RESILIENT ASSESSMENT DASHBOARD

BY

Anurak Jittang 62011091

Chanvee Phittayavanitch 62011102

Phaotong Pongpamorn 62011201

Natchapol Watthanawong 62011302

A PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF BACHELOR OF ENGINEERING IN COMPUTER INNOVATION

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

ACADEMIC YEAR 2022

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

PROJECT CERTIFICATE

Project Title	RESILIENT ASSESSMENT DASHBOARD
Student Name	Mr Anurak Jittang Student ID. 62011091 Mr Chanvee Phittayavanitch Student ID. 62011102 Mr Phaotong Pongpamorn Student ID. 62011201 Mr Natchapol Watthanawong Student ID. 62011302
Degree	Bachelor of Engineering in Computer Innovation
Project Advisor	Signed: Asst. Prof. Dr. Orathai Sangpetch ( )

Project Title	RESILIENT ASSESSMENT DASHBOARD
Student Name	Mr Anurak Jittang
	Student ID. 62011091
	Mr Chanvee Phittayavanitch
	Student ID. 62011102
	Mr Phaotong Pongpamorn
	Student ID. 62011201
	Mr Natchapol Watthanawong
	Student ID. 62011302
Degree	Bachelor of Engineering in Computer Innovation Engineering
Project Advisor	Asst. Prof. Dr. Orathai Sangpetch
Academic Years	2022

ABSTRACT

Resilience assessment dashboard's goal is to create a user friendly interface for the existing as well as new functionalities of the chaos injection process. We designed the UI from the ground up using Figma and then developed it in iterations with user feedback. More backend APIs were also developed in the need of user requirement and new dashboard design, such as - scheduling a task, getting the response time graph and more. Additionally, the frontend was developed in code using React, making individual components reusable and the code base easier to maintain. The end result satisfies the problems of poor user interface design to keep track of chaos injection and its information, no feature of scheduling of tests, lack of configuration functionalities, and clunky user experience. With our good design practices such as modular components and testing and reducing time complexity with tree data structure for scheduling, the resilience assessment dashboard can be developed and maintained further to be used at scale in the future.

ACKNOWLEDGEMENTS

We would like to express my sincere gratitude to all those who have supported and guided us throughout the completion of this project report. It is with immense pleasure that we acknowledge the individuals who have contributed to the successful development of this project.

First and foremost, we would like to extend my heartfelt appreciation to my project advisor, Orathai Sangpetch, for her invaluable guidance, constant encouragement, and insightful feedback. Her expertise and unwavering support have been instrumental in shaping this project and ensuring its successful execution. We are truly grateful for her patience and dedication in mentoring me throughout this endeavor.

We would also like to extend my gratitude to my team members, Anurak, Chanavee, Phaotong, and Natchapol. Their hard work, cooperation, and collaborative spirit have been vital in accomplishing our shared goals. Each team member's unique skills and expertise have contributed to the development of the user-friendly dashboard and the seamless integration of APIs in the backend. Their commitment and enthusiasm have made this project a truly collaborative and enriching experience.

Additionally, we would like to thank all the individuals who provided assistance, advice, and support during the course of this project. Their contributions, whether big or small, have played a significant role in its completion. The collective effort and teamwork demonstrated by everyone involved have made this project a resounding success.

In conclusion, we extend my heartfelt appreciation to my project advisor, Orathai Sangpetch, and the team members, Anurak, Chanavee, Phaotong, and Natchapol, as well as all the individuals who have contributed to the completion of this project. Your guidance, collaboration, and support have been instrumental, and we are truly grateful for the opportunity to work with such remarkable individuals.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS/ABBREVIATIONS

Symbols/Abbreviations	Terms
CIE	Computer Innovation Engineering
SIIE	School of International Interdisciplinary Engineering Programs
UI	User Interface
UX	User Experience
CSS	Cascading Style Sheets
JSON	JavaScript Object Notation
HTTP	Hypertext Transfer Protocol
API	Application Programming Interface
VM	Virtual Machine
[redacted]	Confidential contents

CHAPTER 1
INTRODUCTION

1.1 Background

The Resilient Assessment Dashboard project aims to provide users with a seamless and user-friendly platform for managing and monitoring tests and applications. Consisting of developments on both frontend and backend, this project focuses on the development of a web application with an aesthetic and intuitive user experience while also having a reliable API and scalable backend infrastructure. The API and components are reliable by thoroughly testing and scalable using tree data structure for scheduling.

Chaos engineering is the process of testing a system's resiliency. These tests may include resource exhaustion such as CPU percentage test or increasing errors to a network such as packet loss and more. The primary objective of the Resilient Assessment Dashboard is to empower users with the ability to execute, monitor, and customize these tests intuitively by using good design principles, having function as the priority of the user experience. Users can submit existing tests as well as scheduling them to a targeted time to run. With real time monitoring, users can also see the metrics of each tested endpoint and stop the tests during their execution with ease. Of course, these features are displayed appropriately with real time visibility updates for both the scheduled tests and finished tests. The running tests update display works in conjunction with device monitoring upon injection. Lastly, the configuration page allows users to create customized tests and vms to be used. These dynamic features allow users to actively control and monitor their tests, ensuring optimal performance and the ability to quickly respond to changing requirements.

1.2 Scope

The purposes of this project are as follows:

1. Allow user to submit or stop test during current running test and show current test
2. Able to have the frontend work as expected with the backend APIs and have a seamless UX experience
3. Be able to schedule tests to run ahead of time
4. Show the running test and application health
5. Display the tests results in real time
6. Allow user to configure the endpoint machine in the system
7. Allow user to create a new custom test
8. Allow user to attach tests to an endpoint machine

1.3 Timeline

	January				February				March					April				May			
Week	1	2	3	4	1	2	3	4	1	2	3	4	5	1	2	3	4	1	2	3	4
UX/UI Design																					
Development																					
Testing																					
Debugging																					

1.4 Report Outline

The rest of this report is organized as follows:

- ☐ Chapter 2 reviews the concepts in which we applied in the project.
- ☐ Chapter 3 describes the design and implementation of concepts mentioned in Chapter 2.
- ☐ Chapter 4 demonstrates how we tested the implementation described in Chapter 3.
- ☐ Chapter 5 is the conclusion of this project.

CHAPTER 2

REVIEW OF CONCEPTS, THEORIES AND RELATED RESEARCH

The frontend section of this documentation covers the client-side components of the application, including the UI and interactions between the UI and the backend API.

The backend section of this documentation provides an overview of the backend components of the project, including the server-side technologies and tools that we are using to build and run the application.

2.1 Frontend

2.1.1 UX/UI Design

2.1.1.1 Personalities

According to the article "The Rounded User Experience" by Nick Babich, the border-radius, which refers to the curvature at the corners of a design, plays a significant role in shaping the emotional impact of the design (Babich, n.d.). Designs with little to no border-radius tend to convey a serious and formal ambiance. In contrast, increasing the border-radius introduces a sense of playfulness into the design, influencing the overall user experience. This was used in every component of the web application. A serious format is described in figure 2.1 while playful style is shown in figure 2.2.

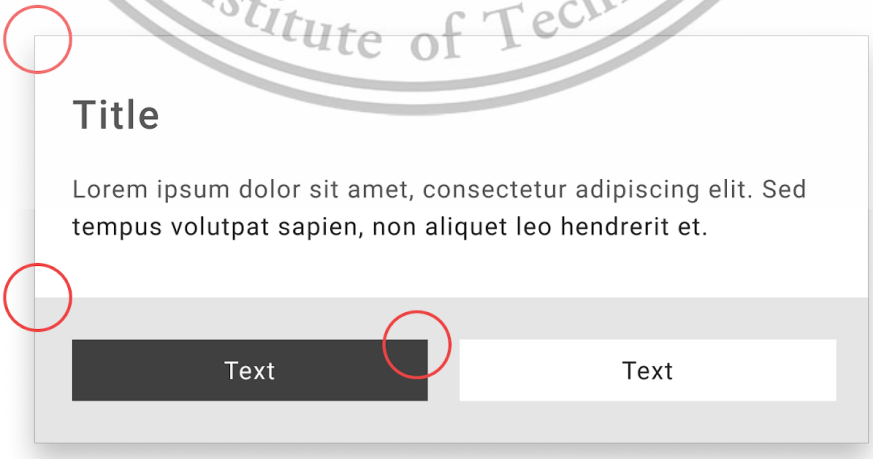

Figure 2.1 No border-radius pop-up

Figure 2.2 Pop-up with border-radius

Each font family is purposefully designed with specific considerations in mind. Fonts typically used for headlines generally feature tighten letter spacing and shorter lowercase letter height. Conversely, body fonts are designed with broader letter spacing and taller lowercase letters. These distinct features contribute to the readability and aesthetics of different sections of text which the comparison is shown in figure 2.3.

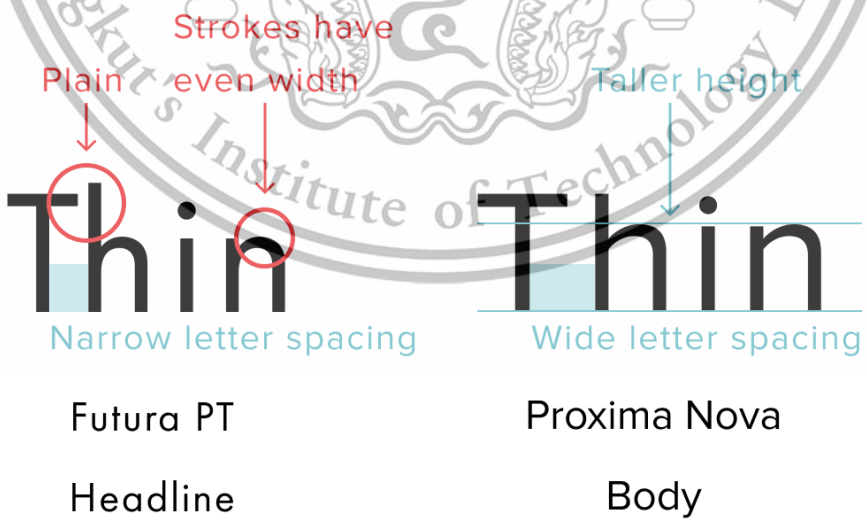

Figure 2.3 Headline and body fonts comparison

2.1.2.2 Hierarchy

According to the Interaction Design Foundation, visual design principles play a crucial role in establishing a visual hierarchy in user interfaces, similar to how we layer clothing to create a stylish ensemble. The concept of visual hierarchy determines the order in which elements are presented in relation to each other within the interface. When a UI becomes overloaded with ideas and information, it can resemble a chaotic and disorderly environment. In such cases, every component competes for the user's attention, leading to an overwhelming experience. The visual hierarchy point of view is depicted in figure 2.4. This visual clutter can obscure the content and make it challenging to discern the most important information. Thus, visual hierarchy is vital in UI design to ensure clarity and maintain focus on the essential elements (Interaction Design Foundation, n.d.).

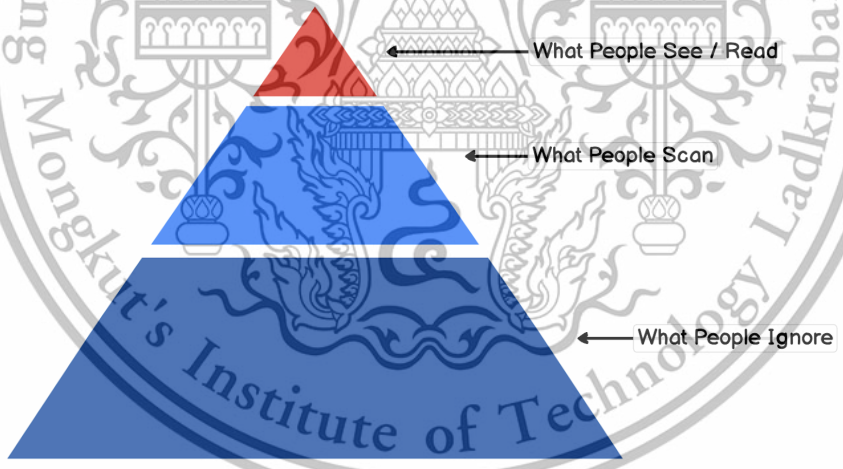

Figure 2.4 Visual hierarchy from article “How to Use Visual Hierarchy and Alignment to Improve UI Design: “Visual Hierarchy.” *Https://Balsamiq.Com/Learn/Articles/Visual-Hierarchy-and-Alignment/*.

It's essential to ensure that a set of components is differentiated by a border or background colors, clearly indicating which group they belong to. This becomes particularly important in the absence of a distinct, visible separator. For instance, when designing and labeling form input components

becomes challenging, margins can be a useful tool. If the margin beneath the label matches the space below the input, it could create a group of components that lacks a sense of connection and contradicts the principles of hierarchy which is described in figure 2.5 while a clear separation is appeared in figure 2.6.

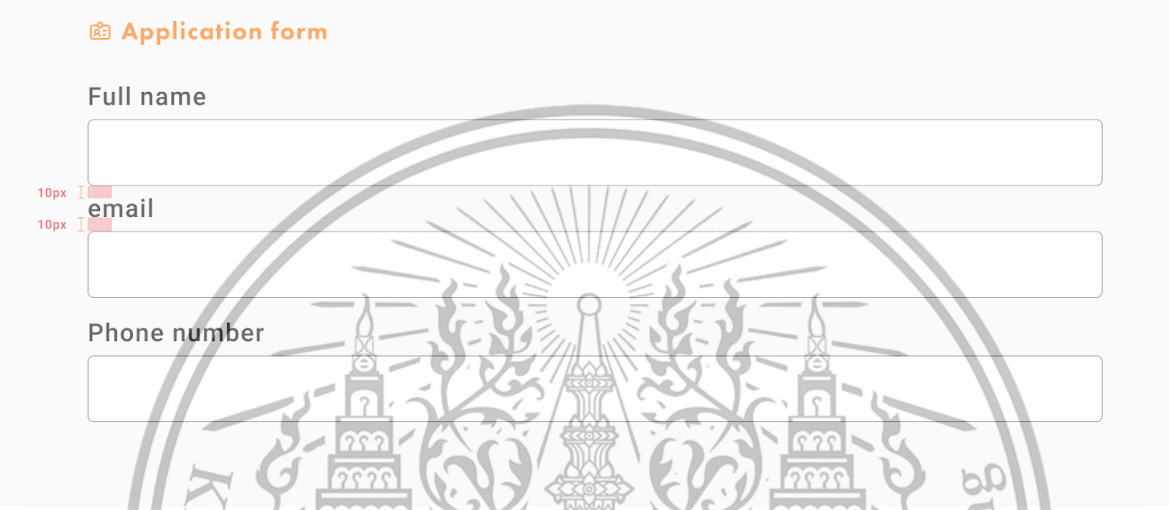

Figure 2.5 Symmetric Spacing

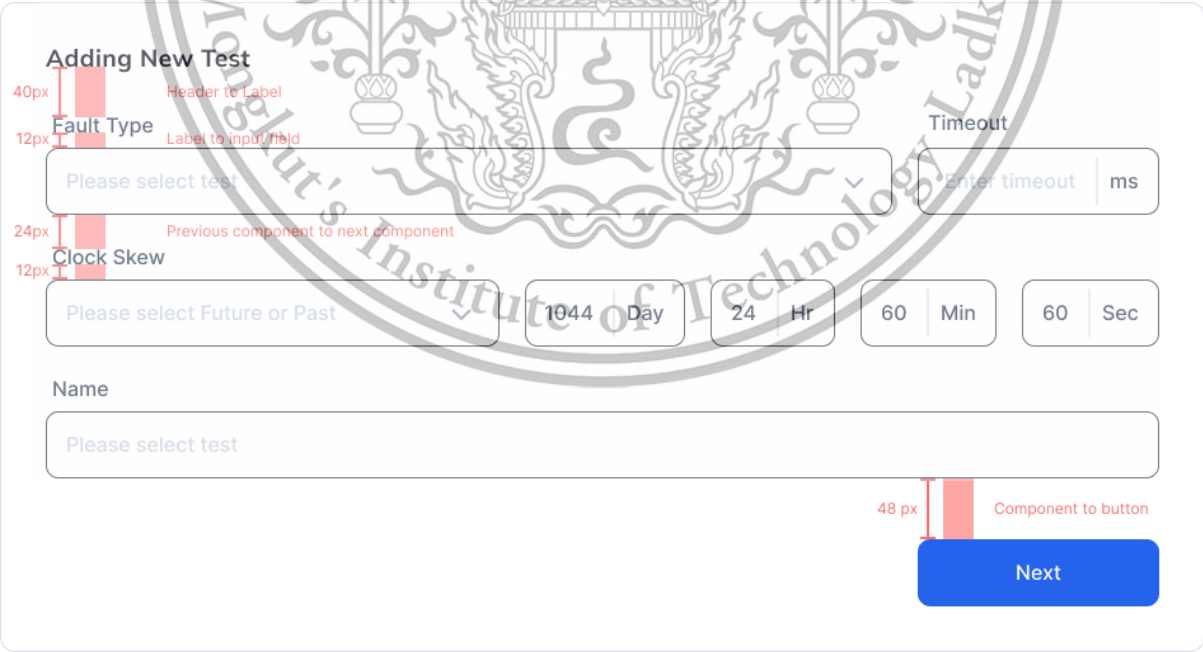

Figure 2.6 Asymmetric spacing

When it comes to text alignment, a common approach is to center-align text vertically. However, this method may not always yield the most aesthetically pleasing results and can potentially create awkward visual moments for the reader. A more effective strategy is to align to the baseline of mixed font sizes. This approach gives the reader a visually superior alignment reference, enhancing the overall reading experience and comparison is shown in figure 2.7.

Figure 2.7 Center and baseline align text comparison

When designing tables that incorporate numerical data, it's advisable to right-align the numbers. This is particularly useful in the case of decimal figures, as the right alignment facilitates easier comparison between these numbers. This uniformity ensures that decimal points across various numbers align perfectly, enhancing readability and comparison displayed in figure 2.8.

Number
12345.148
5460.12
790.01
985564.9
19

Number
12345.148
5460.12
790.01
985564.9
19

Figure 2.8 Left and right align number comparison

2.1.3 Frontend technology stacks

2.1.3.1 ReactJS

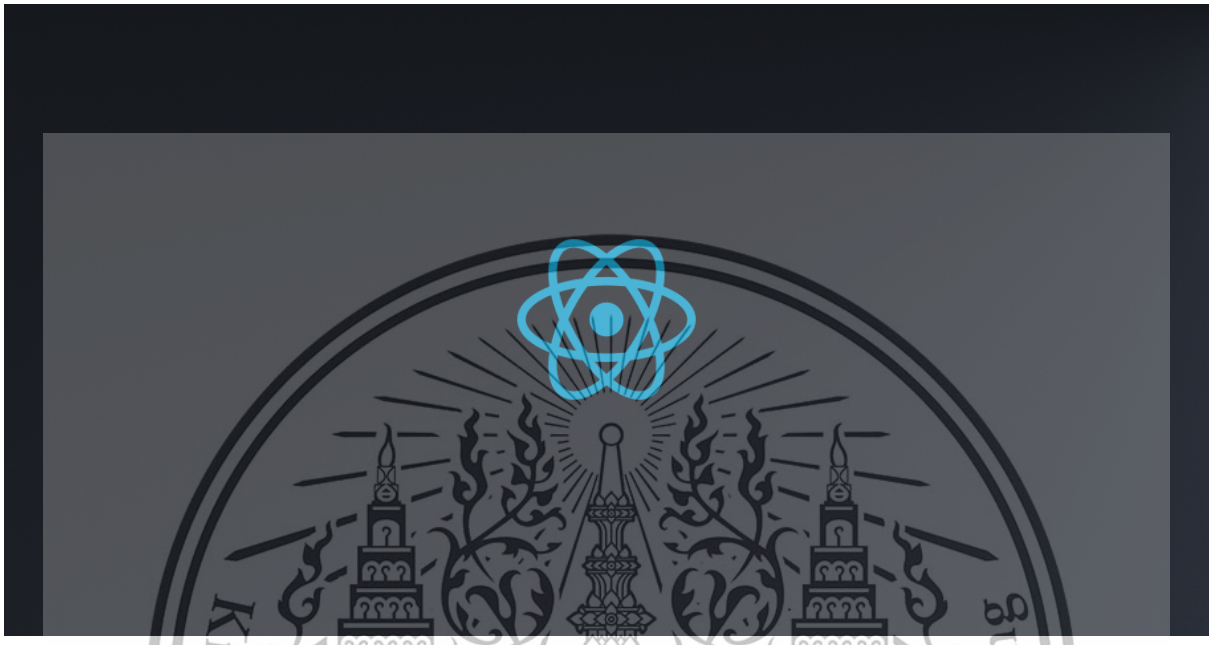

Figure 2.9 (ref:https://react.dev/)

For the frontend development. We use ReactJS as our main framework. ReactJS is an open-source JavaScript framework that allows us to create a component for our web application and provides some tools to help us to create logical functions from the frontend.

2.1.3.2 AxiosJS

Figure 2.10 (ref: https://commons.wikimedia.org/wiki/File:Axios_%28computer_library%29_logo.svg)

For the HTTP requests tools. We use AxiosJS as our main HTTP client. Axios provides the object response data while fetch provides string data that needs to be stringified before being used. In summary, AxiosJS provides more convenience and developer friendly features than normal fetch data.

2.2 Back End

2.2.1 Scheduling concept

We used the binary tree data structure for scheduling for efficiency (as it is faster than regular array structure at scale). The implementation and flow diagram will be discussed in Chapter 3.

2.2.2 Endpoint Configuration concept

We have decided not to directly access the Mangle Controller from the frontend. Instead, we have implemented a system where APIs are created in the chaos-keyspace-controller to manage the endpoints. All the logic pertaining to endpoint management will be applied in the controller. For instance, we can create a new endpoint, update an endpoint group, delete an endpoint, update an endpoint group, and more using the provided APIs.

In addition, we have chosen to parallel write the endpoint data to the chaos controller database. This enables efficient management of tests performed on each endpoint, ensuring comprehensive test coverage and effective monitoring of the testing process.

2.2.3 Additional Fault/Test Configuration concept

In this specific part, two primary additional features have been introduced to the system: the ability to attach tests to an endpoint and the ability to create custom tests. In order to facilitate convenient test management from the frontend, a total of five additional APIs have been developed.

1. The first API, "get all tests," allows the frontend to retrieve a comprehensive list of all available tests within the system. This API provides an overview of all tests that can be associated with endpoints.

2. The second API, "get all test types," enables the frontend to retrieve a list of all available test types. This information is crucial for creating custom tests tailored to specific requirements.

3. The third API, "get unattached test of the selected endpoint," provides the frontend with the capability to retrieve a list of tests that are not currently attached to a selected endpoint. This allows for efficient selection and association of tests to the endpoint.

4. The fourth API, "attach tests to an endpoint," allows the frontend to select unattached tests and attach them to a specific endpoint. This API facilitates the association of tests with endpoints.

5. The fifth API, "create custom test," provides the frontend with the ability to create new custom tests. This API allows users to design and define tests that meet specific requirements or scenarios, enhancing the flexibility and customization options of the testing process.

In summary, these five additional APIs have been created to enhance the frontend's ability to manage tests effectively. By providing functionalities such as retrieving all tests, obtaining test types, accessing unattached tests for selected endpoints, attaching tests to endpoints, and creating custom tests, the system offers improved flexibility and convenience in test management processes.

2.3 Chapter Summary

Much of the theory of our project revolves around UX theory because one of our main goals of this project is to create a seamless experience for the user and therefore focus heavily on the design and aesthetics of each component. Concepts of the backend are also included such as scheduling using the data tree structure and the concept of attaching ‘tests’ to a specific endpoint to be used in the chaos test injection.

CHAPTER 3
METHODOLOGY

3.1 Introduction

The methodology section outlines the systematic approach and methods employed in the project to achieve the defined objectives. It provides a detailed description of the processes, tools, techniques, and data used during the project's lifecycle. This section aims to offer transparency and reproducibility by documenting the steps taken to develop, implement, and evaluate the solution. The following subsections outline the key components covered in the methodology section:

1. Project Planning:

The introduction provides an overview of the project planning phase. It discusses the initial project scope, objectives, and the development of a detailed project plan, including timelines, resource allocation, and task dependencies.

We did these in this step:

- Planning on what we have to do
- Getting the objective and scope of work from the project owner
- Generating Timeline for each task
- Tasks distribution for each member

2. Requirements Gathering:

This section explains the process of gathering and analyzing requirements. It discusses the techniques used to identify the functional and non-functional requirements of the system.

