



REST API Service Middleware

**A dissertation submitted for the Degree of
Master of Information Technology**

D. P. D. Dissanayake

**University of Colombo School of Computing
2018**



Declaration

This thesis is my original work and has not been submitted previously for a degree at this or any other university/institute. To the best of my knowledge, it does not contain any material published or written by another person, except as acknowledged in the text.

Student name: D. P. D. Dissanayake

Registration number: 2015/MIT/013

Index Number: 15550138

.....

Signature

.....

Date

This is to certify that this thesis is based on the work of Mr D. P. D. Dissanayake under my supervision. The thesis has been prepared according to the format stipulated and is of acceptable standard.

Certified by:

Supervisor name: Dr D.A.S. Atukorale

.....

Signature

.....

Date

Abstract

Today world moves into smart world and IOT. For this transition, we have to use lots of web services. Web services are the major methodology of communicating data over the network. Web applications and Mobile apps communicate with API web services. REST API represents over 70% of public APIs. That means the importance of the REST API management system. These API services transfer secret and confidential data. Nowadays these web services are used for communication, security alerts, news broadcasting, social media and many more things. A device calls to several servers at once. All the service calls go to many destination servers. All the services are working like a mesh. Sometimes some services down without aware of a responsible party that only the users identified that their requesting service is not working. After the user complains the responsible party get the action for that failure. This happens because of the lack of resources to monitor that kind of services. We need a proper management system to handle these services. Otherwise, data will be exposed to unauthorized parties, hard to rectify bugs, lack of logging facility and slow response.

All the services all over the world could not feasible to monitor a single party. This is feasible to do by the developer parties of the web services. There are lots of web service developer companies in the world. They are developing services for not only other companies but also for themselves. They need some web services for their internal usage. If there is a way to manage this kind of services as a developing company it will be a positive action for the company. Then they could minify previously mentioned issues. But the main problem is services are deployed on several servers. They are working with deferent ports and different URLs. There should be a centralized middleware system to manage those things. If they are worked with the uniform format, then it will be easy to monitor. Then all the services should be going through that middleware. All the filtrations and monitoring could be done by that middleware system. REST API service middleware is for that kind of management functionalities.

This system is a complete REST API managing middleware as a solution for these issues. From this system admin and other web service, developers can do their functionalities by the using the backend panel. The developer can register their services in the system and assign into their applications later. This assigning is called subscription of services. After that services are using the middleware with the new URL. All the services work with the bearer token which use to authenticate. In here OAuth 2 used as the authorization framework for the services. If developers need to reuse the same API for another application then they have to just subscribe the APIs for a new application. If there was a failure in the system, that can be identified easily by the referring the log files. This provides a very attractive and simple interface to manage whole the system. Since this system is the free and open source, any company can use this without a cost.

Acknowledgements

There are many people who have helped me in preparing this dissertation. First I need to thank, Dr D. A. S. Atukorale for supervising my project. Then people who have contributed to the previous versions and others who have given feedback. I would like to thank all the past students of the BIT program who took this course and helped me fine tune this document. My mother and my father always helped me to complete my thesis. My university friends and my office friends always encourage me to do the things better. I need to thanks to my company for the giving me projects which using web services. That was the root cause to identify this kind of systems.

Table of Content

Declaration	i
Abstract.....	ii
Acknowledgements	iii
Table of Content	iv
List of Tables	vii
List of Abbreviation	viii
Chapter 1 : Introduction.....	1
1.1 Overview	1
1.2 Motivation	1
1.3 Statement of the Problem	2
1.4 Objectives	2
1.5 Scope	3
1.6 List of deliverables	4
1.7 Limitations.....	4
1.8 Structure of the thesis	4
Chapter 2 : Background and Literature Review	5
2.1 Background.....	5
2.2 Related projects of previous students	6
2.3 Similar systems available in productions	7
Chapter 3 : Analysis and Design	9
3.1 Functional Requirements.....	9
3.2 Non Functional Requirements	11
3.3 Use Case Diagram	12
3.4 Use Case Scenario	13
3.5 Database ER Diagram	16
3.6 Wireframes	17
3.7 Tools and Technologies Used for Design.....	18
3.8 Implementation of backend logic	19
Chapter 4 : Implementation	21
4.1 Introduction	21
4.2 Development Tools	21
4.3 Frameworks	22
4.4 Implementation Details	23

4.5 License.....	28
Chapter 5 : Evaluation and Testing	29
5.1 Introduction	29
5.2 Evaluation.....	29
5.3 Testing	35
5.4 Testing Scenarios for the application	36
Chapter 6 : Conclusion and Future works	38
6.1 Conclusion.....	38
6.2 Future works.....	38
References	39
Appendix	40
A: Download and setup the application.....	40
B: The Use Case Scenario	41
C: Implementation of the user interface	44

List of Figures

Figure 1 - Normal System	2
Figure 2 - REST API Middleware System	2
Figure 3 - OAuth2.0 abstract flow	6
Figure 4 - Use case diagram	12
Figure 5 - Database ER diagram.....	16
Figure 6 - Wireframes	17
Figure 7 - Service Request Diagram	20
Figure 8 - Laravel migration file	23
Figure 9 - Laravel data seed file	24
Figure 10 - Database in phpMyAdmin	25
Figure 11 - Login Screen	25
Figure 12 - Register screen.....	26
Figure 13 - Dashboard view of Admin.....	27
Figure 14 - User list view of Admin.....	27
Figure 15 - Google Evaluation Form Page 1	30
Figure 16 - Google Evaluation Form Page 2.....	31
Figure 17 - Google Evaluation Form Page 3	32
Figure 18 - Evaluation result - Application setup	33
Figure 19 - Evaluation result - Backend panel performance	33
Figure 20 - Evaluation result - Functionality	33
Figure 21 - Evaluation result - Service Response time	34
Figure 22 - Evaluation result - Overall performance.....	34
Figure 23 - Selenium IDE test case result	36
Figure 24 - JMeter Load Test result	37
Figure 25 - User create view of Admin	44
Figure 26 - User update view of Admin	44
Figure 27 - Application list view of Admin	45
Figure 28 - Service group list view of Admin	45
Figure 29 - Service list view of Admin	46
Figure 30 - Confirmation Modal	46

List of Tables

Table 1 - Use Case 1.....	13
Table 2 - Use Case 2.....	14
Table 3 - Use Case 3.....	14
Table 4 - Use Case 4.....	15
Table 5 - Use Case 5.....	15
Table 6 - Evaluation result - Rating count.....	32
Table 7 - Test Cases	36
Table 8 - Use Case 6.....	41
Table 9 - Use Case 7.....	42
Table 10 - Use Case 8.....	43

List of Abbreviation

Abbreviation	Definition
API	Application Programming Interface
CRUD	Create, Read, Update and Delete
GUI	Graphical User Interface
HA	High Availability
HTTP	Hyper Text Transfer Protocol
JSON	JavaScript Object Notation
IoT	Internet of Things
MIFE	Mobile Internet & Fulfillment Exchange
OAuth	Open Authorization
ORM	Object Relational Mapping
OS	Operating System
PHP	PHP: Hypertext pre-processor
REST	Representational State Transfer
SOAP	Simple Object Access Protocol
URL	Uniform Resource Locator
WWW	World Wide Web
XML	eXtensible Markup Language

Chapter 1 : Introduction

1.1 Overview

There are many different systems in the world. Many of these systems need to exchange data with each other. We use web services for facilitating this task. A web service is a service offered by an electronic device to another electronic device, communicating with each other via the WWW. In a Web service, Web technology such as HTTP, originally designed for human-to-machine communication, is utilized for machine-to-machine communication, more specifically for transferring machine-readable file formats such as XML and JSON. A Web service, in very broad terms, is a method of communication between two applications or electronic devices over the WWW. Mainly web services are of two kinds SOAP and REST. REST API represents the major part of the industry. For one application, there are many different REST API services. Another application might need the same API services. Multiple services to multiple servers for a single application is complex the system architecture.

If we handle a mechanism to call these services through a common middleware then we can easily manage these things. This project would try to define that common abstract layer for the REST API services. Any admin developer can deploy this application on their servers with basic project deployment knowledge. This gives additional authentication for the public web services. The developer can monitor the services by the panel.

1.2 Motivation

Most of the mobile apps and Front-end web applications use REST APIs for communication. REST APIs are more developer friendly, more flexible and faster than SOAP. I am working as a back-end developer who develops web services for web frontends and mobile applications. I am working with this kind of RESTful web services. I have to work with similar systems in the industry and I have to face several problems with them. That system has many problems. Less flexibility, slowness and lack of user-friendly, unreliable responses give me encouragement to develop this kind of manageable system. Since this happen we need a proper manageable environment to handle those things. There are some similar systems in Sri Lanka but they are not an open source and free. This system is open source and free. Our company developers also encourage me to develop this kind of system to manage all the services developed by our company. There are few similar Java applications in the production environment. They are very hard to configure and they required another third-party application. Then need to purchase that additional application also. But there is no any PHP application related to this functionality. That is the reason for developing this application using PHP language.

1.3 Statement of the Problem

Software development companies create a different kind of REST API services for different kind of applications. Sometimes they use the same service for many apps. It is hard to manage all these services separately. According to the Figure1, it shows how complex the unmanaged systems. Service request goes to multiple servers. This project aim is to solve the above complexity and manage these services allocation and monitoring.

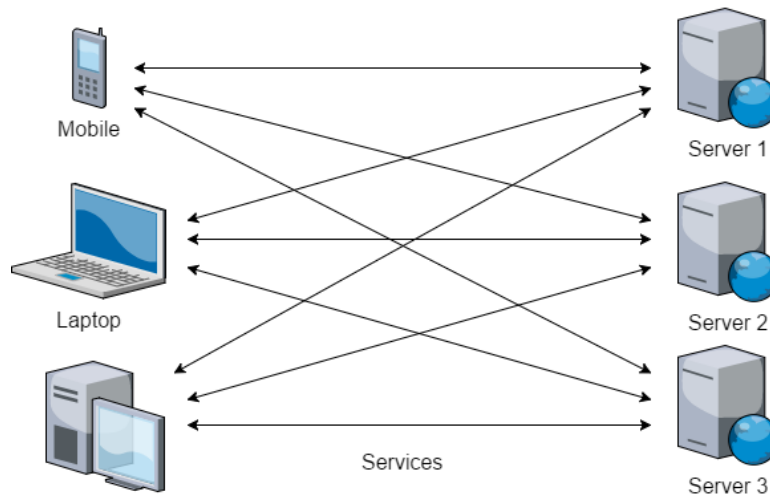


Figure 1 - Normal System

1.4 Objectives

The objective of this system is to provide a proxy middleware and panel to manage all these RESTful APIs in a single place. Technically this kind of system is called an API ecosystem. This is shown in Figure 2. This can easily identify any issue in services. Give analytical results to the all the services and developers can identify the usage of all the services.

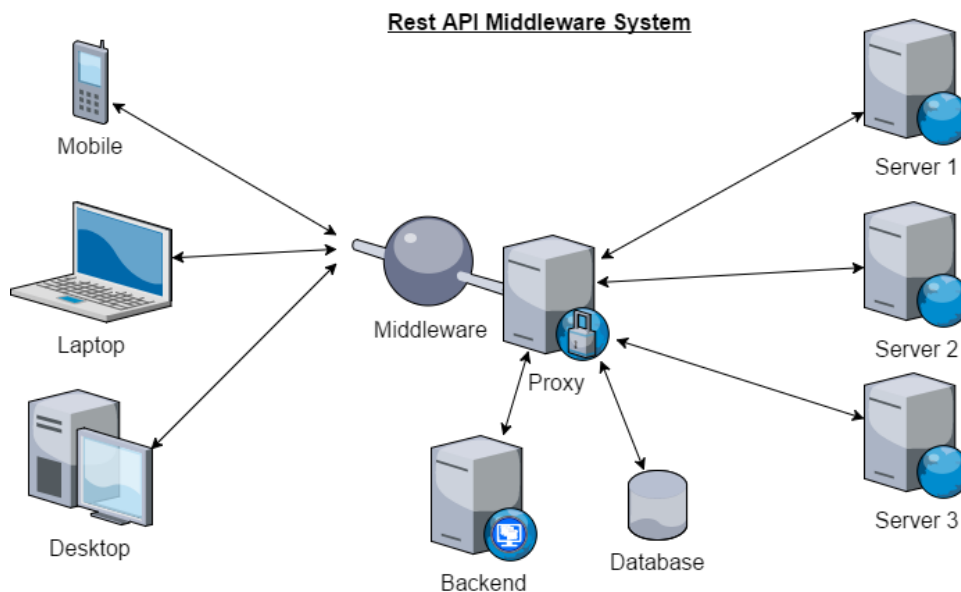


Figure 2 - REST API Middleware System

In this system, first of all, administrators should register the APIs in the API store through the backend panel. The web or mobile app developers need to register first in the system. It is not

only as an individual person, they can register as their company. When they going to build a new app using RESTful APIs, they can use already registered services or they can register new APIs in the API store with the approval of administrators. Then developers need to register the app on the system. The system automatically issues sandbox and live unique app keys and app secret keys which can identify the app services here. Then developers have to subscribe required API services which app needs. Admin panel approves this subscription after checking the app specification. After the approval developers can use this APIs for their apps.

