

AUTOMATIC MARKING SYSTEM FOR PROGRAMMING SUBJECT

BY

Chan Jin Yee

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR INFORMATION SYSTEMS (HONOURS)

INFORMATION SYSTEMS ENGINEERING

Faculty of Information and Communication Technology

(Kampar Campus)

JUNE 2023

AUTOMATIC MARKING SYSTEM FOR PROGRAMMING SUBJECT

BY

Chan Jin Yee

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR INFORMATION SYSTEMS (HONOURS)

INFORMATION SYSTEMS ENGINEERING

Faculty of Information and Communication Technology

(Kampar Campus)

JUNE 2023

REPORT STATUS DECLARATION FORM

Title: Automatic Marking System For Programming Subject

Academic Session: June 2023

I CHAN JIN YEE

(CAPITAL LETTER)

declare that I allow this Final Year Project Report to be kept in

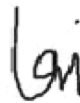
Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.

Verified by,



(Author's signature)



(Supervisor's signature)

Address:

Jalan Universiti, Bandar

Barat, 31900 Kampar,

Perak

Lai Siew Cheng

Supervisor's name

Date: 15 September 2023

Date: 15 September 2023

Universiti Tunku Abdul Rahman			
Form Title : Sample of Submission Sheet for FYP/Dissertation/Thesis			
Form Number: FM-IAD-004	Rev No.: 0	Effective Date: 21 JUNE 2011	Page No.: 1 of 1

Faculty of Information and Communication Technology

UNIVERSITI TUNKU ABDUL RAHMAN

Date: 15 September 2023

SUBMISSION OF FINAL YEAR PROJECT /DISSERTATION/THESIS

It is hereby certified that Chan Jin Yee_(ID No: 19ACB01409) has completed this final year project entitled “Automatically Marking System for Programming Subject” under the supervision of Ms Lai Siew Cheng (Supervisor) from the Department of Computer Science, Faculty of Information and Communication Technology.

I understand that University will upload softcopy of my final year project / dissertation/ thesis* in pdf format into UTAR Institutional Repository, which may be made accessible to UTAR community and public.

Yours truly,



Chan Jin Yee

DECLARATION OF ORIGINALITY

I declare that this report entitled “**AUTOMATIC MARKING SYSTEM FOR PROGRAMMING SUBJECT**” is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature : 

Name : CHAN JIN YEE

Date : 15 September 2023

ACKNOWLEDGEMENTS

I would like to express my sincere thanks and appreciation to my supervisors, Ms Lai Siew Cheng who has given me this bright opportunity to engage in a design system project. It is my first step to establish a career in system design field. A million thanks to you.

ABSTRACT

This project is a web application designed for lecturer that teaching programming subject to automatically mark programming algorithm exercise. Marking programming algorithm is a repetitive process, lecturer will be required to manually key in expected input of algorithm and compare the expected output and the output provided by algorithm done by student again and again. This project create an web application to automatically execution code submitted by student and mark the code according to its correctness. This application handle execution of anonymous code that could be malicious securely by running the code in an isolate environment. To achieve this, this project use cloud compute resources that contain Docker container which act as an isolation for anonymous code execution. This project scale code execution environment and pre-initialized or deallocated code execution environment to execute anonymous code to optimize usage cloud resources. The project had implemented architecture of secure code execution environment which can handle malicious code securely. When the code exceed the resources limit, the code execution environment kill the isolate environment and collect the metadata of why the execution failed. Besides, this project implements and design user interface for user to submit their source code and pass the source code to code execution environment for execution and judgement. Users can add new exercise with test cases to examine the submission program.

Table of Contents

REPORT STATUS DECLARATION FORM	II
DECLARATION OF ORIGINALITY	IV
ACKNOWLEDGEMENTS	V
ABSTRACT.....	VI
TABLE OF CONTENTS	VII
LIST OF FIGURES	XI
LIST OF TABLES	XII
LIST OF ABBREVIATIONS	II
CHAPTER 1 INTRODUCTION	1
1.0 Introduction.....	1
1.1 Problem Statement and Motivation	1
1.2 Project Objectives	2
1.3 Project Scope and Direction.....	3
1.4 Contributions.....	4
1.5 Report Organization.....	5
CHAPTER 2 LITERATURE REVIEW	6
2.0 Literature Reviews	6

2.0.1 Layer Based Approach [3]	6
2.0.2 Microservice Approach [5]	7
2.1 Comparison of Implementation in Existing System with Proposed Approach	12
2.2 Discussion	9
2.3 Limitation of Previous Study	11
2.4 Critical Remark for previous work	13
2.4.1 Experimental Teaching Platform For Computer Software Course [8]	13
2.4.2 HackerRank [9]	16
2.5 Summary of Proposed System with Existing System	18
CHAPTER 3 PROPOSED METHOD/APPROACH	20
3.1 Agile Methodology	20
3.2 Proposed Solution	21
3.3 Sequence Diagram	23
3.4 Networking	25
3.4.1 Proposed Solution to manage data passing	26
3.5 Infrastructure as Code	27
3.6 Front End Development	27
3.6.1 Front End Functional Requirement Refinement	28
3.6.2 Front End Low Fidelity Design	30
3.6.3 Application Database Schema	30
3.7 Clustering with K8s	31
3.7.1 Event Base Autoscaling using KEDA	33
CHAPTER 4 SYSTEM IMPLEMENTATION	34

4.1 Setup Development Environment	34
4.1.1 Load Balancer and Ingress	35
4.1.2 Setup Kind Cluster	35
4.2 Implementation Of CEE	36
Initial Implementation of CEE	36
Improved Version of CEE	37
4.4 Front End	39
4.4.1 Web Server and Web Page Implementation	39
4.4.2 Authentication and Authorization Implementation.	39
4.5 Implementation of system monitoring	41
4.6 Setup Load Testing Tool	44
CHAPTER 5 SYSTEM EVALUATION AND DISCUSSION	45
5.1 Setup Unit Test For CEE	45
5.3.1 Python	46
4.3.2 NodeJS	49
4.3.3 C++	51
4.3.4 Rust	54
5.2 Platform Testing	56
5.3 Load Testing of CEE	66
5.1.1 Performance of CEE in high traffic before scaled	67
5.1.2 Performance of CEE after scaled	69
5.4 Evaluation Discussion	74
5.5 Innovate Features Testing – Code Inject	75
5.5 Project challenges	77
CHAPTER 6 CONCLUSION AND RECOMMENDATIONS	79

6.1 Conclusion	79
6.2 Recommendation	81
REFERENCES.....	82
APPENDIX.....	ERROR! BOOKMARK NOT DEFINED.
FINAL YEAR PROJECT WEEKLY REPORT	1
FINAL YEAR PROJECT WEEKLY REPORT	2
FINAL YEAR PROJECT WEEKLY REPORT	3
FINAL YEAR PROJECT WEEKLY REPORT	4
POSTER.....	1
PLAGIARISM CHECK RESULT.....	1

LIST OF FIGURES

Figure Number	Title	Page
Figure 2.1.1.1	Online Judge architecture	7
Figure 2.1.2.1	Remote Code Execution Engine Architecture	8
Figure 2.1.2.2	Penalty Formula	9
Figure 2.5.1.1	Relationship Between Each Role	14
Figure 2.5.1.2	New Exercise submission interface	15
Figure 2.5.1.3	Submission Interface	15
Figure 2.5.2.1	HackerRank Add Test Case Interface	16
Figure 3.1.0.1	Agile Methodology Diagram	20
Figure 3.2.1	Proposed Architecture	23
Figure 3.3.1	Proposed Sequence Diagram	25
Figure 3.4.0.1	Proposed Network Diagram	26
Figure 3.6.3.1	MongoDB Schema	31
Figure 4.5.1	Prometheus UI	42
Figure 4.5.2	Grafana Dashboard	43
Figure 5.3.1.1	CEE metric before scaled	66
Figure 5.3.1.2	CEE metric after scaled	68
Figure 5.5.1	Code Inject Test Testing	73
Figure 5.5.2	Code Inject Submission Result	74

LIST OF TABLES

Table Number	Title	Page
Table 2.2.1	Comparison of Implementation in Existing System with Proposed System	10
Table 2.6.1	Summary of proposed system with existing system	18
Table 5.1.1.1	Python Test	39
Table 5.1.2.1	NodeJS Test	41
Table 5.1.3.1	CPP Test	43
Table 5.1.4.1	Rust Test	46
Table 5.2.1	Platform testing	59
Table 5.3.2.1	Baseline Load Testing Result	71
Table 5.3.2.2	Spike Testing Result	72
Table 5.3.2.3	Stress Testing Result	73

LIST OF ABBREVIATIONS

CEE	Code Execution Environment
DinD	Docker in Docker
PHPA	Predictive Horizontal Pod Auto-scaler
K8s	Kubernetes
HPA	HorizontalPodAutoscaler
KinD	Kubernetes in Docker

CHAPTER 1 INTRODUCTION

1.0 Introduction

Students that study for computer science or related subject often time involve of coding practice to improve student programming skill and understanding. However, According to [1], lecturers in university that without involve in administration will involve in two to six courses in one semester; moreover, lecturer will require to implement research, entrepreneurship that required by Ministry of Higher Education, another that, lecturer also responsible for supervision of student thesis or final year project. Yet, the paper state that the amount of workload set by university is difficult to achieve for lecturers. Since workload of lecturer is heavy, lecturer will have limited amount of time to give and mark programming algorithm exercise and lead to giving not enough exercise to students. This project is going to propose web application that mainly assist lecturers to mark programming exercises.

1.1 Problem Statement and Motivation

The motivation of this project is to propose automatic marking system for programming subject to increase lecturers' spare time to finish up other workload while able to give enough programming algorithm questions to students and automatically mark the questions so that it is able to increase and evaluate students' understanding about the subject and programming skill. This project also aims to allow lecturers provide flexible teaching style by provide programming exercise that suitable for students' level. The problem involved for this project included:

1. Platform for lecturer to upload their assignment with test cases.

Lecturer required a suitable platform for uploading their algorithm questions with test cases which allow the system to automatically mark accordingly.

2. Lecturer had to carefully execute source code submitted by students.

Lecturer required a secure environment to execute submitted source code from students from being attacked.

3. Assignment submission take too long to execute.

CHAPTER 1

Students often required to have faster response from marking so that they can have time to review or make change according to the result.

Therefore, this study focus develop and design an automatic marking system which allow lecturer to provide programming exercise and automatically mark for the submission base on the test cases provided. [2] Automatic marking system also call judge system often used in competitive programming contest, interview, or educational area. However, this project specially designed for educational purpose. There are many criteria which need to be more emphasize in automatically marking system include:

1. Extensibility. The system can support more programming language.
2. Scalability. The system can scale cloud resources efficiently.
3. Functionality. The system provide functionality to assist student and lecturer. For example, the system help lecturer to mark for the code based on test cases provided.

1.2 Project Objectives

The main objective of this project is to create an automatic marking system with

1. To design automatic marking system which able to execute submission code securely.

The system should be designed in a way that ensures the security of the code execution environment, to prevent any malicious code from affecting the system or compromising its data. The security of the system can be achieved by running the submitted code in an isolated environment using tools such as Docker containers. The automatic marking system should also be designed to scale, allowing it to handle a large number of submissions simultaneously. The aim of this objective is to provide an efficient and effective solution for marking programming assignments while ensuring the security and reliability of the system.

2. To develop platform which allow users to submit their source code for marking.

The platform is a critical component of the system as it will be the primary means by which users interact with the system. The platform should be intuitive and user-friendly to make the submission process easy and seamless for users. Additionally, the platform should provide users with feedback on the submission process, such as confirmation of

CHAPTER 1

successful submission or error messages if there are any issues. Overall, the objective is to create platform that provides a positive user experience while also being functional and effective in facilitating the submission of source code for automated marking.

3. To design scalable microservices architecture which ensure the submission can be done execute in shorter amount of time

The objective of designing a scalable microservices architecture is to ensure that the system can handle an increasing number of users and data without compromising on performance or availability. This can be achieved by designing each microservice to be stateless and horizontally scalable, meaning that additional instances of the service can be added or removed as demand fluctuates. In addition, the objective of ensuring that the modules can connect to each other is critical for the overall functionality of the system. This can be achieved by using well-defined APIs and communication protocols between the services, such as RESTful APIs or messaging queues.

1.3 Project Scope and Direction

This project focus on design a scalable automatic marking system using microservices architecture and scale the cloud resource effectively. Lecturers can use this system to provide assignment which the assignments can be automatically marked by the test cases provided. Besides, instead of restricting the usage of the system to lecturers, students can use this system as note taking platform where they can think in lecturer perspective to provide assignments with test cases.

In this project, there mainly involve of 3 main modules include:

1. Front End Module

This module involve of developing user interface which allow interaction of users with the system. For example, users need to authenticate to use the services, or being authorized to access privileged information, or add new programming exercise, test cases etc. Another that, this module also includes the preparation of relevant data that required by code execution engine module and pass the data to that module.

2. Code Execution Engine Module

This module mainly involve of design of secure and reliable code execution environment that function as isolation from malicious attack. Besides, this module

CHAPTER 1

involves of provide endpoint to front end module to submit their submission to this module for execution.

3. Scaling Module

This module mainly involve of design of cluster monitor system which will track the loads of code execution service and scale on demand.

However, this project has several limitations include:

1. Do not provide in detail scoring based on the test cases. For example, if the submission returns compiled error, the system will mark the result as error instead of 50% mark, 0% mark etc.
2. This system only involves of single source file and don't involve of multiple source file. For example, this system doesn't support of import from another header file in CPP program.
3. This system doesn't provide details analytics into students' performance include time taken to finish the assignment, number of attempts etc.

1.4 Contributions

This project makes several important contributions to the field of automatic marking systems and microservices architecture. Firstly, the system is designed to be scalable and can effectively utilize cloud resources to ensure that it can handle a large number of submissions at once. This scalability ensures that the system can meet the needs of lecturers and students alike, regardless of the size of the user base.

Secondly, the project contributes to the development of microservices architecture. By breaking the system down into three main modules, the system is designed to be modular and extensible. This modular design makes it easy to add new features or modify existing ones, without affecting the other modules.

Thirdly, the system is designed to be user-friendly and can be used by both lecturers and students. This approach ensures that the system can be used by a wide range of users, from experienced programmers to those who are new to coding.

Finally, the project also contributes to the development of secure and reliable code execution environments. The system is designed to be secure and reliable and can

CHAPTER 1

isolate the code execution environment from malicious attacks. This ensures that the system can be used safely and securely by users.

Although the project has several limitations, such as not providing detailed scoring based on test cases and not supporting multiple source files or customizing the code execution environment configuration, the project's contributions are significant and represent an important step forward in the development of automatic marking systems and microservices architecture.

1.5 Report Organization

This report is organized into five main chapters. Chapter 1 provides an introduction to the project, including the background and motivation for the work, as well as an overview of the structure of the report.

Chapter 2 presents a comprehensive literature review that provides a detailed analysis of the existing research and knowledge related to the project topic. This chapter also highlights gaps and limitations in the current literature and identifies opportunities for further research.

Chapter 3 outlines the proposed methodology for the project, including the design system, project planning and analysis methods, and any tools or software that will be used. This chapter also includes a detailed description of the project timeline and resources needed.

Chapter 4 describes the system implementation of what have plan in Chapter 3, including setup development environment or tool.

Chapter 5 presents evaluation of the system implementation including platform testing, code execution performance evaluation, code execution engine unit testing.

Finally, Chapter 6 summarizes the key findings and conclusions of the project and provides recommendations for future research or applications.

CHAPTER 2 LITERATURE REVIEW

2.1 Literature Reviews

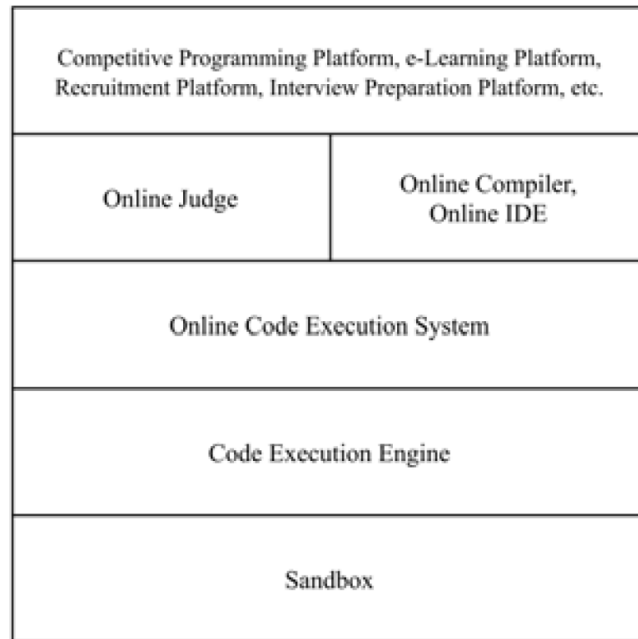
Automated marking systems for programming subject have become increasingly popular in recent years, as they offer numerous benefits for both educators and students. With the rise of online learning and remote education, such systems have become even more important, providing a way to assess students' programming skills and offer feedback at scale. In this literature review, we will examine existing research on automated marking systems for programming exercises, focusing on their design, implementation, and effectiveness. We will explore various approaches and techniques used in these systems, as well as their strengths and limitations. Finally, we will identify gaps in the current literature and suggest areas for future research.