We did these in this step:

- Gather the requirement from the project owner

3. System Design:

The methodology section describes the system design phase, where the architectural and component-level design of the solution was developed. It outlines the design principles, methodologies, and tools used to create an effective and scalable system design.

We did these in this step:

- Designing the front-end pages including the 2 main pages and the pop-ups
- Designing the logic of each APIs in the back-end
- Designing the Database architecture that is capable of handling the data

4. Development:

This subsection focuses on the implementation and development phase of the project. It outlines the programming languages, frameworks, and technologies used to develop the frontend, backend, and any supporting modules or components. It also discusses the iterative or agile development approach employed.

We did these in this step:

- Hosting the web-app and the back-end on the server

5. Testing: The methodology section explains the testing processes employed during the project. It discusses the types of testing conducted, such as unit testing, integration testing, and system testing, along with the tools and frameworks used to ensure the quality and reliability of the developed solution.

We did these in this step:

- Creating test cases for front-end and back-end
- Testing each and every test case that has been created

6. Project Management:

This subsection discusses the project management techniques employed throughout the project, including task tracking, team coordination, and project monitoring. It outlines the project management tools and methodologies used to ensure the project's successful execution.

The methodology section provides an overview of the approach followed in the project, planning, requirements gathering, system design, development, testing, and project management processes.

We did these in this step:

- Using Trello as our project planner and tasks manager

3.2 Design Methodology

3.2.UI Prototype

In the context of table design, the table is crafted to display values that are both informative and readily discernible. The design underscores the ease with which users can quickly scan the table's contents. This intuitive design allows users to instinctively locate the data they're seeking as shown in figure 3.1.

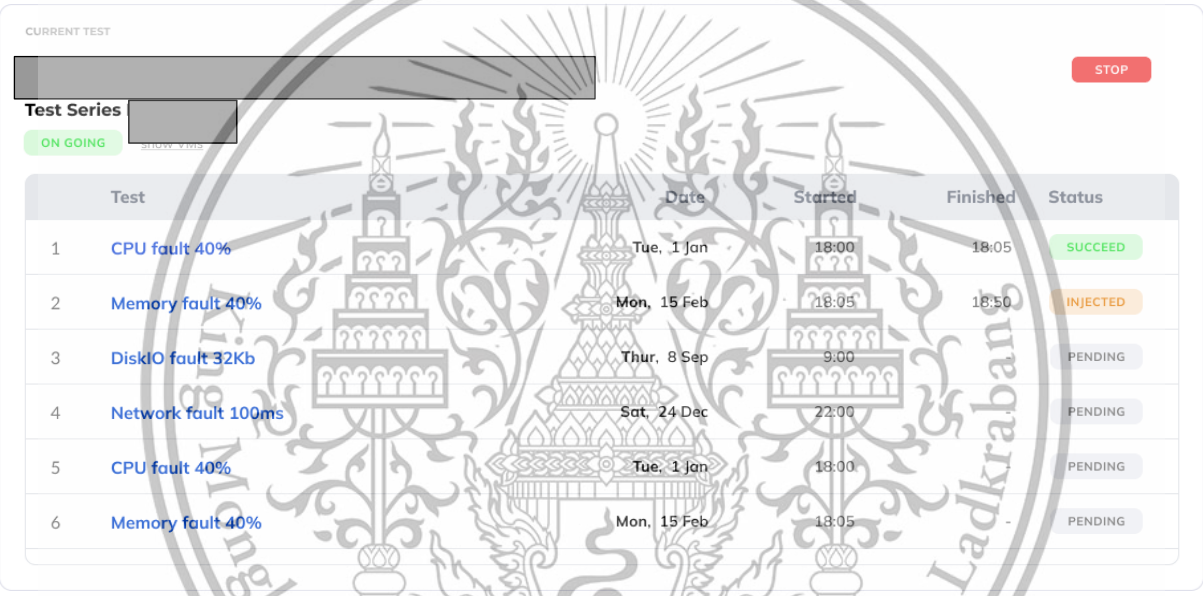

Figure 3.1 Table design

According to the numerical typeface on the table, they are right aligned due to the uncertain volume of number theory as displayed in figure 3.2.

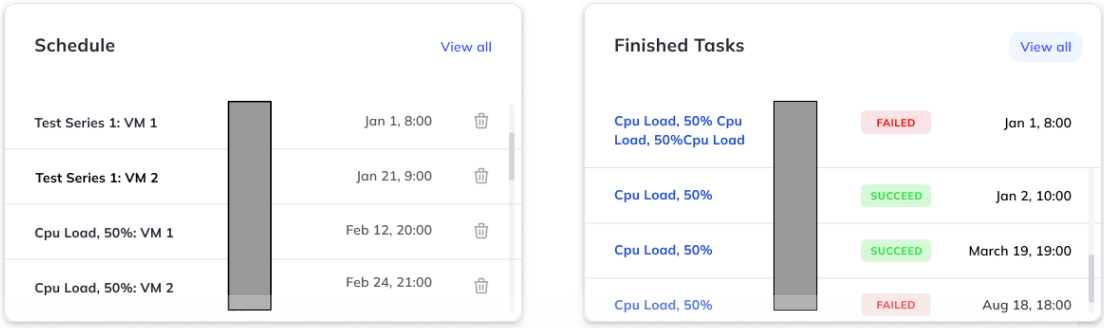

Figure 3.2 Past components

Our Manage test page is designed according to the time period hierarchy. From past, present to future consecutively. Since humans are looking from top to bottom such as figure 3.3, organizing the features hierarchy is an essential aspect of creating a great UX.

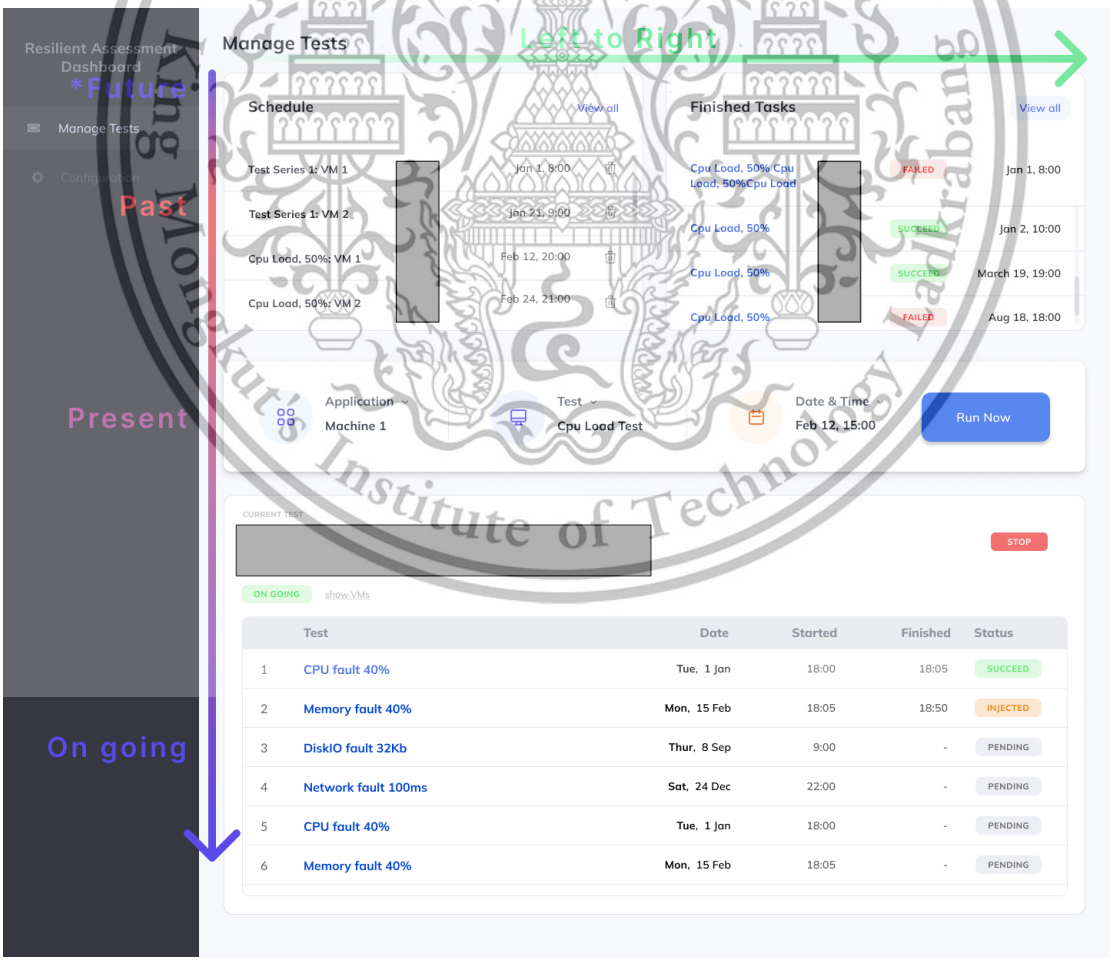

Figure 3.3 Organized components

Every typeface has its unique characteristics and is designed with a specific purpose in mind. Headline fonts generally exhibit tighten letter spacing and shorter lowercase letter heights, while body fonts are characterized by wider letter spacing and taller lowercase letters. The primary typefaces selected for this project are Futura PT and Mulish. Futura PT, with its narrow letter spacing and even stroke width, is an ideal choice for distinct and compelling headlines. On the other hand, Mulish, with its minimalist design, serves the dual purpose of display and text typography. Both are sans serif fonts, hence they present a less ornate effect than typical serif fonts as depict in figure 3.4. This aligns well with the dashboard's technical nature which requires less decorative flair and more emphasis on user-friendly interface design.

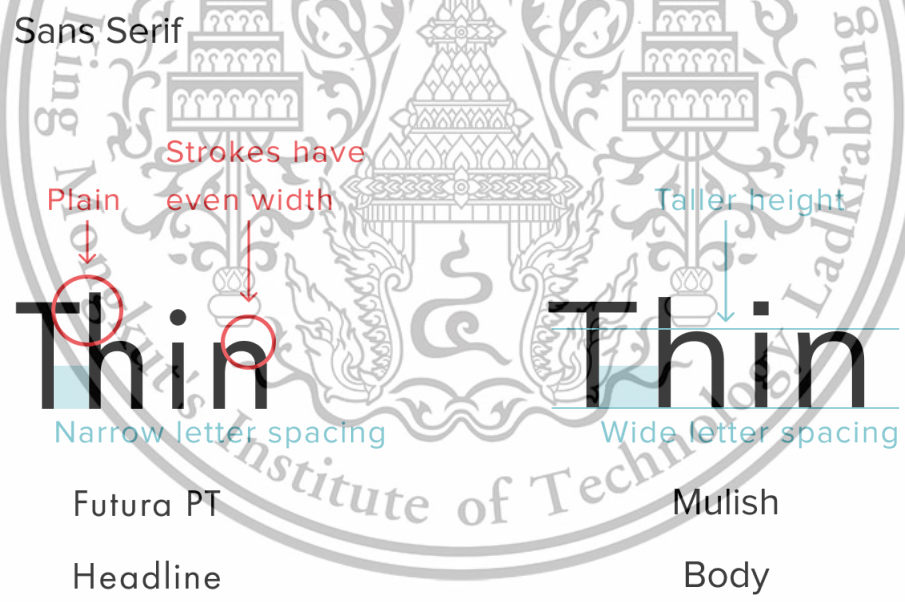

Figure 3.4 San Serif font type

Other typefaces like Inter and Lato that are used in this dashboard are specifically applied on occasional text due to optimizing font and de-emphasizing techniques

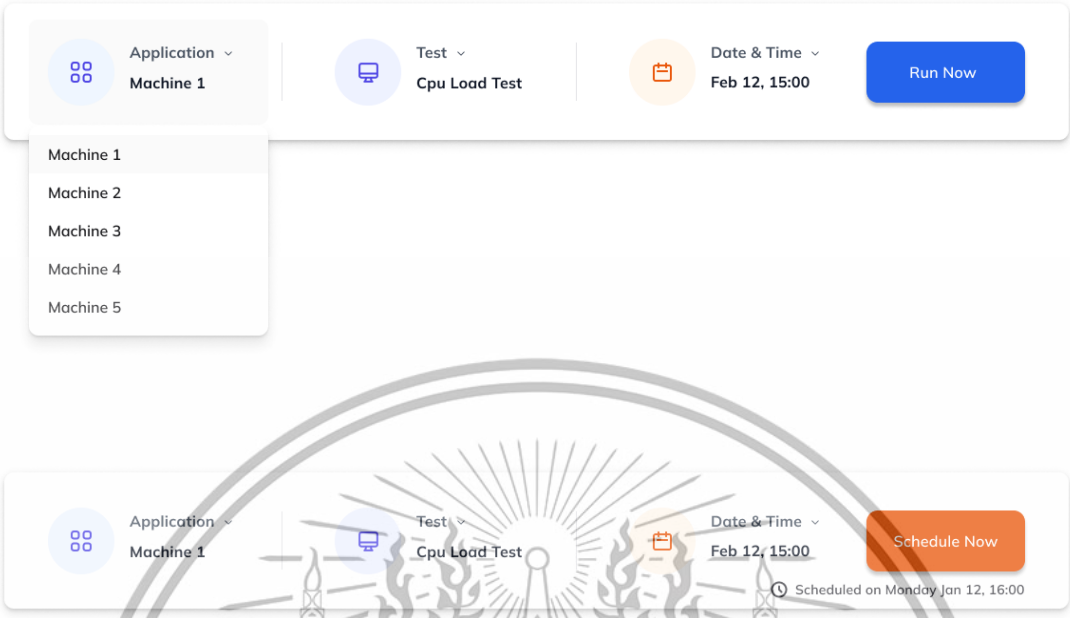

Figure 3.5 Test selection bar

The dominant color chosen for the UI is primary blue no.600, as defined in the design system, or 2563EB in hexadecimal notation. This shade symbolizes a formal and secure appearance, fitting for a technically functional feature. Conversely, the date and time selection and schedule buttons are rendered in an orange theme which both are shown in figure 3.5. This distinct colouration serves to alert the user that these actions differ from the primary "run now" command.

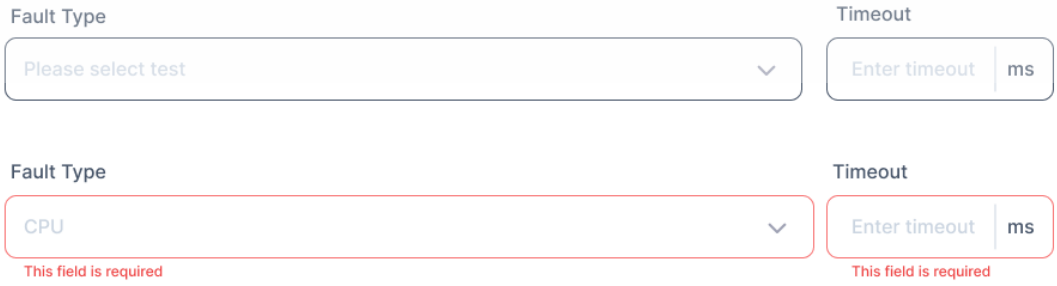

Figure 3.6 Rounded border input field

In terms of creating a user-friendly UI, employing a slightly rounded border-radius adds a playful touch to the design, contributing to its overall user-friendly appeal. However, it is important to strike a balance between playfulness and formality. To achieve this, the border-radius is maintained within the range of 8-12px that are proportional to each other by eyes as in figure 3.6. Since it is varied in appearance, choosing the one that suits its look is better.

Figure 3.7 Baseline alignment of different text size

The positioning of typefaces in the design is carefully aligned with their respective baselines, even when they differ in size. This alignment minimizes contrast and enhances the user-friendliness of the user interface also shown in figure 3.7. This approach is not limited to text alignment alone but also extends to numerical typefaces. In the case of numbers, they are right-aligned to facilitate easy comparison of digits, considering their unpredictable range of values which is compared in figure 3.8.

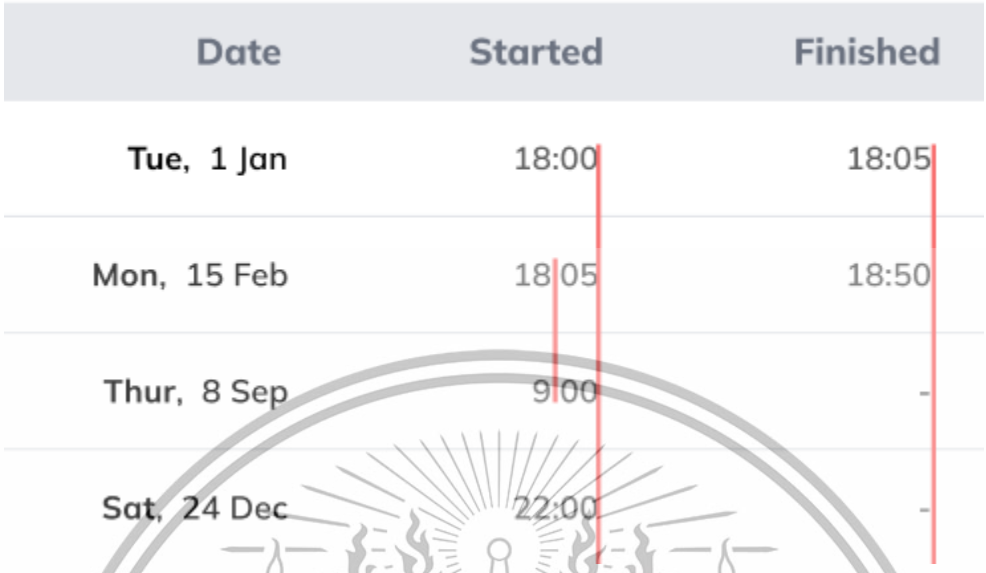

Figure 3.8 Numbers are right aligned

Having variations in digits can potentially confuse users and require more time for them to comprehend and compare numbers. This can hinder the ease of reading and make it more challenging for users to quickly and accurately compare numbers within the same column.

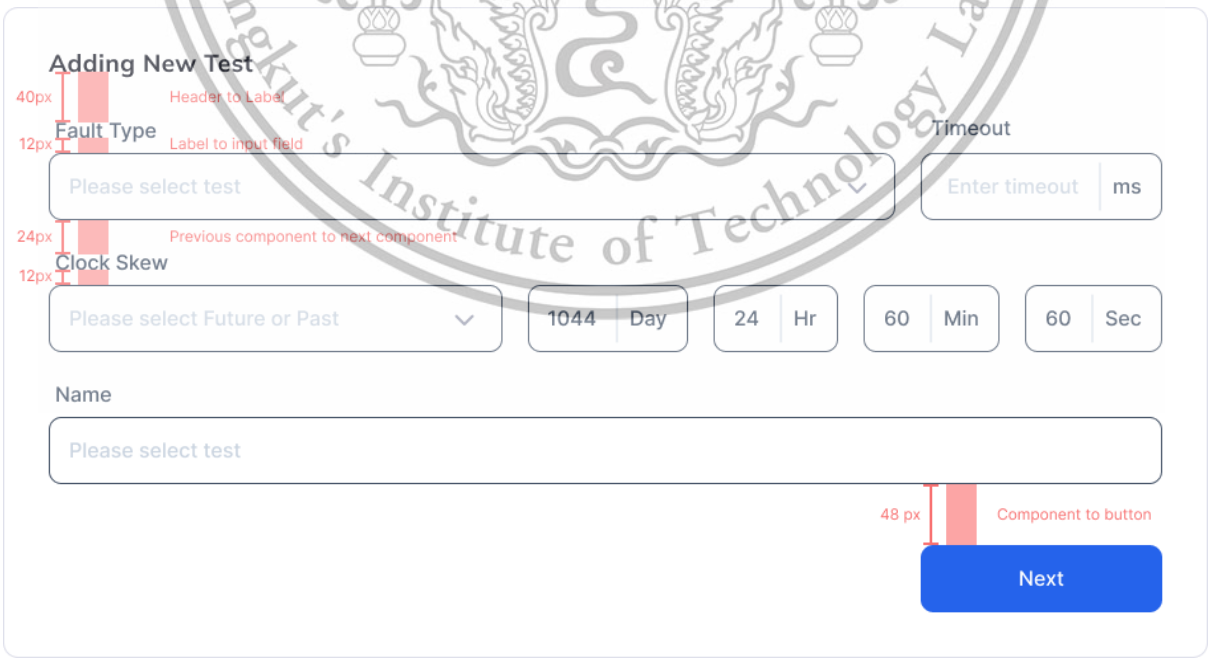

Figure 3.9 Asymmetric spacing

It is crucial to establish clear differentiation among groups of components through the use of borders or background colors. This becomes especially important when there is no visible separator to demarcate these groups. In situations where labeling form input components poses challenges, margins can be employed effectively. By aligning the margin below the label with the space beneath the input, it ensures a cohesive grouping of components, avoiding any disconnect and maintaining a consistent hierarchy as in figure 3.9.

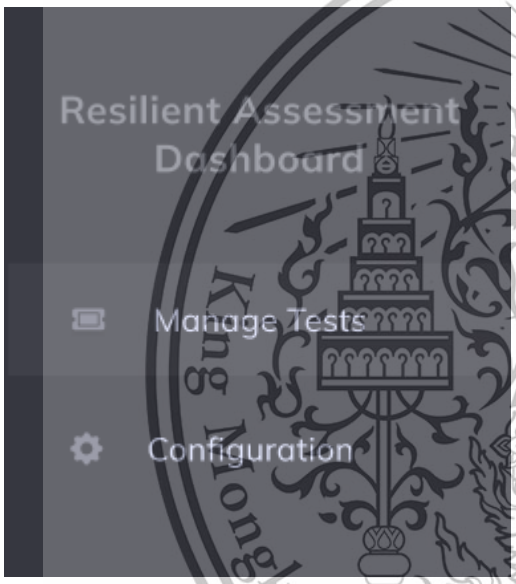

Figure 3.10 Non stand-out active

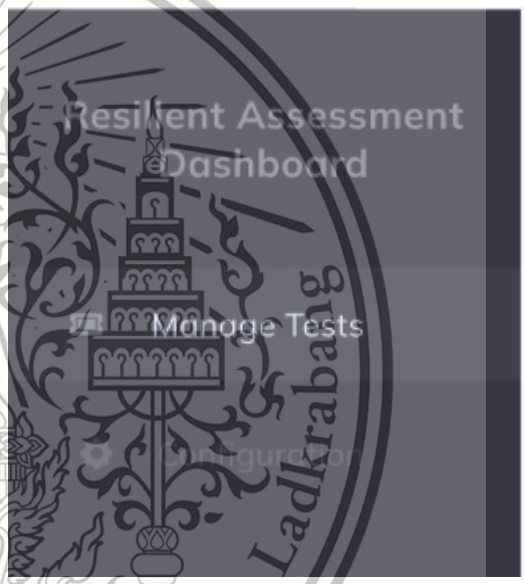

Figure 3.11 Stand-out active

Sometimes, it is challenging to effectively highlight a crucial element in the interface when additional emphasis doesn't yield the desired outcome. For example, simply coloring an active tab may not be enough to make it stand out from the inactive tabs in figure 3.10. In such cases, a de-emphasizing strategy can be beneficial. By reducing the contrast of the inactive tabs and adjusting their color to blend with the background, the primary component can receive more attention in figure 3.11. Unlike the traditional method of emphasizing elements through bold text, de-emphasizing less important components can be more effective in certain situations. This subtle approach creates a noticeable impact and directs focus towards the critical elements.