That middleware log all the services. It can easily identify any issue in the services. Monitor all the services and do analytics. This system help in protecting services back-end systems from overexposure.

- Providing middleware proxy for RESTful API service calls
- Increase the security of services using a proxy.
- Providing logging facility API requests and responses.
- Providing analytics for services.
- Providing a backend panel for users (developers) and app registrations.
- After the app register, Users can subscribe API services for their apps.
- Providing sample RESTful API services

1.5 Scope

This system is developed for Rest API service developers to manage their services. Anyone can download this application from GitHub and change according to their environments. All the users can log in to the system once it is hosted on a web server with PHP 7.0.0 or higher version. The database has been created using Eloquent ORM. Therefore admin can change the database to MySQL, PostgreSQL, SQLite or SQL server as he required. Admin can do all the administration functionalities by using the back-end panel. They can create all the other developer accounts. Developer accounts need to be active before login. Developers need to register their services using a backend panel. That services use in applications which can be a mobile or web application. This application must have the approval of the system admin. Middleware proxy works only for the registered RESTful API service calls. That services group into related categories. This grouping is used to identify the services easily. Logging facility provides for only user preferences.

I have to assume that there were not any hardware problems in the system. Because I am not going to consider network hardware. There must be a load balance server and high availability servers for this kind of system. It is good for the reliability and the best performance of the services. This system is for web service developers, not for the general users. Then there is no better user-guided documentation available for the user. All the guides available in the GitHub repository Wiki. This application is the free and open source with the covering of MIT license.

1.6 List of deliverables

- Finalized Middleware proxy
- Finalized Backend panel (Admin panel and developer panel)
- Complete GitHub repository with the public access.
- Few samples RESTful API services working through middleware

Middleware proxy and the backend panel are consists of this application. Other RESTful API services are created on a separate application.

1.7 Limitations

This system most suitable for JSON content type RESTful web services. The speed of the system depends on the server specifications. This system should be suitable for HA servers. Otherwise if the main system failure cause to full system failure. There should be a method to minimize the network traffic. Otherwise, the response will be a delay. Need a little Laravel framework knowledge to deploy. Working database types are MySQL, PostgreSQL, SQLite or SQL. For Oracle Database, there should be to install an additional library to the application. There is only one admin account for the system. Other accounts are developer accounts. If new developer user register, then admin must active the developer account. After that, the developer can use the account. Applications also need to approve by the admin. This restriction is for the security of the system.

1.8 Structure of the thesis

The second chapter will be the background and literature review. This gives background information with references to published material in research papers, URLs, magazine articles. The third chapter, the design and the methodology of the project. The fourth chapter describes the implementation of the application. Then the fifth chapter describes the test cases and conclusions of the project.

Chapter 2 : Background and Literature Review

The background technologies and related projects are going to discuss in this chapter. Those technologies are mostly used in the industry. There are few similar applications in the industry.

2.1 Background

REST API

REST or RESTful [1] web services is a method of providing interoperability between computer systems on the Internet. This kind of web services allows requesting systems to manipulate and access textual representations of web resources using a predefined and uniform set of stateless operations. “REST” was coined by Roy Fielding in his PhD dissertation [2] to describe a design pattern for implementing networked systems. REST stands for Representational State Transfer, an architectural style for designing distributed systems. It’s not a standard, but rather a set of constraints. It’s not tied to HTTP but is associated most commonly with it.

Web Service Middleware

Web services middleware consists of supporting products that work with primary Web services application or facilitate the functionality of an application and an OS. Sometimes Web services middleware is also known as Web services management. Web services middleware plays the "middleman" role in the overall web architecture. This layer can manage things like security or cross-platform data communications. Web services middleware works like a client and server architecture where the Web services application is the client and the middleware is the server, that is, it provides services to the client. This is how many engineers and developers think about the process when they add Web services middleware to a particular application.

OAuth 2.0

OAuth 2.0 [3] is the industry-standard protocol for authorization. OAuth 2.0 supersedes the work done on the original OAuth protocol created in 2006. OAuth 2.0 focuses on client developer simplicity while providing specific authorization flows for web applications, desktop applications, mobile phones, and living room devices. This OAuth2.0 token issue abstract flow shown in Figure 3.

Abstract Protocol Flow

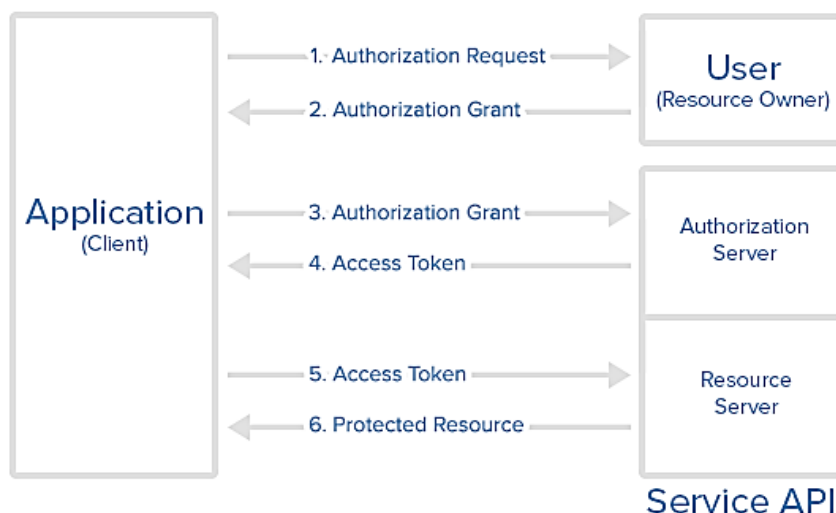


Figure 3 - OAuth2.0 abstract flow

2.2 Related projects of previous students

REST Data Services and API Manager Framework for PHP (PHPDS)

There was an API manager framework to build REST data services called “REST Data Service & API Manager Framework for PHP” in 2014 as the MIT project. This project short name was PHPDS. According to its final report, it was very lightweight, fast, free and open source framework, which allowed developers to create a web service from databases in a few seconds without writing any single PHP code. It provides a very attractive interface and user experience to generate services. The universal SQL was used if developers need custom services other than create, retrieve, update and delete operations. Different types of authentication mechanisms can be applied to created data services including social networking authentications such as Facebook, Twitter and Google authentication. APIs finally can be managed by its versions using this framework. Developers can host it in their own environment to ensure data security. It also facilitates developers to concentrate more on end user experience rather than taking their mind to think out complex backend logic. The main advantage of this framework is it can be hosted in even a shared hosting environment, which is very low cost than expensive cloud hosting solutions. Since it is free and open source anyone can use this software without any cost.

Access Control for RESTful Web Services

There was a project called “Access Control for RESTful Web Services” in 2014 as Master of Science in Information Security project.

Web services are the prominent methodology of communicating data over the network. A web service accepts an XML based or JSON based request and returns a response according to the request. RESTful form of web services is a mechanism of exchanging data in compliance with HTTP protocol. These services may or may not supply confidential information. Information security has three main aspects – Confidentiality, Integrity and Availability. The data services which give out potentially secretive information need to be secure from any unauthorized access. These services can be restricted to a user, a group of users, based on time, etc. This study is about defining and implementing an extensible access control framework for RESTful web services. There are quite a number of security frameworks that provide access control systems in the market. But none of them can be connected to authorize a RESTful web service out of the box. This framework should be able to connect to any authentication, authorization and accounting service. This study will further benchmark the new access control framework's performance in order to position it with the other access control systems in the market.

2.3 Similar systems available in productions

IDEABIZ

IDEABIZ [4] is an online digital enablement API Platform provided by Dialog Axiata PLC, through which business entities can create applications using the given “API” toolkits or publish “API”s to enable the use of other services. IDEABIZ is a product of Ideamart under Dialog Axiata PLC. The Ideamart is a company which tries to promote the latest technology among schools and universities students. They build a new platform called IDEABIZ which helps to use Telco web services for newcomers for the developing. This platform is gathered many web services and users can use them by subscribing to their applications.

MIFE

WSO2.Telco builds on technologies first applied to the successful API-driven solution developed at Dialog, and Axiata’s have launched Mobile Internet & Fulfillment Exchange (MIFE) [5], which connects all Axiata OpCos to a central hub, both powered by WSO2 middleware. The overall solution deployed in Sri Lanka has generated huge interest from thousands of developers whose products and services offered to Dialog’s 9 million customers have realized a new revenue stream, which is growing 20% every month.

These two systems are in a production environment. IDEABIZ is used by developers and non-developers for creating simple apps by using its available APIs. MIFE is used by developers who create internal applications of Dialog Axiata PLC. But there are some issues with them. When considering in IDEABIZ, the backend panel has some UX and functionality issues. Users

hard to find what to do and how to proceed. Some links are broken. When considering in MIFE, it's not reliable. MIFE backend panel is the very slow system. It is hard to use waste lots of time. Many times gives a proxy error for the services and lack of documentation. There is no any logging mechanism. It is hard to rectify a trouble. These two systems are not the free and open source.

REST Data Services and API Manager Framework for PHP (PHPDS) is a framework which can create REST API service by using a panel. It is a simple way to build web service. My project is not exactly matching with this project because I am not going to create web services. I am trying to do manage the already created web services. Access Control for RESTful Web Services project is considered about the only the security of the web service. REST API service middleware system has a security module, but it is not only targeting the security matter. The scope is bigger than the previous project.

Chapter 3 : Analysis and Design

All the analysis and design methodologies are discussed in this chapter. All the functional and non-functional requirements are mentioned in this chapter.

3.1 Functional Requirements

Multiple user roles

There are two main roles for the backend panel. They are developer user role and admin user role. Admin user accounts are predefined user accounts. Admin user should create developer user accounts. Also, the developer user should have a facility to self-register to the system. Then admin user should approve the self-registered developer accounts. After that developer users could log in to the backend panel.

Services

Services mean the REST API services in here. The developer and admin user should have permissions to register these services in the backend panel by using a URL. After services registered on the backend, the admin user should have permission to approve the registered service. Hereafter services should be worked on the system.

Service Group

Set of services related to one single app should be defined as a service group. The user should have to create a service group before register a service. When registering the services, the user should assign a service group to the service. The purpose of this service group is easy the control of services. Every service under one service group should be able to activate and deactivate using service group.

Application

This system could use for multiple applications. The user should have created an application in the backend panel for use the services. Then the admin user should approve before using the created application. Then the user should subscribe required services to that application.

Subscription

The user should assign the service to an application before use it. That is called a subscription. This subscription should be approved by an admin user. This subscription could be used to identify which services are used in an application.

Create middleware for the services

Services should be managed by this middleware. Authentication mechanism and the log should be done by this layer. All the limitations and restrictions should be done by this layer according to the admin approval.

Create backend panel

Admin and developers should use this panel to manage services. Users should log in to this panel. Then they can easily identify the main things what they have to do by navigating side menu. This panel should be created very user-friendly manner even the users are developers. Admin and developers could communicate with each other by using messages on the panel.

Sandbox mode for API services

All the services should have sandbox mode for the development purposes. So testing could be done by using sandbox mode without affecting the production environment.

Applying security for each service

The services should be authenticated with OAuth2.0. This will increase the security of all the services.

Log the services

All the services should be log and that data should be accessed easily. All the logs could access through the panel, no need to login to the server separately. Logs should have written in a human-readable manner. It will be easy to identify an issue in the services.

Do analyze using logs

Well, formatted log entries should be used for the data analyzing part. This also managed by the admin panel. Charts will display it clearly.

3.2 Non Functional Requirements

User friendliness

This backend panel should have a nice and clear user interface, which provides a great user experience for all the users. All the CRUD parts should be easy to identify with the related icons/symbols.

Performance

This system should be created using Laravel 5.5 framework. This is the best framework for the PHP web industry and it is fast and reliable. PHP 7.1(latest stable version) use in the system. That is double the speed than PHP 5.6(previous stable version).

Extensibility

This system could be managed by a huge number of applications and services. Not only for few applications. If the user needs to expand the database using PostgreSQL, SQLite or SQL server then the user could easily do that changing the database driver of the system because system this created on Laravel Eloquent ORM.

Maintainability

This system source should in the GitHub version control system. Then all the versions should be tagged. GitHub is free for open source projects. Fixing bugs and adding new features of the system should be made easy by branching the source repository and fix them separately and later merge it with master repository.

Platform compatibility

This system should run on major operating systems including Linux, Windows, Macintosh and Solaris.

Response Time

The final overall response time of APIs through the middleware should be minimal. This can be achieved using best practices of the software development by reducing the complexity of the code.