2.1.1 Layer Based Approach [3]

A layer-based approach is a software design approach that organizes the system or application into distinct layers or functional components. Each layer is responsible for a specific set of functionalities and interacts with the layers above and below it through well-defined interfaces [4]. This paper focuses on the study of an existing online code execution system called Judge0. The online code execution system is a complex system that involves several components such as sandbox environment, code execution engine, online code execution engine, online compilers, and IDEs, and online judges. Judge0 used layer-based approach to organized their project and each component in this system is being organized into distinct layer as shown in figure below.

The study conducted by [3] presents a comprehensive overview of the different components of a judge system and the challenges that need to be considered during the process of building the system. The study highlights the importance of scalability, robustness, ease of use and deployment, and security and sandboxing in the design and implementation of the system. To achieve high scalability, Judge0 used both vertical and horizontal scaling strategies in its system. In addition, to ensure robustness, Judge0 has the ability to catch unexpected behaviour and mark the submission as "Internal Error" accompanied by an error message.

Moreover, Judge0 has made the submission process easy and efficient for users by exposing endpoints for creating new submissions for one or more available language and another endpoint to fetch submission results. The system has also utilized Docker technology to create and store the sandbox image in an online registry, which ensures the security and sandboxing of the system.



2.1.1.1 Online Judge architecture

Figure above show the architecture of online judge with each component which each component being organized into distinct layer. Besides, each component can be deployed using docker container which allow to scale flexibly according to their resource's consumption. Overall, the study provides a valuable insight into the architecture, design, challenges, and comparison of Judge0 with other online code execution systems and online judge systems.

2.1.2 Microservice Approach [5]

The microservice approach is a software development architectural style that structures an application as a collection of small, independent, and loosely coupled services. Each service runs in its own process and communicates with other services through well-defined APIs [6]. The main idea behind this approach is to break down a complex system into smaller, more manageable and scalable services that can be developed, deployed, and scaled independently of each other.

This paper explores the architecture and concept of building a cloud-based web service for remote code execution to provide workspace to developers to experiment with different programming languages and technology while designing software applications using microservices. The proposed architecture involves a router service responsible for front-end web content that allows users to write and submit their code for execution. The code is then submitted to another service based on the programming language of the code submitted. Each run programming language service is an instance of a docker image that specializes in executing code for a certain language. The paper proposes using predictive horizontal autoscaling as a scaling strategy, which will forecast the usage of services in the future and create the services beforehand. The tool used to achieve this strategy is the Predictive Horizontal Pod Autoscaler (PHPA), built on top of Kubernetes Horizontal Pod Autoscaler.

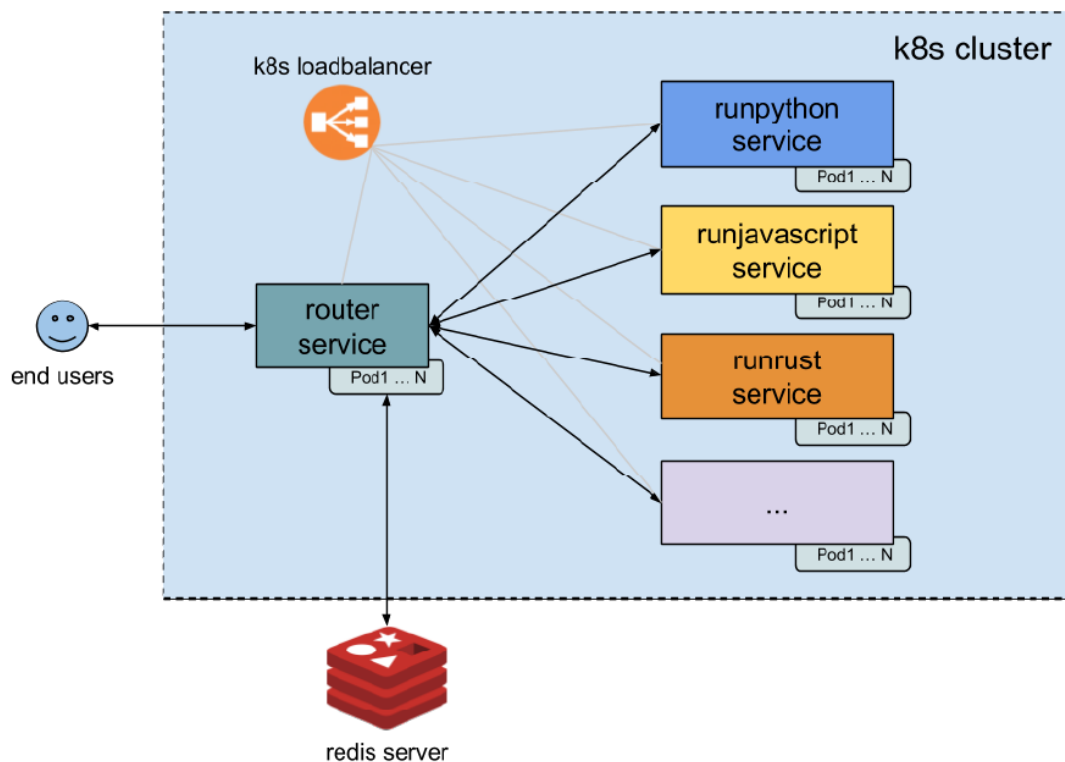


Figure 2.1.2.1: Remote Code Execution Engine Architecture

The paper used EKS service in AWS to set up the infrastructure needed, and Kubernetes was used to configure the services needed in EKS. To provision the infrastructure, the paper used CloudFormation as the Infrastructure as Code tool, which is specially created by AWS to provision infrastructure. The proposed architecture provides

benefits of autoscaling of remote code execution as needed by using a load balancer in Kubernetes services object as shown in above diagram. Each runlang service provides a web server that exposes a RESTful API, which router will redirect the code to runlang server using the RESTful API and runlang server will execute user code in a particular environment.

$$\frac{\sum_{i=0}^{\# \text{ of batches}} (\text{successful request count}(i) * \text{average response time}(i)^2) + (\text{failed request count} * \text{failed request penalty}) + (\text{average pods replica count} * \text{cloud resources scarcity})}{\sum_{i=0}^{\# \text{ of batches}} \text{successful request count}(i)}$$

Figure 2.1.2.2 Penalty Formula

The paper's proposed scaling strategy is evaluated using the PHPA tool, which allows the use of a machine learning model to forecast the future needs of the runlang server and scale up and down accordingly. The effectiveness of the PHPA tool is evaluated through a penalty formula created by the author of the research paper showed in above figure. The testing environment will batch request in five-second periods and asynchronously fire them together to the services. Overall, the paper proposes an architecture and concept for building a cloud-based web service for remote code execution that provides a workspace to developers to experiment with different programming languages and technologies while designing software applications.

2.2 Discussion

Based on paper review in this section, there are several important components that had been summary to build automatic marking system:

1. Sandboxing environment

Sandboxing allows create isolate environment to execute untrusted code with limited system resources to prevent exploitation. In context of automatic marking system, when the untrusted code submitted to code execution environment that build on top of sandbox environment, the code execution environment can execution the code without impact toward computer system. Furthermore, even the submitted code is malicious, sandbox environment can be "clean up" and recreate another sandbox environment for another code execution environment.

2. Scaling code execution environment

In context of automatic marking system, many users may use the code judge system at the same time, if judge system doesn't provide scaling functionality of remote code execution environment, user may need to wait code execution of other users to finish before execution his code, this results long waiting time to get the output of the code where could result in bad user experience. With the help of scaling features, code execution environment can be prepared before users submit their code which can reduce wait time of users to finish the code execution.

There are two type of scaling included horizontal scaling and vertical scaling. Horizontal Scaling is scaling strategy that increase code execution environment in single computer whereas vertical scaling is scaling strategy that create another computer as code execution environment. Decision of scaling up and down of code execution environment can be based on various factor. For example, when the CPU usage of one computer is above 75%, then scale up another computer as code execution environment. With this method, the traffic can be load-balance between these two environments.

3. Code execution environment

Code execution environment will build on top of sandbox environment to avoid from cyber-attack. Furthermore, code execution environment required to have pre-installed code executor to execute code for particular programming language. For example, python interpreter needs to pre-install in the code execution environment before executing the code. However, automatic marking system often not only involve one programming language, so an extensible environment needs to be designed to execute various programming language. There are two ways have been introduced in literature review included:

1. Include all relative interpreter and compiler into one image to execute every supported programming language.
2. Separate code execution environment based on programming language. For example, there are 2 type of code execution environment, 1 for execute python code, another 1 execute cpp code.

2.3 Limitation of Previous Study

There are three important factor that had been discussed in previous section about previous study to build an automatic marking system. This section will discuss on limitation of previous study based on those factors.

Scaling Strategy

Scaling strategy implement in [3] no doubt able to handle multiple requests parallely. However, when reach certain point where the request hit the scaling threshold, users may experience slow response to wait CEE creation. However, this issue can be minimized by using predictive auto-scaling introduce by [5] where scale up and down of CEE based on machine learning model that had been train with previous workload data. This solution will scale up and down based on future need of the CEE and pre-initialized the CEE to achieve better resource management and better user experience. However, this machine learning model is implemented based on past workload data pattern, the model might not be able to accurately predict the users need if the number of users is changed.

CEE Implementation

CEE is sandbox environment that provide safety code execution environment for untrusted code. [3] install all dependencies into one image for code execution environment. When users request for the code execution, the container which contain one or more dependencies will be launch for execute users' code. In other word, same container can execute more than one programming language. However, this method will cause space of image of CEE container increase if the number of supported language increase and lead to waste of cloud resources since each CEE will only execute for one programming language. Besides, [5] mention that this implementation may cause difficulty in maintenance for different language ecosystem. For [5], the implementation is separate programming language in each container. In other word, each container will only handle one programming language execution. Developer needs to manage cloud resource to support CEE for each programming language. In short, implementation of [5] is less manageable since the developer need to manage and maintain every single CEE docker image.

Extensibility

Technology evolve over time and more new programming language may release in future. So, CEE with great flexibility that able to accommodate with new programming language is important. If [5] extend new programming language, it requires to create another new container image with installed compiler or interpreter. This implementation had issue mention in CEE implementation section which this implementation is less manageable because this method will need to create many containers image file that only suit for one programming language. In [3], it requires to create new container image that suit for new programming language environment. However, this implementation will cause container image size to become larger in the CEE need to support more programming language where could lead to waste of cloud resources and increase latency to pull that image to create container.

2.4 Comparison of Implementation in Existing System with Proposed Approach

Table 2.3.1 Table of comparison of implementation of system

	Layer Based [3]	Microservice Based [5]	Proposed System
Scaling Strategy for CEE	CEE scaled horizontally and vertically	Predictive horizontal pod auto scaling	Event-Base horizontal pod autoscaling
CEE implementation	Install compiler and interpreter into one sandbox before running	Each container only supports one interpreter or compiler	Install compiler and interpreter into one sandbox during initiate of sandbox
Extensibility of new programming language	Recreate current CEE image with new environment	Create another CEE with new environment image without change of current image	Recreate CEE deployment which include environment variable of new

			programming language details.
--	--	--	----------------------------------

In the proposed system, Kubernetes event driven horizontal pod autoscaling will be used together with horizontally scaling for CEE. In this project, Kubernetes Event-Driven Autoscaling (KEDA) technology will be used, it is a Kubernetes components to scale any container based on the event needing to be processed [7]. For the implementation of CEE, there will be two components involved, sandbox and worker. Worker will handle the preparation of source code, test case, and the sandbox configuration. Then, worker will connect to sandbox and launch container with desired configuration, pass source code and input to sandbox, check for sandbox status, collect execute output or logs. CEE will be scaled based on the number of task in the queue. This process is to improve the user experience so that user submission code can judge and execute faster. Talk about CEE, the proposed CEE will involve in download docker images for related programming language compiler or interpreter docker images to perform source code compilation or execution, the compiler or interpreter docker images registry will be specified in environment variable along with its language details. Instead of download compiler or interpreter to the sandbox, this approach is much more manageable when the language updated. For example, when python programming updated from version 3.8 to 3.9, there is not involve of reinstall python interpreter in the sandbox and recreate the CEE images. Besides, this approach also make the CEE more flexible on adding or removing supported programming language. This would improve the manageability of CEE compare to [3] and [5]. Proposed system provide better extensible and manageable CEE compare to another reviewed system. Proposed system provide scalable architecture of CEE allow to avoid waste of cloud resources and improve performance.

2.5 Critical Remark for previous work

2.5.1 Experimental Teaching Platform For Computer Software Course [8]

This paper describes the use of an online judge system as a teaching tool for computer programming courses. The authors argue that traditional teaching methods in computer programming may not be sufficient to effectively teach the skills required for

programming, and that an online judge system can help provide students with hands-on programming experience and encourage self-directed learning. This paper concludes some requirement for online judge for different role involve in online judge system. The role includes student, lecturer and administrator. Students' role able to try out the assignment, access the forum. Besides, lecturer able to release assignment with relevant test cases settings. As administrator, management toward the system can be perform like forum management, security settings and system settings. Below figure shown the relationship among different roles.

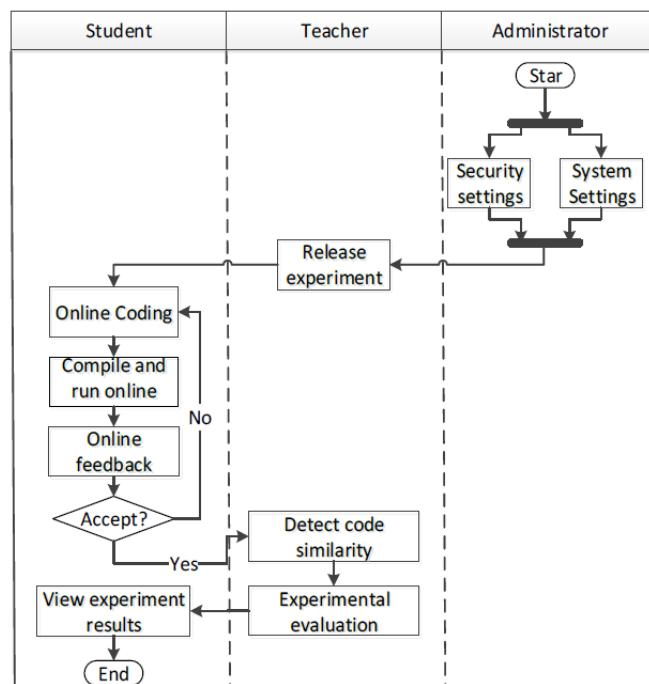


Figure 2.5.1.1 Relationship Between Each Role

Another that, this paper proposed the pipeline of code judging. First, the submission code will submit to background server. The server will later compile and execute the source code by the input specified. The result from the execution will be recorded and return to students for further operations. Below figure shown that the interface provided to by this system to allow lecturer to submit their new exercise. Lecturer can set run time limit and memory limit for the test case. However, lecturer can only set one test case for each exercise which clearly unable to test all possible outcome for correct answer.