3.3. Frontend

3.3.1 Class Diagram

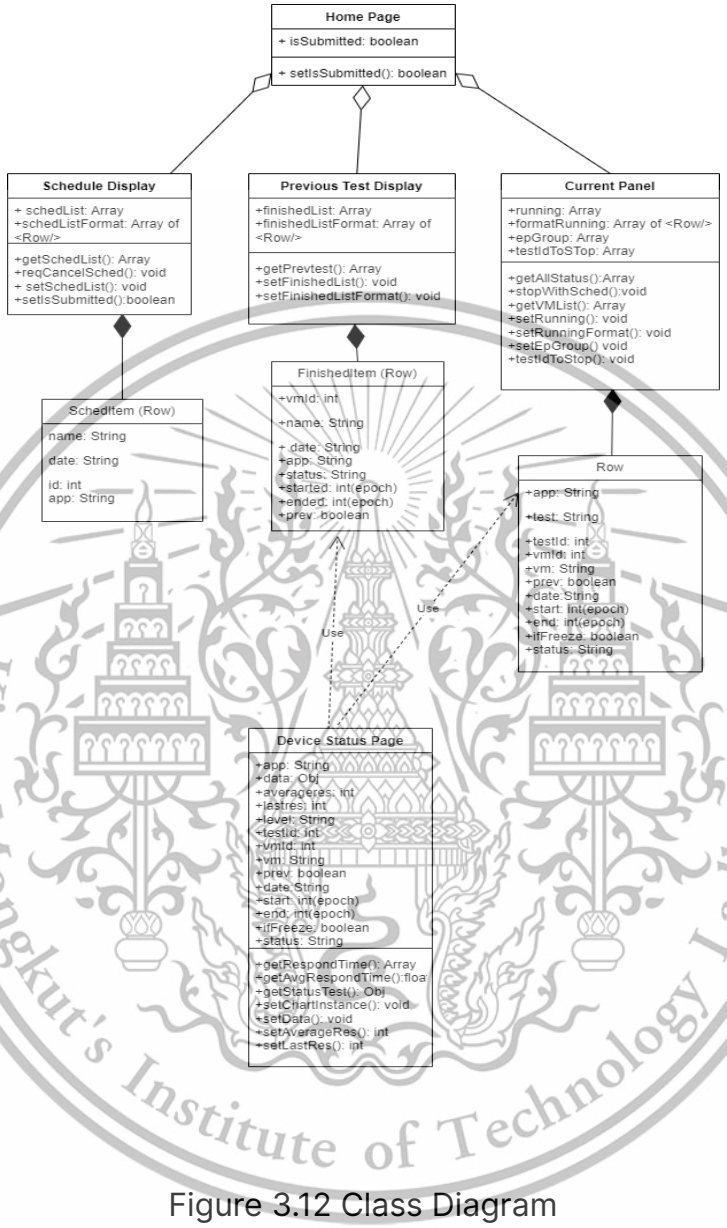

Figure 3.12 Class Diagram

Figure 3.12 shows attributes, methods, and relationships of each main component. Schedule display, previous test display, and current panel component aggregates with the home page component. The rows component is a composition of the corresponding parent, each component will have a row component for items. Lastly, the Device status page depends(uses) on the Finished Item and Current Panel row components: by clicking the items it brings the user to the Device Status page. Each main component has different

attributes and methods but maintains the same pattern- an array of values to use for the component(e.g. scheduled tests array for Scheduled component) and a formatted version of said array of values(formatted time, styled etc.). The methods consist of a getter method to fetch the arrays and a setter method to format those arrays. There are also other methods and attributes not following the pattern such as stop test method for current panel or cancel schedule method for schedule. The row component's attributes basically have the corresponding array item attributes. For example, for Finished Item row it consists of the vm's id, test name, app name, status of the test, the start and end time in epoch, and a boolean value to tell the device status page if it is from the finished item row or not. The attributes of the Finished Item row are then passed on to the Device Status page to be used.

The Device Status page is used to display the attributes such as app name, test name, test status, response time or average response time depending on the test status and more. The main attributes passed on to the device status page is the vm id and start/end time because it is used in conjunction with the component's response time getter method that uses vmId and the start/end time to fetch. The remaining methods are the methods used to set up the graph of the response time.

Lastly, the homepage component only consists of a boolean attribute isSubmitted and a setter method of that boolean value. It is used to jumpstart the other components to fetch and start clock to update values based on if the test is submitted.

3.3.2 Swimlane Diagram

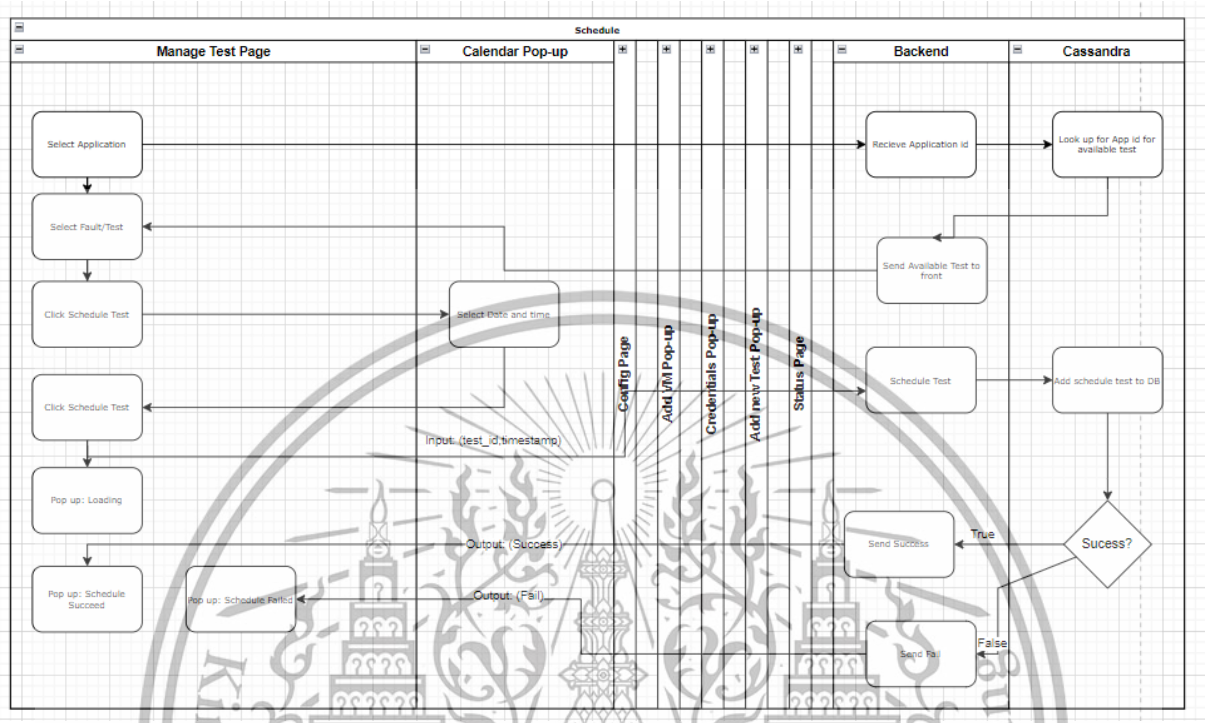

Figure 3.13 Schedule Diagram

Figure 3.13 shows the scheduling process. First the user selects an application to then select the available tests from it. Once the test is selected the user presses on the select date and time button, a calendar component pops up to have the user input their date and time. Thereafter, the user can click the schedule test button to finally send the datetime input to the backend which the arguments consist of the selected test id and the timestamp. The backend then adds that slot to the database. The response is sent back to the user if it is successful or fails.

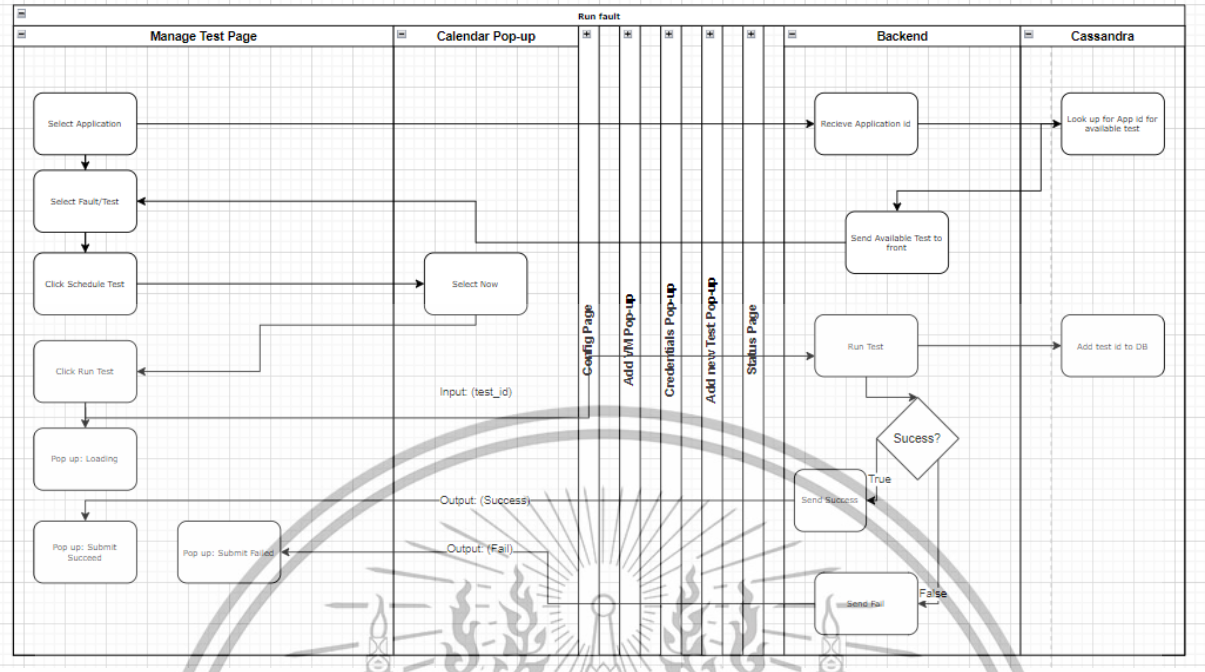

Figure 3.14 Running Faults Diagram

Figure 3.14 shows how the user can run the test immediately as opposed to scheduling. First the user selects an application to then select the available tests from it. Thereafter, the user can click the run test button to send the datetime input to the backend which the arguments consist of only the selected test's id. The slot is then written to the database. If the request is successful the test will be run immediately and a success response to the user, if not, a failure response will be displayed to the user.

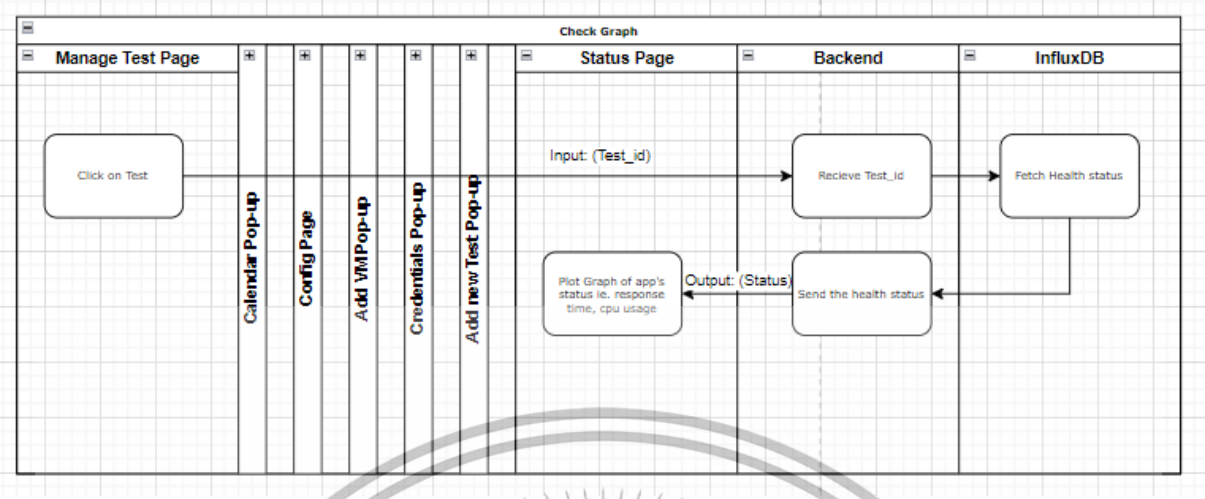

Figure 3.15 Check Status Diagram

Figure 3.15 shows how a user can click on the test to monitor the vm's metrics through a graph plot. First, the user clicks on the test on the Manage test page, then the test_id is used to fetch the health status and plots it to the status page. The health status may vary depending on the type of vm it is.

3.3.3 State Machine Diagram

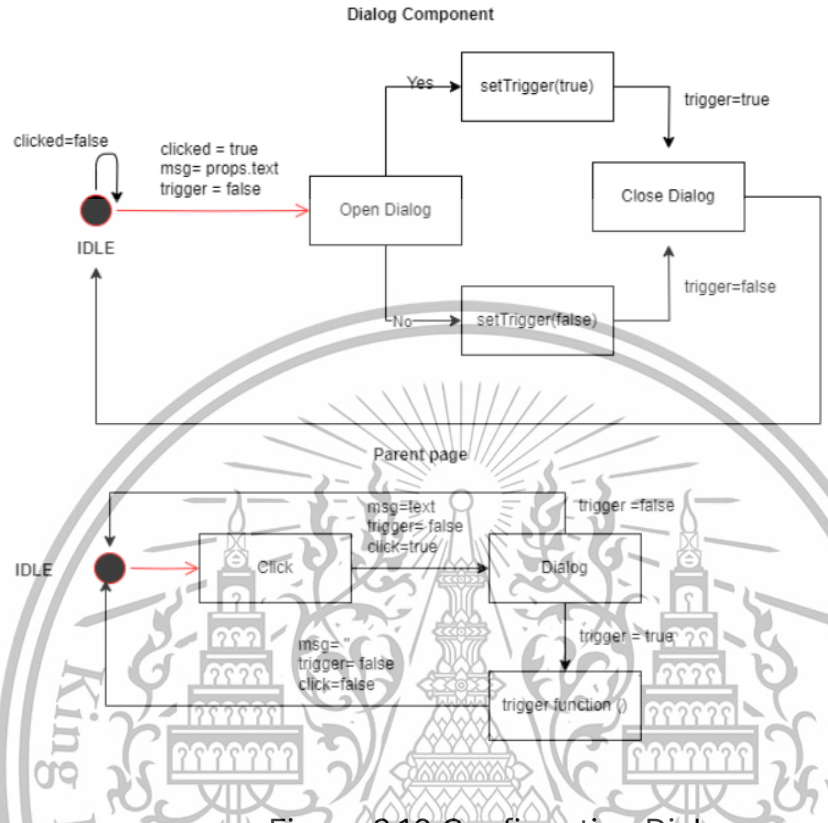

Figure 3.16 Confirmation Dialog

Figure 3.16 shows in detail how the confirmation dialog is implemented in the code. The upper diagram in the figure is the Dialog component, it shows that the component will not show up until clicked is true. Then, depending on yes or no input, the trigger function will be called to consequently call the prop function.

The lower diagram shows how the dialog component is implemented to a page element. When on click to the implemented element, the click state is set to true and a dedicated message is set for the confirmation dialog to display. Then, following the confirmation dialog process, the function will either run or not based on user input, then states are finally reset to then remain idle until that element is clicked again.

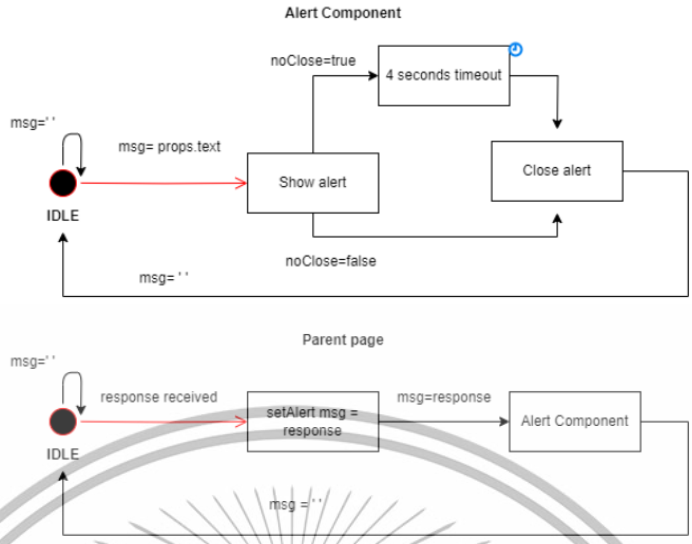

Figure 3.17 Custom Alert

Figure 3.17 The upper diagram shows how the alert components work alone and the lower diagram on how that is implemented into a page element. The alert component only relies on response to set as msg value and display. There is an option to set if it auto closes or the user needs to click on the icon to close it themselves. If auto close alert, it will simply close after a 4 seconds timeout (adjustable) and the msg value resets and remains idle until it is not an empty string.

3.3.4 UI Documentation

3.3.4.1 Manage Test page

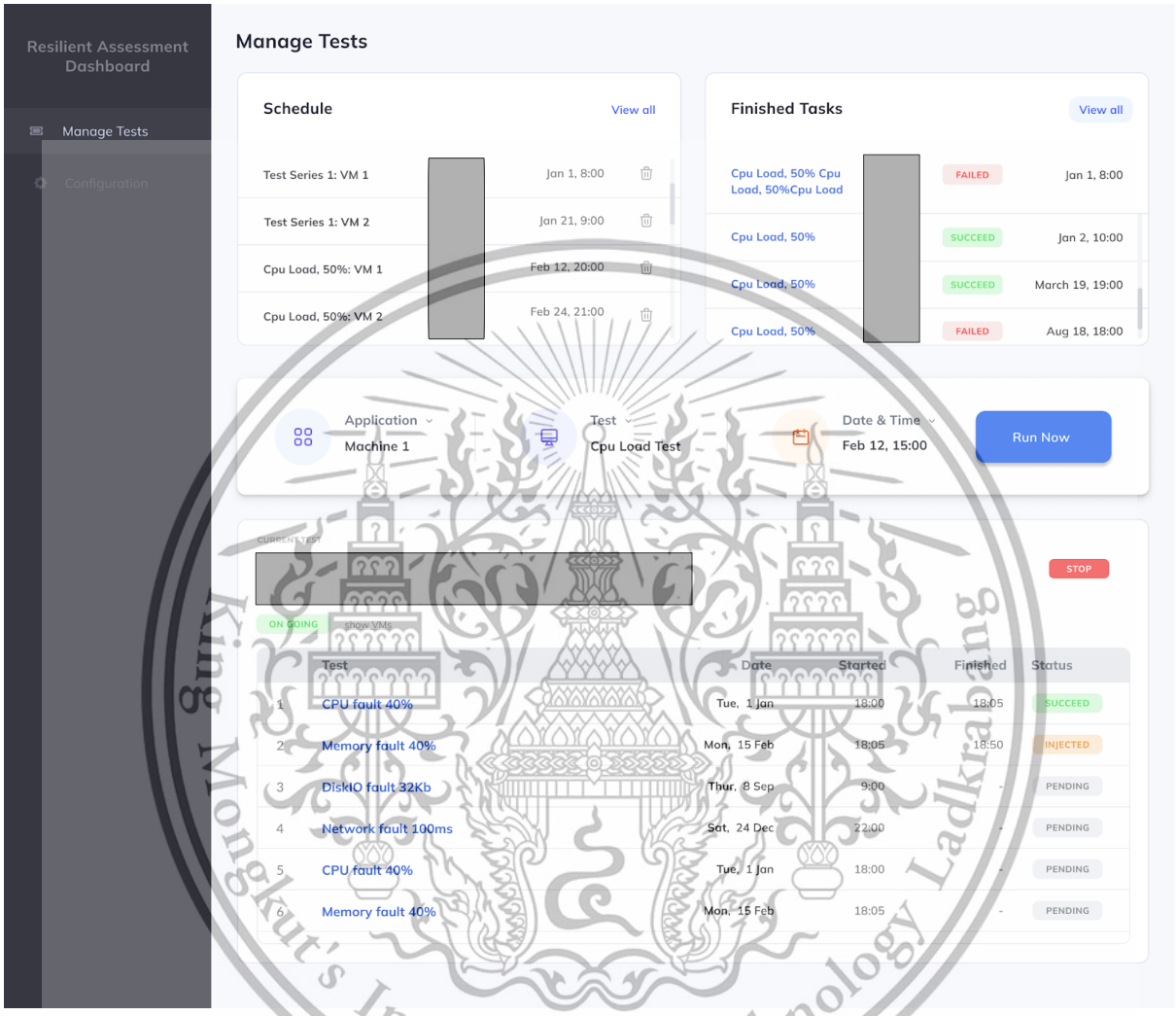

Figure 3.18 Manage Test page

The figure 3.18 shows a screenshot of the Manage Test page. This page is divided into four sections.

The first section is the test schedule. In this section, the tests that were scheduled from the Run Test section are displayed. It includes the test name, VM name, application name, and the scheduled time. Users have the option to cancel a scheduled test by clicking the trash can button associated with it.

Moving on to the second section, we have the finished task section. Here, you can find a list of tests that have been completed.

This section displays the test name, VM name, application name, test status, and the time of completion for each task.

Next, we have the third section called the Run Test section. This section provides the user with the capability to either run a test immediately or schedule it for a specific time. It includes various elements such as a dropdown menu to select the desired application, another dropdown menu to choose the test, a calendar widget, and a submit button. If the user selects a date and time using the calendar widget, the text on the submit button will display “Schedule test”. However, if the user leaves the calendar at its default or selects the current time, the text on the submit button will display "Run Now" to initiate the immediate execution of the test.

Lastly, we come to the fourth section known as the Current test section. This section provides an overview of the tests that have already been completed, the tests currently in progress, and the tests that are yet to be executed. At the top of this section, users can find a button to select the desired application. The tests displayed in the table below are organized and categorized based on their respective application names. This allows for a clear and structured representation of the tests within the section.

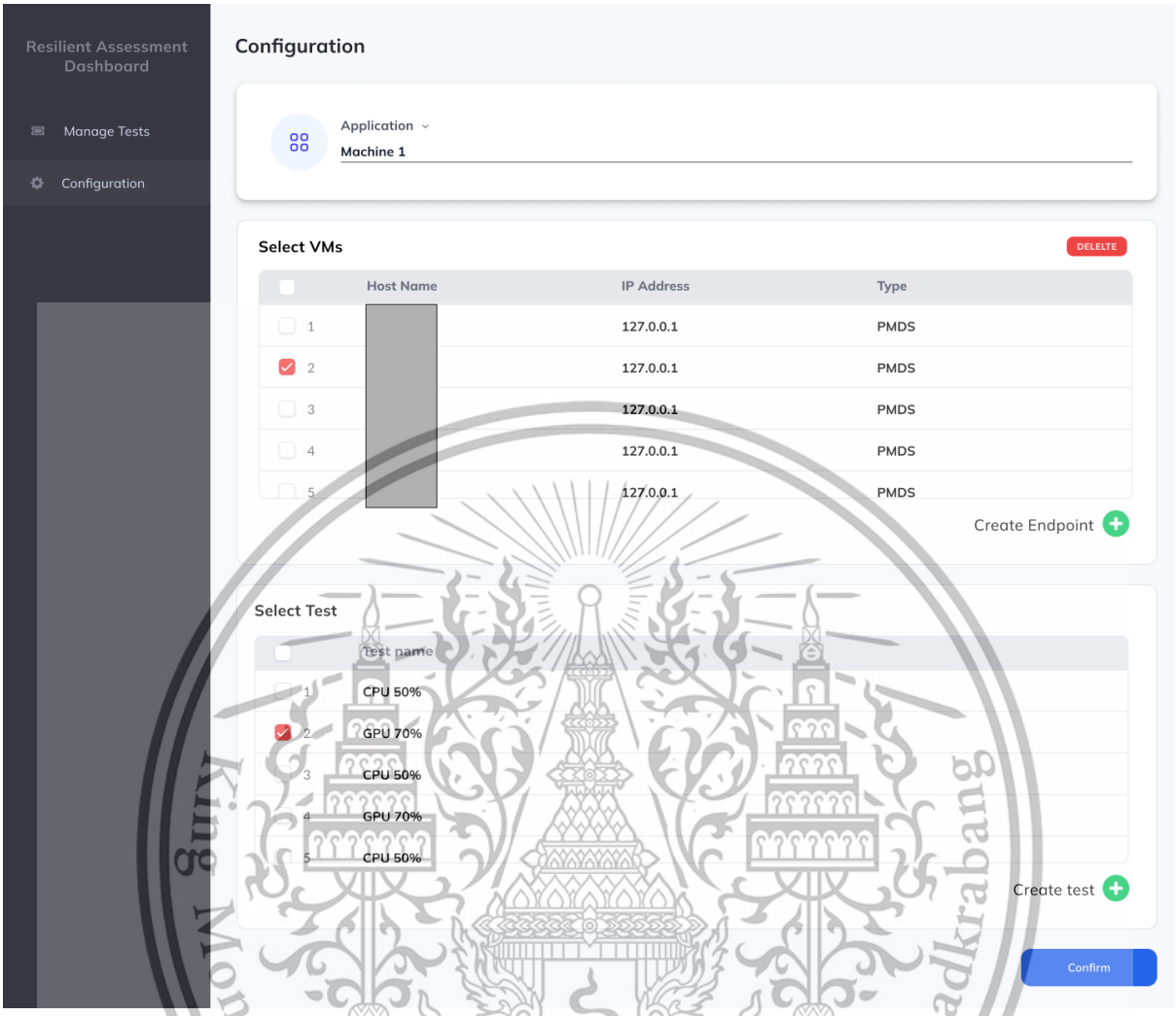

Figure 3.19 Configuration page

The figure 3.19 shows a screenshot of the Configuration page. This page is divided into three sections.