3.3 Use Case Diagram

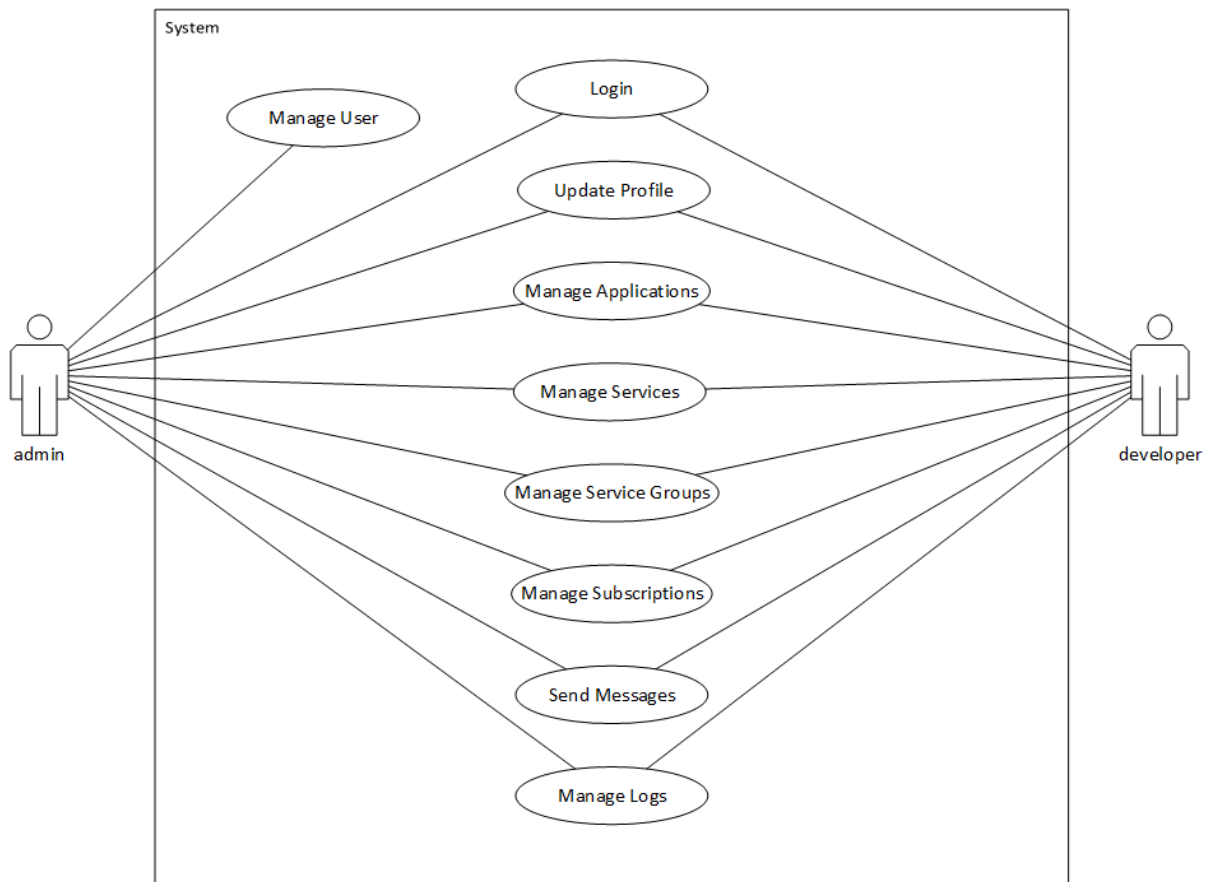


Figure 4 - Use case diagram

In this system, admin and developer are the main actors of the project. Both of them has the same tasks except few functionalities. This Figure 4 diagram shows the high-level use case activities of these actors. Admin can create all the user accounts. Developers could not create their own accounts because that could be an opportunity for unauthorized parties to access the system. Admin and developer have a different dashboard to do their functionalities. Admin has the full responsibility to manage the system.

3.4 Use Case Scenario

Following detail, scenarios describe all use cases in the diagram.

Use Case number	1
Use Case Name	Login
Purpose	Log in to the system
Priority	High
Actors	Admin, Developer
Precondition	-
Basic Flow	<ol style="list-style-type: none"> 1. The user enters username and password. 2. System validate admin username and password. 3. User login to the system. 4. User redirect to their dashboard according to user role
Alternative Flows	<ol style="list-style-type: none"> 2.1 The system rejects to log in to the system. 2.2 Login error message is shown.
Postcondition	-

Table 1 - Use Case 1

Use Case number	2
Use Case Name	Manage User
Purpose	Manage user accounts in the system
Priority	High
Actors	Admin
Precondition	Admin must log in to the system
Basic Flow	<ol style="list-style-type: none"> 1. Create User <ol style="list-style-type: none"> 1. Admin select "USERS" menu inside the menu. 2. Click on "CREATE NEW USER" button. 3. Fill the user creation form. 4. Click on "CREATE" button. 5. Validate the form data. 6. Save new user data. 2. Show User <ol style="list-style-type: none"> 1. Admin select "USERS" menu inside the menu. 2. Click on created user show button in the list. 3. Show user data. 3. Update User <ol style="list-style-type: none"> 1. Admin select "USERS" menu inside menu. 2. Click on created user edit button in the list. 3. Fill the user update form. 4. Click on "UPDATE" button. 5. Validate the form data. 6. Update user data. 4. Delete User <ol style="list-style-type: none"> 1. Admin select "USERS" menu inside menu. 2. Click on "DELETE" button of the user in the list. 3. Show delete confirmation message 4. Click on "DELETE" Button 5. User delete from the system
Alternative Flows	1.5.1 Validation errors in the form.

	1.5.2 Validation errors are shown in the create form. 3.5.1 Validation errors in the form. 3.5.2 Validation errors are shown in the update form.
Postcondition	-

Table 2 - Use Case 2

Use Case number	3
Use Case Name	Update Profile
Purpose	Update current user profile
Priority	Low
Actors	Admin, Developer
Precondition	Admin must log in to the system
Basic Flow	1. User select “My Account” from the user top menu. 2. Fill the profile update form. 3. Click on “UPDATE” button. 4. Validate the form data. 5. Save profile data.
Alternative Flows	4.1 Validation errors in the form. 4.2 Validation errors are shown in the update form.
Postcondition	-

Table 3 - Use Case 3

Use Case number	4
Use Case Name	Manage Application
Purpose	Manage application in the system
Priority	High
Actors	Admin
Precondition	Admin must log in to the system
Basic Flow	<ol style="list-style-type: none"> 1. Create Application <ol style="list-style-type: none"> 1. User select “APPLICATIONS” menu on the side menu. 2. Click on “CREATE NEW APPLICATION” button. 3. Fill the application creation form. 4. Click on “CREATE” button. 5. Validate the form data. 6. Save new application data. 2. Show Application <ol style="list-style-type: none"> 1. User select “APPLICATIONS” menu on the side menu. 2. Click on created application show button in the list. 3. Show application data. 3. Update Application <ol style="list-style-type: none"> 1. User select “APPLICATIONS” menu on the side menu. 2. Click on created application edit button in the list. 3. Fill the application update form. 4. Click on “UPDATE” button. 5. Validate the form data. 6. Update application data. 4. Delete Application <ol style="list-style-type: none"> 1. User select “APPLICATIONS” menu on the side menu. 2. Click on “DELETE” button of the application in the list. 3. Show delete confirmation message 4. Click on “DELETE” Button 5. Application delete from the system

Alternative Flows	1.5.1 Validation errors in the form. 1.5.2 Validation errors are shown in the create form. 3.5.1 Validation errors in the form. 3.5.2 Validation errors are shown in the update form.
Postcondition	-

Table 4 - Use Case 4

Use Case number	5
Use Case Name	Manage Service Groups
Purpose	Manage Service Groups in the system
Priority	High
Actors	Admin, Developer
Precondition	The user must log in to the system
Basic Flow	<ol style="list-style-type: none"> 1. Create Service Group <ol style="list-style-type: none"> 1. User select "API GROUPS" menu on the side menu. 2. Click on "CREATE NEW SERVICE GROUP" button. 3. Fill the service group creation form. 4. Click on "CREATE" button. 5. Validate the form data. 6. Save new service group data. 2. Show Service Group <ol style="list-style-type: none"> 1. User select "API GROUPS" menu on the side menu. 2. Click on created service group show button in the list. 3. Show service group data. 3. Update Service Group <ol style="list-style-type: none"> 1. User select "API GROUPS" menu on the side menu. 2. Click on created service group edit button in the list. 3. Fill the service group update form. 4. Click on "UPDATE" button. 5. Validate the form data. 6. Update service group data. 4. Delete Service Group <ol style="list-style-type: none"> 1. User select "API GROUPS" menu on the side menu. 2. Click on "DELETE" button of the service group in the list. 3. Show delete confirmation message 4. Click on "DELETE" Button 5. Service group delete from the system
Alternative Flows	1.5.3 Validation errors in the form. 1.5.4 Validation errors are shown in the create form. 3.5.3 Validation errors in the form. 3.5.4 Validation errors are shown in the update form.
Postcondition	-

Table 5 - Use Case 5

Rest of the use case scenarios in Appendix B.

3.5 Database ER Diagram

ER diagram was designed by MySQL workbench tool. That diagram is shown in Figure 5

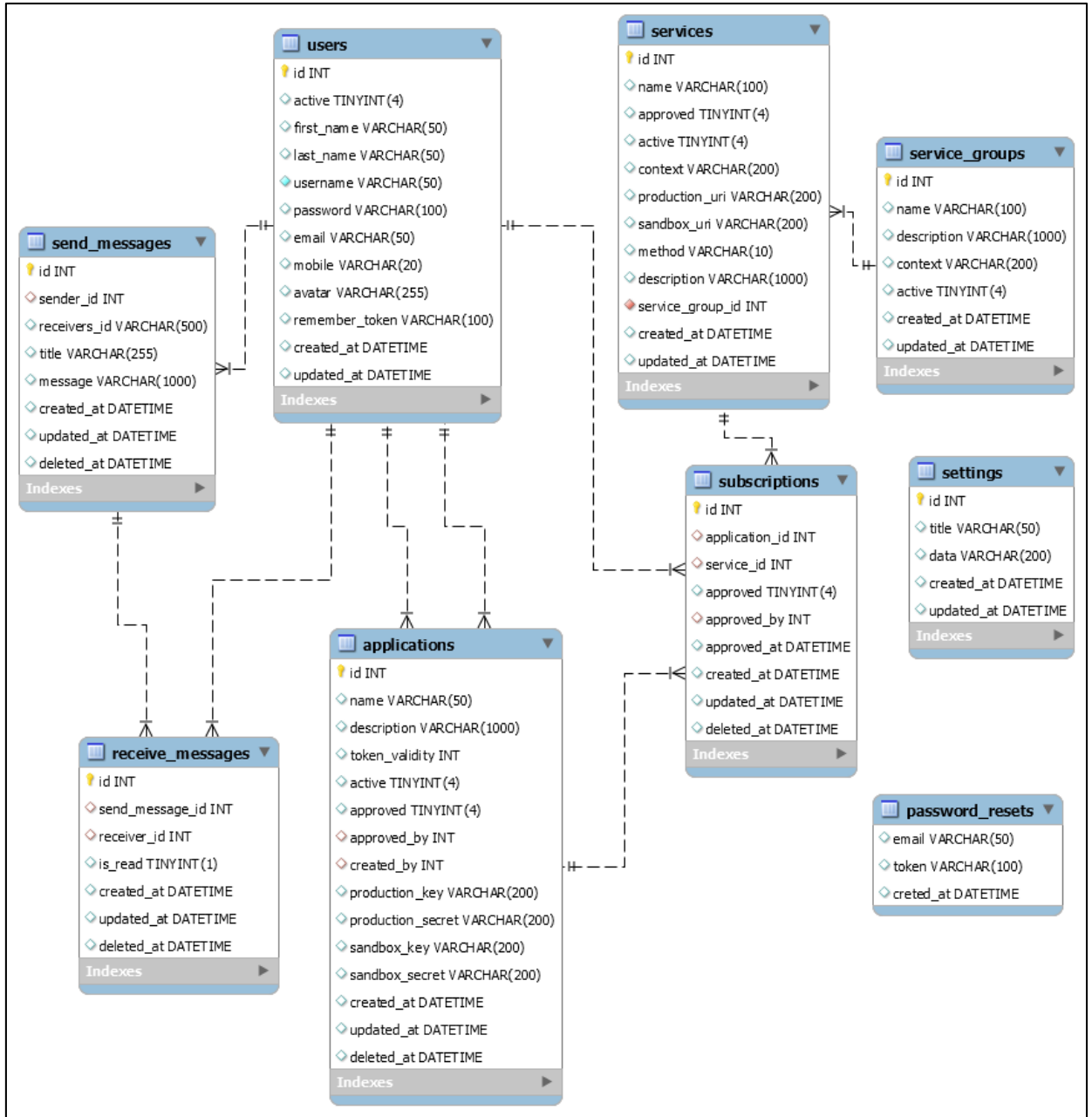


Figure 5 - Database ER diagram

One service group has many services. One service has many subscriptions. One application has many subscriptions.

3.6 Wireframes

Created some wireframes in Figure 6.

