough to prove that the API has met the needs of a system, especially if there are non-functional performance-related requirements such as response time. This comprehensive approach enables a more thorough evaluation of the quality and effectiveness of the developed API.

5. CONCLUSIONS

The results of the research on the implementation of the REST API architecture for FeelsQuest online course feature in FeelsBox application using the Laravel framework, show that the application of the REST API in this feature provides API results that can be accessed from various platforms because API calls can be made by accessing the available access points or web service links. This feature API is functionally appropriate as evidenced by the results of the functionality testing conducted. From the results of functionality testing, each API provides a response that matches the expectations in each test case. In addition, load testing on the FeelsQuest feature API shows that most APIs have met the non-functional needs of the FeelsQuest feature, namely response times under 3 seconds. However, APIs that involve the file upload process require longer response times, exceeding the 3-second limit. This is due to the complexity of the file uploading and processing process which takes longer. Nonetheless, the performance of the API is still acceptable given the specific characteristics of the file upload operation.

In future research, it can use the results of this study as a reference to support research, besides that researchers can add a chunking system to handle the file upload process that has a relatively large size in the Laravel framework to speed up the file upload process time.

6. LITERATURE LIST

[1] R. Choirudin and A. Adil, “Implementasi Rest Api Web Service dalam Membangun Aplikasi Multiplatform untuk Usaha Jasa,” *MATRIK : Jurnal Manajemen, Teknik Informatika dan Rekayasa Komputer*, vol. 18, no. 2, pp. 284–293, May 2019, doi: 10.30812/matrik.v18i2.407.

[2] M. A. Hertiavi, “Penerapan E-Learning dengan Platform Edmodo untuk Meningkatkan Hasil Belajar Mahasiswa,” *Jurnal Komunikasi Pendidikan*, vol. 4, no. 1, pp. 1–8, 2020.

[3] S. A. Achsan and Y. A. Susetyo, “RESTFUL WEB SERVICE IMPLEMENTATION USING SPRING FRAMEWORK IN ROOM ASSETS MANAGEMENT SYSTEM,” *J. Tek. Inform. (JUTIF)*, vol. 3, no. 2, pp. 395–403, Apr. 2022.

[4] M. I. Aulawi, S. Amini, and S. Mulyati, “Implementasi Web Service dengan Metode Restful API dan QR Code untuk Aplikasi Manajemen Inventori pada Toko Indah Jaya Sport,” *Jurnal Ticom:Technology of Information and Communication*, vol. 10, no. 3, pp. 211–217, May 2022.

[5] K. Gowell and Suprihadi, “Perancangan Web Service REST API Menggunakan PHP dan Framework Laravel di Tenta Tour Salatiga,” *Jurnal JTIK (Jurnal Teknologi Informasi dan Komunikasi)*, vol. 8, no. 1, pp. 49–57, Jan. 2024, doi: 10.35870/jtik.v8i1.1269.

[6] K. Prihandani and A. Rizal, “Analisis Perbandingan Kinerja Framework Codeigniter Dengan Express.Js Pada Server RESTful Api,” *Jurnal Ilmiah Wahana Pendidikan*, vol. 8, no. 16, pp. 316–326, Sep. 2022.

[7] W. Hadinata and L. Stianingsih, “ANALISIS PERBANDINGAN PERFORMA RESTFULL API ANTARA

EXPRESS.JS DENGAN LARAVEL FRAMEWORK," *Jurnal Informatika dan Teknik Elektro Terapan*, vol. 12, no. 1, Jan. 2024, doi: 10.23960/jitet.v12i1.3845.

[8] A. S. Perdana and E. Mailoa, "Perancangan Website Penjualan Cupang Menggunakan Laravel(Studi Kasus Salatiga Betta Genetic)," *JATISI (Jurnal Teknik Informatika dan Sistem Informasi)*, vol. 9, no. 2, pp. 1343–1354, Jun. 2022, doi: 10.35957/jatisi.v9i2.2095.

[9] S. Atmojo, R. Utami, S. Dewi, and N. Widhiyanta, "Implementasi Sistem-informasi Desa Berbasis Arsitektur Microservices," *SMATIKA JURNAL*, vol. 12, no. 01, pp. 55–66, Jun. 2022, doi: 10.32664/smatika.v12i01.658.

[10] I. Yatini, F. W. Nurwiyati, and K. Anam, "PERFORMA MICROFRAMEWORK PHP PADA REST API MENGGUNAKAN METODE LOAD TESTING," *Jurnal Informatika Komputer, Bisnis dan Manajemen*, vol. 19, no. 2, pp. 12–20, Nov. 2023, doi: 10.61805/fahma.v19i2.55.

[11] I. Nurjaman, F. S. Utomo, and N. Hermanto, "Penerapan REST API Laravel sebagai Fondasi Back-end Aplikasi G-MOOC 4D," *Journal of Informatics and Interactive Technology*, vol. 1, no. 1, pp. 9–18, Apr. 2024.

[12] R. Choirudin and A. Adil, "Implementasi Rest Api Web Service dalam Membangun Aplikasi Multiplatform untuk Usaha Jasa," *MATRIK : Jurnal Manajemen, Teknik Informatika dan Rekayasa Komputer*, vol. 18, no. 2, pp. 284–293, May 2019, doi: 10.30812/matrik.v18i2.407.

[13] M. A. Novianto and S. Munir, "ANALISIS DAN IMPLEMENTASI RESTFUL API GUNA PENGEMBANGAN SISTEM INFORMASI AKADEMIK PADA PERGURUAN TINGGI," *Jurnal Informatika Terpadu*, vol. 8, no. 1, pp. 47–61, 2022.

[14] D. Akbar, F. Riana, and F. Satrya, "PEMBUATAN WEB SERVICE PADA APLIKASI SIJAB DENGAN METODE REST," *JATI (Jurnal Mahasiswa Teknik Informatika)*, vol. 8, no. 4, pp. 5567–5575, Aug. 2024.

[15] A. Pulodi, Yulianawati, and I. Suwandi, "Implementasi Web Service Restful Dengan Autentikasi JSon Web Token Berbasis Web dan Android," *AJCSR (Academic Journal of Computer Science Research)*, vol. 5, no. 2, pp. 95–103, Jul. 2023.

[16] N. L. A. M. Rahayu Dewi, R. S. Hartati, and Y. Divayana, "Penerapan Metode Prototype dalam Perancangan Sistem Informasi Penerimaan Karyawan Berbasis Website pada Berlian Agency," *Majalah Ilmiah Teknologi Elektro*, vol. 20, no. 1, p. 147, Mar. 2021, doi: 10.24843/MITE.2021.v20i01.P17.

[17] R. J. Romandhondaru and A. Basuki, "Visualisasi Topologi Jaringan berdasarkan Data Routing Border Gateway Protocol," *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, vol. 6, no. 9, pp. 4329–4338, Sep. 2022.

[18] S. S. Raweyai and I. R. Widiasari, "PERFORMANCE TESTING OF ACADEMIC WEBSITE USING LOAD TESTING METHOD SUPPORTED BY APACHE JMETERTM AT XYZ UNIVERSITY," *J. Tek. Inform. (JUTIF)*, vol. 5, no. 3, pp. 721–730, Jun. 2024.