The first section on the Configuration page is the "Select Application" dropdown. Users can choose the application name from the dropdown to proceed to the next step. The selected application name will be utilized to categorize the virtual machines (VMs) associated with that particular application.

Moving on to the second section, we have the "Select VMs" section. In this section, the VMs belonging to the selected application will be displayed. Users have the option to delete a VM by checking the checkbox in front of the order number in the respective VM's row and then clicking the red "Delete" button located above. Additionally, users can click on the plus button labeled "Create Endpoint" to add a new VM to the system. When the "Create Endpoint" button is clicked, the website will present a form that users need to complete and submit in order to create a new VM. If the user provides valid information, the system will proceed accordingly. However, if there are any errors or invalid inputs, an error message will pop up on the screen to alert the user.

Lastly, we have the third section called the "Unattached Test" section. Once the user checks the checkbox in front of the test row, this section will display tests that are not currently attached to the selected VM. Users can select the desired tests by checking the checkboxes and then click the "Attach Test" button to associate the selected tests with the VM. Additionally, users have the option to click the "Create Test" button to generate a new custom test. Upon clicking this button, the website will present a form where users can input details such as the test name, test type, loads, timeout, additional parameters (for specific test types), and path. Once the form is submitted, the newly created test will be added to the unattached pool for all the VMs, making it available for selection and attachment in the future.

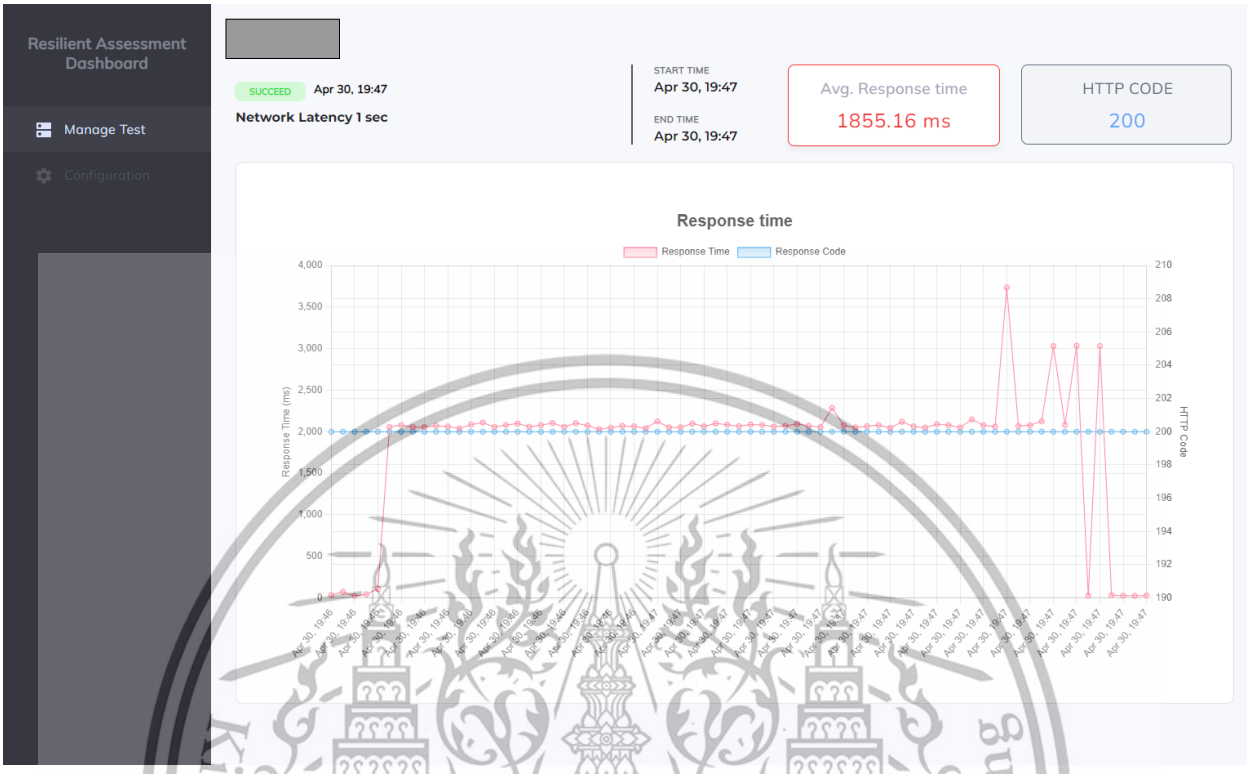

Figure 3.20 Response time graph

Figure 3.20 illustrates the response time graph. The red line in the graph represents response time and the blue line represents response code. On the top left, there is a test name. On the top right, there are the start time, end time and values for response time and response code.

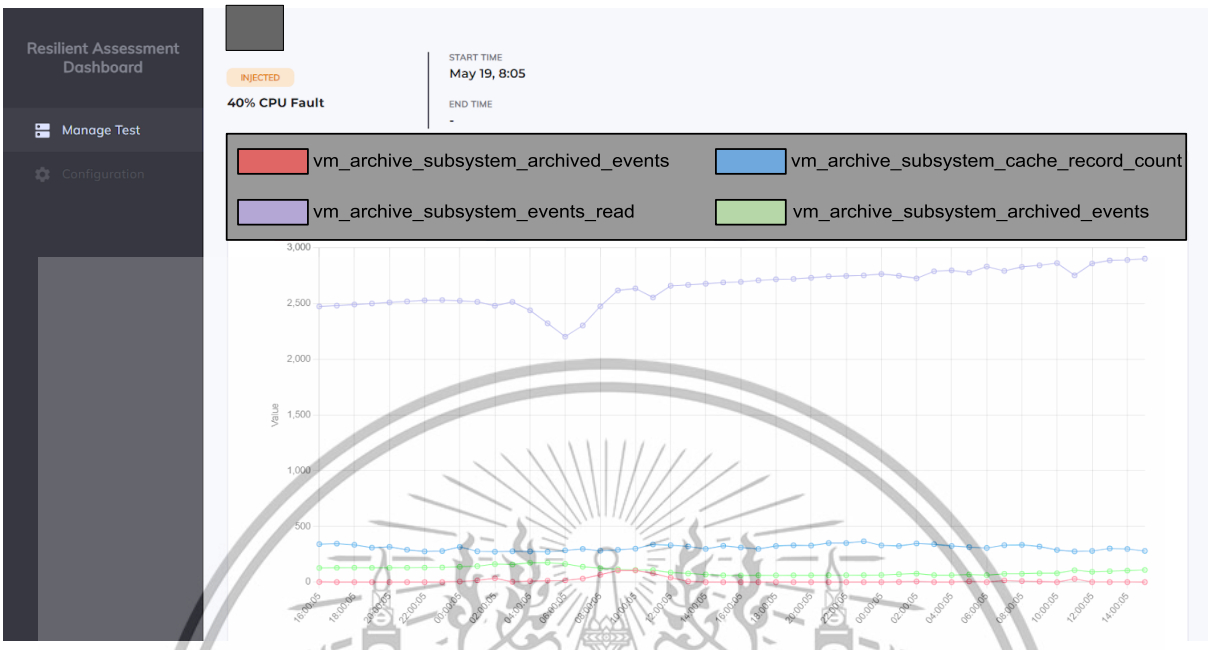

Figure 3.21 Device Status page

Figure 3.21 illustrates the device status graph. There are four graphs on this page, each represents metrics from the device. On the top left, there is a test name, start time, and end time.

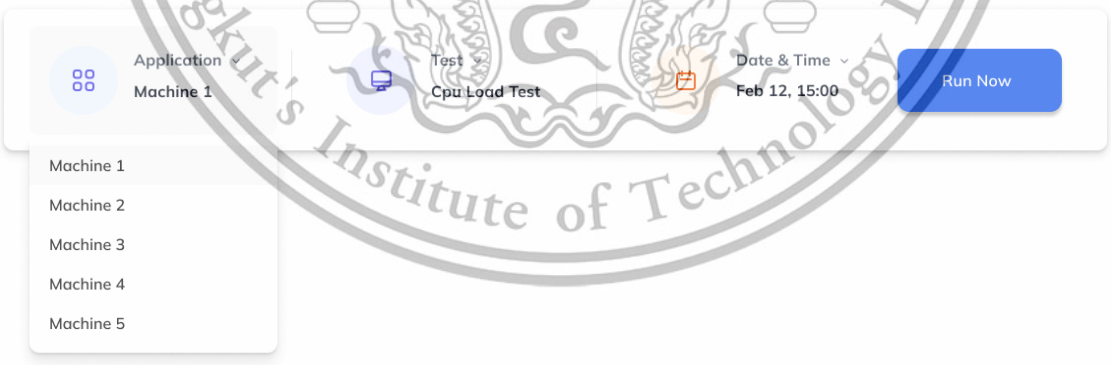

Figure 3.22 Test selection bar (Run test now)

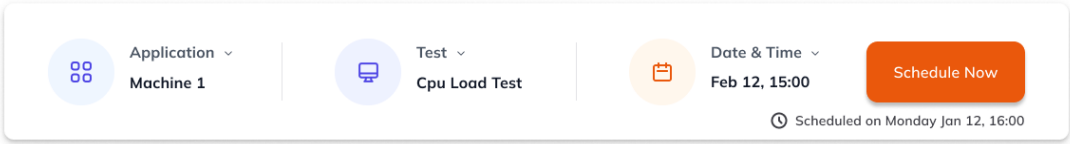

Figure 3.23 Test selection bar (Schedule test)

The test selection bar figures 3.23 shown above is designed to be the dashboard's main action, which has an essential role in controlling the whole dashboard. The dropdown from second to last is intentionally disabled until the previous dropdown is selected. However, there is a schedule option that allows the user either skip the select date and time to run now button or select the date and time to schedule the test due to the time selected.

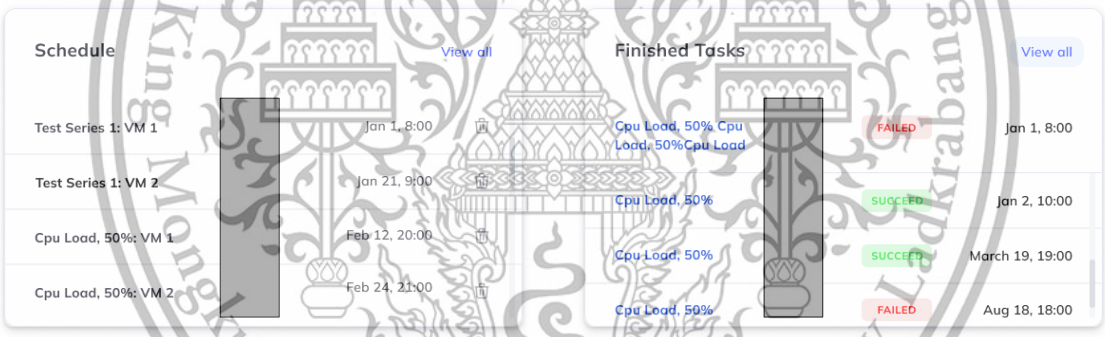

Figure 3.24 Schedule and Finished Tasks Display

The figure 3.24 shown above shows both displays on top of the Manage test page in order to manipulate the user's focus on the past action. The test selection bar appears under them and acts as a current action. For schedule display, it is capable of deleting the unwanted schedule that has been made and also contains an enlarged view of the display itself. For finished tasks, the darker blue acts as a route connecting to the device status page of itself. The display also mends to show the wording in order, such as the status before the date the status was achieved. The lighter blue is classifying the type of endpoint group.

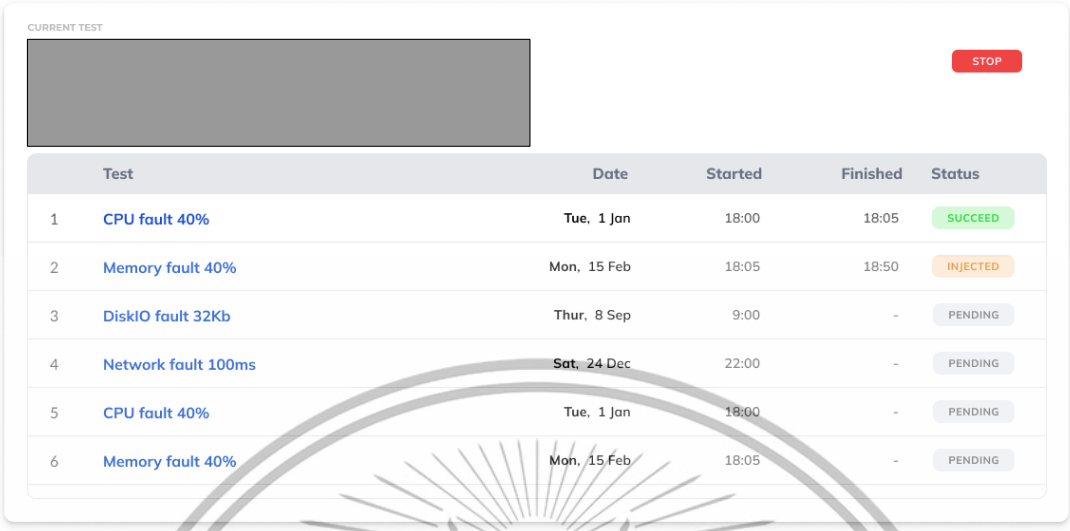

Figure 3.25 Current Test

In this current test, the component contains multiple features according to the selected test as shown in figure 3.25. The gray button "shows VMs" showing all endpoints in the group and in the bottom is the running one. There is a status under test showing the condition of the test. The secondary and tertiary blue buttons indicate the later running test and are capable of switching between tests. In the table show lists of the tests, the date, the time started and finished and also the status. The blue wording tests are routed to its device status page. Lastly, the stop button is to stop the currently running test.

3.3.4.2 Device Status page

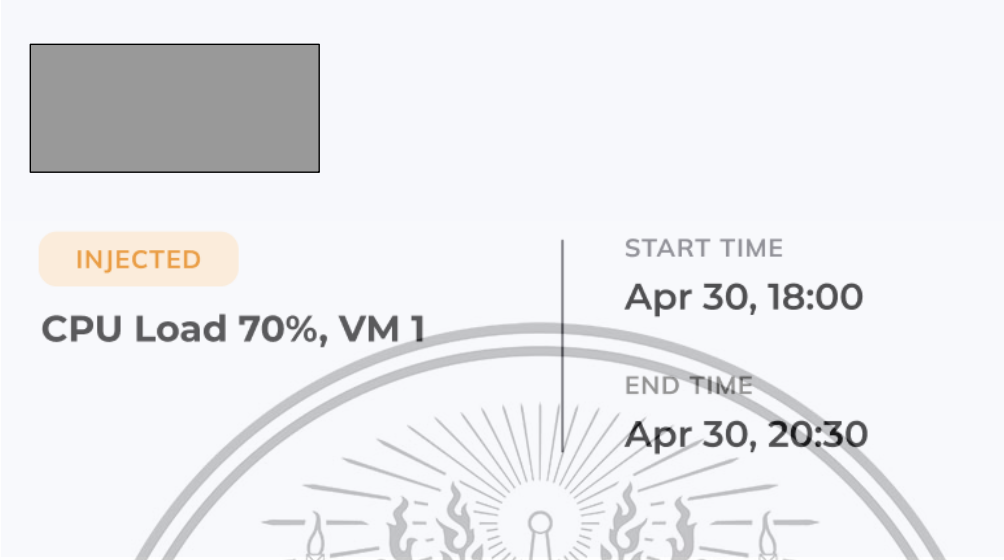

Figure 3.26 Device status header

From the figure 3.26. The header of the device status page includes the application name at the top left and below are the test name and its status. Besides that the start and end time including the date of the test. (As shown in figure 3.33)

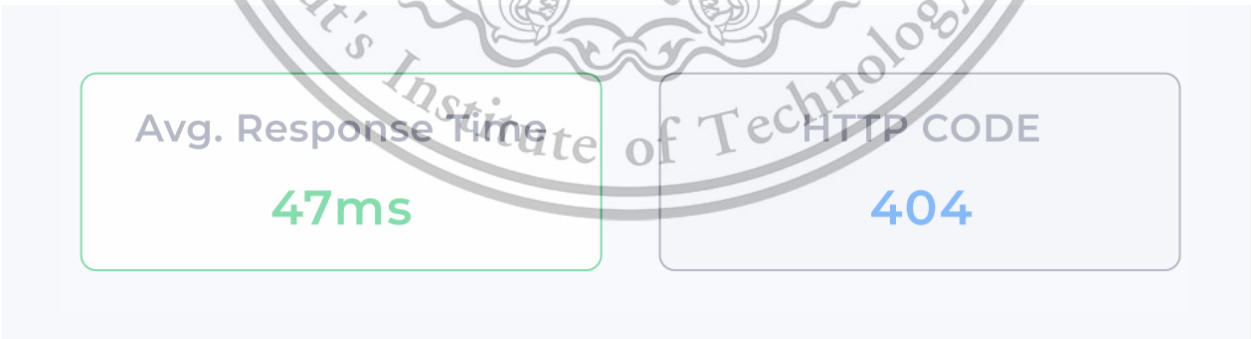

Figure 3.27 Response Time and HTTP Code

From the figure 3.27. Both Response Time and HTTP Code are on the right part of the header indicating average response time in various colors and values of HTTP code.

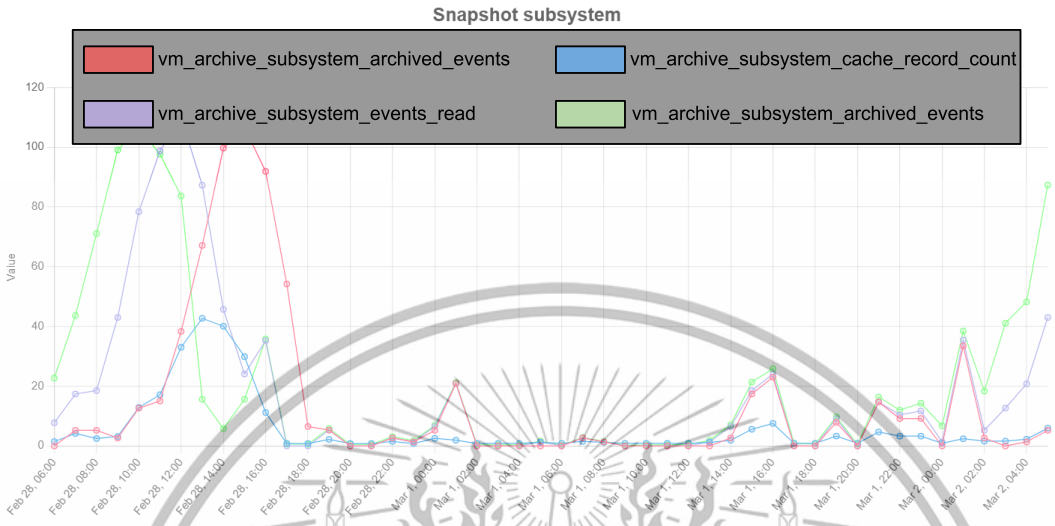

Figure 3.28 Graph

From the figure 3.28. There are two types of graphs for our purpose, shown above is a graph for the device endpoint group. This was plotted with previous data from the InfluxDB. No response time and HTTP Code will be plotted here.

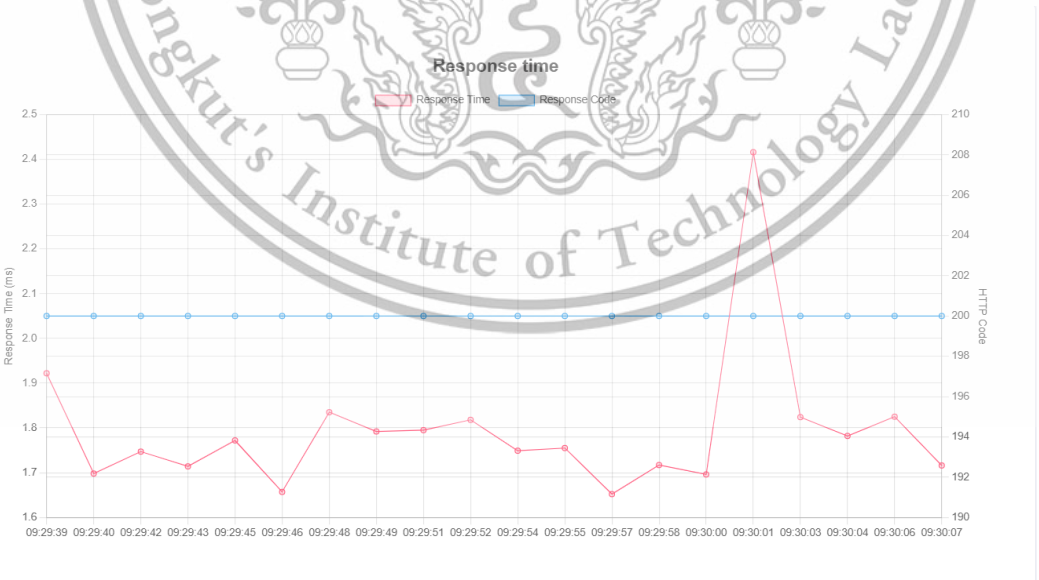

Figure 3.29 Response time graph real time

From the figure 3.29. The response time graph above shows when the status is not yet completed, it updates the response time and HTTP code in real time in ten second intervals.

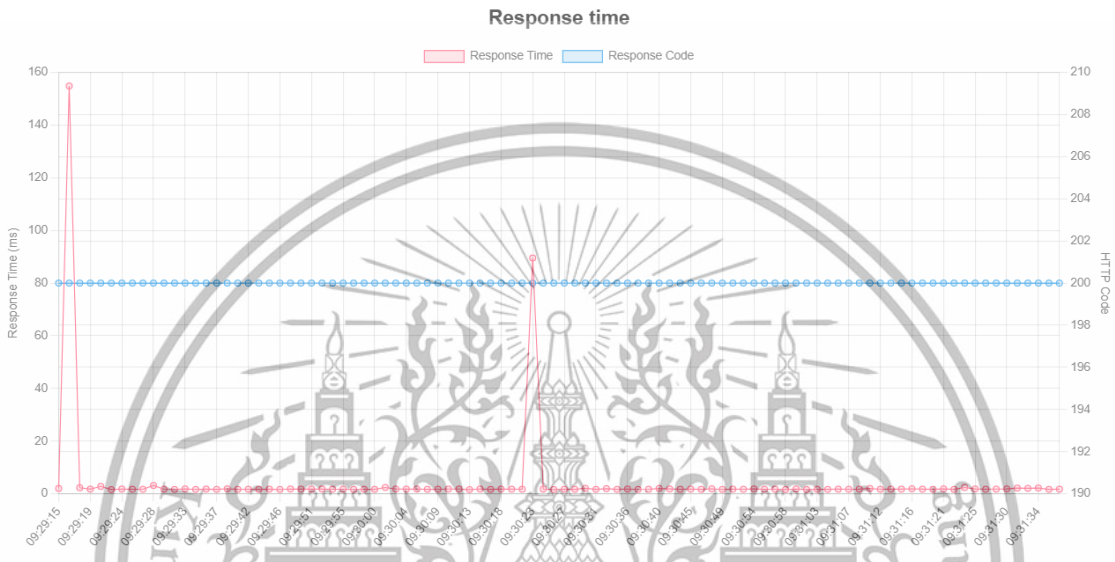

Figure 3.30 Finished Response time graph

In contrast, this is a response time graph when the test is completed it plots the whole data point from starting time to finished time as shown in figure 3.30.

3.3.4.3 Configuration page

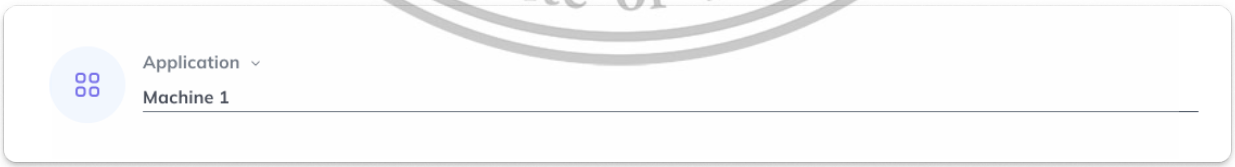

Figure 3.31 Select application

In this configuration page, selecting an application means choosing an application to customize VMs and tests as shown in figure 3.31.