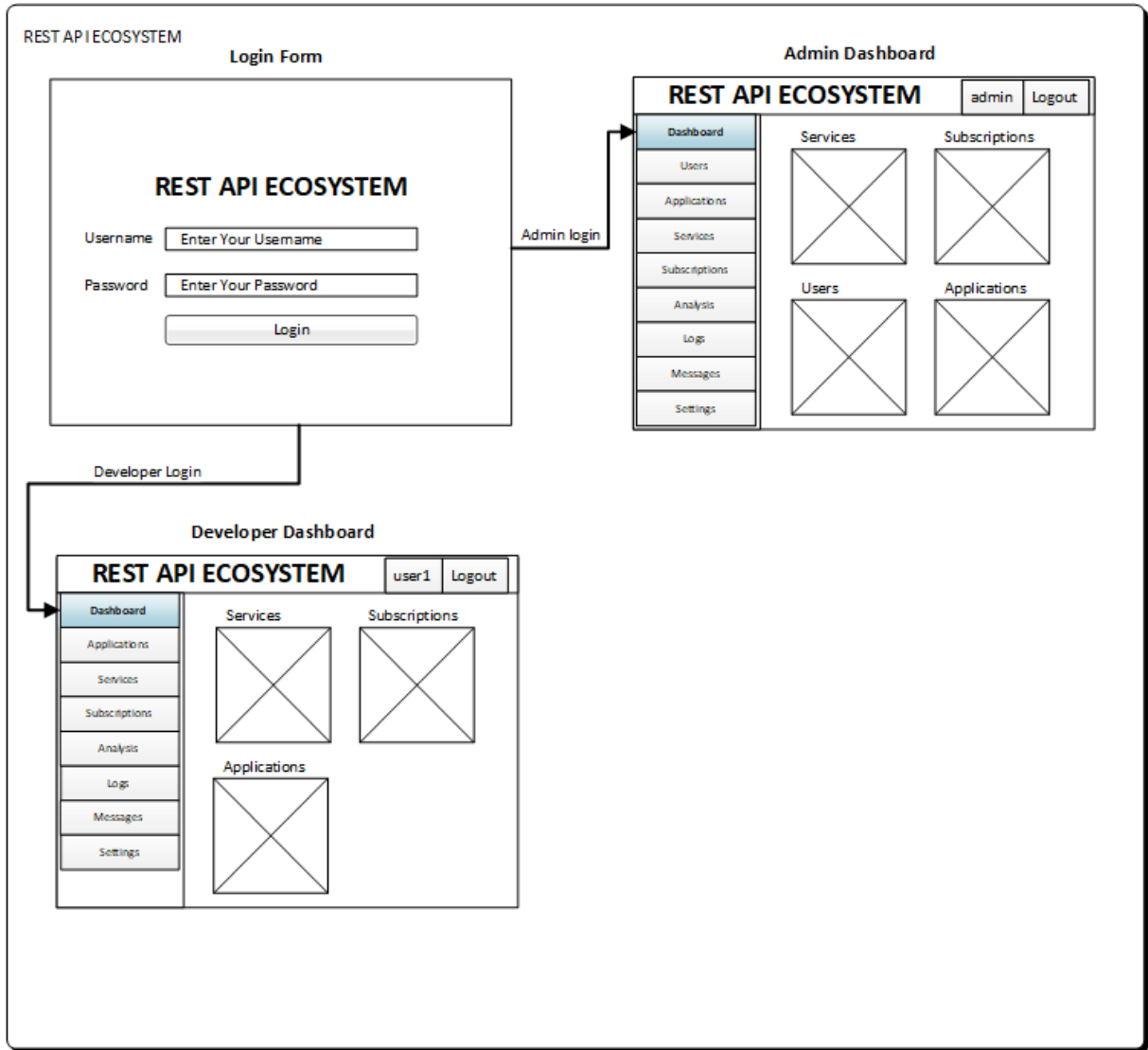


Figure 6 - Wireframes

3.7 Tools and Technologies Used for Design

Draw.io

Draw.io [6] is a free online diagram editor for making flowcharts, UML diagrams, ER diagrams, network diagrams mockups and more. That diagrams can be shared with Google Drive, One Drive, Dropbox, GitHub and Trello. Not only that but also diagram can be exported as images. I have created my network diagram using Draw.io website. Network diagrams have more variety of objects to design the best diagram.

Microsoft Visio

Microsoft Visio [7] is also a software for drawing a variety of design diagrams. This application is more flexible with Microsoft office packages. I have created my use case diagram and wireframe using Microsoft Visio.

MySQL Workbench

MySQL workbench [8] is a unified visual tool for database developers. This is a simple graphical user interface to do the database administrations easily. This provides data modelling, SQL query development, administration tools and much more. MySQL Workbench is available for Windows, Linux and Mac OS. This tool can use not only for forwarding engineering but also reverse engineering. I have created the database ER diagram using MySQL Workbench software.

3.8 Implementation of backend logic

Laravel developments based on MVC pattern. This method is the most used method for application development. MVC architectural pattern divides an application into major three interconnected parts. This is done to divide internal representations of information from the ways information is presented to and accepted by the user. This allowing for efficient code reuse and parallel development.

1. Models
2. Views
3. Controllers

Models

Models have been created for fetching data from the database. There are models for each and every table. For example, the user table has Users model. Table relationship is also managed by the models. These models are used inside of the controllers. Models directly communicate with the database through the database connection. Every table needs a model. Table names should be in plural and model names should be in the singular. For example, the “users” table model name is “User”.

Views

Views are user presentation part. The user can see this part of the code. Laravel blade template engine used to build this part. All the JavaScript are executed in this layer. CSS and styling part is also done by views. View are grouped in a folder according to their module functionality.

Controllers

Controllers are done the middleware functionality of the code. Fetching data from models and pass them to views by using controllers. User submitted data processed here and transfer them to models to store or update. All the UI functionalities are categorized into groups and create a controller. For example user CRUD in the separate file called “UserController”.

In Laravel, there is another functionality called routing. This is done by route file. There are two route files for web and API. The major functionality of this is mapping the URLs with the controller. It identifies the request URL and directs that request to the relevant controller. This request flow described by the Figure 7. The backend panel web request or external API service request first come into route module and then go to the controller module. Then request goes to model for fetch data from the database. Then request goes to view or external web services.

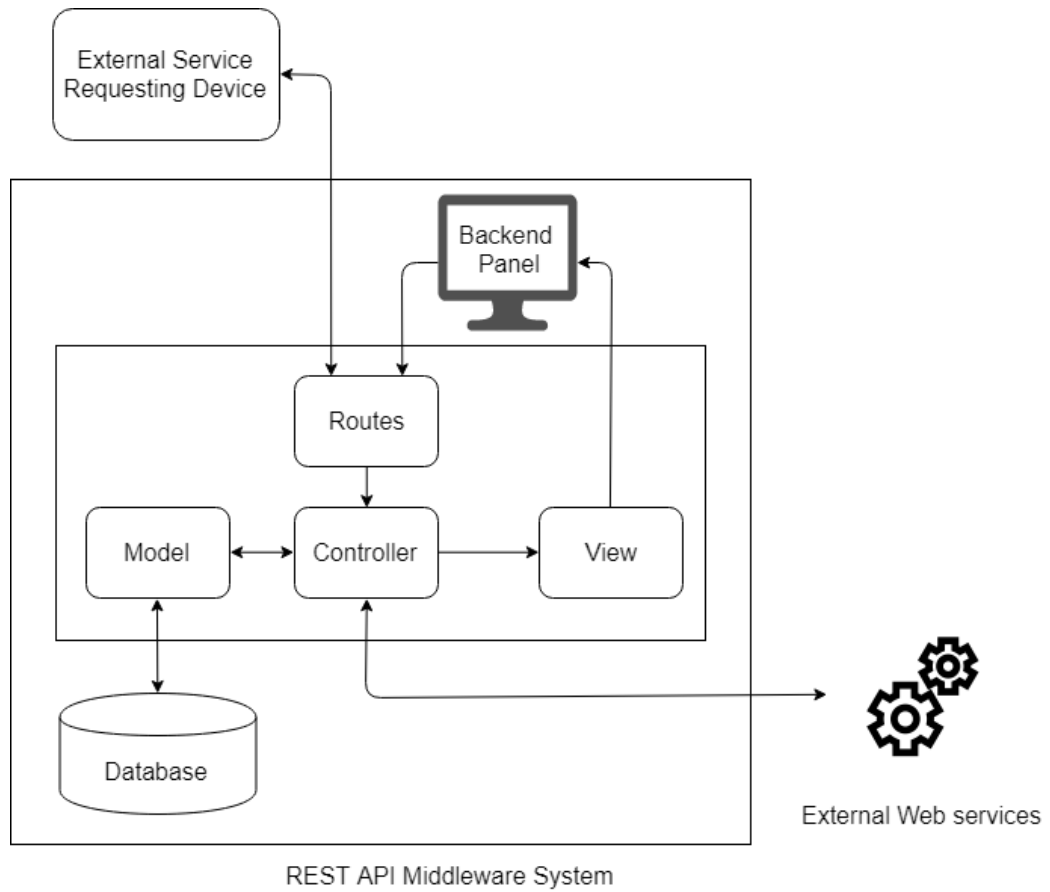


Figure 7 - Service Request Diagram

Between routes and controller, there is another functionality called middleware in Laravel. This middleware filters the request and creates the log entries for every route.

Chapter 4 : Implementation

4.1 Introduction

In this chapter discussed the development tools and what are the frameworks used to develop this REST API service middleware application. It has been much faster and easier to change the developments by using the following tools.

4.2 Development Tools

Following software tools had been identified to develop the REST API service middleware application.

1. Apache 2 Web server
2. PHP 7.1
3. MariaDB 10.1
4. PHPMysqlAdmin
5. PHP extensions : OpenSSL, PDO, Mbstring, Tokenizer, XML
6. PhpStorm IDE
7. Git Version control system

XAMPP

The XAMPP [9] stands for Cross-Platform (X), Apache (A), MariaDB (M), PHP (P) and Perl (P). It is a simple, lightweight Apache distribution that makes it extremely easy for developers to create a local web server for testing and deployment purposes. This is very easy to set up the environment without any installation overhead.

PhpStorm IDE

JetBrains PhpStorm [10] is a commercial, cross-platform IDE for PHP built on JetBrains' IntelliJ IDEA platform. PhpStorm provides an editor for PHP, HTML and JavaScript with on-the-fly code analysis, error prevention and automated refactoring for PHP and JavaScript code. PhpStorm's code completion supports latest PHP versions (modern and legacy projects), including generators, co-routines, the final keyword, list of foreach, namespaces, closures, traits and short array syntax. It includes a full-fledged SQL editor with editable query results. PhpStorm is built on IntelliJ IDEA, which is written in Java. Users can extend the IDE by installing plugins created for the IntelliJ Platform or write their own plugins.

Git

Git [11] [12] is a free and open source distributed version control system designed to handle all from small to very large projects with speed and efficiency. Git is very easy to learn and also has a tiny footprint with lightning fast performance. Every Git working directory is a full-fledged repository with complete history and full version tracking capabilities, not dependent on network access or a central server.

4.3 Frameworks

Laravel 5.5

Laravel [13], [14] is a free, open-source PHP web framework and intended for the development of web applications following the model-view-controller (MVC) architectural pattern. Some of the features of Laravel are a modular packaging system with a dedicated dependency manager, different ways for accessing relational databases it called Eloquent ORM, utilities that aid in application deployment and maintenance, and its orientation toward syntactic sugar. Laravel framework can create amazing layouts with dynamic content, owing to its lightweight templates and widgets including JavaScript & CSS Code with solid structures. Laravel framework offers numerous inbuilt functions, which simplifies the development process making it quick and accessible.

Bootstrap 3

Bootstrap [15] is a free and open-source web front-end library for designing websites and web applications. Bootstrap's responsive CSS adjusts to phones, tablets, and desktops. It contains HTML and CSS based design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions.

jQuery

jQuery [16] is a lightweight, "write less, do more", JavaScript library. The purpose of jQuery is to make it much easier to use JavaScript on the website. jQuery takes a lot of common tasks that require many lines of JavaScript code to accomplish and wraps them into methods that you can call with a single line of code. jQuery also simplifies a lot of the complicated things from JavaScript, like AJAX calls and DOM manipulation.

4.4 Implementation Details

Implementation was started after the designing phase. The code was written in PHP language. Laravel is the best framework for PHP language which use to develop this application. Database creation has done by Laravel migrations. The view was created using Laravel blade template engine. I have used Bootstrap 3 to create the site responsiveness. Then customize the style of Bootstrap classes. It was developed and tested using XAMPP. Once the module was completed, source code was committed to GitHub using Git.