Lab Number	<input type="text" value="1.2"/>		
Lab Title	<input type="text" value="Linked list merging"/>		
Lab Content	<p>Create two ascending ordered linked lists, merge the two lists into an ascending ordered list and output it. Input: 3 6 8 -1 2 4 9 -1 Output: 2 3 4 6 8 9 Notice : The entry of the linked list should end with - 1</p>		
Input case	<input type="text" value="3 6 8 -1 2 4 9 -1"/>	Output case	<input type="text" value="2 3 4 6 8 9"/>
Run time limit(S)	<input type="text" value="10"/>	Memory limit(B)	<input type="text" value="65535"/>
Programing language	<input type="text" value="C++"/>	Compiler	<input type="text" value="VS"/>
<input type="button" value="Submit"/>			<input type="button" value="Back"/>

Figure 2.5.1.2 New Exercise submission interface

Besides, if this project does provide code editor for students to write their code before submitting to backend for execution and judgement. However, this project only allows students to submit single page code for execution. Below figure shown the platform provided to students to write their answers or submission code.

```

#include "stdio.h"
#include "stdlib.h"
#include <iostream>
using namespace std;

typedef struct node{
    int data;
    struct node *next;
}Lnode,*Linklist;

Linklist A,B,C;

Linklist init_linklist(){
    Lnode *s,*r,*L;
    L=new Lnode;
    L->next=NULL;
    r=L;

```

Figure 2.5.1.3 Submission Interface

The role classification in paper [8] reduce the flexibility which only allow lecturer to set questions. Instead, each student can be lecturer where student have the capability to provide questions with test cases. This can make the learning process more interactive

Chapter 2

where student can think like lecturer to provide questions and learn through the process during questions brainstorming.

Advantages

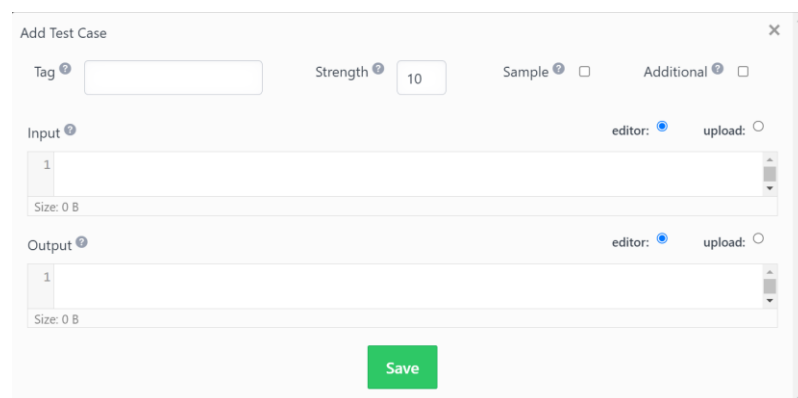
1. Allow lecturer to manage the test case configuration like memory usage, run time limit etc.
2. Multiple programming language support for execution.

Disadvantages

1. Role classification made the students' learning process less flexible which not allow students to create their own assignment.
2. Only support single programming language in one questions
3. Only support one test case.

2.5.2 HackerRank [9]

HackerRank is a popular platform used by many organizations for conducting coding assessments and evaluating programming skills. As an automatic marking system, HackerRank provides several benefits to clients. HackerRank saves users the time and effort required to manually evaluate candidates' coding submissions. Instead, the system automatically evaluates the code and provides a score and feedback based on predefined criteria. Below figures shown how users can add test case in HackerRank, users can set the test case as sample test case to help user to debug their source code. If the test case is categorized as sample test case, the test case will not be executed in test run but will be executed in submission.



The screenshot shows the 'Add Test Case' interface in HackerRank. It features a 'Tag' input field, a 'Strength' dropdown set to '10', and two checkboxes for 'Sample' and 'Additional'. Below these are two sections for 'Input' and 'Output', each with a text area containing the number '1' and a 'Size: 0 B' label. Each section has radio buttons for 'editor' (selected) and 'upload'. A green 'Save' button is located at the bottom center of the form.

Figure 2.4.2.1 HackerRank Add Test Case Interface

Chapter 2

Another advantage of HackerRank is its ability to provide standardized assessments to all candidates. Users can create custom tests with specific questions and criteria, which ensures that all candidates are evaluated on the same basis. This makes the evaluation process fairer and more transparent.

Furthermore, the platform provides detailed analytics and insights into the candidates' performance, including the time taken to complete the test, the number of attempts made, and the overall score. This data can help clients identify the top performers and make data-driven decisions when selecting candidates.

From a users' perspective, the use of HackerRank as an automatic marking system also ensures that the evaluation process is consistent across all candidates. This reduces the risk of errors or biases that may arise in manual evaluation.

However, HackerRank only allow input through stdin to examine the output. This can be trouble for certain scenarios where the program doesn't need stdin to examine the output. For example, if user need to test for a function and different parameter, user need to purposely write stdin and pass the value to function parameter. This could reduce the users experience where users need figure out the proper format for the input which would possibly lead to unexpected output.

In conclusion, HackerRank is a valuable tool for clients looking to evaluate the coding skills of candidates. As an automatic marking system, it provides standardized assessments, saves time and effort, and ensures consistency in the evaluation process. However, it should be used in conjunction with other evaluation methods to ensure a holistic assessment of candidates' skills.

Advantages

1. provides detailed analytics and insights into the candidates' performance, including the time taken to complete the test, the number of attempts made, and the overall score.
2. Users can create custom tests with specific questions and criteria, which ensures that all candidates are evaluated on the same basis.
3. Support multiple language in one question.
4. Allow to customize time limit and memory limit for test the program.

Disadvantages

1. Only allow input through stdin to examine the output.

2.6 Summary of Proposed System with Existing System

Table 2.6.1 Summary of proposed system with existing system

Features	Experiment Teaching Platform[8]	HackerRank [9]	Proposed System
Create custom test cases with input and output	Yes	Yes	Yes, test case with input, code inject and output.
Support multiple programming language	Yes	Yes	Yes
Support multiple programming language in one question	No	Yes	No
Customizable test environment	Yes	Yes	Yes
Support multiple test cases	No	Yes	Yes

In summary, proposed system will allow users to set custom and multiple test cases for test the correctness of the submission. Instead, proposed system allow user to inject

Chapter 2

code to test the submission program. For example, users can set inject value to function parameter to test the correctness of the submission as shown below in nodejs program:

```
function max(a, b){  
  // User code here  
}  
console.log(max("<replace-a>", "<replace-a>"));
```

Users can set to inject value "<replace-a>" and "<replace-b >" to another value to test the correctness of the submission. This can reduce the burden of users to design suitable stdin format and pass the value to submission program. Besides, proposed system has multiple programming language availability. Besides, proposed system allow users to customize the execution environment like time limit and memory limit, the execution result will tell if the submission is over time limit or memory limit. Proposed system only support one programming language in one exercise, users need to create another exercise with another programming language to create exercise with another language. Proposed system will provide tool to users to test the submission program.

CHAPTER 3 PROPOSED METHOD/APPROACH

3.1 Agile Methodology

Agile methodology is a software development approach that emphasizes iterative, incremental, and flexible delivery of working software [10]. It breaks projects down into several dynamic phases, commonly known as sprints, and emphasizes adaptability, quick delivery, and collaboration.

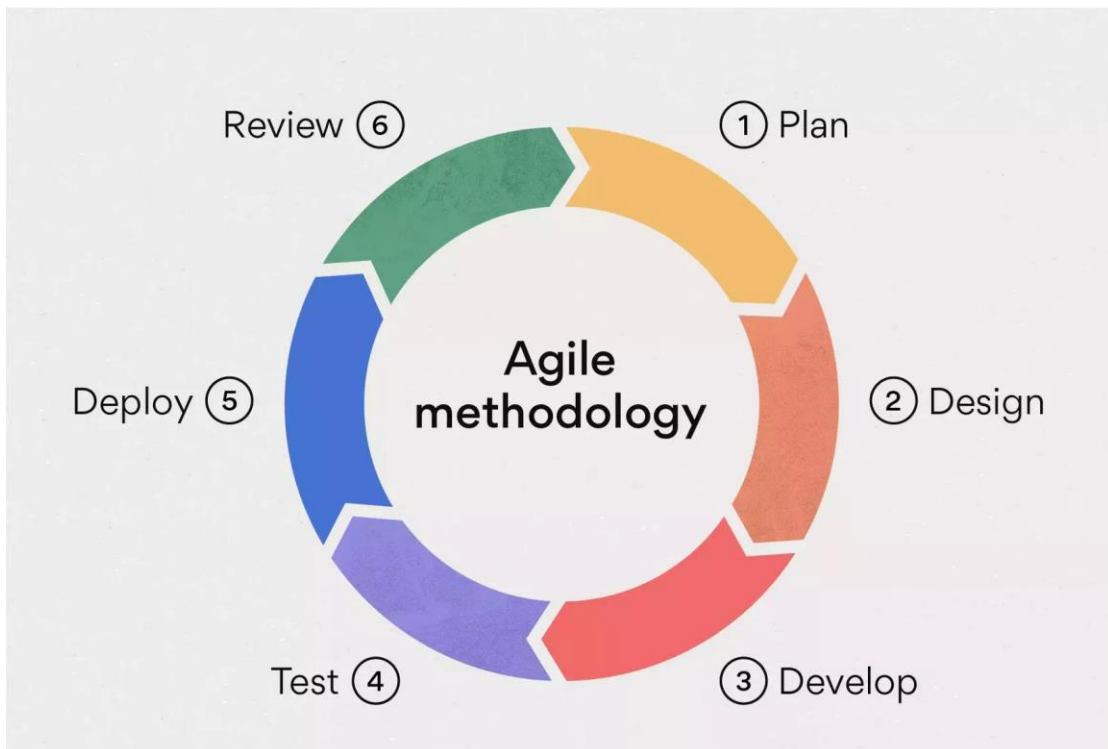


Figure 3.1.0.1 Agile Methodology Diagram [10]

Agile methodology is characterized by the following practices:

1. Iterative and incremental development - software is delivered in small increments, allowing for frequent feedback and adjustments.
2. Cross-functional teams - a team of developers, testers, and stakeholders works together to deliver software.
3. Empirical process control - progress is monitored and controlled through continuous inspection and adaptation.
4. Continuous delivery - working software is delivered frequently, allowing for faster feedback and continuous improvement.

- Customer involvement - customers are closely involved in the development process, providing feedback and guidance.

For this project, Agile methodology is chosen because it aligns well with the goals of my project. One of the main objectives of my project is to design a scalable automatic marking system using microservices architecture, and Agile methodology provides a flexible framework that allows me to continuously refine and improve the design based on feedback.

3.2 Proposed Solution

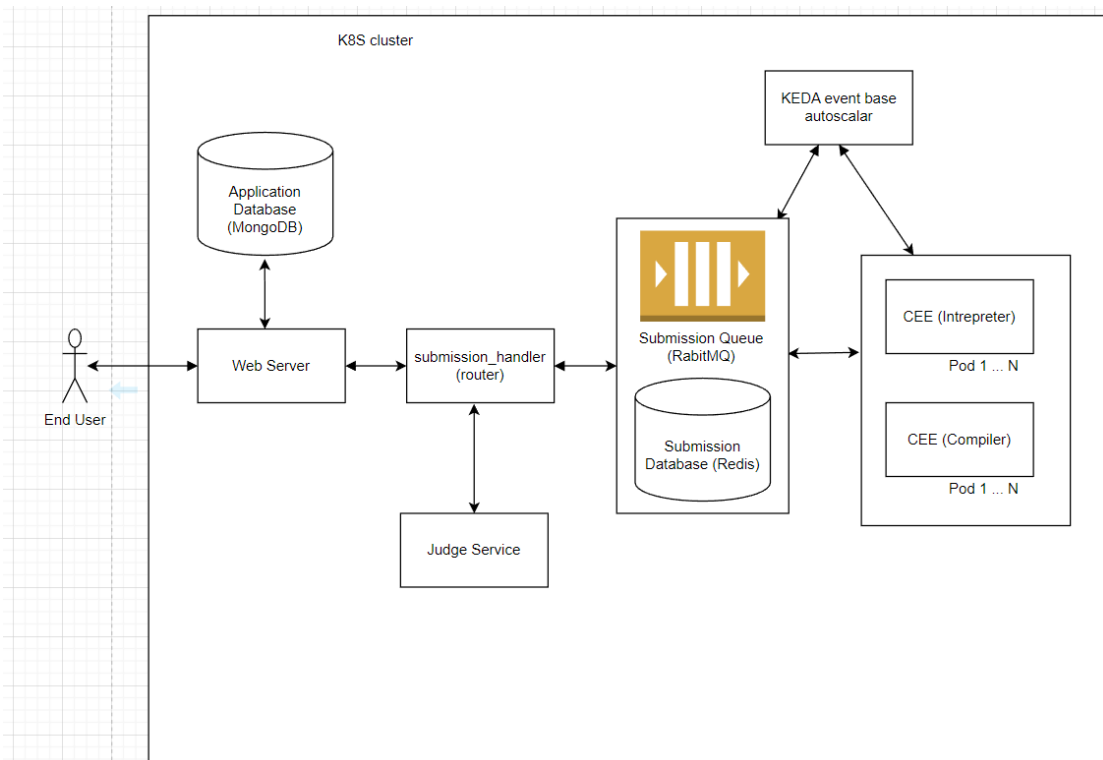


Figure 3.2.1 Proposed System Diagram

To solve problem mention in previous section, proposed solution had been made to build an automatic marking system. Above is system design diagram of proposed system. There are components included:

- Submission Handler (Router)

This component will mainly be responsible for interact with front end web server with backend. In the other hand, this component is also function as an entry point for submission code to execute in code execution environment. This component include endpoint to submit new code for available programming for execution, endpoint to fetch

Chapter 3

submission result, and endpoint to check the correctness of the code based on test cases. Endpoint to submit code execution will pass test case input (stdin, stdout) which retrieve from database along with users' code to CEE to execute code. Besides, endpoint of check correctness of code will pass expected output which retrieve from database to judge service. Lastly, endpoint for fetch submission result will get output from submission database about the status of the execution either in pending mode or finish executed mode.

2. Web Server

Web server will expose to end user directly and handle front end logic include authentication and authorization. When end users try to request to the web application, web server will be responsible to response with resources required. Yet, web server will interact with backend with router component by make HTTP request.

3. Judge Service

Upon code execution environment finish execute users' code and response the output to web server, web server will pass the output and test case output to judge service to examine the correctness of the code base on test case input provided. In context of programming subject, judge will be different from competitive programming, judge for educational context prioritize on making sure the output is same as expected. So, judge service will compare output from users' code with test case output.

4. Submission Queue (RabbitMQ) & Submission Database (Redis)

Execution of code often take longer time to execute, once web server sent create new submission request to submission queue service, the service will store the record into queue. Once CEE being initialized and ready, the record in queue will be pull for execution. Web Server will act as producer where will pass users' code and test case input to the queue; CEE will act as consumer where will wait the message sent by producer. Submission Queue will dispatch submission to CEE right after process submission before.

Another that, the submission will submit to submission database, it will generate token that represent the submission to web server. This token can be used to view the status of the execution, either still in pending, done execution or finished. The submission record will be deleted after 10 minutes to save space for another submission. While CEE finish executes the code, CEE will update output of the code to database.

5. Code Execution Environment (CEE)

This component will continuously fetch next submission from submission queue service once being initialized. CEE will start executing code with specified input right after received submission record and return the output to submission queue service. After execution of users' code with test case input, the CEE will be restarted to reinitialize the sandbox. CEE will be horizontal and vertical scale up and down depend on using predictive auto-scaler which will be discuss in next section.

6. Application Database

This component will store application data for programming subject such as programming question, test case, supported language etc.

7. CEE Event Based Auto-Scaling Service

This component will analyze the queue length or message rate in submission queue then used that information to scale the number of CEE.