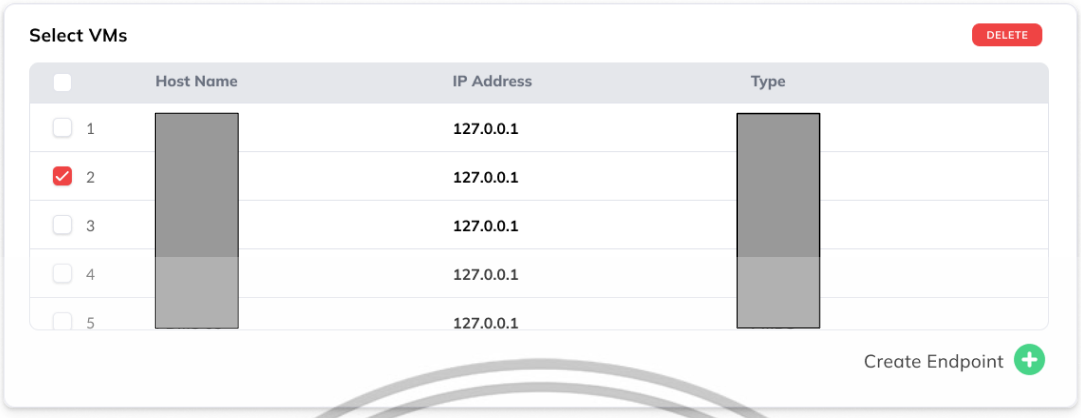

Figure 3.32 Select VMs

Choosing a VM to edit tests below or delete the selected VM from the application as shown in figure 3.32. This component is also able to create a new VM for the application by entering information in the popups. In a popup, each field has an input validation which also comes with helper text letting users know what to type in the field as shown in figure 3.33.

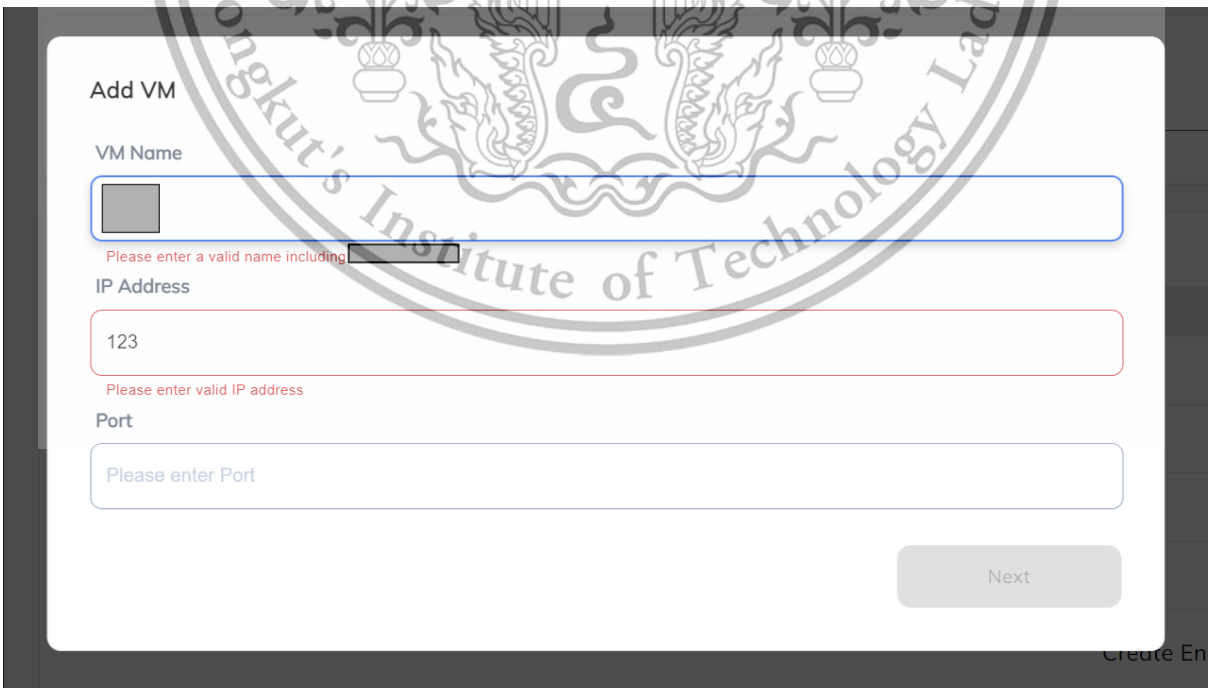

Figure 3.33 Add a new VM

Users must input the VM name for a new VM which must include the keyword in the name which also lets the user know by helper text. IP address has a format error check and same for port number.

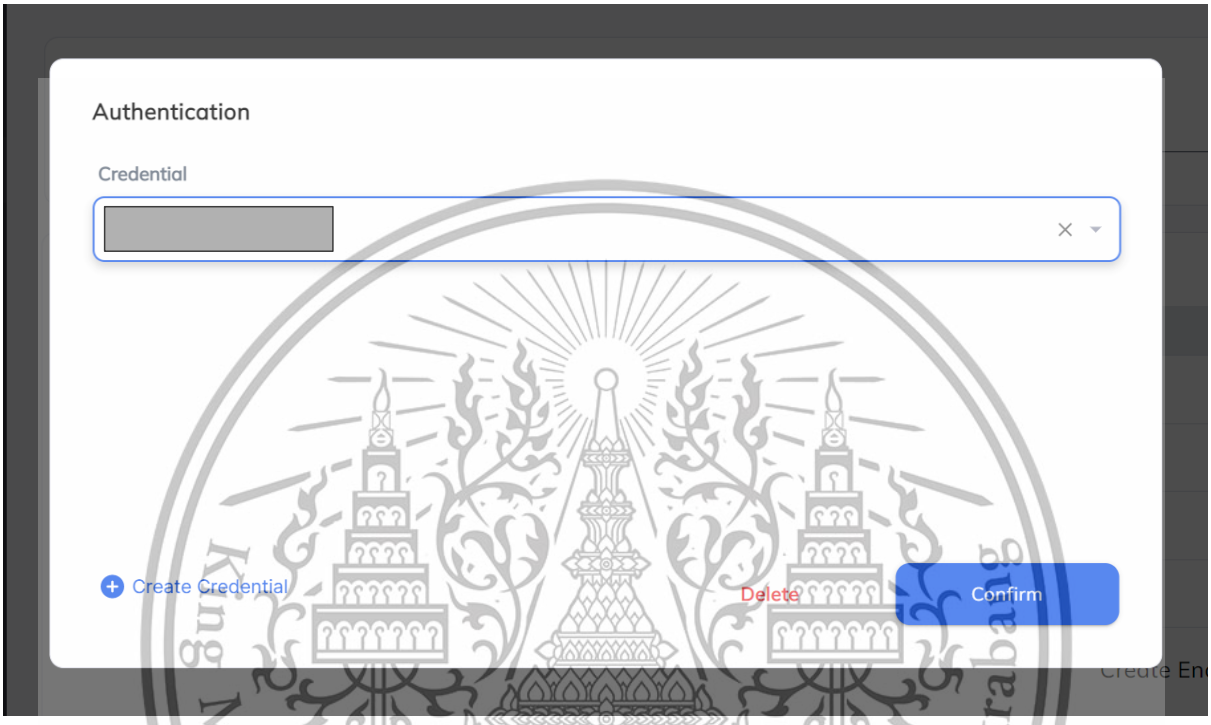

Figure 3.34 Authentication

After adding a new VM, the user must select a credential for authentication which can either delete the current credential or create a new one as shown in figure 3.34.

Figure 3.35 Create a credential

Creating credentials in figure 3.35 are available when new users are created in VM

Figure 3.36 Select test

Choosing tests to add tests to the selected VM in figure 3.36. This component is also able to create a new test for the VM by entering information in the popups. In a popup, each field has an input validation which also comes with

helper text letting users know what to type in the field. *If the IP is not online then the action of creating a VM will be rejected.

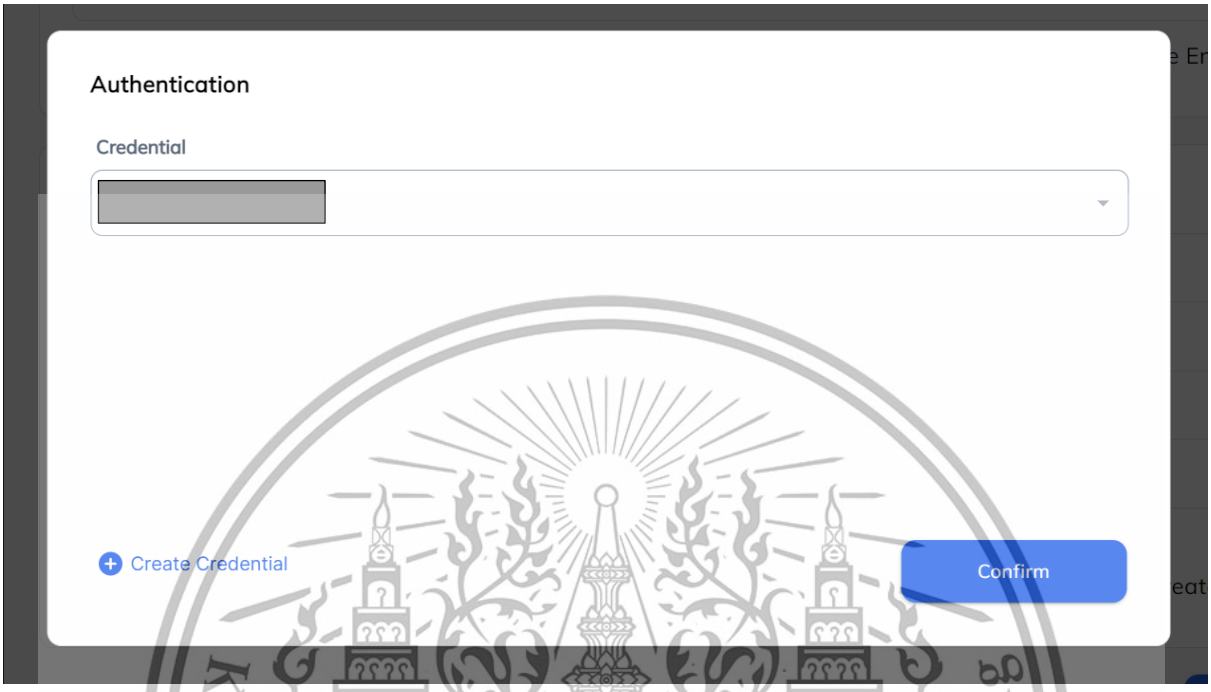

Figure 3.37 Authentication

Credentials are selected to be used for test injection path in figure 3.37

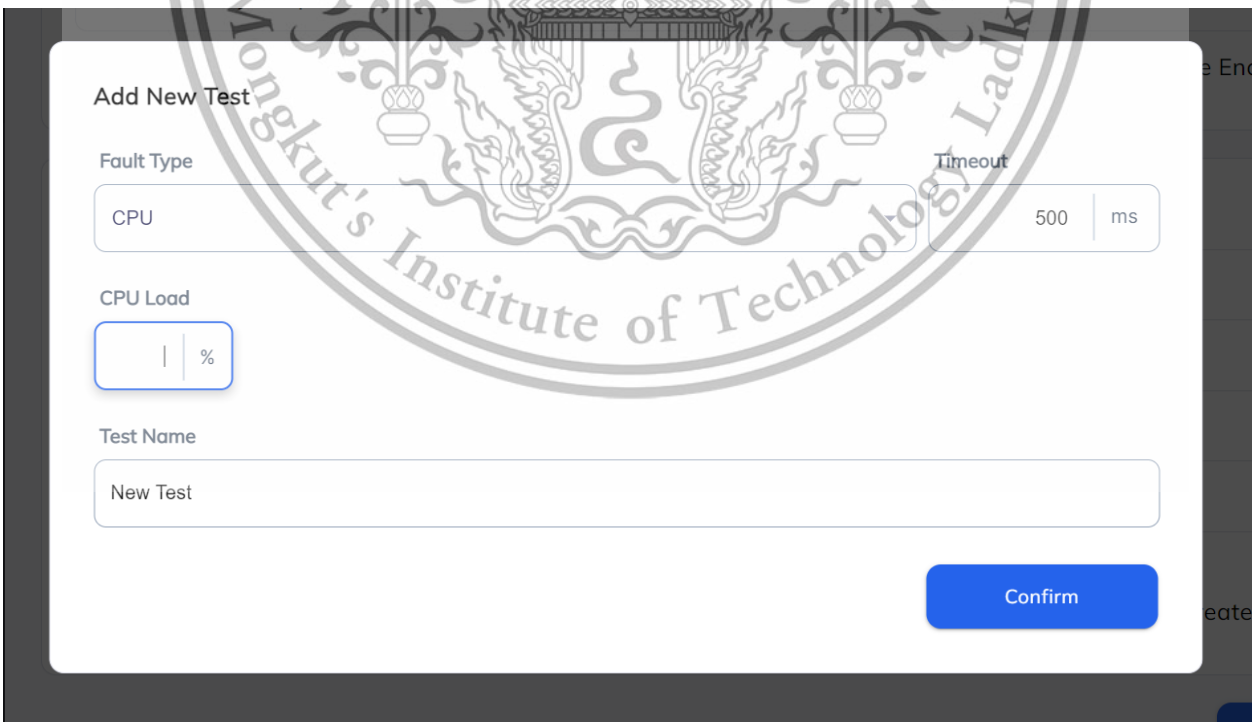

Figure 3.38 Add new test (Fault Type = CPU)

The CPU fault type lets the user input Cpu load in figure 3.38.

Adding a new test is corresponding to the fault type selected which will let the user input a different input variable due it specific fault type.

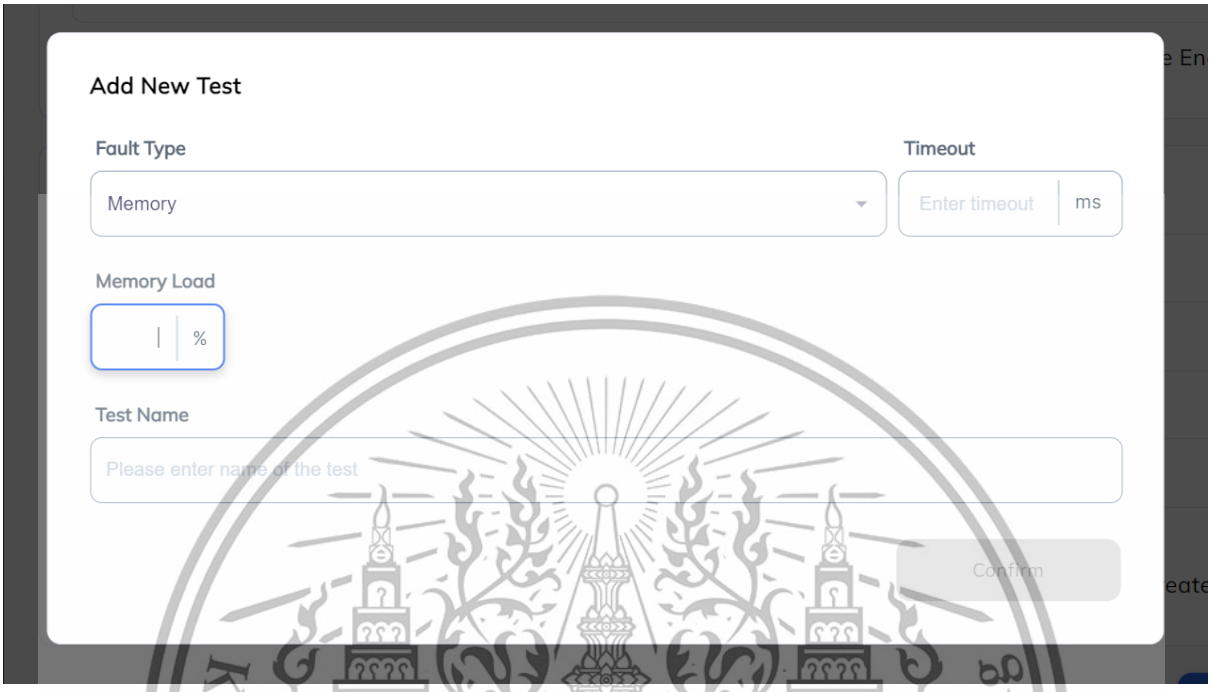

Figure 3.39 Add new test (Fault Type = Memory)

Memory fault type lets the user input memory load in figure 3.39.

Figure 3.40 Add new test (Fault Type = DiskIO)

Disk IO fault type lets the user input a block size of Disk IO in figure 3.40.

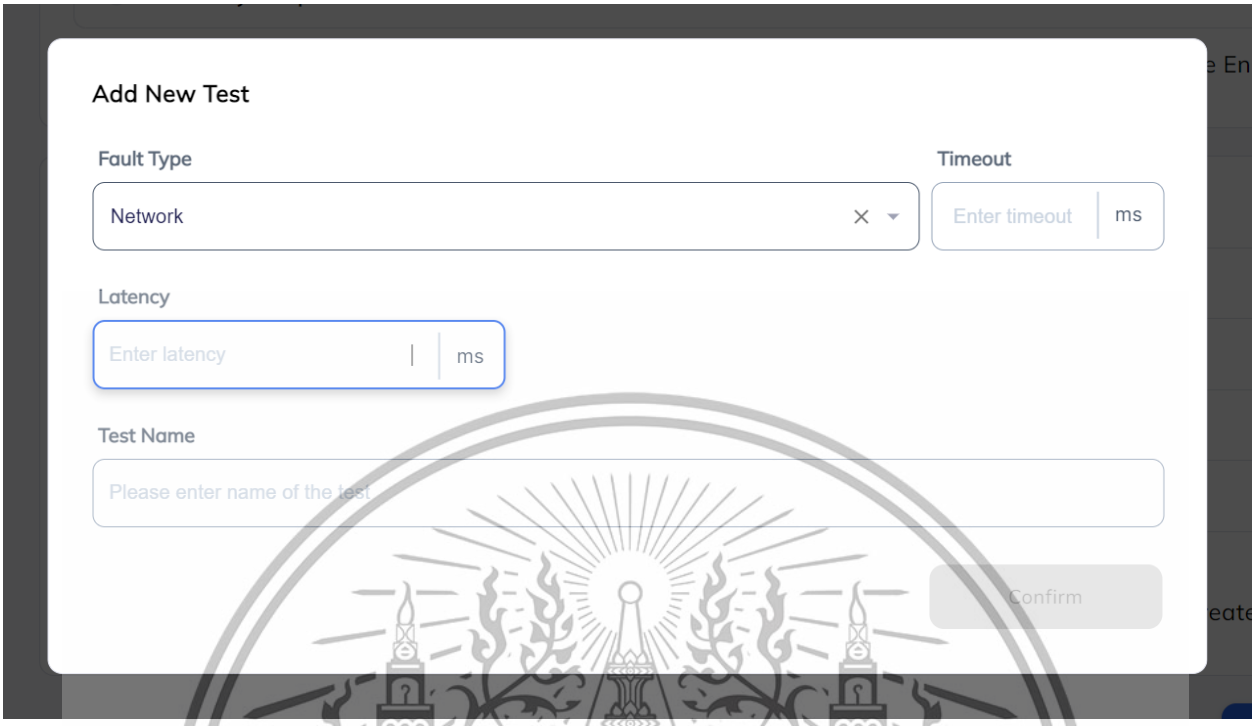

Figure 3.41 Add new test (Fault Type = Network)

Network fault type lets the user input a latency in figure 3.41.

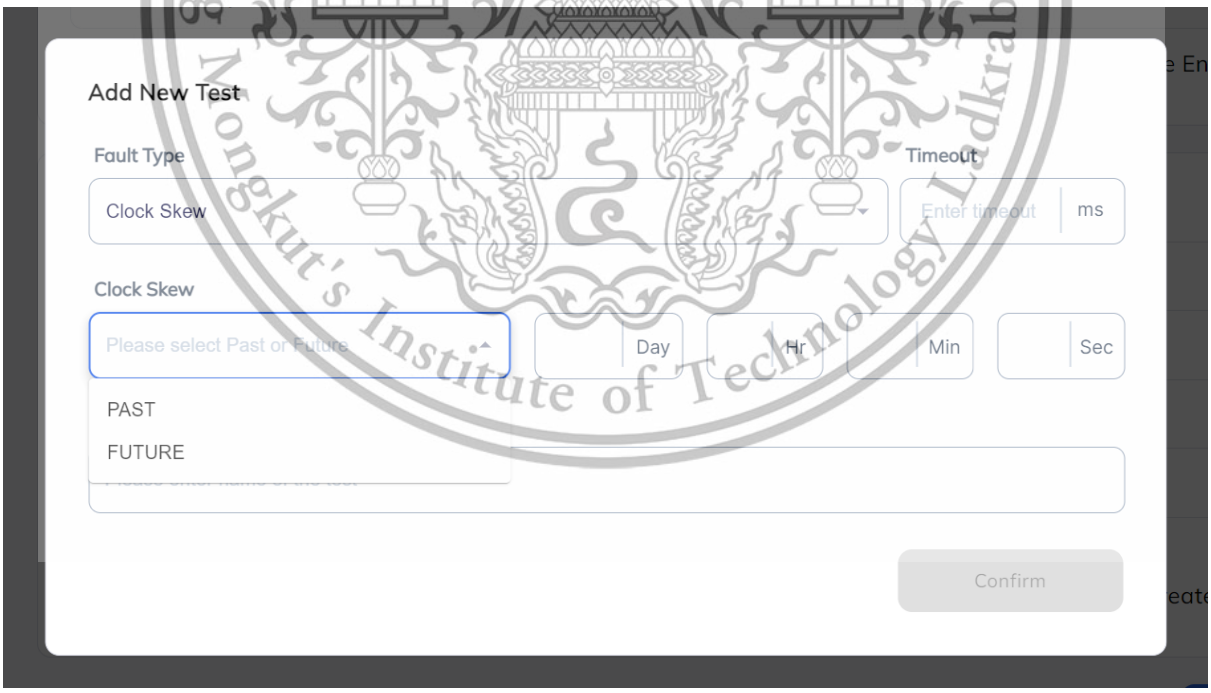

Figure 3.42 Add new test (Fault Type = Clock Skew)

This fault type is able to select clock skew either setting time in past or future in figure 3.42.

3.3.4.4 Miscellaneous

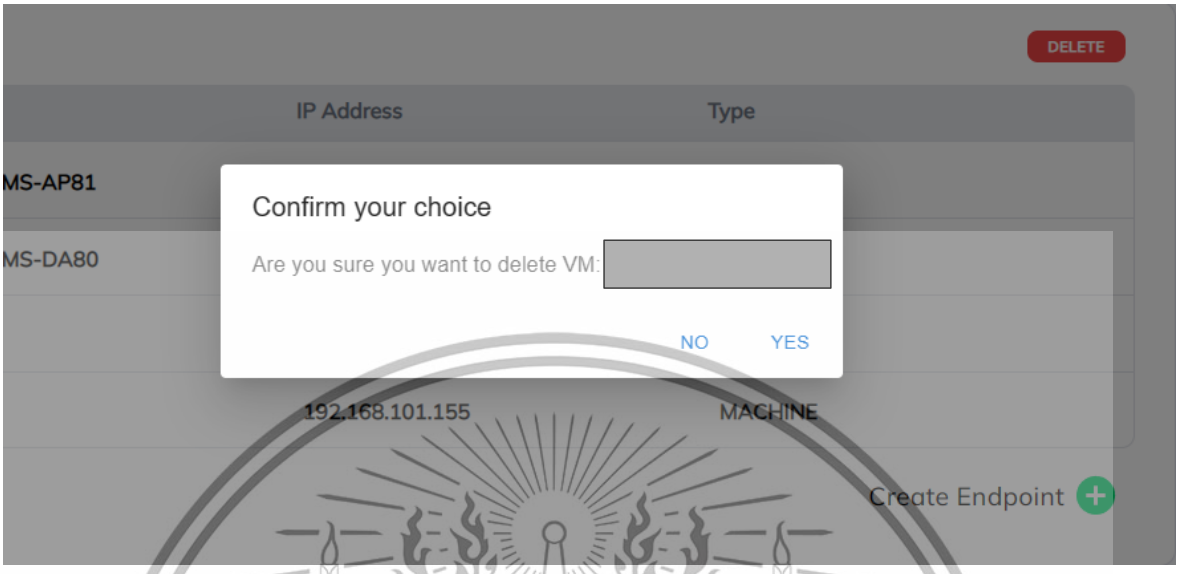

Figure 3.43 Confirmation Dialog

Confirmation dialogs in figure 3.43 are used to confirm user actions before a crucial consequence eg. delete endpoints, stop running tests etc. Confirmation Dialogs are coded to be a reusable component, however, the complexity arises when making the component appear,disappear, and keep track of the yes or no state. Therefore, the React states are manipulated to create the dialogs. This can be seen above in the state machine diagram(figure ci). Essentially the component's mechanism is to trigger the parent page(implementer) function eg. delete endpoint, delete schedule etc. and then reset the state.