4.4.1 Implementation of the database

According to the ER diagram which created using MySQL Workbench, created the Laravel database migrations. Then run the migrations and create tables in the database. Testing data seeded by using Laravel seeder files. Then phpMyAdmin was used to view the database while developing. Figure 8 shows the user table migration file.

```
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateUsersTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create( table: 'users', function (Blueprint $table) {
            $table->increments( column: 'id');
            $table->tinyInteger( column: 'active')->default(0);
            $table->string( column: 'first_name', length: 50)->nullable()->default(null);
            $table->string( column: 'last_name', length: 50)->nullable()->default(null);
            $table->string( column: 'username', length: 50)->unique();
            $table->string( column: 'password', length: 100);
            $table->string( column: 'email', length: 50);
            $table->string( column: 'mobile', length: 20)->nullable()->default(null);
            $table->string( column: 'avatar', length: 255)->nullable()->default(null);
            $table->rememberToken();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
}
```

Figure 8 - Laravel migration file

After table migration test data can be inserted by the seed files. I have created some data seed files. Figure 9 shows the user table seed file. For the seeding, first of all, truncate the table and then insert data by using an array. “bcrypt” hash function used for password encryption. Carbon library which builds in Laravel framework used for date time format.

```
1 <?php
2
3 use Illuminate\Database\Seeder;
4 use Illuminate\Support\Facades\DB;
5
6 class UsersTableSeeder extends Seeder
7 {
8     /**
9      * Run the database seeds.
10     *
11     * @return void
12     */
13     public function run()
14     {
15         DB::table('users')->truncate();
16         DB::table('users')->insert([
17             [
18                 'active' => 1,
19                 'first_name' => 'Dasun',
20                 'last_name' => 'Dissanayake',
21                 'username' => 'admin',
22                 'password' => bcrypt( value: '1qaz2wsx'),
23                 'email' => 'dpdasun@gmail.com',
24                 'mobile' => '0710474824',
25                 'avatar' => '/assets/avatar/admin.jpg',
26                 'created_at' => \Carbon\Carbon::now(),
27                 'updated_at' => \Carbon\Carbon::now(),
28             ], [
29                 'active' => 1,
30                 'first_name' => 'Shalinda',
31                 'last_name' => 'Suresh',
32                 'username' => 'shalinda',
33                 'password' => bcrypt( value: '1qaz2wsx')
```

Figure 9 - Laravel data seed file

Created databases were changed by using the phpMyAdmin tool when I needed for developing. Figure 10 shows the database architecture in the phpMyAdmin tool. It is very flexible Graphical User Interface (GUI) tool for manage databases. The database has been created type is InnoDB and collation is utf8mb4_unicode_ci which support Unicode characters with emoji icons.

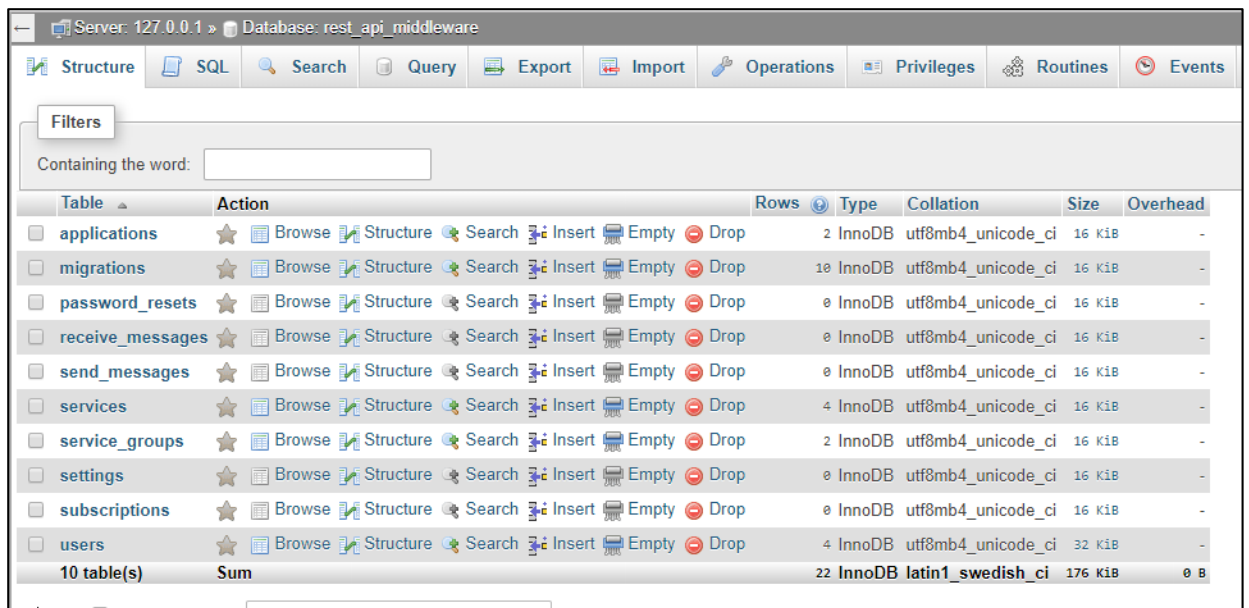


Figure 10 - Database in phpMyAdmin

4.4.2 Implementation of the user interface

HTML, JavaScript and CSS were used to create basic user interfaces. Bootstrap and jQuery used to extend the design and implement a more user-friendly way. AJAX was used to do requests without browser refreshments

The Login page has shown in Figure 11 registered and approved users can log in to the backend panel by using username and password. New developers can register to the system by using the register button. It directs the user to register page.

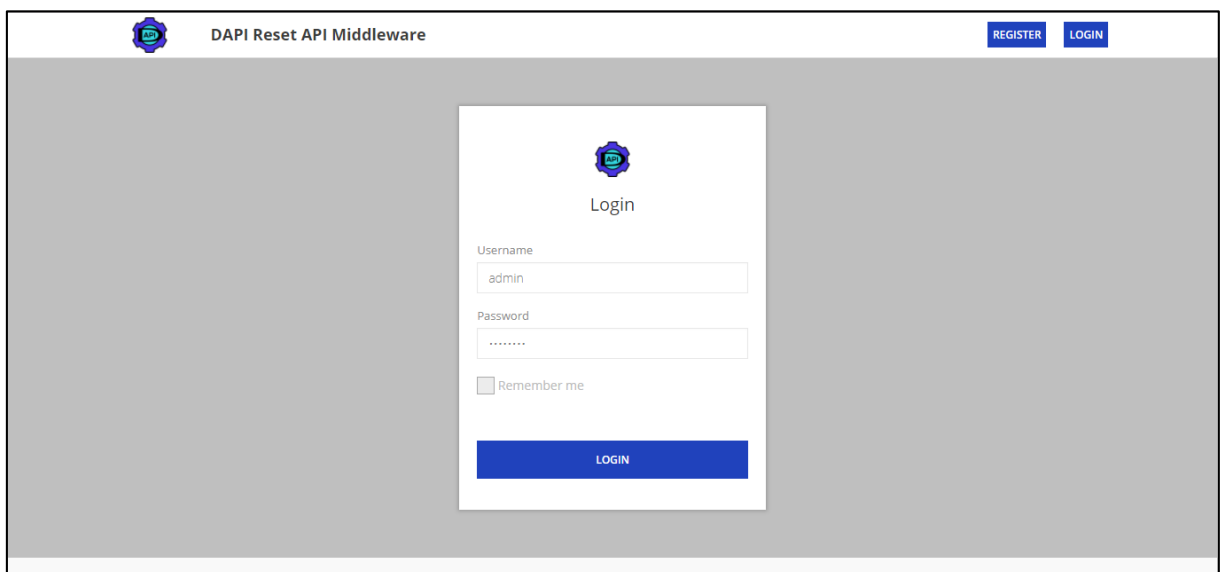


Figure 11 - Login Screen

New user registration page screen is in Figure 12. First name, last name, username, email, mobile number, password and password confirmation are the registration details. The user can add own avatar image also.

The screenshot shows a web application interface for user registration. At the top left, there is a logo and the text "DAPI Reset API Middleware". At the top right, there are two buttons: "REGISTER" and "LOGIN". The main content area is a white card titled "Register" with a blue circular avatar icon. Below the icon are seven input fields: "FIRST NAME", "LAST NAME", "USERNAME *", "E-MAIL ADDRESS *", "MOBILE NUMBER", "PASSWORD *", and "CONFIRM PASSWORD *". A blue "REGISTER" button is located at the bottom of the form.

Figure 12 - Register screen

Successfully registered users cannot login to the system at once. The admin must active the registered user. After that user can log in to the system.

The admin can login to the system by using his credentials at the login page. When admin user login to the system, he will direct to Dashboard which shows in Figure 13. The admin dashboard is completed with pending items. Then the admin can easily identify what he wants to do first.

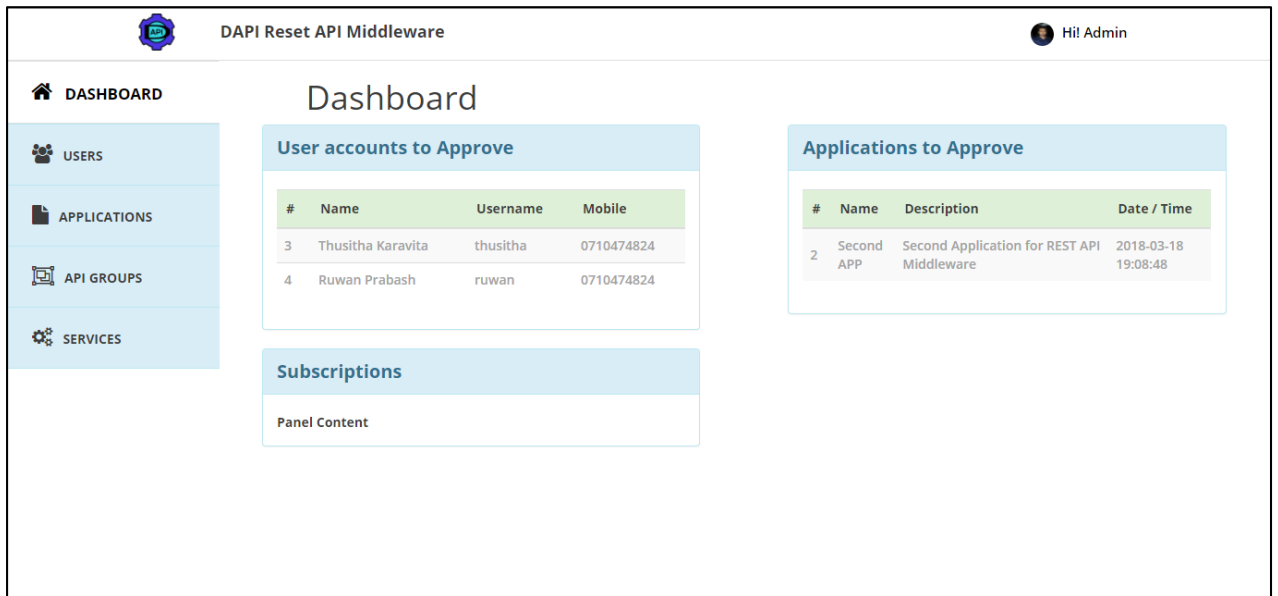


Figure 13 - Dashboard view of Admin

All the users are listed in this user list page. Admin user can manage all the developer user accounts by using the user list view page in Figure 14. There is an active button for each user to activate the account. If the new user in the pending list then admin can activate their accounts by click on active switch button. Admin can create a new developer account by the using “CREATE NEW USER” button on the top of the user list view.

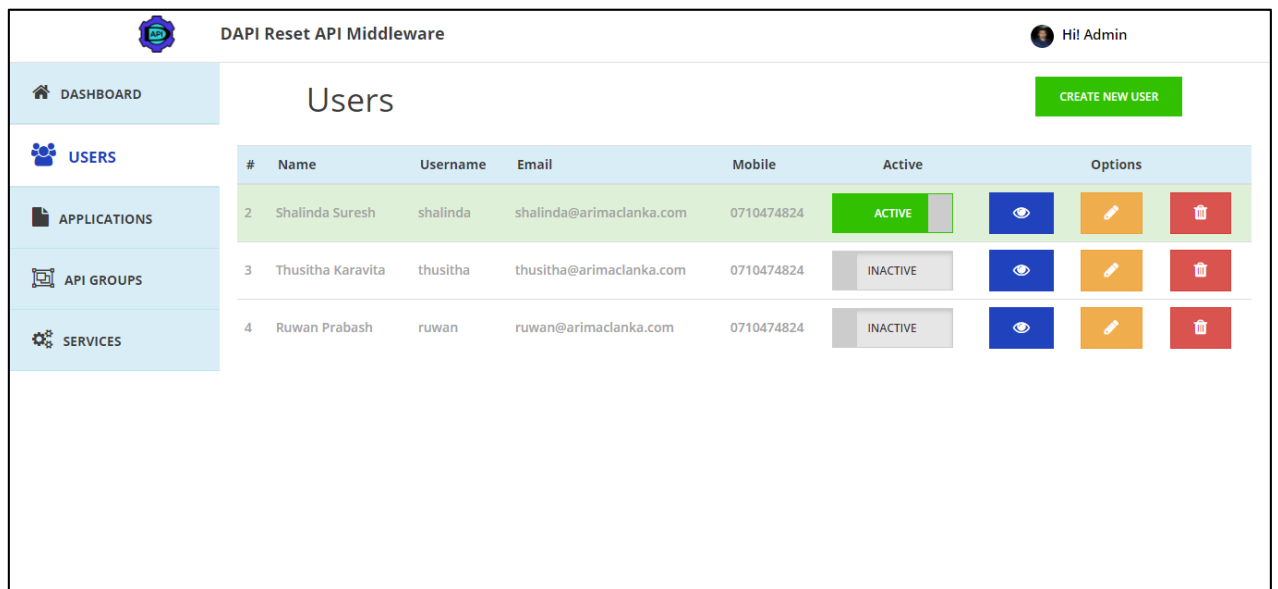


Figure 14 - User list view of Admin

Rest of the user interfaces in Appendix C.