3.3 Sequence Diagram

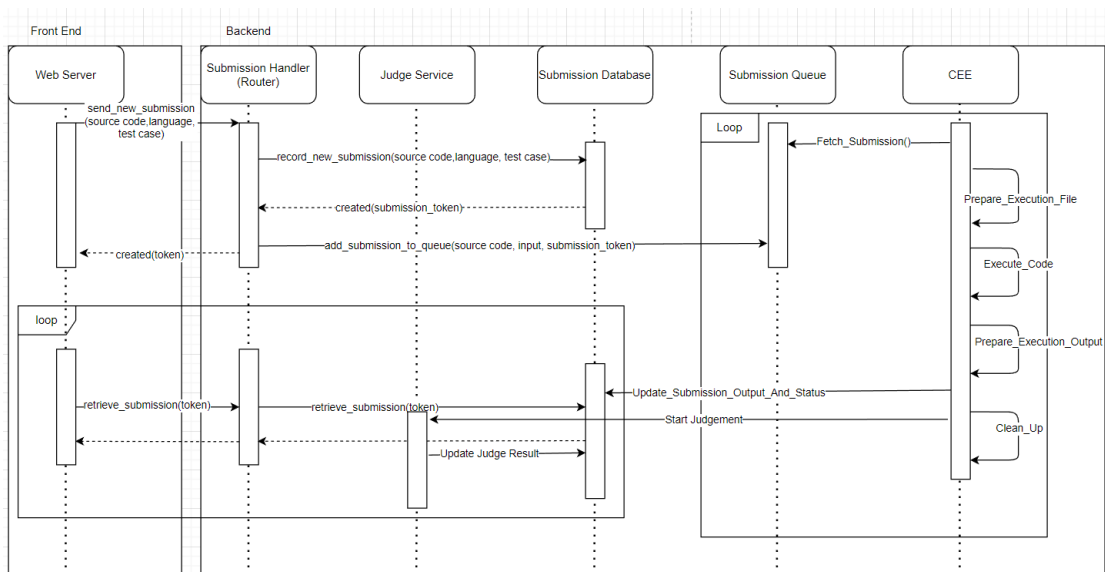


Figure 3.3.1 Proposed Sequence Diagram

Recall from previous section, the procedure of execution of users' submission code are as show in sequence diagram above. First, when users submit code with selected programming language to the backend of the system, users will get token that represent the submission of the code whereas the code had been store in submission database. Meanwhile, front end will retrieve test cases from application database for the question

Chapter 3

for related programming questions for that users' code represent then send to backend. Then router will save the submission details to submission database which will generate a unique ID that represent the submission whereas the ID will represent as token send back to users. This token will be used by users to check the status of code submissions. Once submission code saves to submission database, then the code will queue up in submission queue. Submission in queue will be fetch by CEE once CEE is ready to handle submission code. CEE will prepare the required file based on language type, then execute the code, prepare the output then save the output to submission database. After done the execution, CEE will be restarted to clean up make the CEE to initial state which for security purpose. User can continually retrieve the detail of the code execution progress based on the token provided.

3.4 Networking

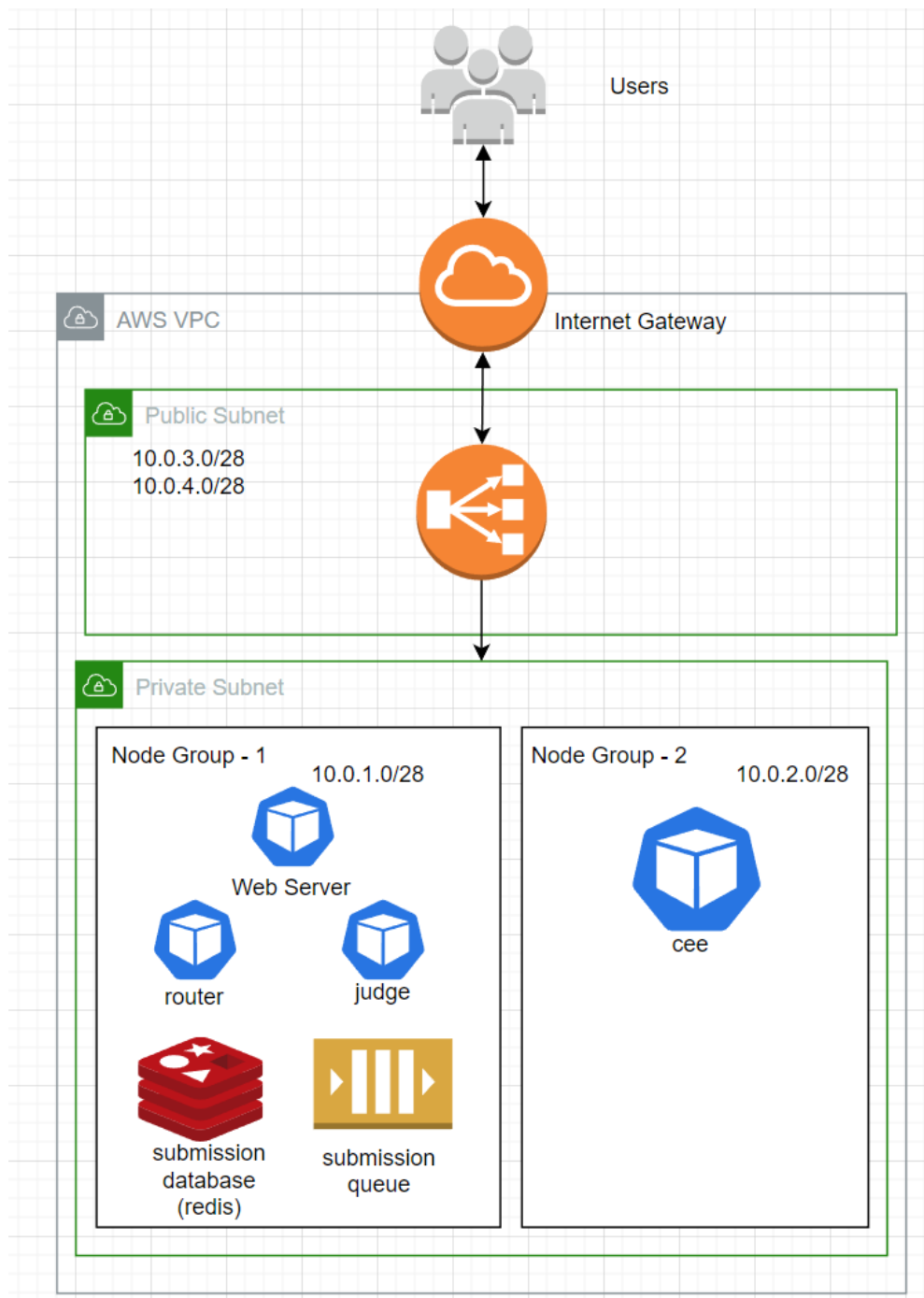


Figure 3.4.0.1: Proposed Network Diagram

This system proposed to build with 2 public subnet and 2 private subnets. There are several components in this network diagram included:

Load Balancer

[11] Load balancer is a component in cloud computing that distributes incoming traffic across multiple resources, such as servers, containers, or virtual machines. The load balancer acts as a reverse proxy, directing incoming traffic to the appropriate resource based on the load balancing algorithm configured.

Advance Messaging Queue Protocol (AMPQ)

[12] Advanced Message Queuing Protocol (AMQP) is an open standard application layer protocol for messaging. It defines a wire-level protocol for communication between clients and brokers (message brokers) in a messaging system. AMQP provides a robust, reliable, and scalable messaging infrastructure for applications that need to exchange messages. It provides features such as message durability, flow control, and routing, making it suitable for a wide range of use cases.

In this system, the submission code will submit to web server. Web server will handle front end application logic and submit the submission code to router using REST API where HTTP protocol is being used. Then the code will enqueue in submission queue (RabbitMQ) using AMPQ protocol. Once CEE ready, the code inside submission queue will be fetch using AMPQ protocol. RabbitMQ will default expose port 5672 where the client can connect to the message broker using the port. However, RabbitMQ created by Amazon MQ service will expose port 5671. For Redis submission database, port 6379 will be exposed where the client can instantiate connection to the database. However, Redis created in Amazon Elastic Cache service will not be able to instantiate connection outside the VPC.

3.4.1 Proposed Solution to manage data passing.

Problem

All data can be passed through each service using JSON format. However, when attempt to dump raw submission code into JSON value, it turns out the string value does not attempt to encode Unicode with “/”. For example, there had python code as shown as below:

```
print(“hello \n\n”)
```


Chapter 3

When client dump above code to JSON, the code doesn't encode Unicode from "\n" to "\\n" and this will lead to incorrect code preparation in CEE where the CEE will treat "\n" as new line instead of treat "\n" as part of the string to be print out. Below is result what will get when save above code.

```
print("hello  
"
```

Solution

Instead of dumps raw submission code into JSON, encode the submission code using base64 encoding will solve the problem. [13] Base64 encoding is a method of encoding binary data using a set of 64 characters, including letters, digits, and common symbols. This method ensure data transferred uncorrupted and same with what the client sent.

3.5 Infrastructure as Code

To ease the deployment process, infrastructure as code using Terraform is being proposed to use in the project. Terraform [14] is a tool for building, changing, and versioning infrastructure safely and efficiently. It can be used to manage popular service providers, such as Amazon Web Services (AWS), Google Cloud Platform (GCP), and Microsoft Azure, as well as custom in-house solutions. Terraform operates using a declarative syntax, allowing users to describe their desired infrastructure state, and Terraform will build and manage the necessary resources to reach that state. In the context of cloud computing, Terraform can be used to automate the provisioning and management of cloud resources, such as virtual machines, databases, load balancers, and storage.

3.6 Front End Development

Front-End involve of two component included web server and application. For web server, it mainly responsible for handle HTTP request by application to request for certain resources, web server for this project is built by ExpressJS framework. [15] Express.js is a fast, minimalist web framework for Node.js, which is used to build web applications and APIs. For the application, it mainly handle for user level logic and is built by React JS. [16] React is a popular open-source JavaScript library for building user interfaces (UIs) or single-page applications. React allows developers to build

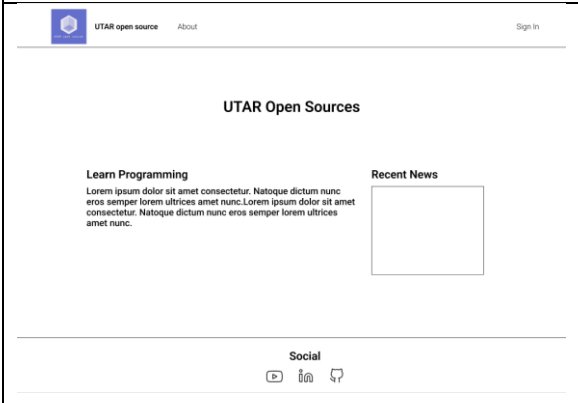
Chapter 3

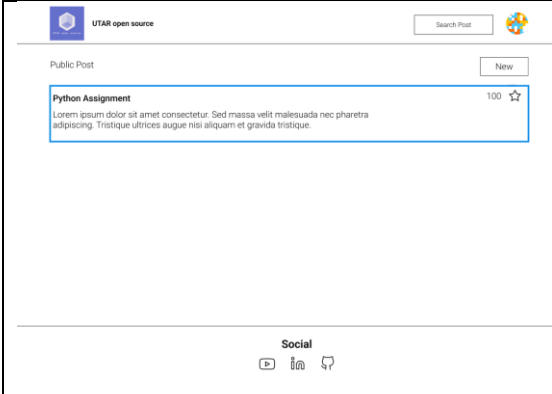

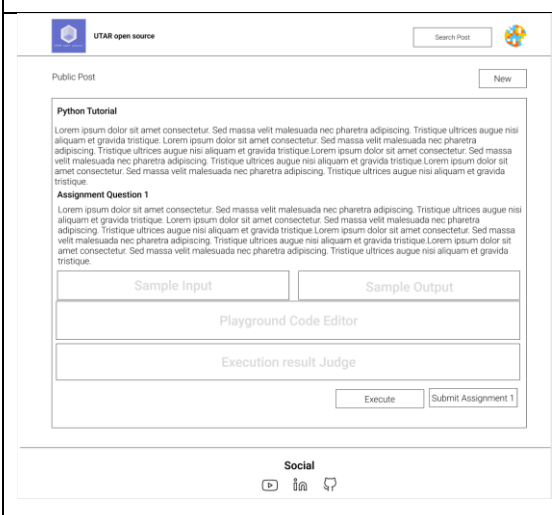
reusable UI components and manage the state of these components efficiently. Yet, to Redux JS is used to maintain the state of application. [17] Redux is a predictable state container for JavaScript applications, mostly used with web frameworks like React or Angular for managing the state of the application. It provides a central place to store the application state and ensures that the state can only be modified in a predictable way through a mechanism called "reducers". This helps to simplify the application logic and can make it easier to reason about and maintain complex application states over time. The application will perform API fetching with express server when required to access certain resources.

3.6.1 Front End Functional Requirement Refinement

1. User can upload tutorial in markdown mode format.
2. User can upload assignment question for tutorial in markdown format.
3. When writing the tutorial, user can use online markdown editor to write question.
4. User can set assignment practice for another user to practice.
5. User can set test cases for the assignment so that the system can automatically mark for the answers.

3.6.2 Front End Low Fidelity Design

Design	Explanation
	Users need to sign in before they can access to the functionality of the system to restrict the CEE being exploit.

	<p>After successfully sign in, users will be redirected to main page where user can search post, add new post, view public post</p>
	<p>When users want to create new post, user need to fill in post details include post title and description. User can choose whether to make this post public or private. User can add assignment for the post with the assignment questions and assignment test cases for the CEE to automatic judge the code submitted.</p>
	<p>When users view their post created, users can now submit or executed their code, if users submit their code, the code will submit to CEE and judge with test case. If users only need to execute the code, the code will not be judged.</p>

4.

3.6.3 Application Database Schema

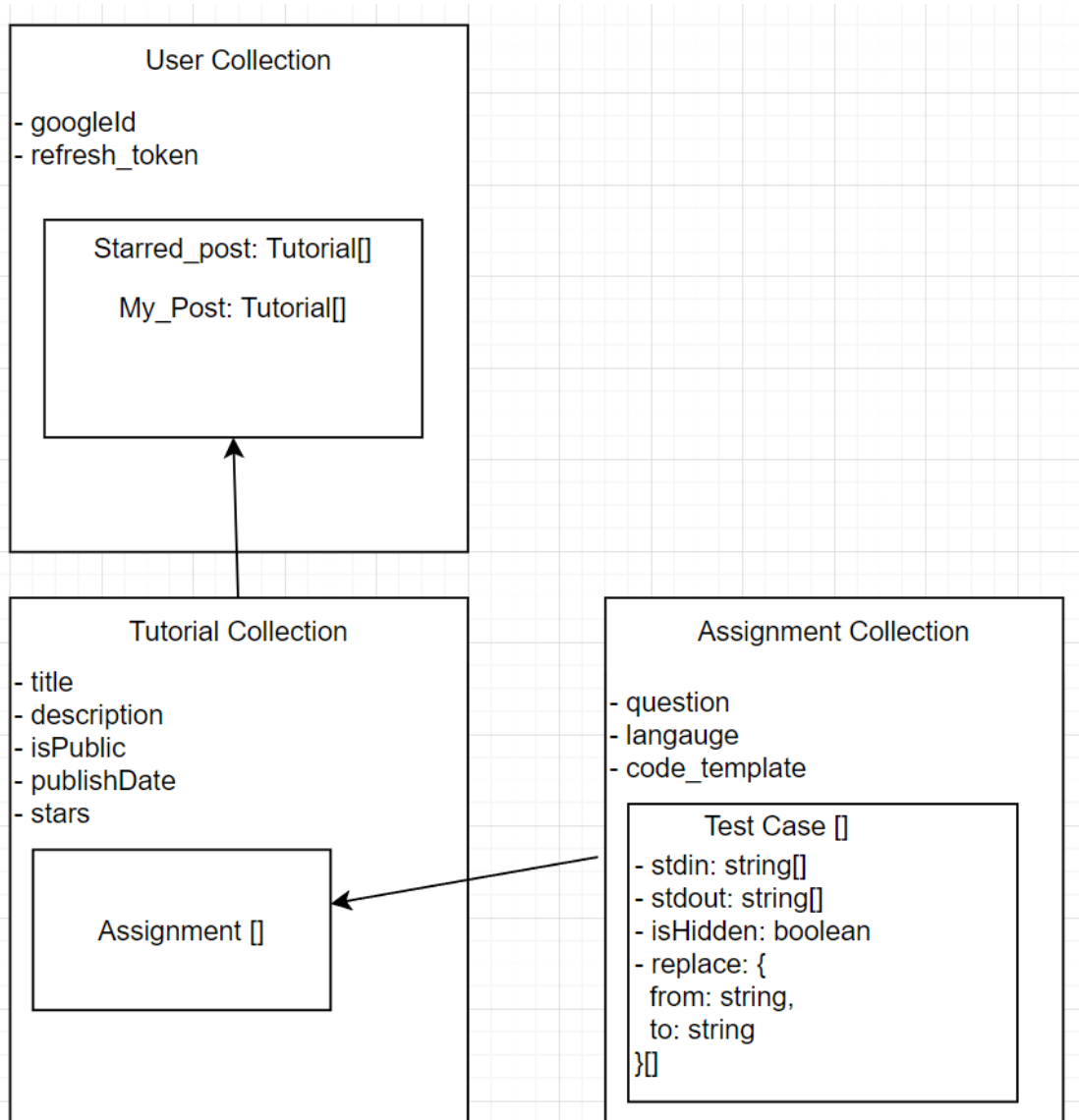


Figure 3.6.3.1 MongoDB Schema

MongoDB will be used as application which to store data related to client-side logic. There are several collections will be used in this project included user collection, tutorial collection, assignment collection and test case collection.