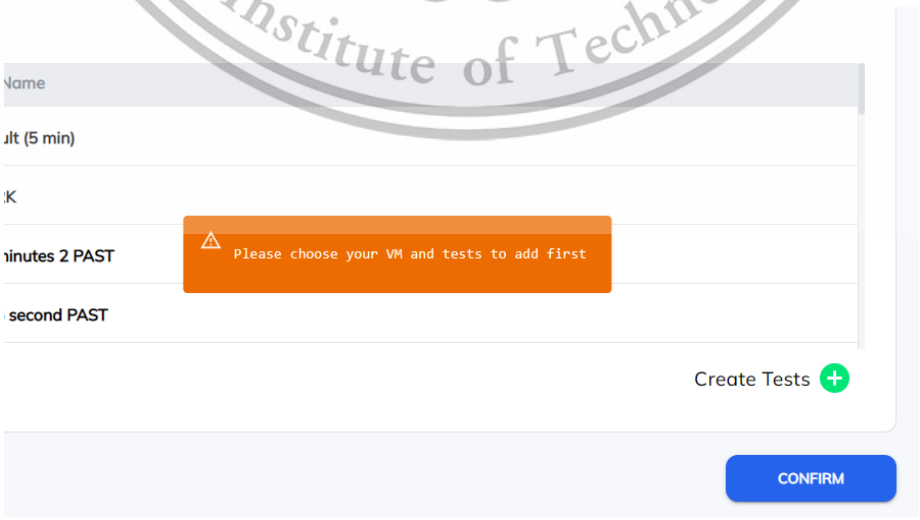

Figure 3.44 Caution Alert

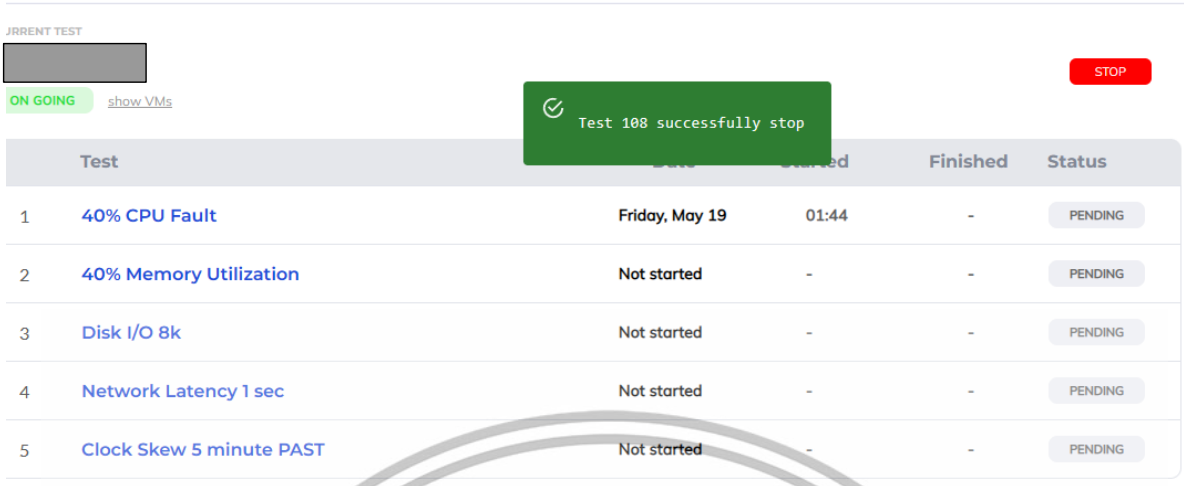

Figure 3.45 Success Alert

Custom alerts are used to handle actions where conditions are not met or used to alert users the outcome of the action in both figure 3.44 and 3.45 ie. API requests The same complexity of the alerts can be said to be the same as the confirmation dialog, albeit simpler. The alert does not have to keep track of when it is being clicked or not. it only appears typically when the response api returns.

3.4 Backend

3.4.1 Overall diagram of the system

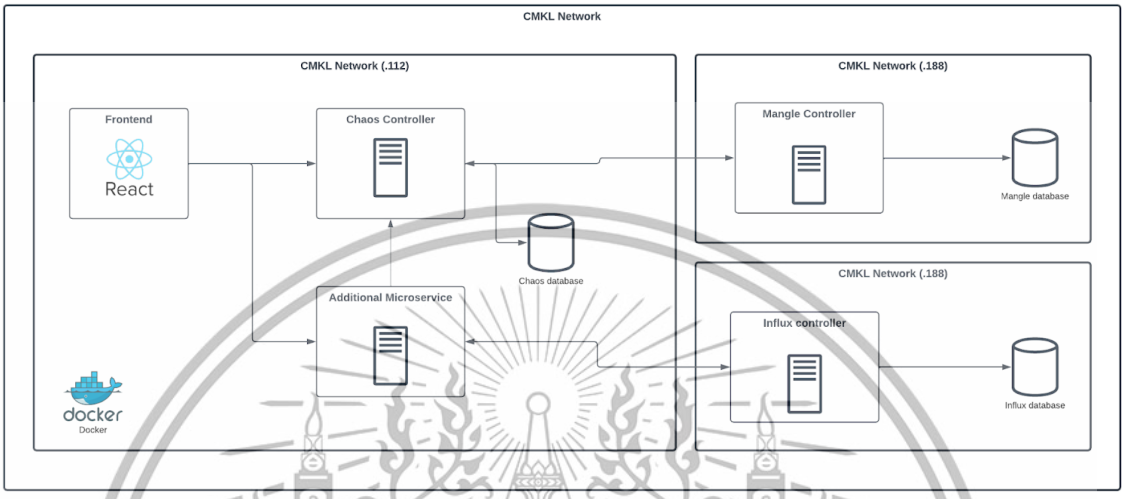

Figure 3.46 Overall diagram of the system

Figure 3.46 illustrates the overall diagram of the whole system. We used React.JS for our framework in frontend development and Python flask for the additional microservice and Django for the chaos controller. The chaos database and the Mangle database are operated by Apache Cassandra. The InfluxDB is used for any real time data, In such cases, real time test status. The frontend server, Chaos controller, the microservice and the chaos database are hosted in the same IP using Docker but in a different port.

3.4.2 Backend APIs Flowchart Diagram

Scheduling function

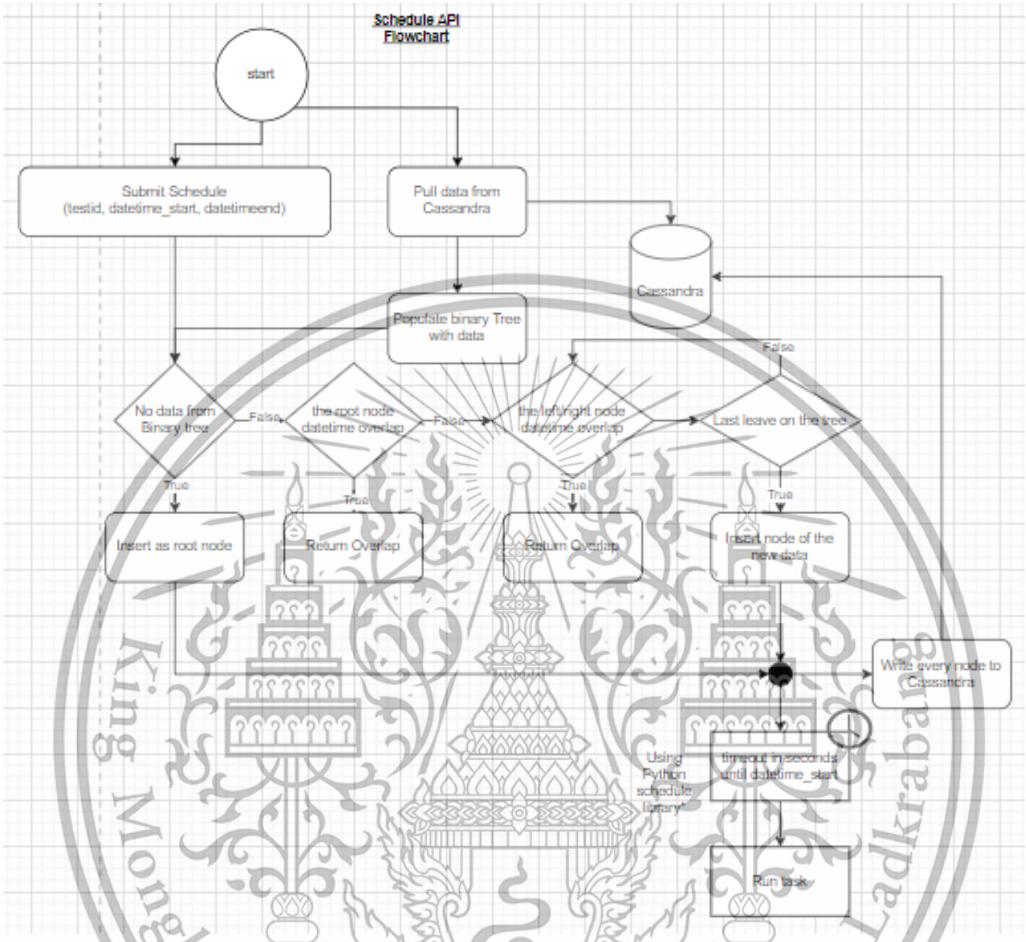

Figure 3.47 Scheduling function

Figure 3.47 shows the algorithm of the scheduling function. The algorithm in this code performs several tasks, including retrieving VM scheduling data, managing failed test entries, and creating new scheduling entries based on provided parameters. It utilizes HTTP requests to retrieve VM and test data, parses JSON responses, and interacts with a database. The algorithm ensures proper scheduling and execution of tests, handling various scenarios such as past time checks, overlapping time slots, and missing parameters.

Get Schedule function

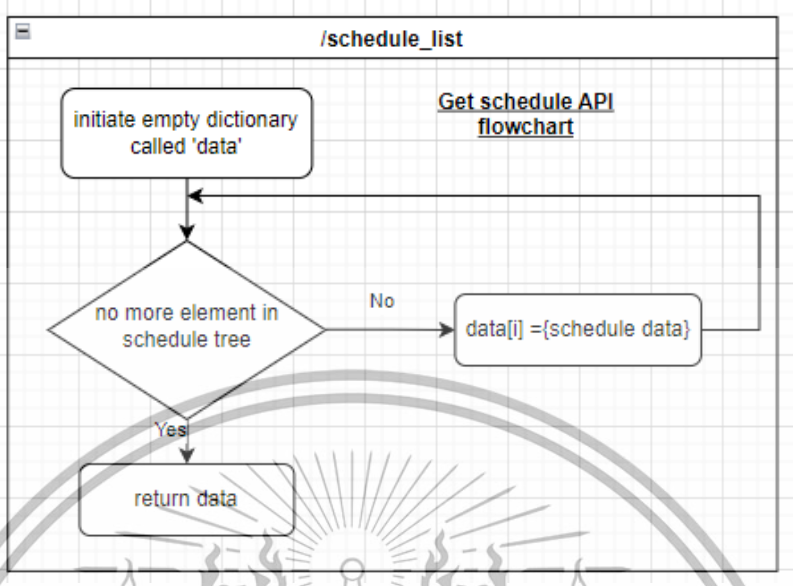

Figure 3.48 Get Schedule function

Figure 3.48 shows how the Get Schedule function works. Firstly, it initializes empty dictionaries. Then it generates a list of scheduled data for each element in the schedule tree, filtering and sorting the data based on certain conditions. Convert the resulting list to a specific data format and associate it with relevant information from another data source. Then it returns the data with scheduled tasks.

Delete Schedule function

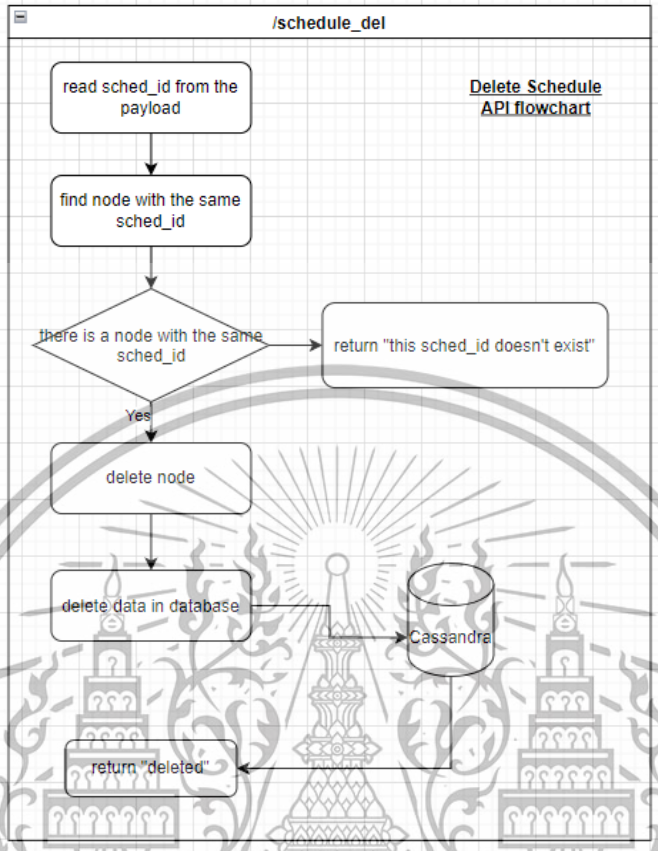

Figure 3.49 Delete schedule function

Figure 3.49 shows the algorithm in the delete schedule function. It provides functionality to delete a specific scheduling entry based on the given 'sched_id'. It uses an HTTP GET request to retrieve the VM list, parses the response, and iterates over each VM to retrieve and store its scheduling data. When a 'sched_id' is provided, the algorithm searches for the corresponding entry in 'rootnode', deletes it from both 'rootnode' and the database, and returns a success message. If the 'sched_id' is not found, a not found message is returned.

Get Previous function

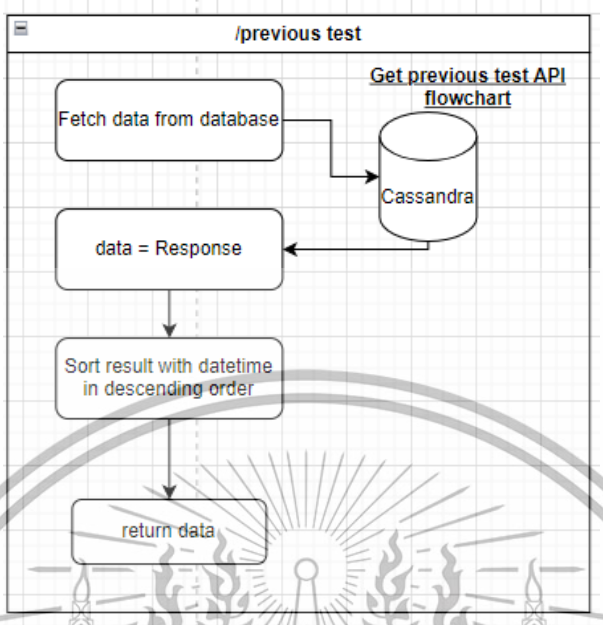

Figure 3.50 Get previous function

Figure 3.50 shows how the Get previous function works. It initializes an empty list. It executes a SELECT query on a database table and retrieves the result set. It iterates over each row in the result set and processes the data. Inside the loop, it extracts relevant information, and creates a dictionary 'q' containing the extracted data. The dictionary is appended to the list.

Stop Test function

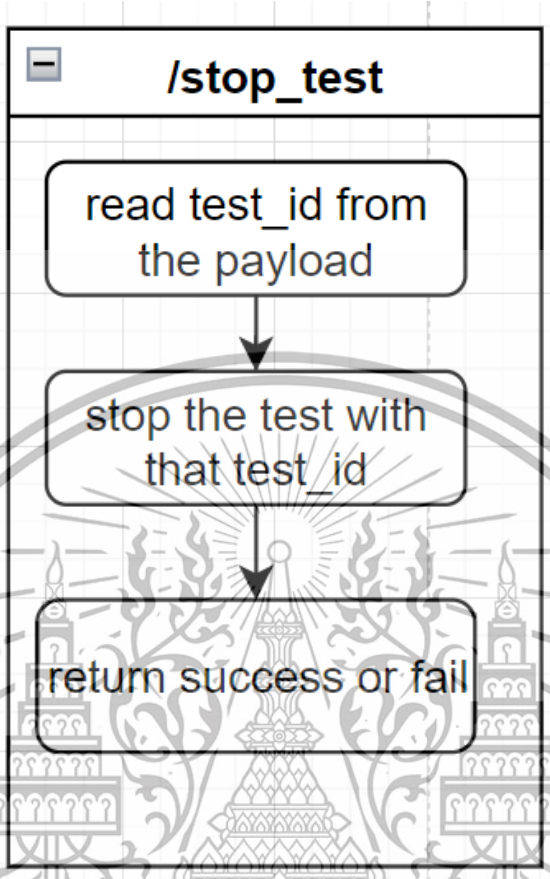

Figure 3.51 Stop test function

Figure 3.51 shows how the Stop test function works. First, the function reads the test_id from the payload. Then, it stops the test with the test_id read earlier.

Get Test status function

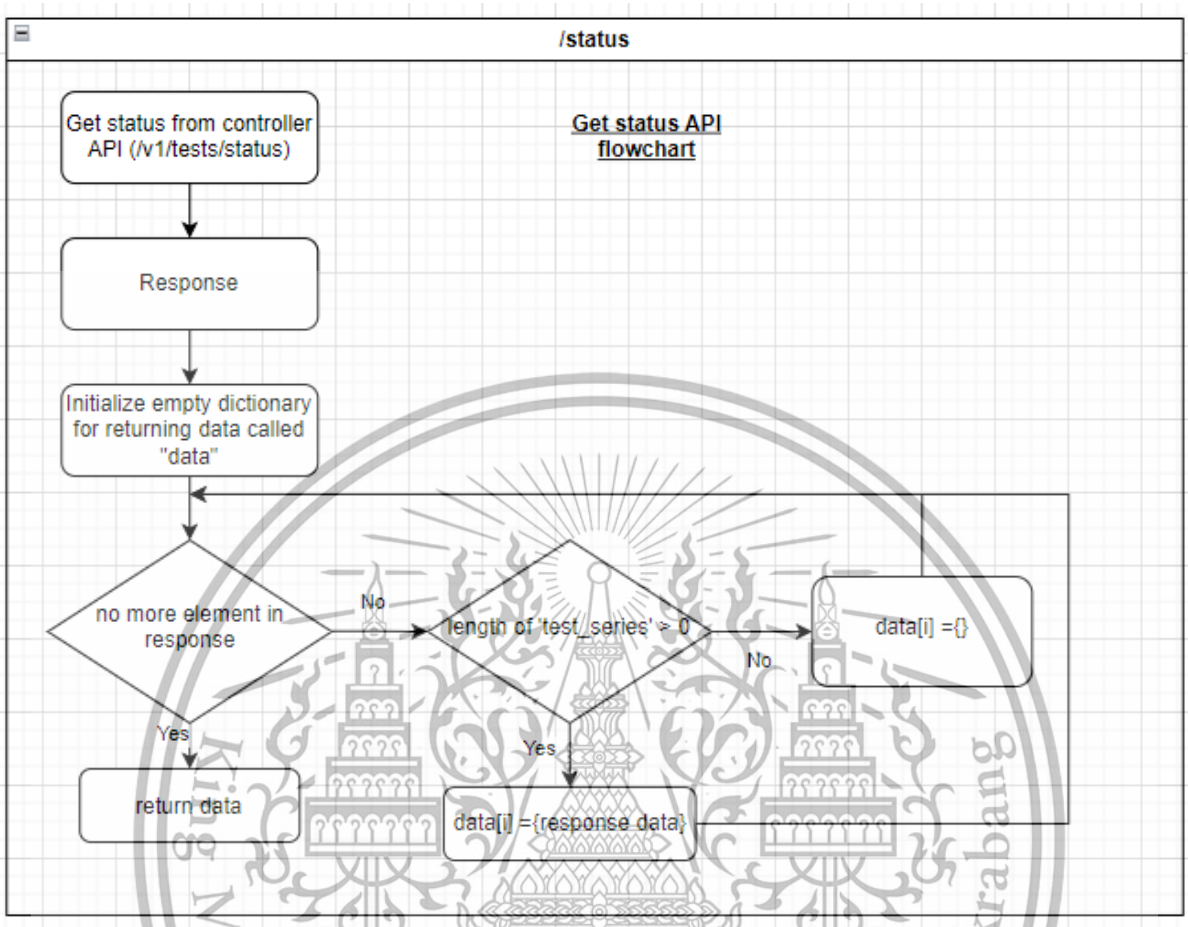

Figure 3.52 Get Test status function

Figure 3.52 shows how the Get Test status function works. It makes an HTTP GET request to a specified URL and retrieves the response data, which is expected to be in JSON format. The response data is processed and organized into a structured dictionary called 'data'. The algorithm iterates over the relevant information in the response and extracts specific data points to populate the 'data' dictionary. The resulting 'data' dictionary represents a summarized view of the retrieved information.

Get Influx Data function

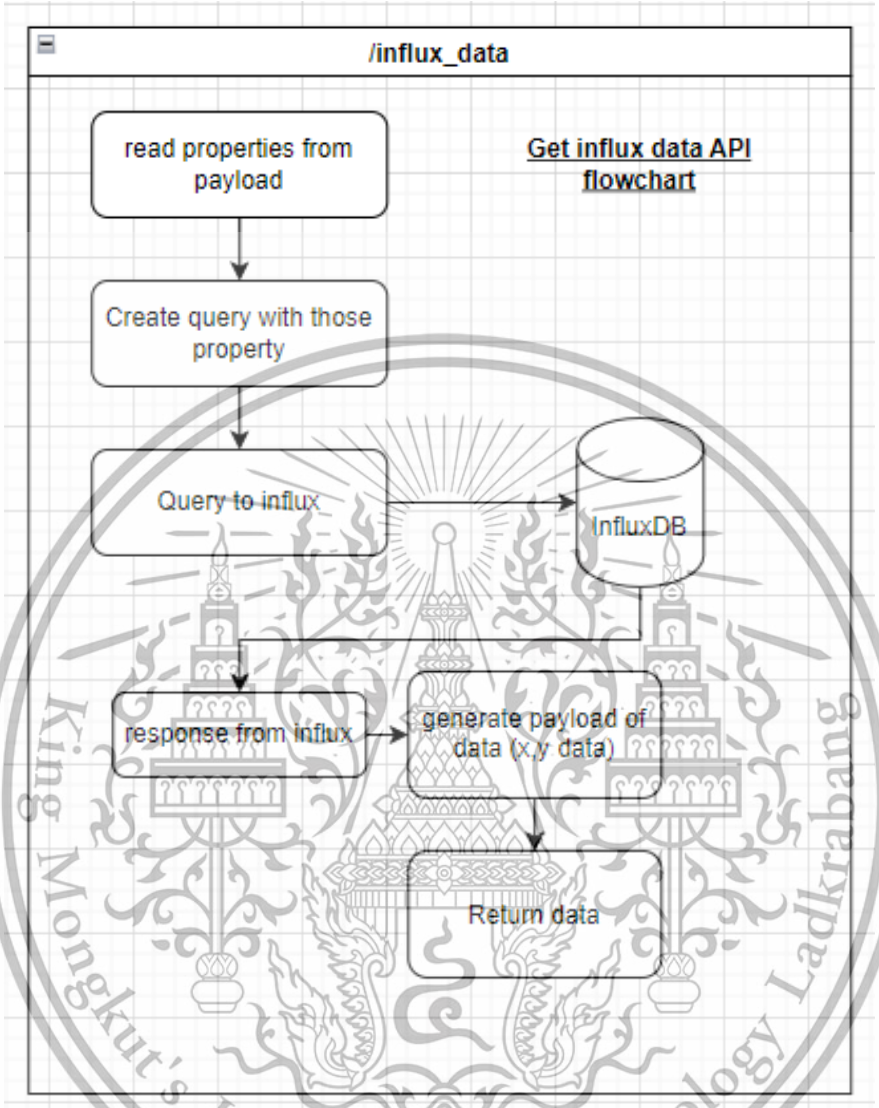

Figure 3.53 Get influx data function

Figure 3.53 shows how the function of Get influx data works. The code initializes a dictionary with predefined graph names and empty lists. It then calls a function to query InfluxDB passing the extracted time range, the dictionary, an empty list, and the measurement. The function populates the dictionary and appends data to the list. The updated dictionary is returned as the response along with a 200 status code.