4.5 License

This application is free and open source. Any person can get the source code and modify according to their usage. The MIT License is a permissive free software license originating at the Massachusetts Institute of Technology (MIT). As a permissive license, it puts only very limited restriction on reuse and has, therefore, an excellent license compatibility.

MIT License

Copyright (c) 2017 Dasun Dissanayake

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Chapter 5 : Evaluation and Testing

5.1 Introduction

The REST API middleware system evaluation is done by the sharing the system with my co-workers. There was 23 people have participated in my evaluation. The REST API middleware system was tested by the black box and white box testing. White box testing was done at the time of the source code was written including the unit and integration testing. Black box testing was done using test scenarios.

5.2 Evaluation

A user evaluation is carried out after the completing the project. Following results have been identified by the users who are working as developers in my office, evaluation which was done in recent. Application setup, Service Response time, Backend panel performance, Functionality and Overall performance evaluated by the evaluation form.

I informed my co-workers about the system and I have shared the GitHub link with them and inform them to use it. Then I create a form for gathering evaluation results from the users. That form consists of some evaluation questions which can identify the user experience. After collection the real user data, I came to a conclusion about the system.

Evaluation form structure

REST API Service Middleware System Evaluation Form

Name:

Designation:

Industry Working experience:

Please rate the following questions in regards to this evaluation. Scale 1-5;

(1)-bad, (2)-poor, (3)-average, (4)-good, (5)-excellent

	1	2	3	4	5
Easy to the download the source code					
Easy of the installation					
Operating System Compatibility					
Backend panel loading time					
The consistency of interface of backend panel					
User friendliness of interfaces					

Overall backend panel appearance					
Easy of the use of backend panel					
The response time of the backend panel					
The overall functionality of backend panel					
The response time of the services which use the middleware					
The overall idea of the system					

Comments

.....

Finally, I have created a Google form [17] for the getting the evaluation results using 23 user results. The evaluation form is shown in Figure 15, 16 and 17. That was easy to collecting results than a printed form.

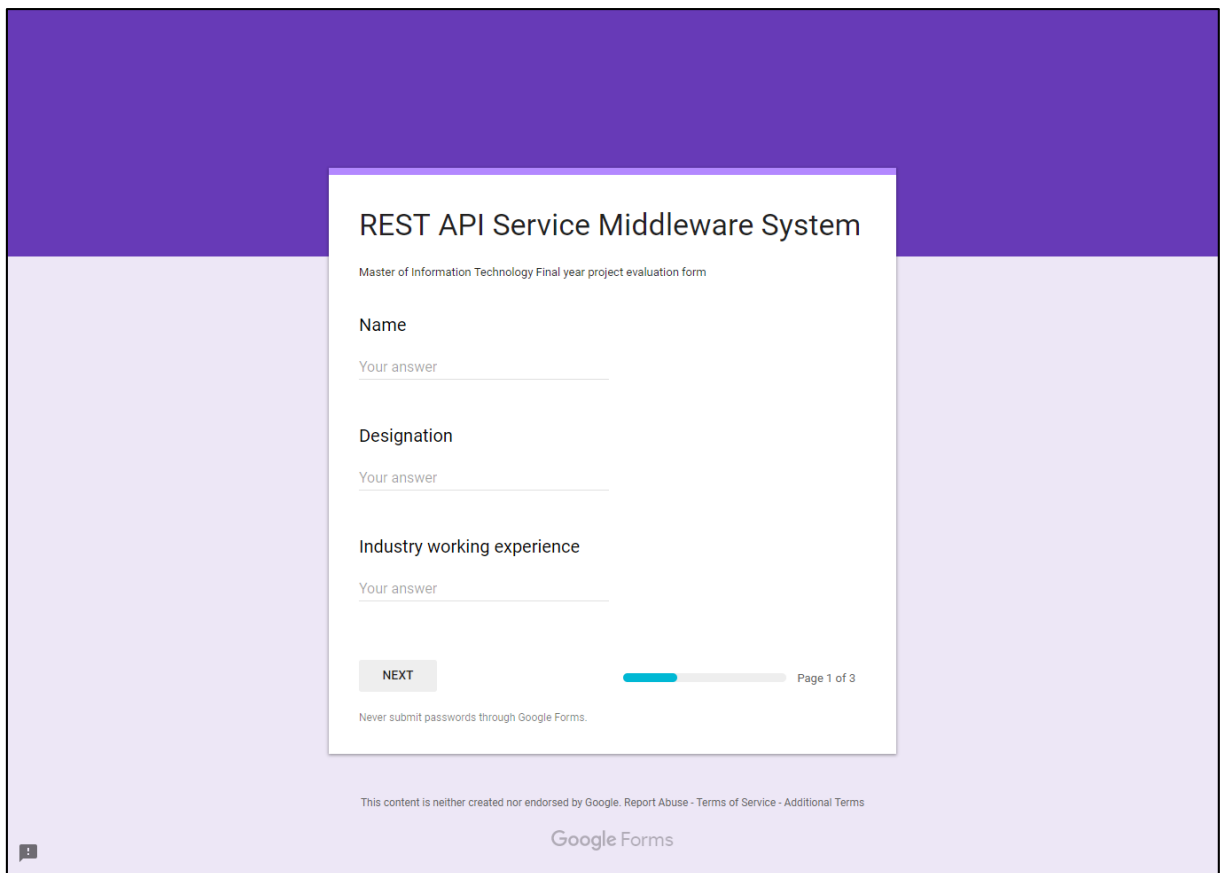


Figure 15 - Google Evaluation Form Page 1

REST API Service Middleware System

* Required

Rating Questions

Please rate the following questions in regards to this evaluation. Scale 1-5;
(1)-bad, (2)-poor, (3)-average, (4)-good, (5)-excellent

Easy of the download the source code *

1 2 3 4 5
Bad Excellent

Easy of the installation *

1 2 3 4 5
Bad Excellent

Operating System Compatibility *

1 2 3 4 5
Bad Excellent

Backend panel loading time *

1 2 3 4 5
Bad Excellent

Consistency of interface of backend panel *

1 2 3 4 5
Bad Excellent

User friendliness of interfaces *

1 2 3 4 5
Bad Excellent

Overall backend panel appearance *

1 2 3 4 5
Bad Excellent

Easy of the use of backend panel *

1 2 3 4 5
Bad Excellent

Response time of the backend panel *

1 2 3 4 5
Bad Excellent

Overall functionality of backend panel *

1 2 3 4 5
Bad Excellent

Response time of the services which use the middleware *

1 2 3 4 5
Bad Excellent

Overall idea of the system *

1 2 3 4 5
Bad Excellent

BACK NEXT Page 2 of 3

Never submit passwords through Google Forms.

This content is neither created nor endorsed by Google. Report Abuse - Terms of Service - Additional Terms

Google Forms

Figure 16 - Google Evaluation Form Page 2

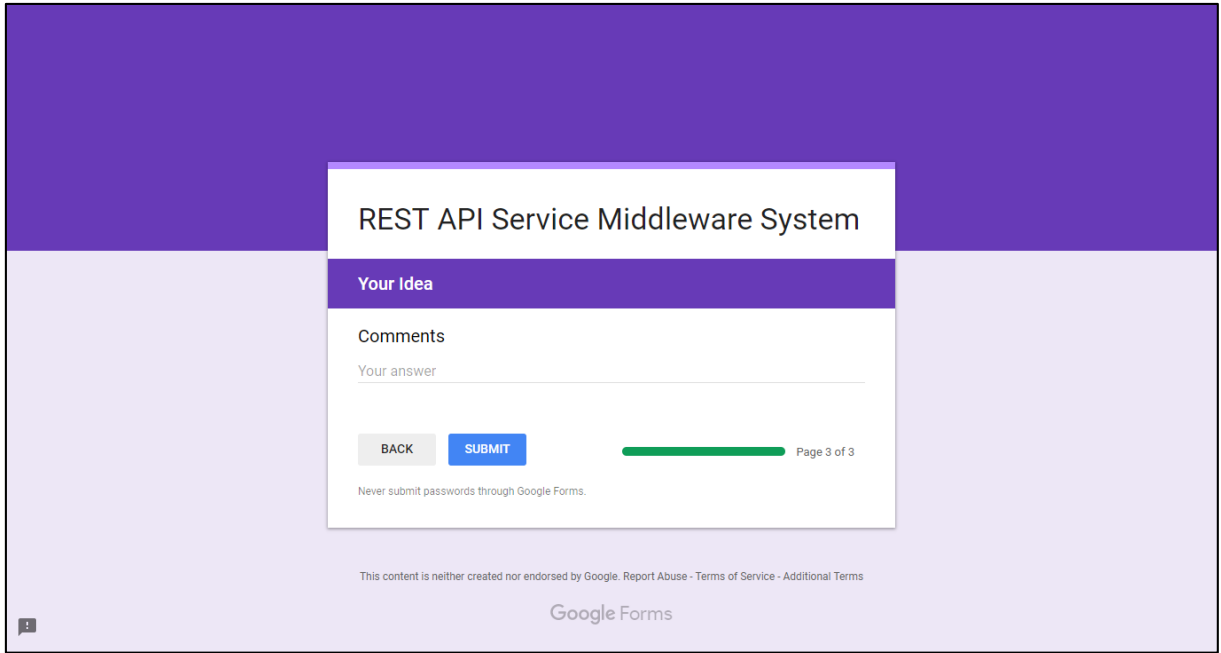


Figure 17 - Google Evaluation Form Page 3

The evaluation results as follows. Table 9 shows the total rating count of the evaluation results. Figure 18, 19, 20, 21 and 22 show the percentage value of the evaluation results.

Questions	Rating				
	1	2	3	4	5
Easy to the download the source code	0	0	2	3	18
Easy of the installation	0	0	1	4	18
Operating System Compatibility	0	0	2	4	17
Backend panel loading time	0	0	1	6	16
Consistency of interface of backend panel	0	0	2	3	18
User friendliness of interfaces	0	0	0	6	17
Overall backend panel appearance	0	0	1	5	17
Easy of the use of backend panel	0	0	0	7	16
Response time of the backend panel	0	0	1	5	17
Overall functionality of backend panel	0	0	1	5	17
The response time of the services which use the middleware	0	0	2	4	17
Overall idea of the system	0	0	1	6	16
Total	0	0	14	58	204
Average	0%	0%	5%	21%	74%