User collection will store authentication details and tutorial details. Tutorial collection will store tutorial details and this collection will reference to assignment collection to represent what assignment this tutorial record belongs to. Lastly, assignment collection will store detail about assignment while test case will store test case detail which the

test case record will be referenced by assignment collection. The test case has property of replace which will handle for “code inject” features. The reason array data type is used is because every test case can have many code inject instead of just one.

3.7 Clustering with K8s

[18] Kubernetes (k8s) clustering is a technique used to manage and orchestrate multiple containers running in a distributed environment. It is a popular open-source platform for automating deployment, scaling, and management of containerized applications. Kubernetes clusters consist of one or more master nodes that control the overall state of the cluster and multiple worker nodes that run the applications.

Recall from previous section, there will be several components with respective functionality connect to each other to build the system. In this section, proposed K8s configuration will be discussed to achieve this microservice system. All configurations will be deployed toward local cluster using kubectl. Below is proposed configuration for each service or component.

Submission Database

1. Persistence volume mount with /data folder to perform backup.
2. Expose port 6379 with ClusterIP service type.
3. Deploy using StatefulSet instead of deployment to provide stable persistence storage with Redis Official Image.

Submission Queue

1. Persistence volume mount with /var/lib/rabbitmq folder to perform backup.
2. Expose port 15672 and port 5672 as ClusterIP service type. Connection through port 15672 will provide RabbitMQ web interface to manage the queue. Port 5672 will provide connection toward RabbitMQ for consumer or producer.
3. Deploy using StatefulSet instead of deployment to provide stable persistence storage with RabbitMQ Official Image.

Router

1. Expose port 80 as ClusterIP Service type where the port will be map to port 8080 of the application.

Chapter 3

2. Deploy using deployment K8s object with docker images build on top of router logic written in Python.
3. Setup environment variable through config map and secret k8s object with submission database DNS, submission queue DNS, submission database secret and other required by router application.

CEE

1. **For initial planning**, CEE will contain two containers separately include sandbox (DinD) and worker. So, the CEE will deploy using deployment K8s object include two containers.
2. Setup empty dir K8s volume to share DinD connection socket toward worker.
3. CEE don't expose port instead CEE require environment variable of submission database DNS, submission queue DNS, submission database secret and other required by router application.

Judge

1. Expose port 80 as ClusterIP Service type where the port will be map to port 8080 of the application.
2. Deploy using deployment K8s object with docker images build on top of "Judge" application logic written in Python.
3. Setup environment variable through config map and secret k8s object with submission database DNS, submission database secret and other required by router application.

Application database

1. Persistence volume mount with /data/db folder to persistence perform backup.
2. Expose port 27017 with ClusterIP service type and headless service.
3. To mount persistence volume toward deployment, StatefulSet K8s object will be used.

Client

1. Client will expose port 80 which will map to port 3000 in the application with load balancer service type.

2. Deployment will be done by deployment K8s object with environment of application database cluster DNS

3.7.1 Event Base Autoscaling using KEDA

Kubernetes Event-Driven Autoscaling (KEDA) [19] is a Kubernetes components like Horizontal Pod Autoscaler where have functionality to specify the application event to monitor and scale target resources based on the application event. KEDA support many application event include Prometheus, RabbitMQ, Redis, MongoDB etc. However, in this project, RabbitMQ's queue length event will be monitor and scaled according to the queue length. For example, when the value of queue length of KEDA is set to 1, this mean that each consumer (CEE) will handle one task. When the queue length is more than 1, KEDA will scale the consumer to another one to maintain the queue length into one. However, there is another event call message rate can be monitored in RabbitMQ. KEDA can scale your application based on this rate. This is particularly useful when your application needs to process a large number of messages and you want to ensure that the processing keeps up with the rate of incoming messages. For example, if monitoring value of message rate set to 100, KEDA will trigger autoscaling when the messages per second in RabbitMQ reach the threshold of 100 per seconds. In this project, queue length event is more appropriate compared to message rate and different queue length value configuration had been used as experiment for the CEE performance include every consumer handle one task and every consumer handle 0.5 task (Each task use 2 consumer). Besides, KEDA allow to set initial number of minimum replicas, maximum number of replicas, polling interval and cooldown period. In this project, there will be different configuration for different maximum and minimum replicas to test the efficiency and performance of CEE, polling period is set to 2 seconds which mean every 2 seconds interval, KEDA will check RabbitMQ event and decide to scale up or scale down. This project don't set cooldown period where this parameter will scale the resource to 0 when it triggered, it similar to serverless. The reason this project not implement cooldown period even it can improve the cloud resources usage efficiency is because each time CEE is launched, CEE will download all compiler and interpreter docker images and remove all these information when scaled down or removed, this would cause longer time for CEE to initialize each time new CEE has been scaled up which lead to poor performance.

CHAPTER 4 System Implementation

This chapter will discuss about the implementation with details based on previous chapter.

4.1 Setup Development Environment

Setup Kubernetes Cluster Locally

To develop an microservices system using Kubernetes in local environment, local development cluster such as Minikube or Docker Desktop is required. Docker Desktop and Minikube are both tools to run single node cluster. Yet Docker Desktop also provide development environment for both Docker and Kubernetes. So, in this project, Docker Desktop had been used for a while to create local cluster.

However, when the project grow, development environment had been changed to Kubernetes In The Docker (Kind). Due to bug mentioned in this stack overflow thread [20] where the node exporter is not working due to mount propagation of Docker Desktop when implement cluster monitoring system using Prometheus. Therefore, Kind had been introduced. [21] Kind (Kubernetes IN Docker) is a tool that allows you to run local Kubernetes clusters using Docker container "nodes". It was primarily designed for testing Kubernetes itself, but can also be used for local development or continuous integration (CI)

Setup AWS Command Line Interface (AWS CLI)

[22] AWS CLI is a unified tool to manage your AWS services. With AWS CLI, you can control multiple AWS services from the command line and automate them through scripts. Using the AWS CLI, you can perform tasks such as launching and managing EC2 instances, uploading and downloading files to S3, managing databases in RDS, and much more.

Setup Terraform CLI

[23] Terraform CLI (Command Line Interface) is a tool that allows users to interact with Terraform to manage their infrastructure as code. With Terraform CLI, users can initialize, plan, apply, and monitor Terraform configurations from the terminal or command line. The Terraform CLI takes Terraform configuration files (written in the

HashiCorp Configuration Language or HCL), which define the desired infrastructure, and applies the changes to the desired target environment (such as AWS, Google Cloud, or other cloud providers).

Setup Python Programming Language

[24] Python is a high-level, interpreted, and general-purpose programming language. The reason Python being used is because Python is known for its readability and ease of use. It is used for a wide range of tasks, including web development, scientific computing, data analysis, artificial intelligence, and more. In this project, Flask library from Python programming language will be used as tool to create endpoint for services. Besides, Python will be used to connect Message Broker (RabbitMQ), Submission database (Redis) for CEE and Router.

4.1.1 Setup KinD Cluster

To setup Kind Kubernetes cluster, it required to setup configuration file for Kind Cluster. Then apply the configuration file using below command:

```
kind create cluster --name alorint --config kind-config.yml
```

This command will use the configuration wrote inside kind-config.yml file with the cluster name of “alorint”. For the configuration, this cluster required for extra port mapping for port 80 to port 80 to expose the port so that the ingress is accessible externally and discuss in previous section.

4.1.2 Load Balancer and Ingress

To expose endpoint inside the cluster externally using Kind for development cluster, load balancer and ingress need to be set. [25] Ingress is a Kubernetes API object that manages external access to the services in a cluster. Load balancer as mentioned before is a tool to distribute incoming traffics to different resources. So, when specific load balancer type services, load balancer infrastructure needs to be ready so that the services can locate and use the load balancer. After set the load balancer, it still not accessible externally, ingress come to this place as solutions.

More specifically, this project use metal load balancer (MetalLB) [26]. To setup the load balancer, IP address range need to specifically assign to the load balancer so that the load balancer can assign these IP address to its deployment. For ingress setup, it

involve of mapping of external exposed endpoint to internal exposed endpoint. In this project, only web application is exposed publicly, so the ingress will exposed web application externally. In short, when users accessing the external exposed endpoint, the ingress will redirect the traffic to web application lying inside the cluster.

4.2 Implementation Of CEE

Initial Implementation of CEE

Each CEE uses Alpine Linux as base operating system. Alpine Linux [27] is a lightweight, security-oriented, and resource-efficient Linux distribution that is commonly used as a base image for containers. CEE will pull images for interpreter language for program execution. For example, CEE will pull the python docker images to run Python script. CEE will act as a consumer that keep fetching submission in submission queue. After receive submission data, CEE will save the code into file. Then, CEE will execute the code with the input as stdin provided using **process**. After the execution, CEE will save the stdout or stderr. For security purpose, CEE will clean up all file created during execution include executable file, code, stdout or stderr file. Besides, CEE will limit the amount of CPU and memory resources that can be used by the **process** to execute the code using **resources** standard python library [28]. Initially the resources limit setting for the CPU limit is 10 second and memory limit is 100mb.

Each CEE will execute certain programming language which only allow for using standard library for each programming language details specified in environment variable. Each submission will be handled in first in first out sequence.

Below is the format of environment to setup to allow CEE to handle the submission for particular programming languages.

Format:

```
<language name>@<language docker images>@<language type>@<compilation command>@<execution command>,<language name>@<language docker images>@<language type>@<compilation command>@<execution command>
```

Example:

```
python@python:alpine3.16@.py@interpreter@python
code.py,nodejs@node:alpine3.18@.js@interpreter@node  code.js,c@frolvlad/alpine-
gxx@.c@compiler@gcc --static code.c -o code@./code,cpp@frolvlad/alpine-
gxx@.cpp@compiler@c++ --static code.cpp -o code@./code
```

Problem Encounter for Initial CEE implementation

Since this implementation is relying on docker container and Alpine Linux, [28] Resources library not work as expected to limit resources. Due to this issue, if the code submitted by users used resources more than allowed, the container will be killed and restart to avoid the program excess even more resources. So, this process will raise an issue where the record or metadata of previous execution will be lost.

Improved Version of CEE

Instead of using process to execute code, [29] Docker is much better environment to execute code safely. So, after brainstorm, [30] Docker in Docker, DinD had been introduce and the architecture of CEE had been amended. DinD is a technique for running Docker containers within a Docker container, it's a way to create a Docker runtime environment inside another Docker container. To achieve DinD, there will be one container will hold the docker engine and another worker container that will connect to the docker engine through Python Docker SDK [31] using docker.sock file. Then the worker will launch docker container as **sandbox** to execute the code. In summary, CEE will include two components, worker and DinD. Worker will be responsible to prepare the file needed and launch docker container to execute code. The file that worker prepared will store in “code” folder inside worker, when launch **sandbox container**, the “code” folder will mount inside the **sandbox container** for execution.

However, **separate CEE into two components has disadvantages in scaling**. This is because worker can only connect to DinD using docker.sock file, this make the scaling very difficult when DinD scaled more than one instance. When new DinD being scaled up, new DinD will generate new docker.sock file, worker need to decide to use which docker.sock to connect to which DinD instance. This make it very hard to balance the load between each DinD.

To solve this problem, instead of separate CEE into two components, **embed worker into DinD** will be better choice in term of scaling. Using this method, CEE can utilize the advantage of RabbitMQ prefetch features, multithreading and KEDA event base auto scalar to improve scalability. Basically, RabbitMQ prefetch features will prefetch certain amount of task into consumer. In short, this methodology will prefetch certain amount of task into CEE and execute the task concurrently, then the CEE instances will scaled based on the queue length and message rate for task in RabbitMQ to achieve greater scalability. Lastly, to further optimize the code execution for different test cases, there will be involve of multithreading in launching sandbox for executing different test cases' stdin.

To further optimize the CEE performance and images size, instead of using Python, Golang will be used and will replace Python. Golang has the Go Routine features which will make the multithreading easier. Golang can compile the source code to binary which make the executable lighter where it don't required Golang compiler installation compare to Python where it rely on interpreter for execution. Before talk about last optimization, it is better to clarify that each submission can have many test case, each test case need to be run independently. For example, a submission has 3 test case with different input, then the submission need to run 3 times to get output from 3 different test case input. To optimize the execution of each test case, instead of running each test case synchronously, each test case is running asynchronously using multi-threading.

How CEE Detect Memory or Run Time Limit Exceeded

When launching the docker container, there will be configuration made where to limit the amount of memory, CPU and Run Time limit, process limit etc. Whenever the container exceeds the limit set, the container will automatically stop, and the status of the container will be collected. The status can be parsed to identify what error had been occur that makes the container being stopped. For example, if the container status show that "OOMKilled" is true, that mean of the container is over the limit of memory allocated. To detect time limit, it will be tricky, it involve of tracking of time period and remove the sandbox when the time limit is reached.

Procedure of Worker in Code Execution

1. Received submission token from Message Broker (RabbitMQ).

2. Retrieve submission data from Submission Database (Redis) using token.
3. Store code and input into file with relative extension.
4. Transform The code with relative code replacement.
5. Execute the code by launching new sandbox.
6. Retrieve sandbox log and parse it into stdout and stderr.
7. Save result to submission database and acknowledge the message.

4.4 Front End

4.4.1 Web Server and Web Page Implementation

The web server is implemented using Express JS, then the web server connects to application database (MongoDB) using mongoose to perform CRUD operation. For web page implementation, this project used React as client side application. To integrate web server with React application, web server will send compiled React file to client when client accessing the “index” endpoint. In order for React application communicate with web server, it will send HTTP request to server to perform server side action including authentication, make submission.

The React application has a code editor for users to write their answer and make submission. To achieve this, React Monaco is used. [32] React Monaco is browser-based code editor wrapped around React which made it easier to integrate with React application. Besides, there is feature that allow lecturer to write the tutorial description and assignment questions with markdown format. To achieve this, [33] React markdown editor has be used for this project to let user write content in markdown.

Lastly, since the client side application is written in TypeScript, the client side application will be compiled to JavaScript during the containerize process.

4.4.2 Authentication and Authorization Implementation.

To avoid the CEE being exploit by bot, this project will require user to login to access the CEE features. In this project, Google OAuth 2 will be used to perform login. To make implementation of Google OAuth 2 easier with React, [34] React OAuth library will be used to connect with Google Authorization Server with React application. Whenever user login, React OAuth library will connect with Google Authorization server, return user detail, and access code. Then, access token will send to Web Server

Chapter 4

for further process. In web server, the access token will be used to exchange for JSON web token. The returned token included access token and refresh token. Access token is a token that is issued by the authentication server and is used to authenticate the user. The access token typically contains information about the user. Instead, the refresh token is used to obtain a new access token when the current access token has expired. Refresh token will then store in application database and access token will be return to React application as cookie and JSON value. Whenever user try to request endpoint from Web Server, the access token from cookie pass from React Application will be examined for its expired date and validity (token not being modified). Web Server return error to web application if the token is invalid, while return the requested resources if the token is valid. When the React Application received error from verify token, it will request to refresh its access token from Web Server. For the refresh token endpoint, Web Server will retrieve refresh token stored in application database for that particular user and used that refresh token to request new access token from Google Authorization Server. The reason of perform refresh token action is to improve user experience where users don't need to keep login when the token expired.

4.5 Implementation of system monitoring

This project is using microservices architecture in its primary implementation. Due to the nature of this architecture, monitoring services is required to monitor the resources usage of each services in the architecture. This monitoring service is used to detecting issues and fixing them, it also important to tell how the system is behaving, which parts are under stress, and which parts could be optimized for better performance. According to [35] system monitoring in microservice architecture is important in

1. **Detecting Service Problems:** In a microservices architecture, each service is an independent unit that interacts with external users and other microservices. A problem with one service can disrupt the entire application, therefore, teams need to know about an issue as soon as it arises to take corrective action. Monitoring helps in identifying what happened, when, why, and under what conditions, enabling teams to identify the root cause and recover the malfunctioning service quickly.
2. **Maintaining Service Health:** Monitoring helps maintain the health of the microservices. If an issue is detected, it can be resolved quickly without impacting other microservices or the application as a whole. This is particularly critical because each microservice is small and self-contained, making it easier to identify problems and resolve them.
3. **Platform Metrics Monitoring:** Monitoring platform metrics is crucial to keeping microservices infrastructure running smoothly. This low-level data can indicate problems in the underlying compute, storage, or networking equipment. Careful monitoring of these metrics can highlight performance degradation and prevent system-wide failures

The tool used for system monitoring for this project is Prometheus and Grafana. Prometheus is an open-source systems monitoring and alerting toolkit. It collects and stores its metrics as time-series data. Prometheus can scrape metrics from each services and store all scraped metric locally. On the other hand, Grafana is an open-source tool to easily visualize information. It can take information from server monitoring tools like Prometheus and display it in easy-to-understand dashboards. To achieve this, [36] kube-prometheus open source project is used, this project collects Kubernetes manifests, Grafana dashboards, and Prometheus rules combined with documentation

Chapter 4

and scripts to provide easy to operate end-to-end Kubernetes cluster monitoring with Prometheus using the Prometheus Operator.