Get Response Time function

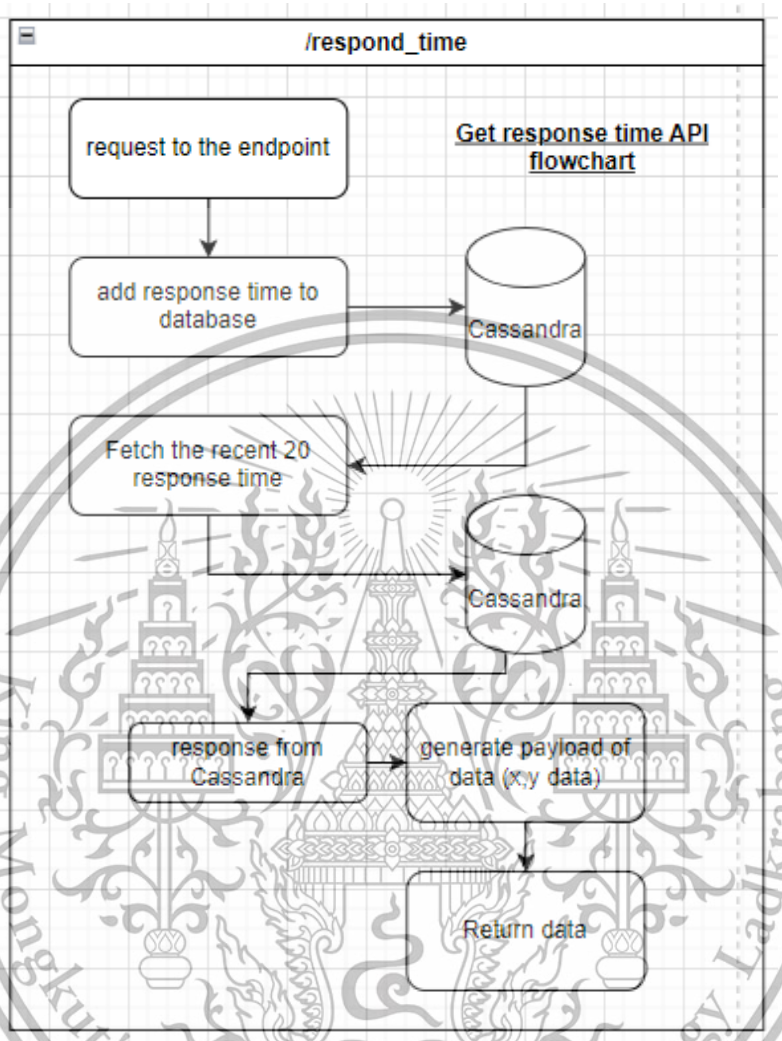

Figure 3.54 Get response time function

Figure 3.54 shows how the Get response time function works. It calls the function 'retrieve_respond_time_func' to retrieve response time data for the given 'app_id'. It then performs data trimming to ensure that the response data has a maximum length of 20. The response data is organized into the 'dict_data' dictionary, which includes 'listx', 'listy', and 'response code'.

3.4.3 Configuration features Activity Diagram

Enter configuration page

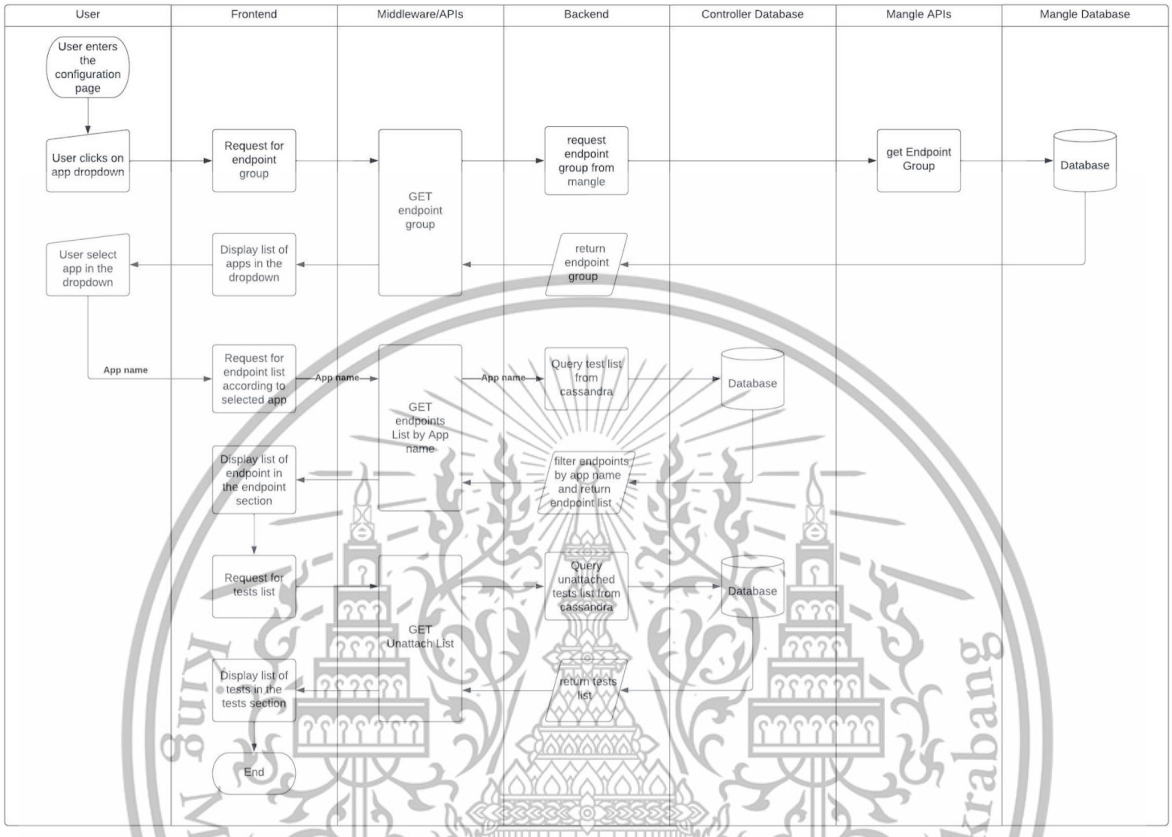

Figure 3.55 Enter configuration page activity diagram

The activity diagram in figure 3.55 Illustrates when the user enters the Configuration page. Once the user clicks the drop down, the frontend will request the application list (endpoint group) for the user to select. The user has to select the application in order to get the list of the VMs endpoint With the same application group. After that, If the user selects the VMs shown in the Configuration page, the frontend will request the test that is unattached to the selected VMs, allowing the user to attach the tests to the VM.

Add a new Endpoint feature

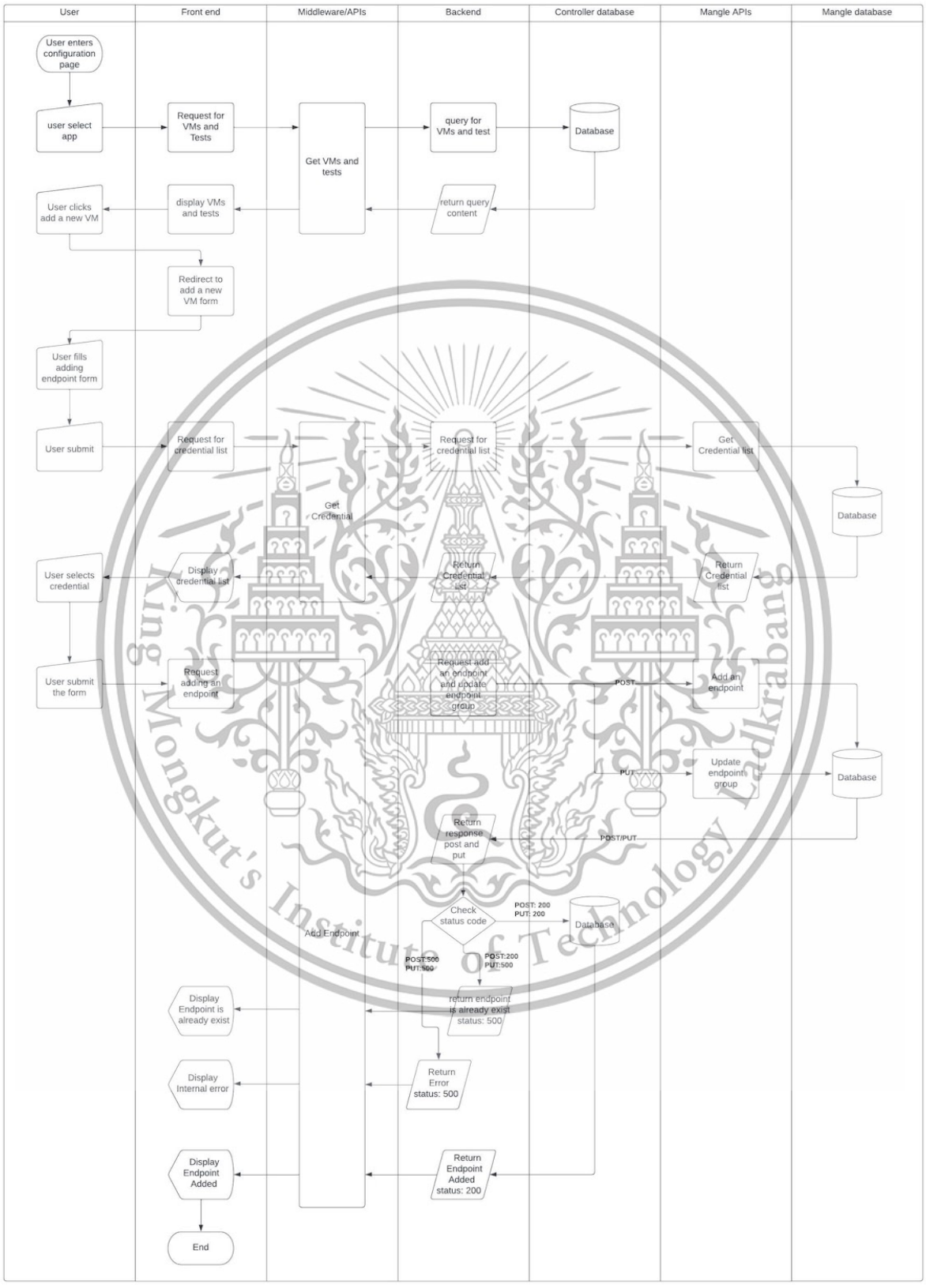

Figure 3.56 Add new endpoint feature activity diagram

The activity diagram in figure 3.56 illustrates when the user clicks add a new endpoint button. The frontend will display the form for the user to fill the settings and configurations of the new endpoint. After the user fills the form and clicks submit, the frontend will request for the endpoint credential list and display in a drop down for the user to select the endpoint credential for a newly added endpoint. Once the user finished filling the form, the frontend will send the request to add a new endpoint form. The backend will do the input validation process. If the input settings are valid, The backend will send a request to the mangle server to add a new endpoint and update the endpoint group member and create the endpoint in the chaos controller database and send a success response to the frontend. If not, The backend will send the error response according to the invalid settings such as no connection established (Wrong IP or Port), wrong endpoint credential, endpoint already exists, and etc.

Remove an Endpoint feature

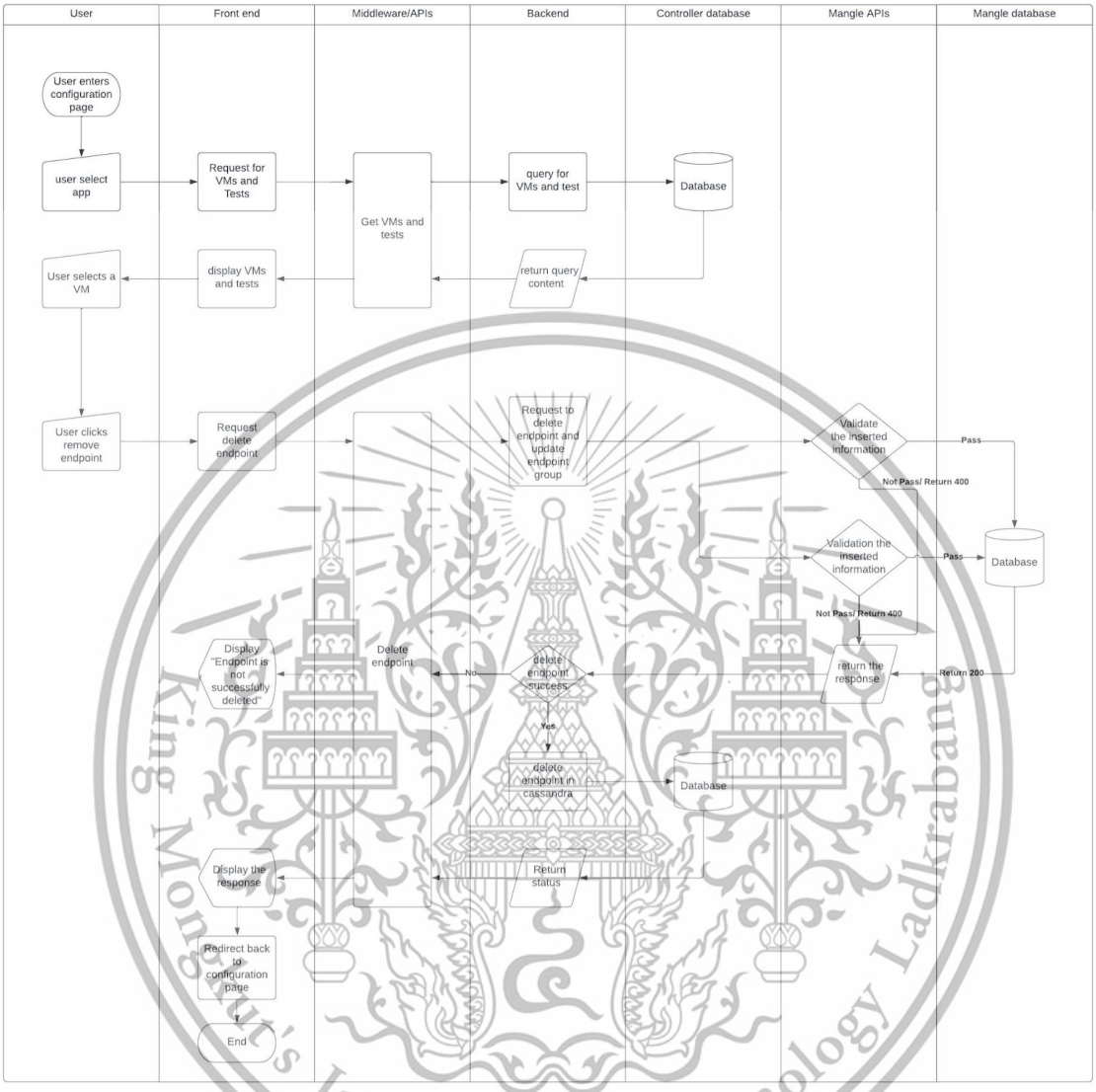

Figure 3.57 Remove an endpoint feature activity diagram

For the remove endpoint in figure 3.57, the user has to select the application and the VM in that application group then clicks delete endpoint. After doing the confirmation, the frontend will send a request to the chaos controller to do the input validation. If the input is valid, the chaos controller will send a request to the mangle server to delete a selected endpoint and delete the endpoint in its database then it will send the success response back to the frontend. If not, It will send the error message to the frontend.

Create Endpoint Credential feature

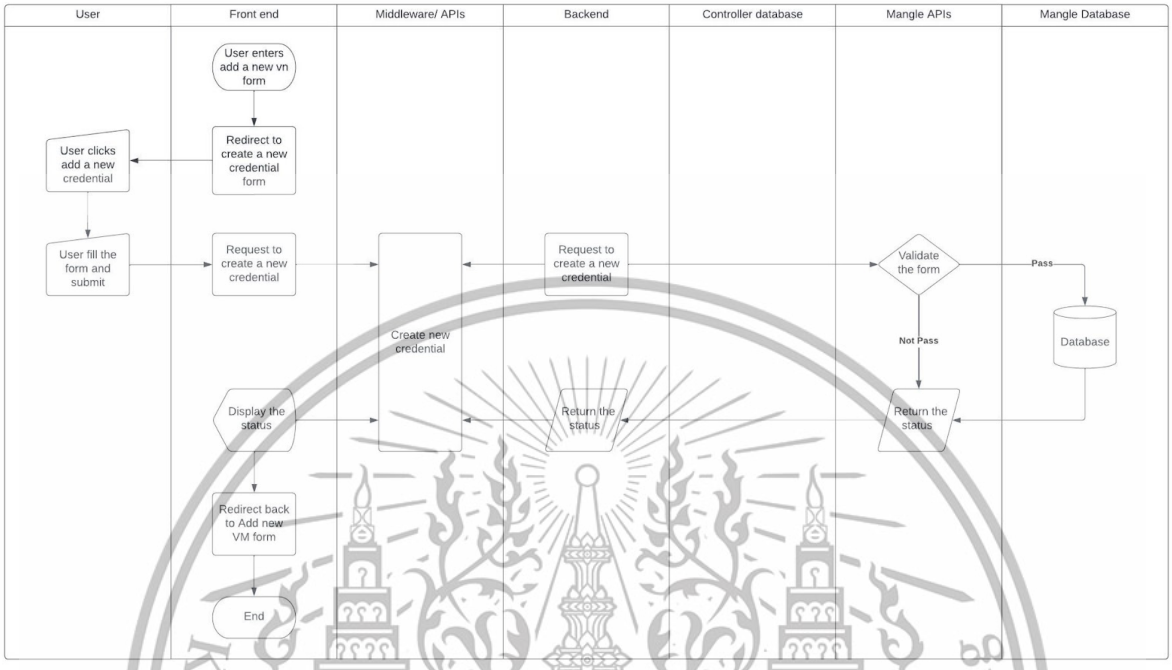

Figure 3.58 Create endpoint credential feature activity diagram

The user clicks create a new credential in the create a new endpoint form displayed on the screen in figure 3.58, The front end will display the create a new credential form. Once the form is submitted, the frontend will send a request to the chaos controller to do the input validation. If passed, the request will be sent to the mangle server and return a success response to the frontend. If not, the controller will return the fail status and the detail to the frontend.

Delete an Endpoint Credential feature

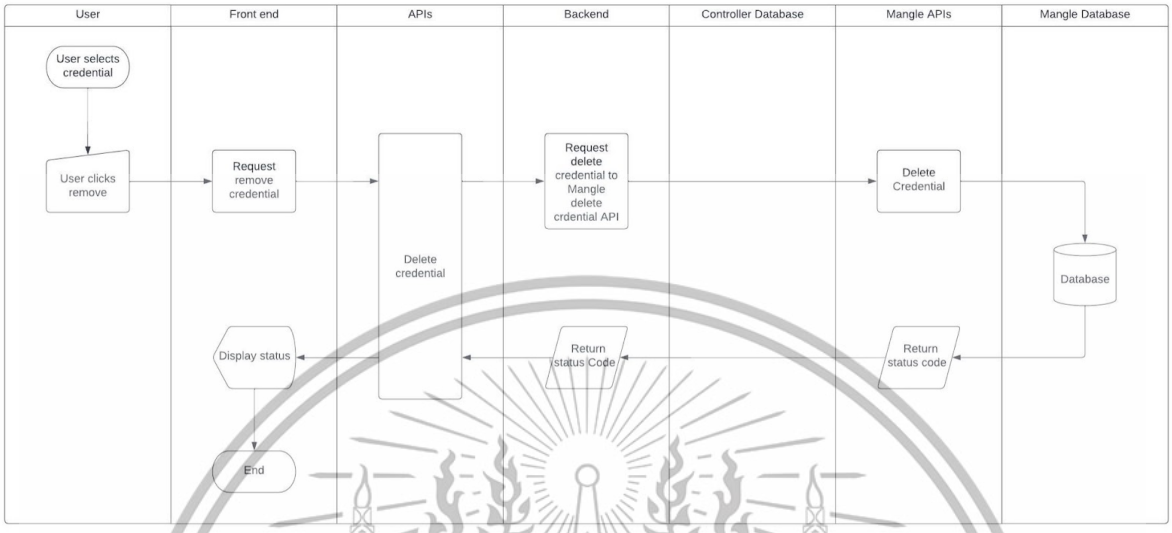

Figure 3.59 Delete an endpoint credential feature activity diagram

The user can select the endpoint credential and clicks remove to delete the endpoint credential in figure 3.59. Once the user clicks remove the credential, the frontend will send the request to the chaos controller to validate the input. If passed, the controller will send the request to mangle server to delete the credential and send the success response to the frontend, If not, it will send the error message to the frontend.

Attach Tests to an Endpoint feature

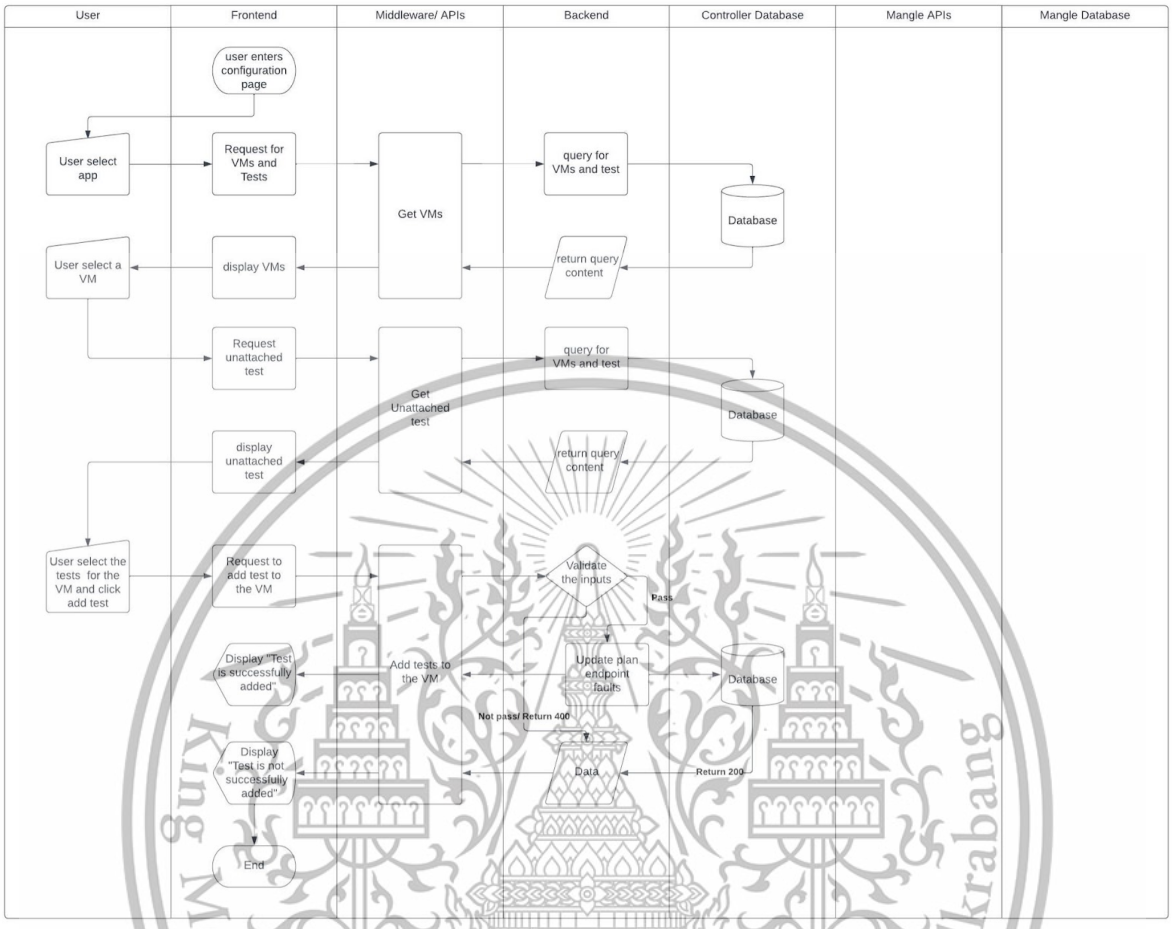

Figure 3.60 Attach tests to endpoint feature activity diagram

Once the user selects the endpoint VM in the configuration page in figure 3.60, the frontend will send a request for the selected VM unattached test, the chaos controller will filter out all the attached tests to the VM and send back the unattached test. This feature allows the user to select multiple tests to attach to the VM. Once the user clicks attach tests, the frontend will send the VM_id and list of test_id to the chaos controller and the chaos controller will update the database accordingly.