Table 6 - Evaluation result - Rating count

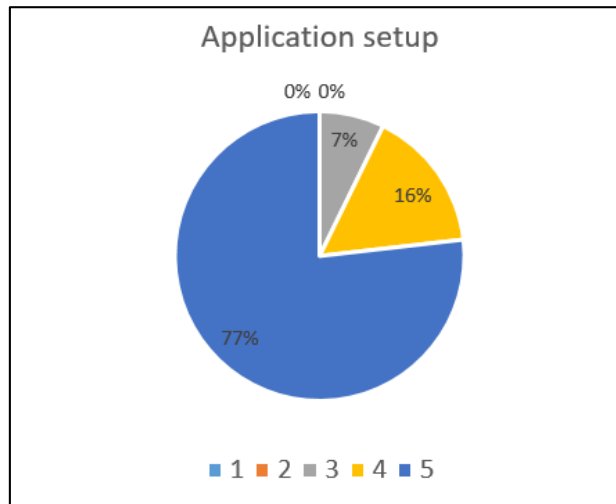


Figure 19 - Evaluation result - Application setup

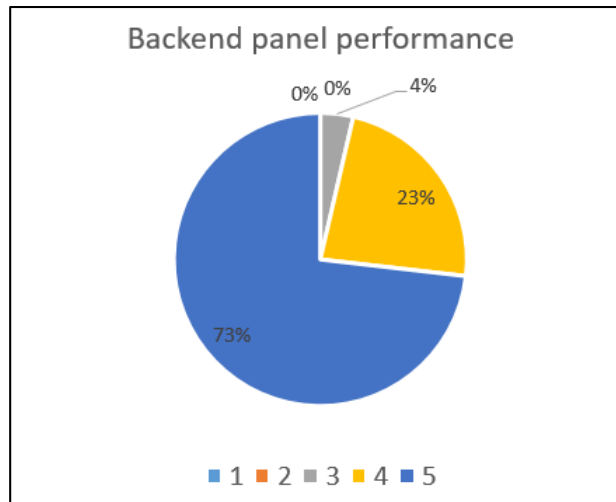


Figure 18 - Evaluation result - Backend panel performance

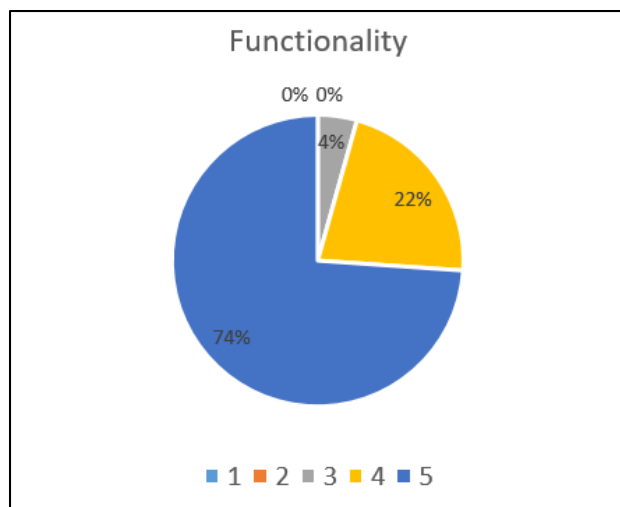


Figure 20 - Evaluation result - Functionality

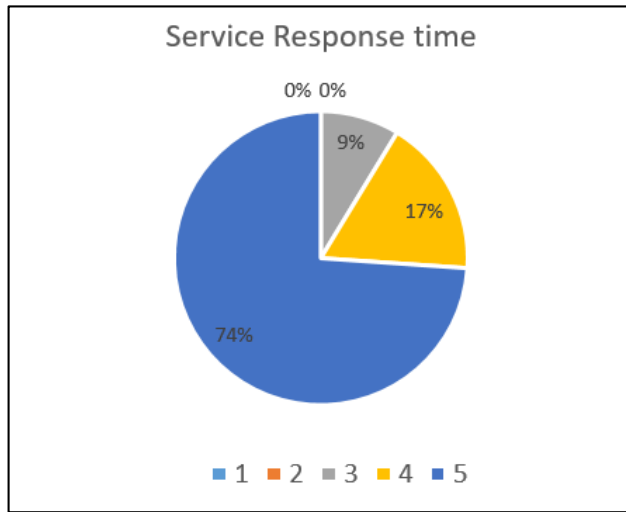


Figure 21 - Evaluation result - Service Response time

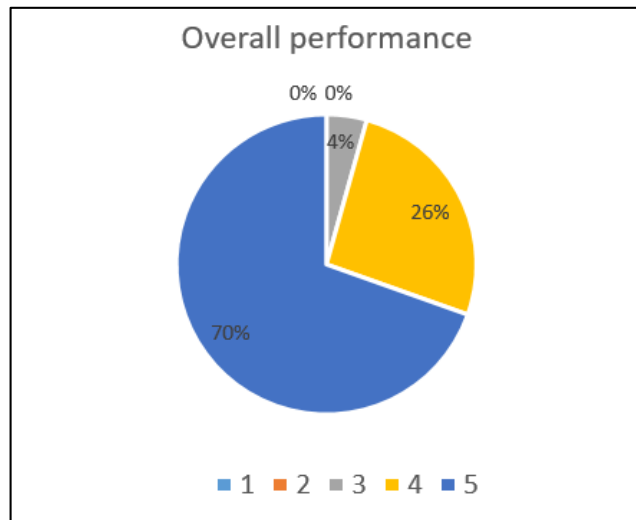


Figure 22 - Evaluation result - Overall performance

5.3 Testing

Introduction

Testing is a very important part of system development methodology. The objective of testing executes a program with the intention of finding present errors in the system which has not been discovered, in a minimum time and effort. It also helps to identify the correctness, completeness, security and quality of the developed software.

Unit Testing

Unit testing is the method which smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation. This can be done manually or automated. Unit testing is a component of test-driven development. Test-driven development requires that developers first write failing unit tests. Then they write code and refactor the application until the test passes.

White box Testing

White box testing is a software testing method in which the internal structure, design, implementation of the item being tested is known to the tester. It focuses primarily on strengthening security, the flow of inputs and outputs through the application, and improving design and usability. White box testing is also known as Clear Box testing, Open Box testing, Structural testing, Transparent Box testing, Code-Based testing, and Glass Box testing.

Black box Testing

This testing method is known as behavioural testing. That examines the functionality of an application based on the specifications. It is also known as Specifications based testing. Independent Testing team usually performs the black box testing. This method of test can be applied virtually to every level of software testing. Testing, either functional or non-functional, without reference to the internal structure of the component or system.

- Unit testing
- Integration Testing
- System Testing
- Acceptance Testing

Load Testing

It's a type of performance testing that simulates real-world load on any software, application, or website. It examines how the system behaves during normal and high loads and determines if a system, piece of software, or computing device can handle high loads given a high demand of end users.

5.4 Testing Scenarios for the application

Test cases

Selenium [18] is the best tool for the web application test automation. Then I decided to use it for the test automation functionalities. Selenium IDE used in the automation testing. It is very easy to use. Then I have executed the following test cases in Table 8 by using Selenium IDE. One of the test results is shown in Figure 23.

#	Test case	Test case description	Status
1	Register	Register a new user	Pass
2	Login	Log in to the system as a registered user	Pass
3	Show dashboard	Show dashboard for the logged in user.	Pass
4	CRUD Users	Create/View/Update/Delete users	Pass
5	CRUD Applications	Create/View/Update/Delete applications	Pass
6	CRUD Service groups	Create/View/Update/Delete service groups	Pass
7	CRUD Service	Create/View/Update/Delete services	Pass
8	CRUD Subscriptions	Create/View/Update/Delete subscriptions	Pass

Table 7 - Test Cases

The screenshot shows the Selenium IDE interface for a test case named 'Login*'. The test is running on the URL 'http://localhost'. The test case consists of seven steps:

Command	Target	Value
1. open	/MY/rest-api-middlew.../public/login	
2. type	id=username	admin
3. type	id=password	1qaz2wsx
4. click at	css=div.row.login_bg	313,285
5. click at	//button[@type='submit']	178,28
6. click at	css=div.col-sm-12 > div.col-sm-8	42,9
7. click at	css=li > a.text-center	87,12

The log at the bottom shows the following execution results:

```

1. Trying to execute open on /MY/rest-api-middlew.../public/login... Success
2. Trying to execute type on id=username with value admin... Success
3. Trying to execute type on id=password with value 1qaz2wsx... Success
4. Trying to execute clickAt on css=div.row.login_bg with value 313,285... Success
5. Trying to execute clickAt on //button[@type='submit'] with value 178,28... Success
6. Trying to execute clickAt on css=div.col-sm-12 > div.col-sm-8 with value 42,9... Success
7. Trying to execute clickAt on css=li > a.text-center with value 87,12... Success
'Login' completed successfully
  
```

Figure 23 - Selenium IDE test case result

Sample REST API could not test using Selenium IDE. For that, there should be a tool to create request. REST Client applications can generate requests. Advanced REST Client and Postman REST client are popular rest client tools available in the software field. Then Postman REST Client software was used for the API testing. It is a very flexible tool to create a request.

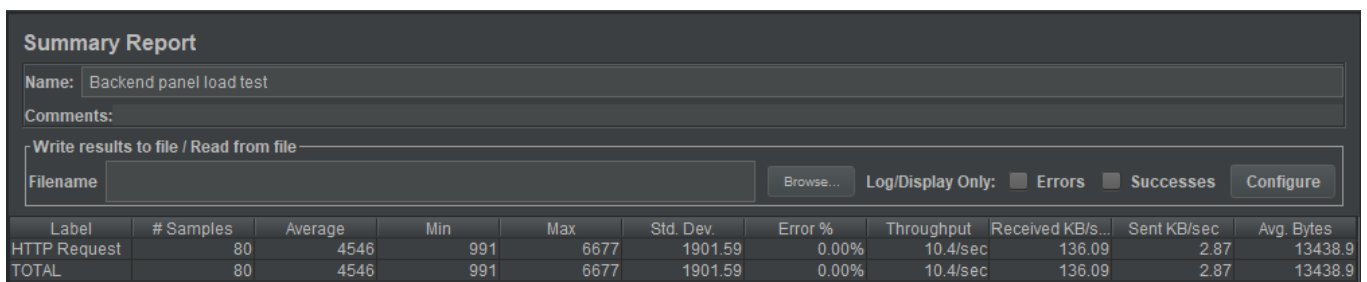
Load Test

This application mainly works with REST API web services. Then the load test is the most important test scenario for the web services. Then run a load test using JMeter tool. I have started the number of threads 50. Then gradually increase the number of treats until 80. It works fine without any issue for all the scenarios. This test is done on my local machine. Actually, this load test depends on the server which the application deploy. My local machine the last result is in Figure 24.

Number of threads (users): 80

Ramp-up Period (seconds): 1

Loop count: 1



The screenshot shows the JMeter Summary Report interface. At the top, the test name is 'Backend panel load test'. Below that, there are fields for 'Comments' and 'Write results to file / Read from file'. The 'Filename' field is empty, and there is a 'Browse...' button. To the right, there are checkboxes for 'Log/Display Only', 'Errors', and 'Successes', along with a 'Configure' button. Below these controls is a table with the following data:

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/s...	Sent KB/sec	Avg. Bytes
HTTP Request	80	4546	991	6677	1901.59	0.00%	10.4/sec	136.09	2.87	13438.9
TOTAL	80	4546	991	6677	1901.59	0.00%	10.4/sec	136.09	2.87	13438.9

Figure 24 - JMeter Load Test result

Result:

Error percentage: 0%

Throughput: 10.4/sec

This means the system can handle 80 user request within 1 second without any issue. But the actual throughput is 10.4/sec.

Chapter 6 : Conclusion and Future works

6.1 Conclusion

The main objective of this project is given a manageable middleware application for unmanageable REST API web services. That target was achieved by developing REST API service Middleware. This application is not developed for a single client. This project is a narrow one. This solution is for all the web service developing companies who wish to organize their web services. This application is now available in the GitHub repository as a public repository. Anyone can download the application, hosted on any web hosting server and then use it. Not only that they can customize the application as they preferred.

This system gives additional security for the public RESTful web services. Log all the request through the middleware proxy. Then the developers no need to worry about the web services monitoring.

6.2 Future works

This application needs more optimization for performance required web services. This will be available with the future release. Report generation is still not a function of this application. That also will be available in next release. Web service lives monitoring system is a more effective way to identify the system failures. There are many tools available in the market. Nagios [19] application is one of the best tools for that kind of tasks. These future releases can be added to the same GitHub repository. And the users can easily update their repositories.

References

- [1] S. Patni, Pro RESTful APIs, Apress Media, 2017.
- [2] R. T. Fielding, 2000. [Online].
Available: https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm.
- [3] "oauth.net," [Online]. Available: <https://oauth.net/2/>.
- [4] IDEABIZ. [Online]. Available: <http://docs.ideabiz.lk/welcome>.
- [5] WSO2. [Online]. Available: <https://wso2.com/about/news/wso2-and-axiata-launch-radical-new-open-source-digital-enablement-platform-for-mobile-network-operators-wso2-telco/>.
- [6] "Draw.io," [Online]. Available: <https://www.draw.io/>.
- [7] "Microsoft Visio," [Online]. Available: https://en.wikipedia.org/wiki/Microsoft_Visio.
- [8] "MySQL Workbench," [Online]. Available: <https://dev.mysql.com/doc/workbench/en>.
- [9] "XAMPP," [Online]. Available: <https://en.wikipedia.org/wiki/XAMPP>.
- [10] "PhpStorm," [Online]. Available: <https://www.jetbrains.com/phpstorm/>.
- [11] "Git," [Online]. Available: <https://git-scm.com/>.
- [12] S. Chacon, Pro Git, Apress Media, 2009.
- [13] "Laravel," [Online]. Available: <https://laravel.com/docs/5.5>.
- [14] M. Stauffer, "Laravel Book," in *Laravel: Up and Running: A Framework for Building Modern PHP Apps*, O'Reilly Media, 2016.
- [15] "Bootstrap 3," [Online]. Available: <https://getbootstrap.com/docs/3.3/>.
- [16] "jQuery," [Online]. Available: <https://jquery.com/>.
- [17] D. Dissanayake, "Google form of evaluation," [Online].
Available: <https://docs.google.com/forms/d/e/1FAIpQLSex22UCWc3mUPqhSKjxTNnImvIjdTK9fi7G4zLr98hWX1NsdA/viewform>.
- [18] "Selenium," [Online]. Available: <https://www.seleniumhq.org>.
- [19] "Nagios," [Online]. Available: <https://www.nagios.org/>.

Appendix

A: Download and setup the application

REST API Middleware can be checked out at GitHub following link

<https://github.com/dasun4u/rest-api-middleware>

The binary releases of the product can be download from the following link

<https://github.com/dasun4u/rest-api-middleware/releases>

After downloading the product then setup in apache server with PHP and MySQL. Then download and setup composer using this link <https://getcomposer.org/>. Then go to the root folder and run “composer install” command. Then it will download all the dependency packages.

Then create a new “.env” file by coping “.env.example” file. Then setup the DB connection and do the configuration in that file. Then run “php artisan migrate”. This command will create all the tables in the database. Then you should change the database seed files inside “database/seeds” folder and run “php artisan db:seed” command to insert default data to the database tables.