To implement Prometheus and Grafana using the kube-prometheus open source project, it involve of download of manifest file from the project and apply it to the cluster. The Prometheus monitoring service will monitor resources for all namespace by default. Since the Prometheus and Grafana is host inside the cluster, port forward toward the monitoring service is required to access the service.

To access Prometheus service, it required of port forward toward the monitoring service in port 9090. Below is the command to access the service using kubectl

```
kubectl -n monitoring port-forward svc/prometheus-operated 9090
```

Below is the figure of Prometheus user interface.

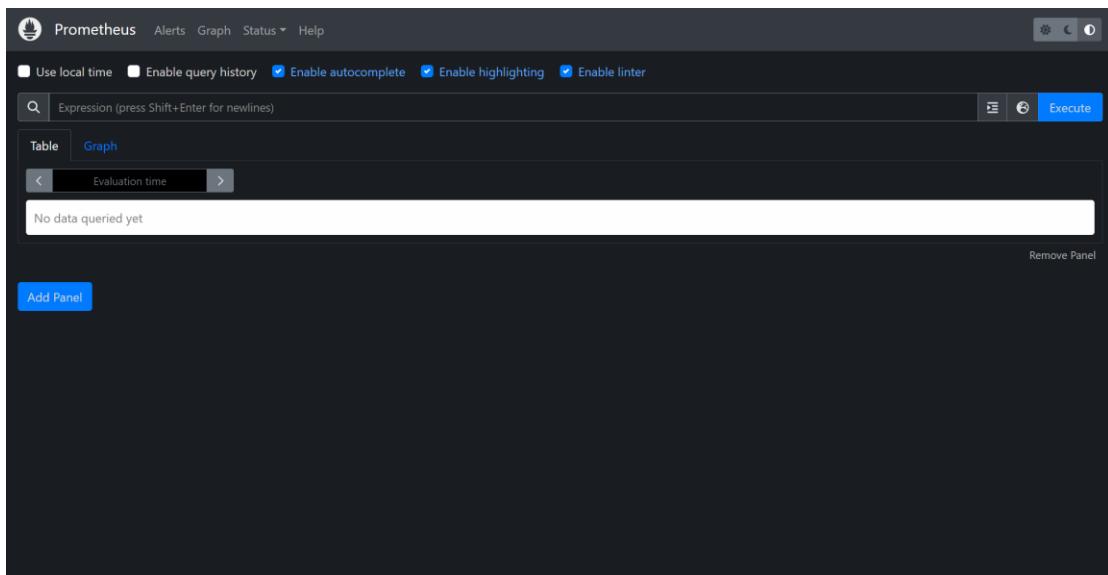


Figure 4.5.1 Prometheus UI

To access Grafana dashboard, it required of port forward toward the dashboard service in port 3000. Below is the command to access the service using kubectl

```
kubectl -n monitoring port-forward svc/grafana 8000:3000
```

With Grafana dashboard, it is easy to monitor resource usage include CPU and memory usage of each pod from different namespace. Below is example of Grafana monitoring dashboard of CEE pod resources usage.

Chapter 4

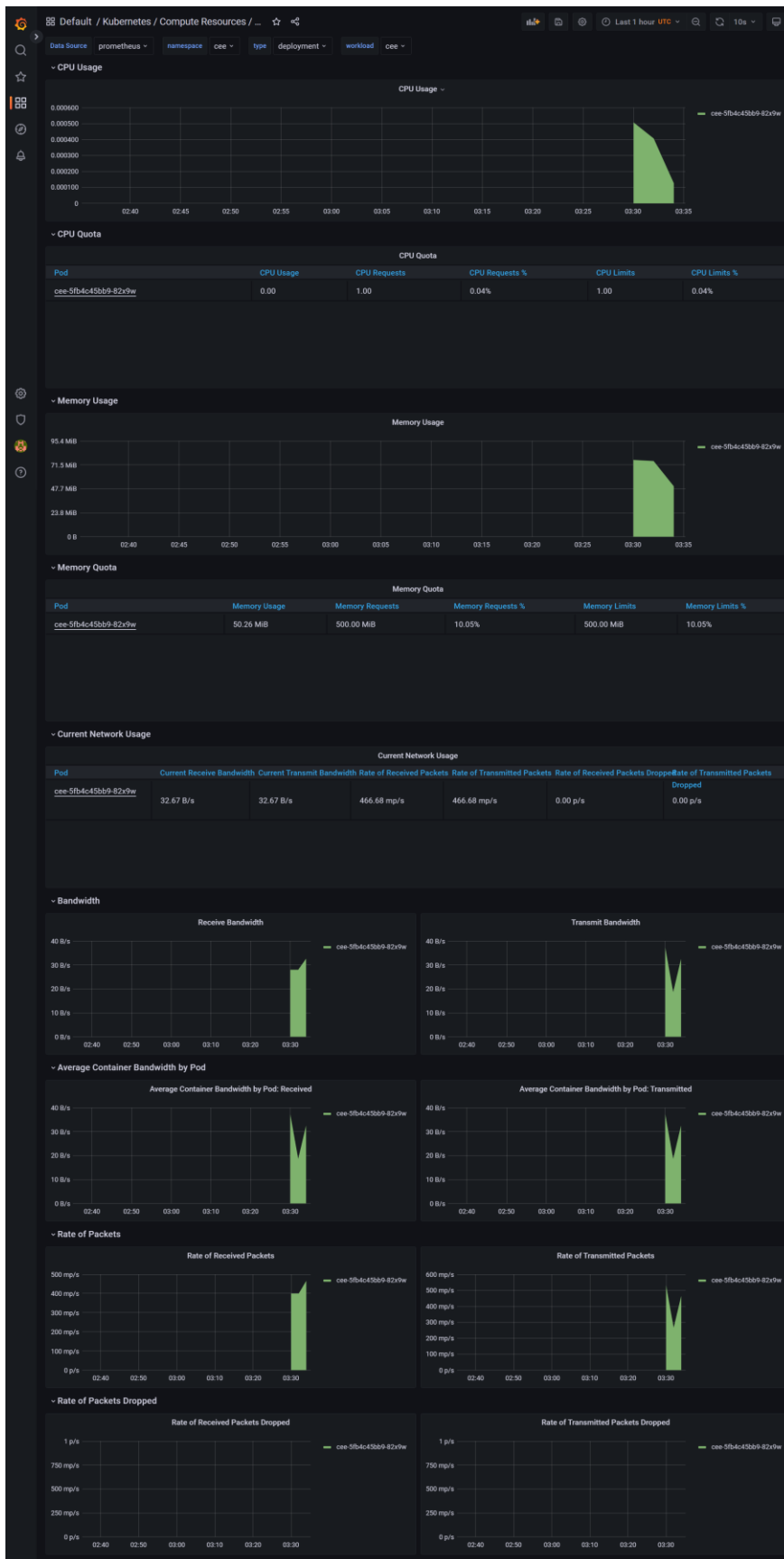


Figure 4.5.2 Grafana Dashboard

4.6 Setup Load Testing Tool

[37] K6 is an open source tool mainly focus on load testing. It used to simulate multiple users accessing a software program concurrently to understand how the software behaves under different conditions, helping to identify bottlenecks, uncover performance issues, and ensure the software can handle real-world user demands. To use K6, it involve of outline workloads using JavaScript, then run the JavaScript using K6 interpreter. After running the load test, the result will be collected and saved to Grafana Cloud. To setup K6 involve of installation of its interpreter and run the JavaScript load test script with output to cloud.

CHAPTER 5 SYSTEM EVALUATION AND DISCUSSION

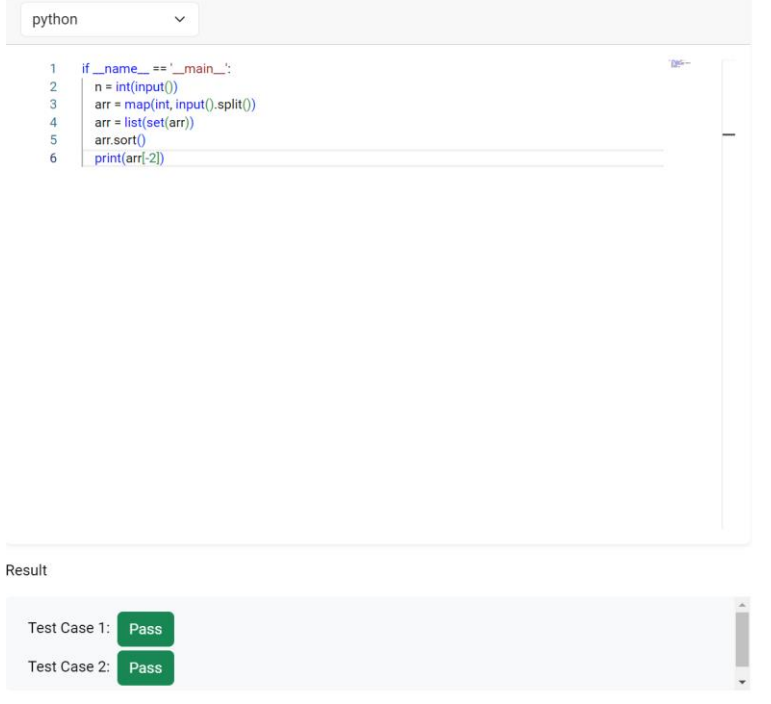
5.1 Setup Unit Test For CEE

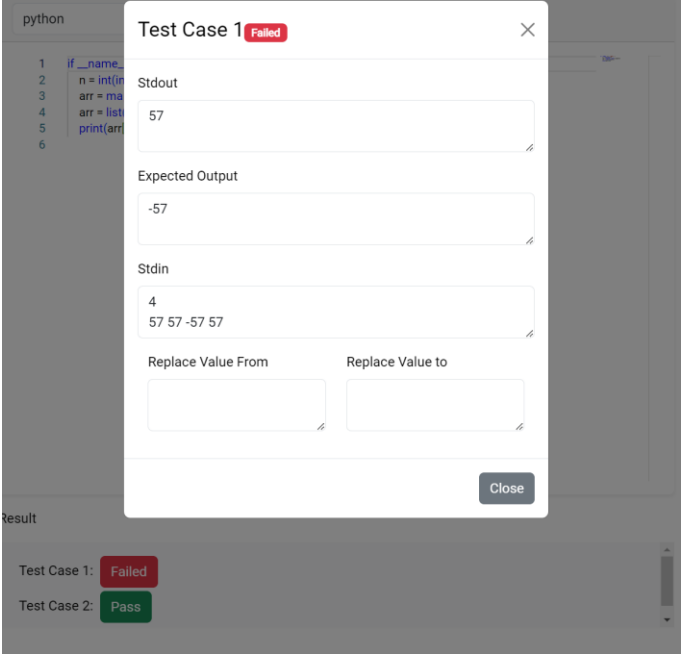
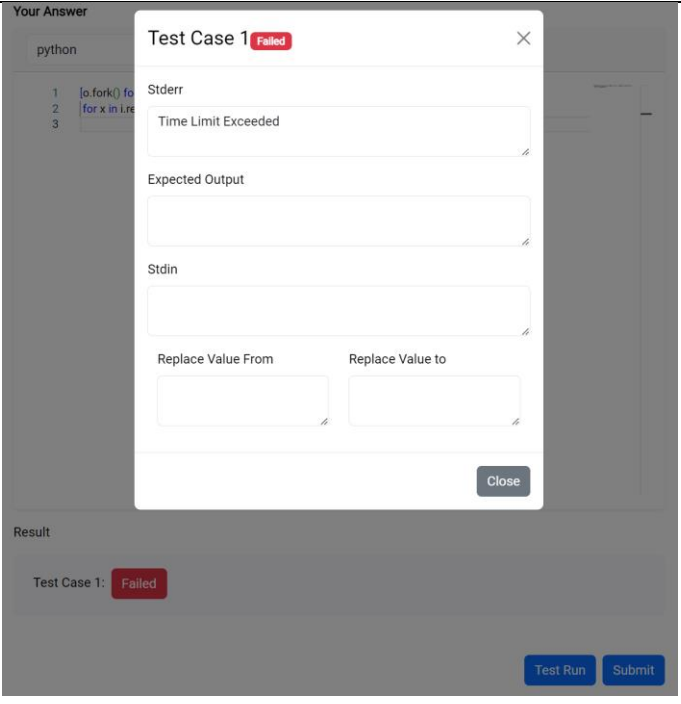
This testing mainly involve of identify either the objective “to design automatic marking system which able to execute submission code securely” is met. There are several types of error in programming, well function CEE must have the ability to identify the error occur in the code. According to [38], the error include

1. Syntax error: When the program contains incorrect or invalid syntax, this type of error is thrown.
2. Logical error: This error occurs when the code runs but doesn't produce the desired outcome. This error sometimes referred to logical error.
3. Run-time error: This error occurs when the code runs and encounters a problem during execution, such as division by zero or accessing an array index that is out of bounds.
4. Compile-time error: This error occurs when the code cannot be compiled, due to missing files, incorrect declarations, or other issues. Interpreter will not experience this error.
5. Resource error: This error occurs when a program is unable to access required resources, such as memory or files, due to insufficient resources or incorrect permissions.

5.1.1 Python

Table 5.1.1.1 Python Test

Test Case	Explanation	Code & Output
<p>Python Program Without Error</p>	<p>The program fit for every test case.</p>	
<p>Python Program with syntax error</p>	<p>The syntax error will appear in dialog, and it will show that the error in stderr.</p>	

<p>Python With Logical Error</p>	<p>The test case result will show “fail”. However, the program returns valid stdout instead of stderr. In short, the program doesn’t have error, but the output doesn’t meet the test case.</p>	
<p>Python Resources error. (Time Limit)</p>	<p>The return result will tell that the program exceeds the run time limit which is 2 seconds.</p>	

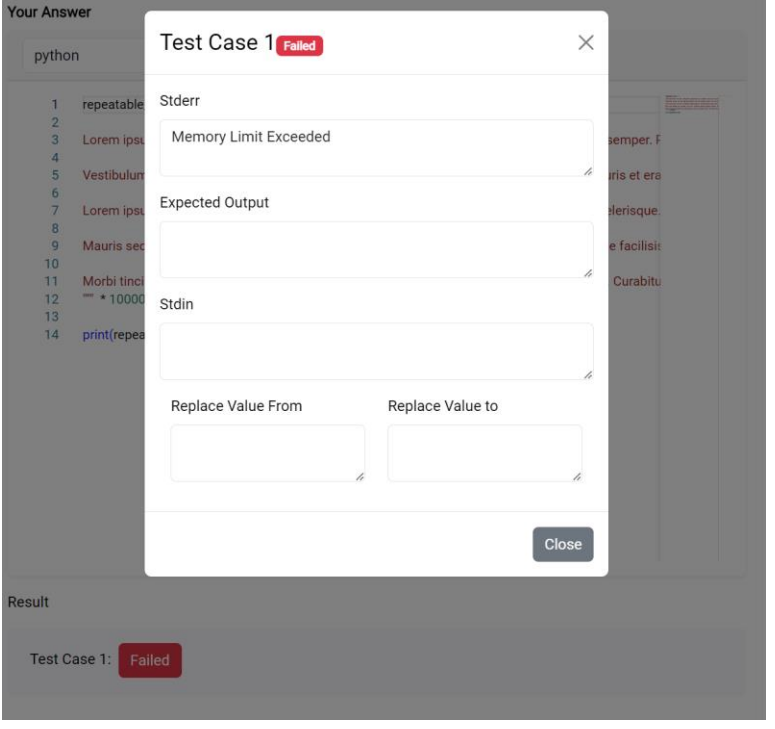
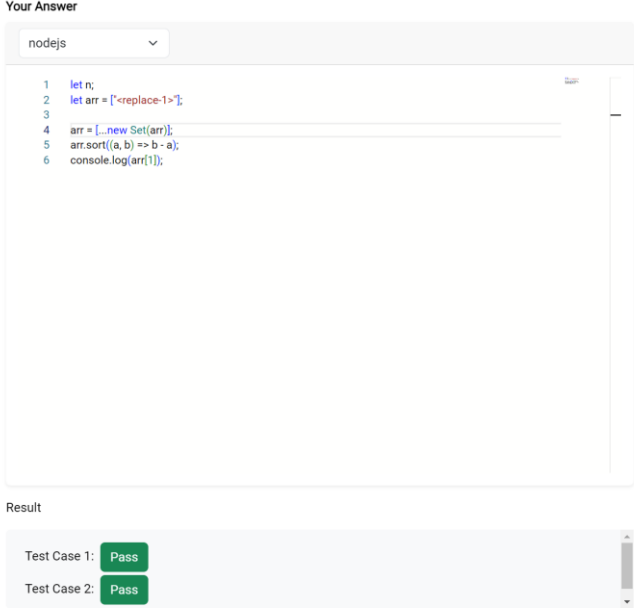
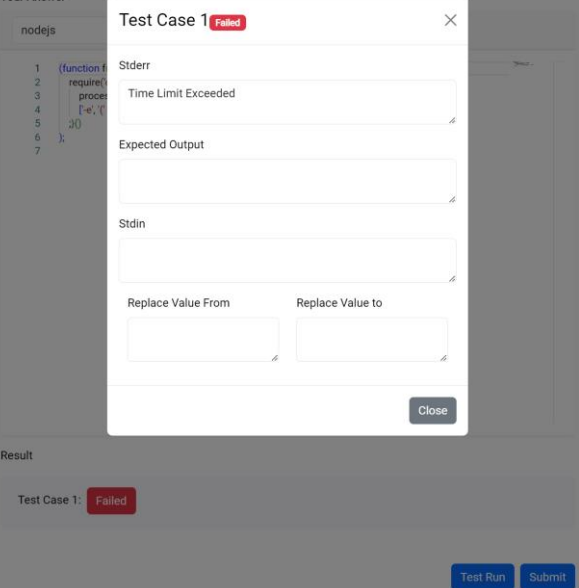
<p>Python Resources error. (Memory Limit)</p>	<p>The return result will tell that the program exceeds the memory limit which is 50mb.</p>	
---	---	--

Table 5.1.1.1 is unit test done to CEE test the usage of Python programming language. Due to Python is interpreter language, it will never experience compile time error. So compile time error will never be tested. [39] Official Python Interpreter docker image will be used to execute the program.