CHAPTER 4
EXPERIMENTAL RESULT

4.1 Introduction

This section provides an overview of the experimental results obtained from the project. The experiments were conducted to evaluate the performance, functionality, and effectiveness of the implemented features and system components. The results offer insights into the system's capabilities and highlight its success in meeting the project's objectives. The following subsections outline the key areas covered in the experimental analysis:

1. Test Management Evaluation:

The experimental evaluation focuses on assessing the system's test management capabilities, including the ability to submit or stop tests during their execution and the display of current running tests. The results shed light on the efficiency, reliability, and responsiveness of the test management functionalities.

2. Frontend-Backend Integration Assessment:

The evaluation of frontend-backend integration examines the seamless interaction between the frontend and backend components of the system. The experimental results analyze the performance and effectiveness of the APIs, ensuring smooth communication and a seamless user experience.

3. Test Scheduling Performance:

The experimental evaluation of test scheduling explores the system's ability to execute tests at predetermined times. The results assess the accuracy of test scheduling, the reliability of automated test executions, and the efficiency of resource allocation.

4. Real-time Test Monitoring Analysis:

The analysis of real-time test monitoring investigates the system's capability to display running tests and monitor the health of the application. The experimental results highlight the system's responsiveness, accuracy in monitoring test progress, and effectiveness in identifying potential issues.

5. Resiliency Score Integration Assessment:

The assessment of resiliency score integration focuses on evaluating the system's ability to incorporate resiliency scores derived from response time graphs. The results provide insights into the system's performance, resilience, and ability to handle varying workloads.

6. Dynamic Test Result Display Evaluation:

The evaluation of dynamic test result display analyzes the system's effectiveness in providing real-time feedback on test outcomes. The experimental results assess the accuracy, timeliness, and comprehensibility of the displayed test results.

7. Endpoint Machine Configuration Performance:

The experimental evaluation of endpoint machine configuration assesses the system's flexibility and reliability in configuring the testing environment. The results examine the ease of endpoint configuration, the accuracy of applied configurations, and the system's ability to adapt to diverse endpoint settings.

8. Custom Test Creation and Attachment Analysis:

The analysis of custom test creation and attachment evaluates the system's capability to create custom tests and attach them to specific endpoint machines. The experimental results assess the usability, accuracy, and versatility of the custom test creation process and test attachment functionality.

The experimental results obtained from these evaluations provide valuable insights into the project's achievements and the system's performance. They demonstrate the successful implementation of key features and validate the effectiveness of the solution in meeting the project's objectives.

4.2 Testing Summary

This section presents the evaluation of API test cases conducted to assess the functionality, reliability, and performance of the APIs under test. It is essential to ensure that APIs are thoroughly tested to verify their adherence to requirements and to detect and address any potential issues or vulnerabilities. The objective of this evaluation was to validate the proper functioning of the APIs by executing a comprehensive set of test cases, covering a range of scenarios and conditions.

4.2.1 Backend and APIs Unit tests.

4.2.1.1 Configuration APIs

4.2.1.1.1 Get Endpoint

Tests	Input Parameters	Expected result	Actual result	Result
Request get endpoint	appName (Endpoint_group)	Return list of endpoints that matched with the inserted endpoint_group status code 200	Return list of endpoints that matched with the inserted endpoint_group status code 200	PASS
Request get endpoint (While server is down)	appName (Endpoint_group)	Return internal server error (500)	Return internal server error (500)	PASS

4.2.1.1.2 Add a new endpoint

Tests	Input Parameters	Expected result	Actual result	Result
Request add a new endpoint	appName (Endpoint group), epName (endpoint name), credentialName (endpoint credential name), hostIP (IP address of the endpoint machine), sshPort(Port of the endpoint machine)	Add a new endpoint and update endpoint group members in Mangle DB. Add a new endpoint in the controller database. return status code "Create" 200 and "Update" 200	Add a new endpoint and update endpoint group members in Mangle DB. Add a new endpoint in the controller database. return status code "Create" 200 and "Update" 200	PASS
Request add a new endpoint from non establish socket machine	appName (Endpoint group), epName (endpoint name), credentialName (endpoint credential name), hostIP (IP address of the endpoint machine), sshPort(Port of the endpoint machine)	Return Connection Failure: timeout: socket is not established. return status code 500	Return Connection Failure: timeout: socket is not established. return status code 500	PASS
Request add an existing endpoint	appName (Endpoint group), epName (endpoint name), credentialName (endpoint credential name), hostIP (IP address of the endpoint machine), sshPort(Port of the endpoint machine)	Return endpoint is already exist status code "Create" 200 and "Update" 500	Return endpoint is already exist and status code "Create" 200 and "Update" 500	PASS
Request add an endpoint (While server is down)	appName (Endpoint group), epName (endpoint name), credentialName (endpoint	Return internal server error (500)	Return internal server error (500)	PASS

	credential name), hostIP (IP address of the endpoint machine), sshPort(Port of the endpoint machine)			

4.2.1.1.3 Delete an endpoint

Tests	Input Parameters	Expected result	Actual result	Result
Request delete an endpoint	appName (Endpoint Group), epName (Endpoint name)	Delete an endpoint and update endpoint group member in Mangle DB. delete an endpoint in the controller database. return status code "Delete" 204 and "Update" 200	Delete an endpoint and update endpoint group member in Mangle DB. delete an endpoint in the controller database. return status code "Delete" 204 and "Update" 200	PASS
Request delete non existing endpoint	appName (Endpoint Group), epName (Endpoint name)	Return endpoint is not exist status code "Delete" 500	Return endpoint is not exist status code "Delete" 500	PASS
Request delete an endpoint (While server is down)	appName (Endpoint Group), epName (Endpoint name)	Return internal server error (500)	Return internal server error (500)	PASS

4.2.1.1.4 Get Endpoint Credential

Tests	Input Parameters	Expected result	Actual result	Result
Request get Credential		Return list of credentials status code 200	Return list of credentials status code 200	PASS
Request get Credential (While server is down)		Return internal server error (500)	Return internal server error (500)	PASS

4.2.1.1.5 Add a new endpoint credential

Tests	Input Parameters	Expected result	Actual result	Result
Request add a new endpoint credential	name (credential name), username (credential username), password (credential password)	Create endpoint credential in Mangle DB return status code 200	Create endpoint credential in Mangle DB return status code 200	PASS
Request existed endpoint credential	name (credential name), username (credential username), password (credential password)	Create endpoint credential in Mangle DB (overwrite) return status code 200	Create endpoint credential in Mangle DB (overwrite) return status code 200	PASS
Request add a new endpoint (while server is down)	name (credential name), username (credential username), password (credential password)	Return internal server error (500)	Return internal server error (500)	PASS

4.2.1.1.6 Delete an endpoint credential

Tests	Input Parameters	Expected result	Actual result	Result
Request delete an endpoint credential	name (credential name)	Return Credential is successfully deleted status code 204	Return Credential is successfully deleted status code 204	PASS
Request delete non existing endpoint credential	name (credential name)	Return Credential is not successfully deleted status code 500	Return Credential is not successfully deleted status code 500	PASS
Request delete an endpoint credential (while server is down)	name (credential name)	Return internal server error (500)	Return internal server error (500)	PASS

4.2.1.1.7 Get Tests list

Tests	Input Parameters	Expected result	Actual result	Result
Request get fault list		Return list of fault status code 200	Return list of fault status code 200	PASS
Request get Credential (While server is down)		Return internal server error (500)	Return internal server error (500)	PASS

4.2.1.1.8 Attach tests to endpoint

Tests	Input Parameters	Expected result	Actual result	Result
Request attach tests to an endpoint	list of tests	Return Tests attached status code 200	Return Tests attached status code 200	PASS
Request attach an already attached tests to an endpoint	list of tests	Return Tests attached (Does not have any update) status code 200	Return Tests attached (Does not have any update) status code 200	PASS

4.2.1.1.9 Create a new custom tests

Tests	Input Parameters	Expected result	Actual result	Result
Request create a new test	fault_name (test name), fault_name (type of the test), timeout (timeout), loads (load), params (optional parameter), path (path to test)	return Fault added status code 200	return Fault added status code 200	PASS
Request create a existing new test	fault_name (test name), fault_name (type of the test), timeout (timeout), loads (load), params (optional parameter), path (path to test)	return Fault is already exist status code 500	return Fault is already exist status code 500	PASS

4.2.1.2 Device status page APIs

4.2.1.2.1 Get Response Time tests

Tests	Input Parameters	Expected result	Actual result	Result
Get response time	app_id, typeofdata, start, end	return listx, listy, respondcode status code 200	return listx, listy, respondcode status code 200	PASS
Get average response time	app_id, typeofdata, start, end	return listx, listy, respondcode, avg_respond_time status code 201	return listx, listy, respondcode, avg_respond_time status code 201	PASS
Get response time without or not-existing app_id	app_id, typeofdata, start, end	return app_id not found status code 404	return app_id not found status code 404	PASS
Internal service Error	app_id, typeofdata, start, end	return Network Error status code 500	return Network Error status code 500	PASS

4.2.1.2.2 Get Influx Data tests

Tests	Input Parameters	Expected result	Actual result	Result
Get influx data	measurement, time	return graph1, graph2, graph3, graph4, listx status code 200	return Fault added status code 200	PASS
Get influx data with incorrect measurement	measurement, time	return measurement not found status code 400	return measurement not found status code 400	PASS
Get influx data with incorrect time end before time start	measurement, time	return Incorrect Datetime, Start time is greater than Stop time status code 400	return Incorrect Datetime, Start time is greater than Stop time status code 400	PASS
Get influx data with no datetime range	measurement, time	return Incorrect Datetime, No range status code 400	return Incorrect Datetime, No range status code 400	PASS
Internal service Error	measurement, time	return Network Error status code 500	return Network Error status code 500	PASS

4.2.1.3 Homepage APIs

4.2.1.3.1 Get Previous Tests tests

Tests	Input Parameters	Expected result	Actual result	Result
Get Previous Test		return prev_test_result status code 200	return prev_test_result status code 200	PASS
Internal service Error		return Network Error status code 500	return Network Error status code 500	PASS

4.2.1.3.2 Delete Schedule tests

Tests	Input Parameters	Expected result	Actual result	Result
Delete schedule	sched_id	return Schedule {sched_id} deleted status code 200	return Schedule {sched_id} deleted status code 200	PASS
Delete schedule with not-existing sched_id	sched_id	return Schedule {sched_id} not found status code 404	return Schedule {sched_id} not found status code 404	PASS
Delete schedule without sched_id	sched_id	return Sched_id not found, Please provide sched_id status code 400	return Sched_id not found, Please provide sched_id status code 400	PASS
Internal service Error	sched_id	return Network Error status code 500	return Network Error status code 500	PASS

4.2.1.3.3 Schedule tests

Tests	Input Parameters	Expected result	Actual result	Result
Schedule a test	test_id, time_start, typeofrun, vm_id	return Schedule added status code 200	return Schedule added status code 200	PASS
Run a test	test_id, time_start, typeofrun, vm_id	return Running added status code 201	return Running added status code 201	PASS
Schedule test with incorrect test_id, typeofrun, or vm_id	test_id, time_start, typeofrun, vm_id	return Fault is already exist status code 400	return Fault is already exist status code 400	PASS
Schedule test without specifying typeofrun	test_id, time_start, typeofrun, vm_id	return Please Specify type of run (Schedule/Run now) status code 412	return Please Specify type of run (Schedule/Run now) status code 412	PASS
Schedule test with time_start in the past	test_id, time_start, typeofrun, vm_id	return That time is in the past status code 413	return That time is in the past status code 413	PASS
Schedule test with overlapping time	test_id, time_start, typeofrun, vm_id	return Run/Schedule failed - overlapping time status code 414	return Run/Schedule failed - overlapping time status code 414	PASS
Internal service Error	test_id, time_start, typeofrun, vm_id	return Network Error status code 500	return Network Error status code 500	PASS

4.2.1.3.4 Get Schedule List tests

Tests	Input Parameters	Expected result	Actual result	Result
Get Schedule list		return data status code 200	return data status code 200	PASS
Internal service Error		return Network Error status code 500	return Network Error status code 500	PASS

4.2.1.3.5 Stop Test tests

Tests	Input Parameters	Expected result	Actual result	Result
Stop test	test_id	return Test {test_id} successfully stop status code 200	return Test {test_id} successfully stop status code 200	PASS
Stop test with incorrect test_id	test_id	return test_id is incorrect, cannot find this test_id status code 500	return test_id is incorrect, cannot find this test_id status code 500	PASS
Internal service Error	test_id	return Network Error status code 500	return Network Error status code 500	PASS

4.2.1.3.6 Get Status tests

Tests	Input Parameters	Expected result	Actual result	Result
Get status		return test_status status code 200	return test_status status code 200	PASS
Internal service Error		return Network Error status code 500	return Network Error status code 500	PASS

4.2.2 Front end Unit Tests

4.2.2.1 Frontend - Homepage Unit tests

Tests	Input Parameters	Expected result	Actual result	Result
(Home page) Dropdowns on selection bar	Action: click on select test before app is selected	no change	no change	PASS
	Action: click on select date before app is selected	no change	no change	PASS
	Action: click on select app, test and date in that order	actions corresponding to that to be visible	actions corresponding to that to be visible	PASS
	Action: select test and reselect app	selected test will reset to text= “Choose Test”	selected test will reset to text= “Choose Test”	PASS

(Home page) Pop ups on schedule and previous test panel	Action: on click show all on schedule panel	pop up to be visible on schedule panel	pop up to be visible on schedule panel	PASS
	Action: on click show all on previous test panel	pop up to be visible on previous test panel	pop up to be visible on previous test panel	PASS
(Home page) Navigation of Navbar	Action: on click Coniguration	to navigate to Configuration page on click	to navigate to Configuration page on click	PASS
	Action: on click Manage Test	to navigate to Home page on click	to navigate to Home page on click	PASS

4.2.2.1 Frontend - Configuration page Unit tests

Tests	Input Parameters	Expected result	Actual result	Result
(Config page) Radio button and checkbox on select VM and tests:	Action: click on checkbox on select vm	checkbox click to be truthy on select vm	checkbox click to be truthy on select vm	PASS
	Action: click on checkbox on select vm	checkbox click to be truthy on select vm	checkbox click to be truthy on select vm	PASS
	Action: click on checkbox all on select vm	check all box click to check all	check all box click to check all	PASS

(Config page) Create credential button and input fills:	Action: on click create credential button	credential button to navigate to pop up	credential button to navigate to pop up	PASS
	Action: type input for all three boxes	input fill in boxes	input fill in boxes	PASS
	Action: delete input for all three boxes	error text for non-input	error text for non-input	PASS
	Action: type in correct inputs and click next button	next button to enable upon no error and be truthy	next button to enable upon no error and be truthy	PASS
(Config page) Create test button and input box	Action: on click create tests button and input fills	test button to navigate to pop up	test button to navigate to pop up	PASS
	Action: change fault type upon drop down	changes in box label for test types correspondingly	changes in box label for test types correspondingly	PASS
	Action: type input for all boxes	input fill in boxes to be truthy	input fill in boxes to be truthy	PASS
	Action: type invalid input for timeout	error text for nonpositive number	error text for nonpositive number	PASS
	Action: type invalid input for second box	error text for second box	error text for second box	PASS

	Action: type in correct inputs and click next button	next button to enable upon no error and click next to be truthy	next button to enable upon no error and click next to be truthy	PASS
(Config page) Submit button without selection	Action: click on submit button	alert to be truthy with warning text	alert to be truthy with warning text	PASS

4.2.3 Frontend-Backend Integration Tests

Tests	Input Parameters	Expected result	Actual result	Result
(Home page) Dropdowns on selection bar with API response	Action: Click on drop downs	length equal to API response length, includes JSON of API response	length equal to API response length, includes JSON of API response	PASS
(Home page) Scheduled test panel with API response	upon load	length equal to API response length, includes JSON of API response	length equal to API response length, includes JSON of API response	PASS
(Home page) Scheduled test panel with API response	upon load	length equal to API response length, includes JSON of API response	length equal to API response length, includes JSON of API response	PASS
(Home page) Current test panel with API response	upon load	length equal to API response length, includes JSON of API response	length equal to API response length, includes JSON of API response	PASS

(Home page) Stop button	Action: click on stop button	confirmation dialog to be truthy	confirmation dialog to be truthy	PASS
(Home page) showVMbutton	Action: click on showVM button	alert to be truthy	alert to be truthy	PASS
(Device status) Check on vm test on Current Panel	Action: Click on vm injected test on Current Panel	vm title with status injected	vm title with status injected	PASS
(Config page) Select app panel with API response	Action: on click select app	length equal to API response length, includes JSON of API response	length equal to API response length, includes JSON of API response	PASS
(Config page) API response for vm	Action: on app select	length equal to API response length, includes JSON of API response	length equal to API response length, includes JSON of API response	PASS
(Config page) API response for unattached tests	Action: on vm select	length equal to API response length, includes JSON of API response	length equal to API response length, includes JSON of API response	PASS
(Config page) Delete vm	Action: click delete when vm selected	confirmation dialog to be truthy	confirmation dialog to be truthy	PASS

4.2.4 End-to-end Tests

Tests	Input Parameters	Expected result	Actual result	Result
Create and run tests	Action: Create Network latency test with 30000 ms and 200ms latency	Alert to be truthy upon creation. Upon refresh, test name will be expected	Alert to be truthy upon creation. Upon refresh, test name will be expected	PASS
	Action: Run said created test	Alert and current test panel to be truthy, expect test name to be truthy on drop down and current panel	Alert and current test panel to be truthy, expect test name to be truthy on drop down and current panel	PASS
	Action: Click on test name to device status	Navigate to device status page	Navigate to device status page	PASS
	Action: On timeout after 30seconds +/- 10 seconds	Status to change to succeed	Status to change to succeed	PASS

4.3 Testing Summary

The testing described in APIs test cases demonstrates that all APIs have been tested and are working properly. The testing in frontend test cases also means that the components work in expectation by itself and with integration of the API as well.

The test execution process involved rigorous validation of API responses against expected outcomes. Different scenarios, including valid and invalid data inputs, Mock up data, and non-existed settings were tested to cover a wide range of use cases. The test results demonstrated that all test cases passed successfully, confirming that the APIs are working properly and delivering the desired results.

CHAPTER 5
CONCLUSION

The aim of this project was to build more upon the chaos engineering project. Our purpose is to build a UI and UX that is more pleasant to use and the functions are all integrated without a third party software like Grafana. We also were appointed extra features to code such as configuration and device status. Furthermore, the code base has been improved in maintainibility by having testable units and using the popular React.js framework.

We then designed and implemented an aesthetic web app that could:

- Schedule a test/tests to an endpoint
- Run a test/tests to an endpoint
- View the metrics of the endpoint
- Configure the above features
- Loading and writing data to cassandra cluster database
- Integration with existing controller code

These combined capabilities result in the resilience dashboard.

Challenges were faced during development including poor backend planning, inexperience in setting specific goals, and lack of knowledge in technical areas. This was all summed up to be a mess, specifically speaking extending the time of the project due date. But these mistakes are now learned and the project has been accomplished and ready in the given scope.

5.2.1 Future remarks

An area that can be improved on in the future is the area of deployment. This can be more optimized using macro architectures for load balancing, scaling, concurrency. Security can also be improved with tokenization of API requests and having a login page and a session duration. Tokenization of sensitive data such as IP and credential information is also a good idea.

5.2.2 Challenges

Challenges were faced during development including poor backend planning, inexperience in setting specific goals, and lack of knowledge in technical areas. This was all summed up to be a mess, specifically speaking extending the time of the project due date. But these mistakes are now learned and the project has been accomplished and ready in the given scope.

5.2.3 Key takeaways

- Good planning makes your life easier
- It is best, when working as a team, to work synergistically with your team, advisor, and mentor
- Get requirements as detailed as possible

REFERENCES

1. “Readability: The Optimal Line Length – Articles – Baymard Institute.” *Baymard Institute*, https://baymard.com/blog/line-length-readability. Accessed 2 May. 2023.

2. “What Is User Experience (UX) Design? | IxDF.” *The Interaction Design Foundation*, https://www.interaction-design.org/literature/topics/ux-design. Accessed 15 May. 2023.

3. “What Is User Interface (UI) Design? | IxDF.” *The Interaction Design Foundation*, https://www.interaction-design.org/literature/topics/ui-design. Accessed 15 May. 2023.

4. “7 Gestalt Principles of Visual Perception: Cognitive Psychology for UX.” *UserTesting*, https://www.usertesting.com/blog/gestalt-principles. Accessed 15 May. 2023.

5. “Gestalt Principles in UX Design.” *United Perfectum*, https://unitedperfectum.com/news/gestalt-principles-in-ux-design/. Accessed 17 May. 2023.

6. Yablonski, Jon. “Home | Laws of UX.” *Laws of UX*, https://lawsofux.com/. Accessed 17 May. 2023.

7. “Designing Personality :: UXmatters.” *UXmatters :: Insights and Inspiration for the User Experience Community*, https://www.uxmatters.com/mt/archives/2013/12/designing-personality.php. Accessed 17 May. 2023.

8. “Learning Center for Everything Web Design, System Design, and Prototyping. Find out the Latest Articles, Tutorials, and Resources.” *Bunin’s Design Blog*, https://buninux.com/learn/typography-alignment. Accessed 17 May. 2023.

9. Rekhi, Sachin. “Don Norman’s Principles of Interaction Design | by Sachin Rekhi | Medium.” *Medium*, Medium, 20 May. 2017, https://medium.com/@sachinrekhi/don-normans-principles-of-interaction-design-51025a2c0f33.

10. “Warm vs. Cool Grays.” *TheVirtualInstructor Blog |*, 20 May. 2019, https://thevirtualinstructor.com/blog/warm-vs-cool-grays.

11. “How To Use Typography In UI Design: A Beginner’s Guide.” *CareerFoundry*, https://careerfoundry.com/en/blog/ui-design/typography-ui-design/. Accessed 20 May. 2023.

12. “Spacing, Grids, and Layouts.” *DesignSystems.Com*, https://www.designsystems.com/space-grids-and-layouts/. Accessed 20 May. 2023.

13. Tapaniya, Dalsukh. “Colors in UI Design — Theory, Psychology & Practice | by Dalsukh Tapaniya | Iconscout - Design Assets Marketplace | Medium.” *Medium*, Iconscout - Design Assets Marketplace, 20 May 2023, https://medium.com/iconscout/colors-in-ui-design-theory-psychology-practice-f6d6a5e6e04d.

14. “The Designer’s Guide to Letter-Spacing | Webdesigner Depot Webdesigner Depot » Blog Archive.” *Webdesigner Depot*, 20 May 2023, https://www.webdesignerdepot.com/2020/07/the-designers-guide-to-letter-spacing/.

15. “What Is Visual Hierarchy? | IxDF.” *The Interaction Design Foundation*, https://www.interaction-design.org/literature/topics/visual-hierarchy. Accessed 20 May. 2023.

16. Groeneveld, Nick. “How to Use Shadows in UI Design - The Designer’s Toolbox.” *The Designer’s Toolbox*, 20 May. 2023, https://www.thedesignerstoolbox.com/design-system/ui-design-shadows/.

17. Chapman, C. (2018). The Role of Color in UX. *Toptal Design Blog*. https://www.toptal.com/designers/ux/color-in-ux. Accessed 20 May. 2023.

18. Chapman, C. (2019). Influence with Design – A Guide to Color and Emotions. *Toptal Design Blog*. https://www.toptal.com/designers/ux/colors-and-emotions Accessed 20 May. 2023.

19. Hamann, J. (2021, December 15). Design for personalities, not “users” - UX Collective. *Medium*. https://uxdesign.cc/design-for-personalities-not-users-8db0b5c60511 Accessed 20 May. 2023.

20.