B: The Use Case Scenario

Use Case number	6
Use Case Name	Manage Service
Purpose	Manage Service in the system
Priority	High
Actors	Admin, Developer
Precondition	The user must log in to the system
Basic Flow	<ol style="list-style-type: none"> 1. Create Service <ol style="list-style-type: none"> 1. User select “SERVICES” menu on the side menu. 2. Click on “CREATE NEW SERVICE” button. 3. Fill the service creation form. 4. Click on “CREATE” button. 5. Validate the form data. 6. Save new service data. 2. Show Service <ol style="list-style-type: none"> 1. User select “SERVICES” menu on the side menu. 2. Click on created service show button in the list. 3. Show service data. 3. Update Service <ol style="list-style-type: none"> 1. User select “SERVICES” menu on the side menu. 2. Click on created service edit button in the list. 3. Fill the service update form. 4. Click on “UPDATE” button. 5. Validate the form data. 6. Update service data. 4. Delete Service <ol style="list-style-type: none"> 1. User select “SERVICE” menu on the side menu. 2. Click on “DELETE” button of the service in the list. 3. Show delete confirmation message 4. Click on “DELETE” Button 5. Service delete from the system
Alternative Flows	<ol style="list-style-type: none"> 1.5.5 Validation errors in the form. 1.5.6 Validation errors are shown in the create form. 3.5.5 Validation errors in the form. 3.5.6 Validation errors are shown in the update form.
Postcondition	-

Table 8 - Use Case 6

Use Case number	7
Use Case Name	Manage Subscription
Purpose	Manage Subscription in the system
Priority	High
Actors	Admin, Developer
Precondition	The user must log in to the system
Basic Flow	<ol style="list-style-type: none"> 1. Create Subscription <ol style="list-style-type: none"> 1. User select “SUBSCRIPTION” menu on the side menu. 2. Click on “CREATE NEW SUBSCRIPTION” button. 3. Fill the subscription creation form. 4. Click on “CREATE” button. 5. Validate the form data. 6. Save new subscription data. 2. Show Subscription <ol style="list-style-type: none"> 1. User select “SUBSCRIPTION” menu on the side menu. 2. Click on created subscription show button in the list. 3. Show subscription data. 3. Update Subscription <ol style="list-style-type: none"> 1. User select “SUBSCRIPTION” menu on the side menu. 2. Click on created subscription edit button in the list. 3. Fill the subscription update form. 4. Click on “UPDATE” button. 5. Validate the form data. 6. Update subscription data. 4. Delete Subscription <ol style="list-style-type: none"> 1. The user selects “SUBSCRIPTION” menu on the side menu. 2. Click on “DELETE” button of the subscription in the list. 3. Show delete confirmation message 4. Click on “DELETE” Button 5. Subscription delete from the system
Alternative Flows	<ol style="list-style-type: none"> 1.5.7 Validation errors in the form. 1.5.8 Validation errors are shown in the create form. 3.5.7 Validation errors in the form. 3.5.8 Validation errors are shown in the update form.
Postcondition	-

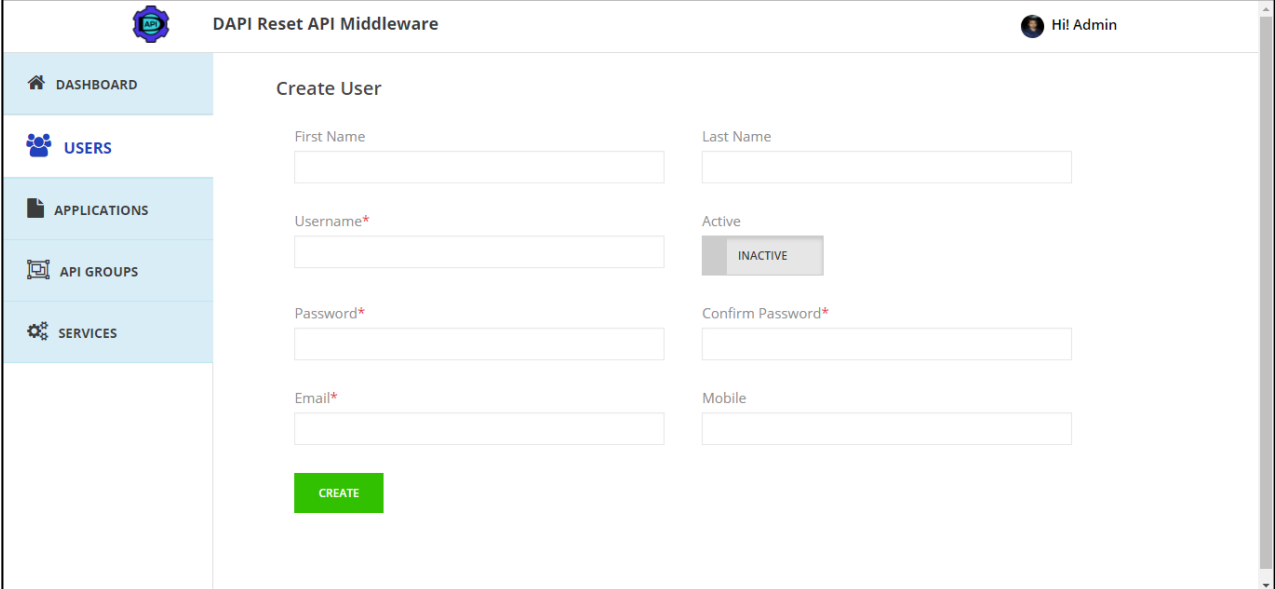
Table 9 - Use Case 7

Use Case number	8
Use Case Name	Send Messages
Purpose	Communication between the user accounts in the system
Priority	Medium
Actors	Admin, Developer
Precondition	The user must log in to the system
Basic Flow	<ol style="list-style-type: none"> 1. Create Message <ol style="list-style-type: none"> 1. User select “MESSAGES” menu on the side menu. 2. Click on “CREATE NEW MESSAGE” button. 3. Fill the message creation form. 4. Click on “CREATE” button. 5. Validate the form data. 6. Save new message data. 7. Send to others 2. Show Message <ol style="list-style-type: none"> 1. User select “MESSAGES” menu on the side menu. 2. Click on received message show button in the list. 3. Show message data. 3. Delete Message <ol style="list-style-type: none"> 1. The user select “MESSAGES” menu on the side menu. 2. Click on “DELETE” button of the message in the list. 3. Show delete confirmation message 4. Click on “DELETE” Button 5. Message delete from the system
Alternative Flows	<p>1.5.9 Validation errors in the form.</p> <p>1.5.10 Validation errors are shown in the create form.</p>
Postcondition	-

Table 10 - Use Case 8

C: Implementation of the user interface

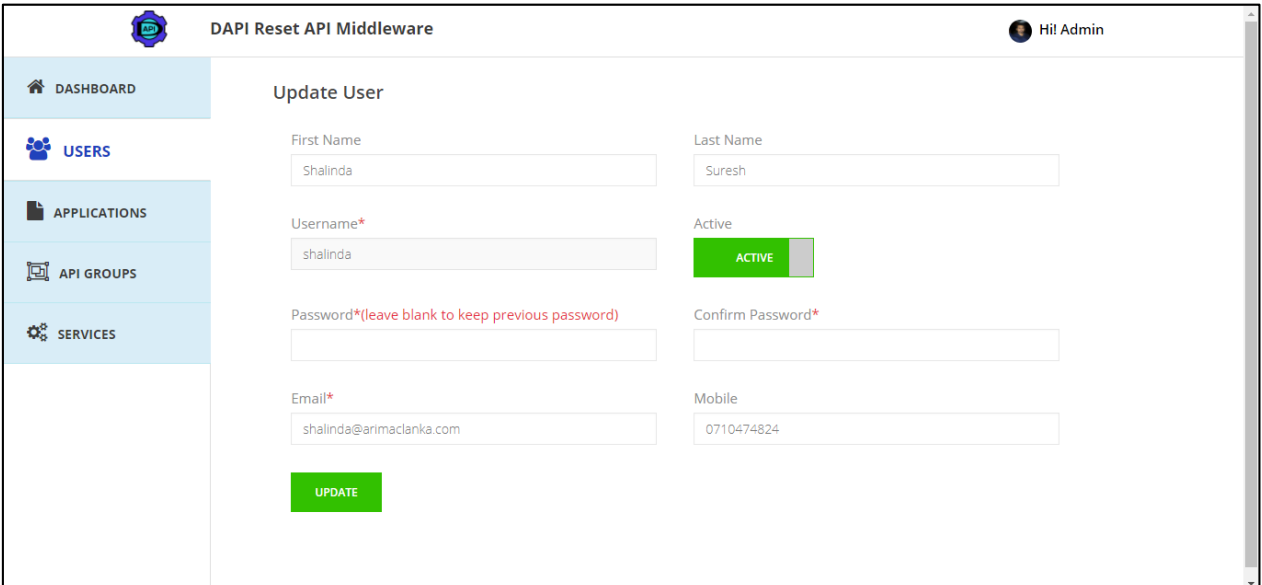
Admin user creation page is shown in Figure 25. Admin can create a developer user using “Create User” page.



The screenshot shows the 'Create User' page in the DAPI Reset API Middleware admin interface. The page has a sidebar with navigation options: DASHBOARD, USERS, APPLICATIONS, API GROUPS, and SERVICES. The main content area is titled 'Create User' and contains several input fields: First Name, Last Name, Username*, Password*, Confirm Password*, Email*, and Mobile. There is also an 'Active' toggle switch currently set to 'INACTIVE'. A green 'CREATE' button is located at the bottom of the form.

Figure 25 - User create view of Admin

Admin can update created users by using the page in Figure 26. Password reset also can do this page. If the admin does not need to change the user password then he can leave it blank and submit.



The screenshot shows the 'Update User' page in the DAPI Reset API Middleware admin interface. The page has a sidebar with navigation options: DASHBOARD, USERS, APPLICATIONS, API GROUPS, and SERVICES. The main content area is titled 'Update User' and contains several input fields: First Name (Shalinda), Last Name (Suresh), Username* (shalinda), Password*(leave blank to keep previous password), Confirm Password*, Email* (shalinda@arimaclanka.com), and Mobile (0710474824). There is also an 'Active' toggle switch currently set to 'ACTIVE'. A green 'UPDATE' button is located at the bottom of the form.

Figure 26 - User update view of Admin

Application list view is shown in Figure 27.







#	Name	Description	Created By	Created Time	Active	Approve	Options
1	First APP	First Application for REST API Middleware	shalinda	2018-03-18 19:08:48	ACTIVE	APPROVED	  
2	Second APP	Second Application for REST API Middleware	shalinda	2018-03-18 19:08:48	INACTIVE	NOT APPROVED	  

Figure 27 - Application list view of Admin

Service group list view is shown in Figure 28







#	Name	Context	Created Time	Active	Options
1	API Group 1	group1	2018-03-18 19:08:49	ACTIVE	  
2	API Group 2	group2	2018-03-18 19:08:49	ACTIVE	  

Figure 28 - Service group list view of Admin

Service list in Figure 29.

#	Name	Service Group	Method	Created Time	Active	Options
1	Service 1	API Group 1	GET	2018-03-18 19:08:49	ACTIVE	NOT APPROVED, View, Edit, Delete
2	Service 2	API Group 1	POST	2018-03-18 19:08:49	ACTIVE	APPROVED, View, Edit, Delete
4	Service 4	API Group 2	POST	2018-03-18 19:08:49	ACTIVE	APPROVED, View, Edit, Delete
5	Test 123	API Group 1	GET	2018-03-18 20:16:07	INACTIVE	NOT APPROVED, View, Edit, Delete

Figure 29 - Service list view of Admin

A confirmation message is shown in Figure 30.

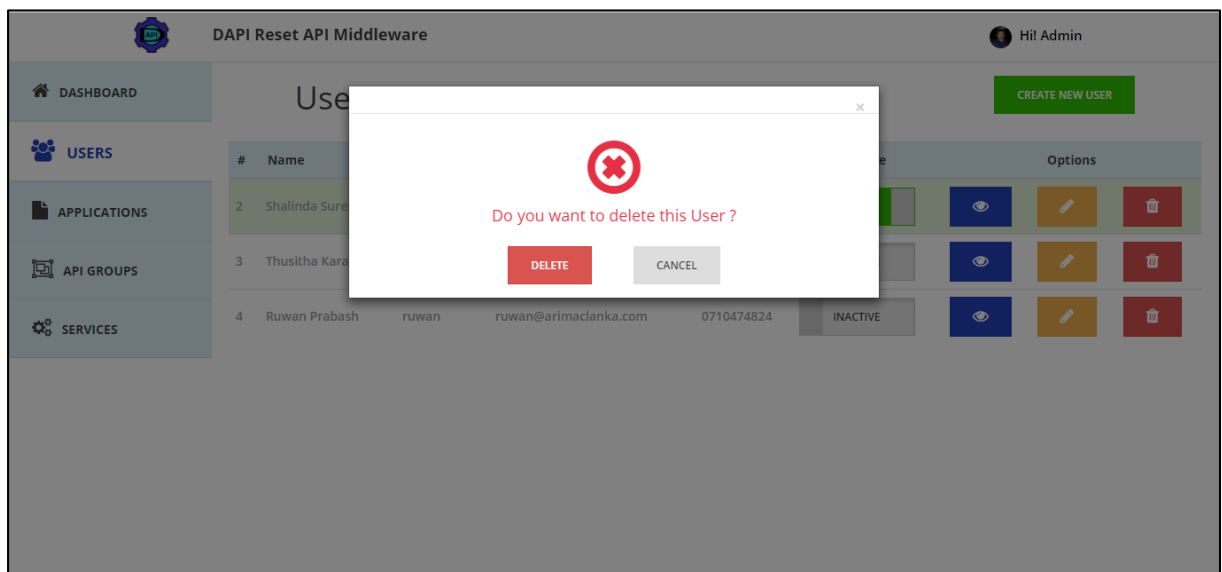


Figure 30 - Confirmation Modal