Refer to appendix [A-1](#) for Python Program and configuration used in the test.

5.1.2 NodeJS

Table 5.1.2.1 NodeJS Test

Test Case	Explanation	Code & Output
NodeJS program	The program fit for every test case.	 <p>The screenshot shows a code editor with the following code:</p> <pre> 1 let n; 2 let arr = ["<replace-1>"]; 3 4 arr = [...new Set(arr)]; 5 arr.sort((a,b) => b - a); 6 console.log(arr[1]); </pre> <p>Below the code, the 'Result' section displays:</p> <ul style="list-style-type: none"> Test Case 1: Pass Test Case 2: Pass
NodeJS resources Error	The return result will tell that the program exceeds the run time limit which is 2 seconds.	 <p>The screenshot shows a code editor with the following code:</p> <pre> 1 (function f 2 require(3 proces 4 ['e', 'f 5], 30 6); 7 </pre> <p>Below the code, the 'Result' section displays:</p> <ul style="list-style-type: none"> Test Case 1: Failed <p>A modal dialog titled 'Test Case 1 Failed' is open, showing the error message: 'Time Limit Exceeded'. The dialog also includes fields for 'Expected Output', 'Stdin', and 'Replace Value From' / 'Replace Value to'.</p>

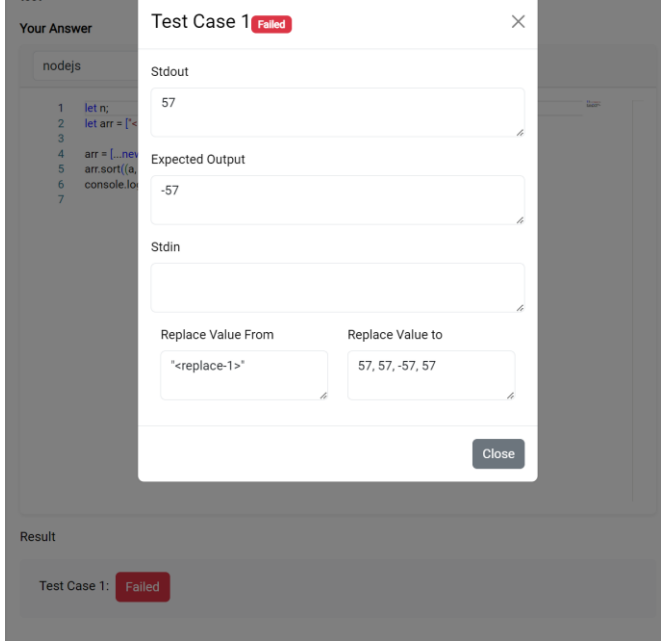
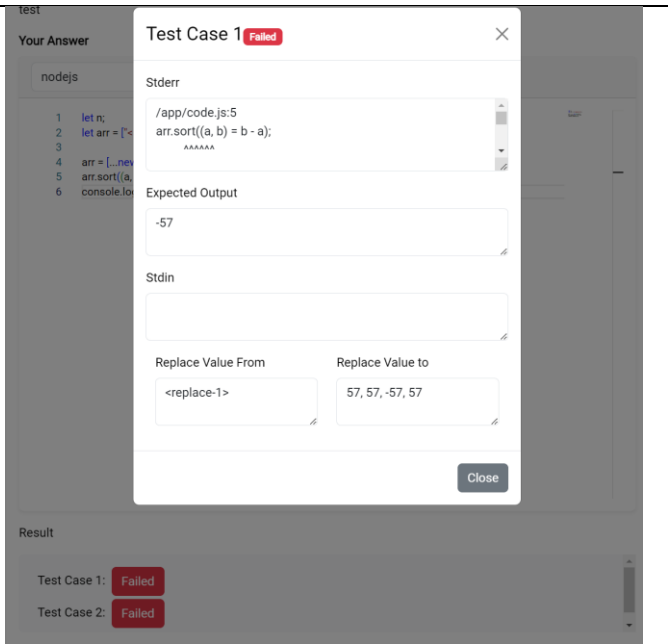
<p>NodeJS with logical error</p>	<p>The test case result will show “fail”. However, the program returns valid stdout instead of stderr. In short, the program doesn’t have error, but the output doesn’t meet the test case.</p>	
<p>NodeJS with run time error, syntax error</p>	<p>The syntax error will appear in popover, and it will show that the error is stderr.</p>	

Table 5.1.2.1 is unit test done to CEE interpreter to test the usage of NodeJS programming language. Due to NodeJS is interpreter language, it will never experience compile time error. So compile time error will never be tested. [40] Official NodeJS Interpreter docker image will be used to execute the program.

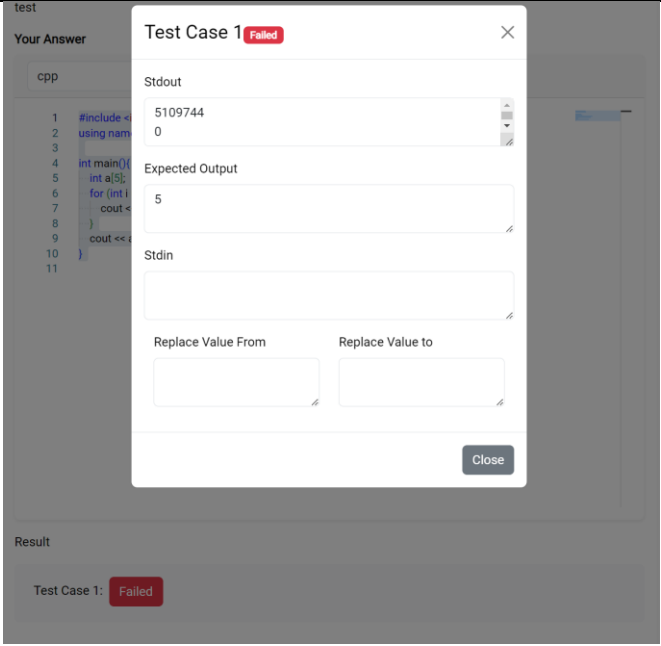
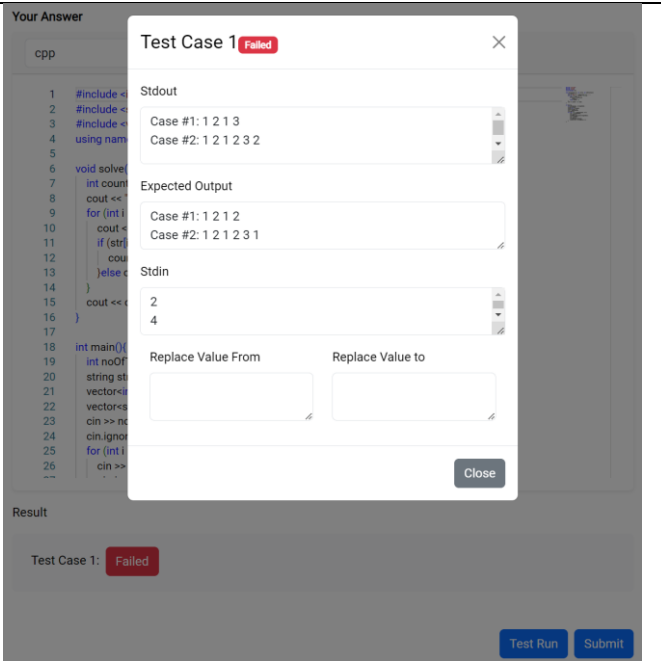
Refer to appendix [A-2](#) for NodeJS Program and configuration used in the test.

Chapter 5

5.1.3 C++

Table 5.1.3.1 CPP Test

Test Case	Explanation	Code & Output
<p>CPP program without error</p>	<p>The program fit for every test case.</p>	<p>Your Answer</p> <pre> cpp 1 #include <iostream> 2 #include <string> 3 #include <vector> 4 using namespace std; 5 6 void solve(string str, int strlen, int noOftestCase){ 7 int counter = 1; 8 cout << "Case #" << noOftestCase + 1 << " "; 9 for (int i = 0; i < strlen - 1; i++){ 10 cout << counter << " "; 11 if (str[i] < str[i+1]){ 12 counter++; 13 } else counter = 1; 14 } 15 cout << counter << endl; 16 } 17 18 int main(){ 19 int noOfTestCase, strlenHold; 20 string strHold; 21 vector<int> strlen; 22 vector<string> str; 23 cin >> noOfTestCase; 24 cin.ignore(); 25 for (int i = 0; i < noOfTestCase; i++){ 26 cin >> strlenHold; </pre> <p>Result</p> <p>Test Case 1: Pass</p>
<p>CPP with compile error (syntax error)</p>	<p>The syntax error will appear in popover, and it will show that the error is stderr.</p>	<p>test</p> <p>Your Answer</p> <pre> cpp 5 void solve 6 int count 7 cout << 8 9 for (int i 10 11 12 13 14 15 16 17 18 int main() 19 int noOf 20 string str 21 vector<int> 22 vector<string> 23 cin >> noOf 24 cin.ignore() 25 for (int i 26 cin >> 27 cin.ignore() 28 strlen 29 cin >> 30 cin.ignore() </pre> <p>Result</p> <p>Test Case 1: Failed</p> <div style="border: 1px solid gray; padding: 5px; margin-top: 10px;"> <p>Test Case 1 Failed</p> <p>Stderr</p> <pre> code.cpp: In function 'void solve(std::string, int, int)': code.cpp:8:47: error: expected ';' before 'for' 8 cout << "Case #" << noOftestCase + 1 << " "; ^ ; 9 for (int i = 0; i < strlen - 1; i++){ ~~~~ code.cpp:9:21: error: 'i' was not declared in this scope 9 for (int i = 0; i < strlen - 1; i++){ ^ </pre> <p>Expected Output</p> <p>Case #1: 1 2 1 2 Case #2: 1 2 1 2 3 1</p> <p>Stdin</p> <p>2 4</p> <p>Replace Value From: <input type="text"/></p> <p>Replace Value to: <input type="text"/></p> <p>Close</p> </div>

<p>CPP with run time error</p>	<p>Runtime error happen index out of range array in CPP. However, in CPP access array out of index would not trigger stderr, so the return result will be stdout which don't match with test case.</p>	
<p>CPP with logical error</p>	<p>The test case result will show “fail”. However, the program returns valid stdout instead of stderr. In short, the program doesn't have error, but the output doesn't meet the test case.</p>	

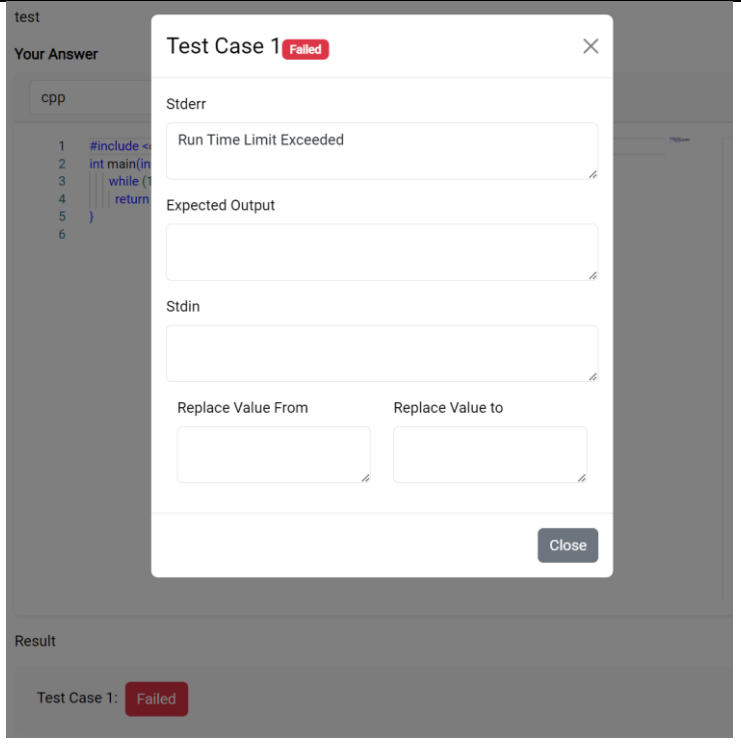
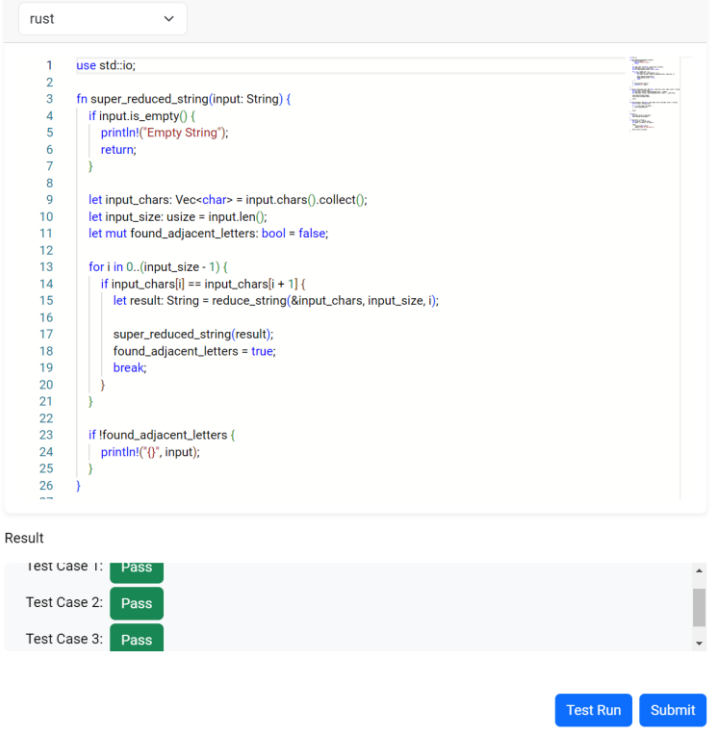
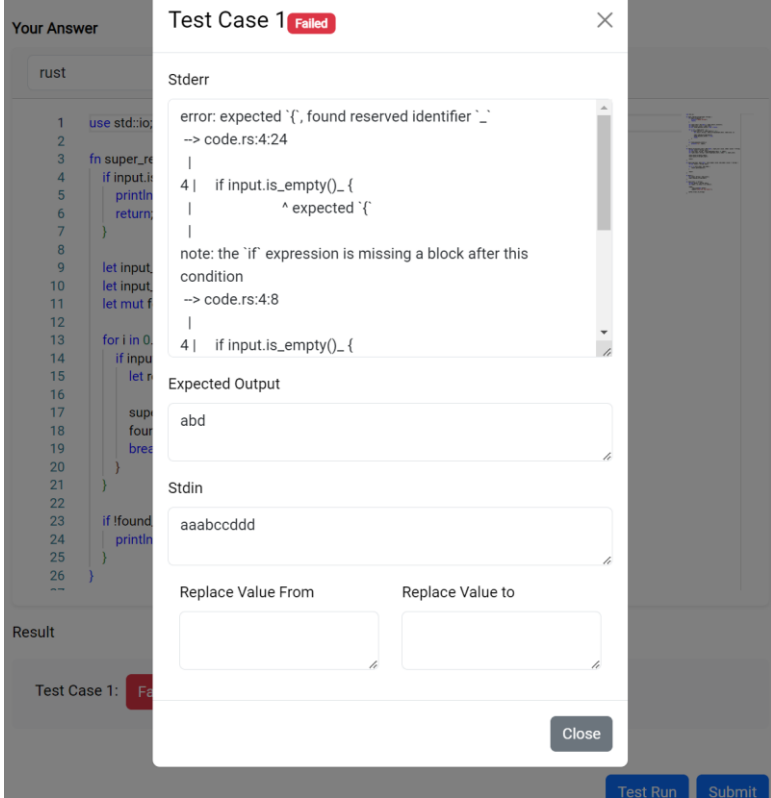
<p>CPP resources error</p>	<p>The return result will tell that the program exceeds the run time limit which is 2 seconds.</p>	
----------------------------	--	--

Table 5.1.3.1 is unit test done to CEE compiler to test the usage of CPP programming language. [41] docker image with C++ and C compiler will be used to compile the program to executable. During this process, compiler time error will be generated such as syntax error. Besides, resources error may happen during runtime.

Refer to appendix [A-3](#) for C++ Program and configuration used in the test.

5.1.4 Rust

Table 5.1.4.1 Rust Test

Test Case	Explanation	Code & Output
<p>Rust Program without Error</p>	<p>The program fit for every test case.</p>	 <pre> rust 1 use std::io; 2 3 fn super_reduced_string(input: String) { 4 if input.is_empty() { 5 println!("Empty String"); 6 return; 7 } 8 9 let input_chars: Vec<char> = input.chars().collect(); 10 let input_size = input.len(); 11 let mut found_adjacent_letters = false; 12 13 for i in 0..(input_size - 1) { 14 if input_chars[i] == input_chars[i + 1] { 15 let result: String = reduce_string(&input_chars, input_size, i); 16 17 super_reduced_string(result); 18 found_adjacent_letters = true; 19 break; 20 } 21 } 22 23 if !found_adjacent_letters { 24 println!("{}", input); 25 } 26 } </pre> <p>Result</p> <p>Test Case 1: Pass</p> <p>Test Case 2: Pass</p> <p>Test Case 3: Pass</p> <p style="text-align: right;">Test Run Submit</p>
<p>Rust with compile time error, syntax error</p>	<p>The syntax error will appear in popover, and it will show that the error is stderr.</p>	 <p>Your Answer</p> <pre> rust 1 use std::io; 2 3 fn super_reduced_string(input: String) { 4 if input.is_empty() { 5 println!("Empty String"); 6 return; 7 } 8 9 let input_chars: Vec<char> = input.chars().collect(); 10 let input_size = input.len(); 11 let mut found_adjacent_letters = false; 12 13 for i in 0..(input_size - 1) { 14 if input_chars[i] == input_chars[i + 1] { 15 let result: String = reduce_string(&input_chars, input_size, i); 16 17 super_reduced_string(result); 18 found_adjacent_letters = true; 19 break; 20 } 21 } 22 23 if !found_adjacent_letters { 24 println!("{}", input); 25 } 26 } </pre> <p>Result</p> <p>Test Case 1: Failed</p> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> <p>Test Case 1 Failed</p> <p>Stderr</p> <pre> error: expected `{`, found reserved identifier `_' --> code.rs:4:24 4 if input.is_empty()_ { ^ expected `{` note: the `if` expression is missing a block after this condition --> code.rs:4:8 4 if input.is_empty()_ { </pre> <p>Expected Output</p> <p>abd</p> <p>Stdin</p> <p>aaabccddd</p> <p>Replace Value From: <input type="text"/></p> <p>Replace Value to: <input type="text"/></p> <p style="text-align: right;">Close</p> </div> <p style="text-align: right;">Test Run Submit</p>

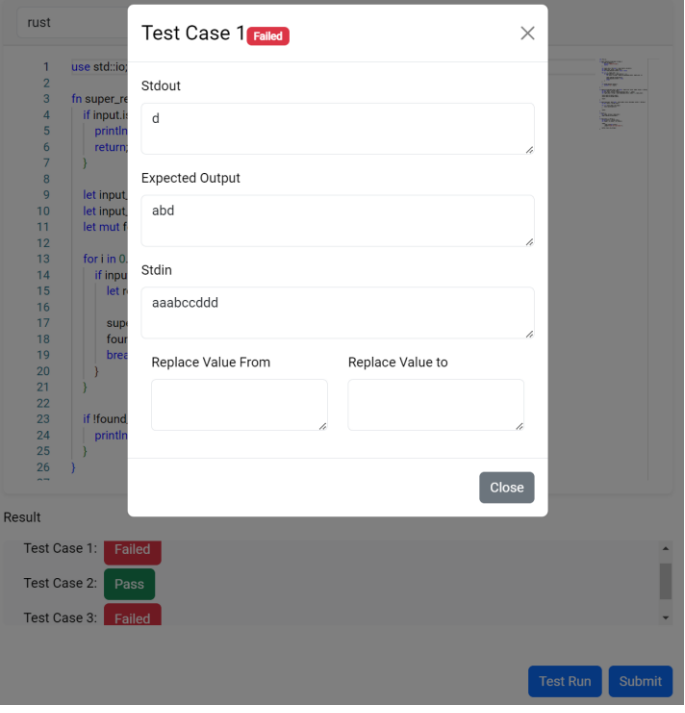
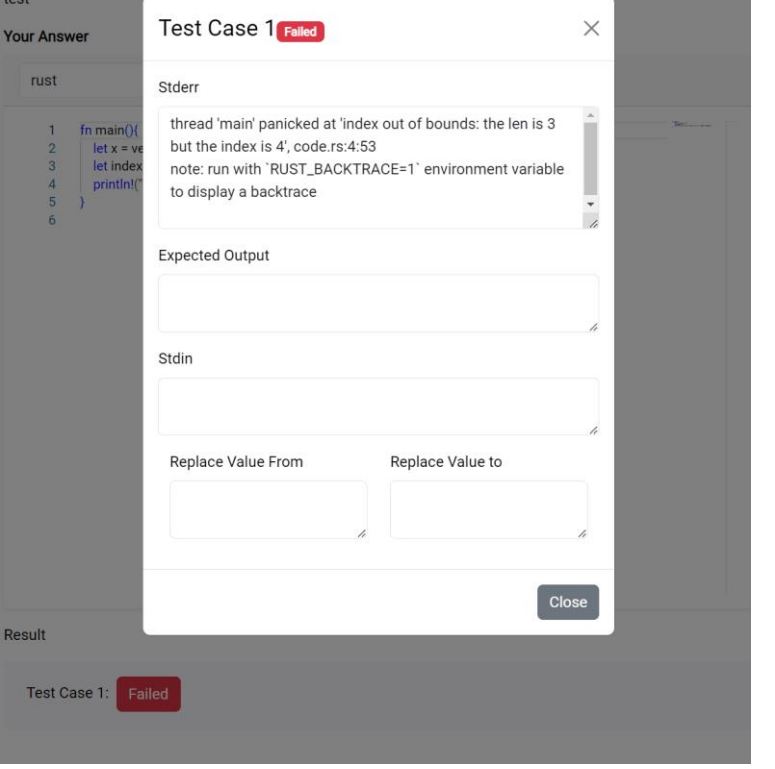
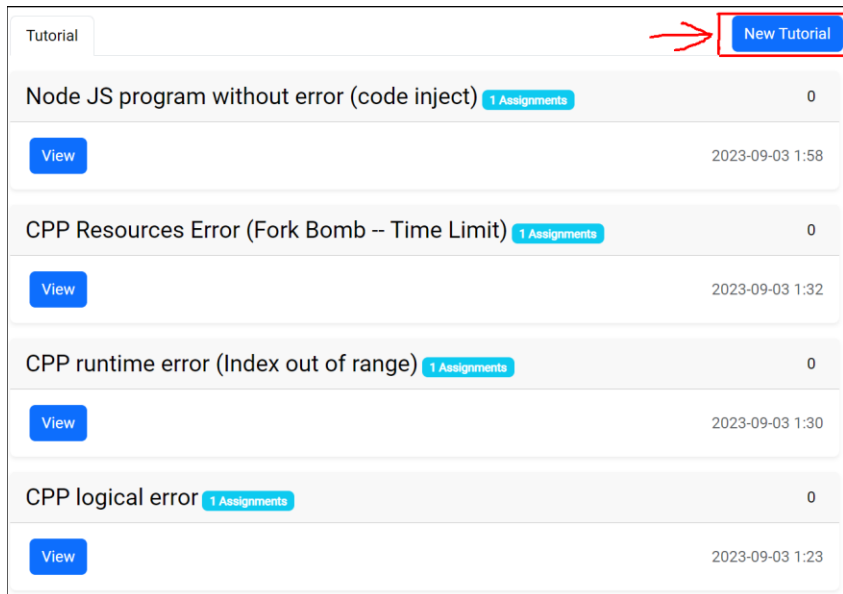
<p>Rust with Logical error</p>	<p>The test case result will show “fail”. However, the program returns valid stdout instead of stderr. In short, the program doesn’t have error, but the output doesn’t meet the test case. In this case, the logic only able to produce output match with test case 2.</p>	
<p>Rust run time error</p>	<p>Accessing index out of range will trigger error in Rust</p>	

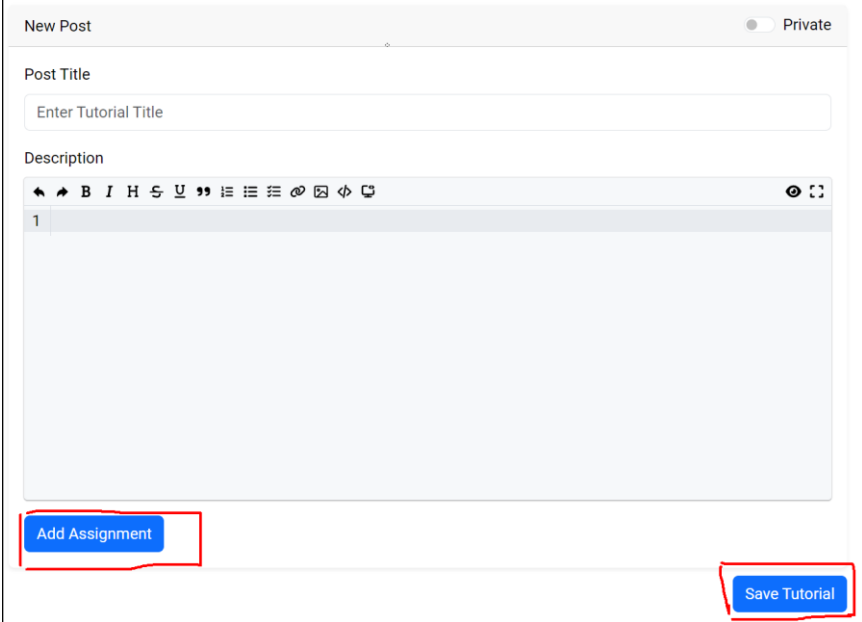
Table 5.1.4.1 is unit test done to CEE compiler to test the usage of Rust programming language. [42] docker image with rust compiler will be used to compile the program to executable. During this process, compiler time error will be generated such as syntax error. However, Rust has nice thread handling and unwrapping so there is no way to fork bomb using rust only. Refer to appendix [A-4](#) for Rust Program and configuration used in the test.

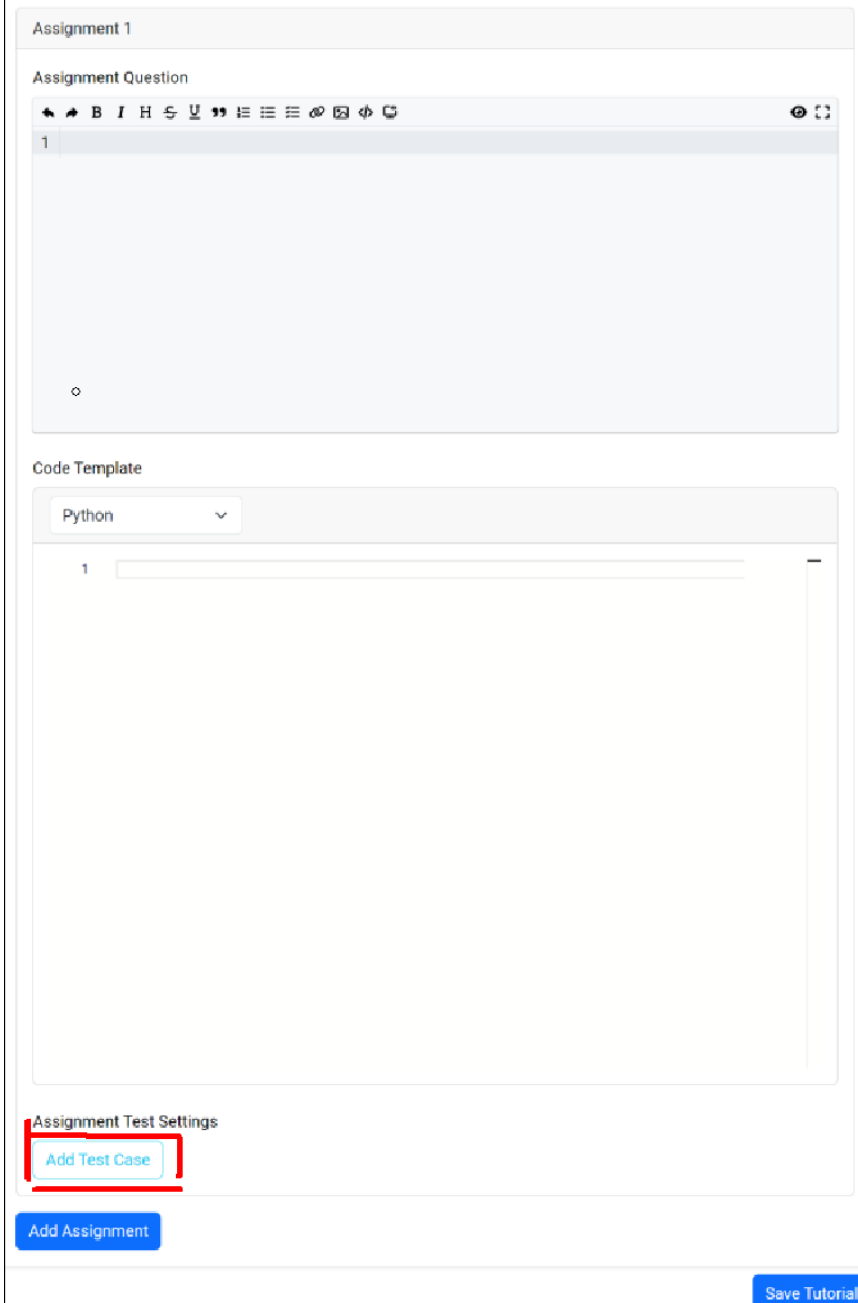
5.2 Platform Testing

This testing primary involve of user testing. It involve of user interaction with the system platform. This testing mainly involve evaluate either the object “To develop platform which allow users to submit their source code for marking” is achieved.

Table 5.2.1 Platform testing

Criteria	Result
User can add tutorial article with assignment.	<p>1. User can click “New Tutorial” to add new tutorial either include assignment or not.</p>  <p>The screenshot shows a search bar labeled 'Tutorial' on the left. To its right is a blue button labeled 'New Tutorial' which is highlighted with a red box. A red arrow points from the left towards this button. Below the search bar is a list of four tutorial entries, each with a 'View' button on the left and a '1 Assignments' badge and a count of '0' on the right. The entries are: 'Node JS program without error (code inject)', 'CPP Resources Error (Fork Bomb -- Time Limit)', 'CPP runtime error (Index out of range)', and 'CPP logical error'. Each entry also has a timestamp on the right.</p> <p>2. User can fill in the details of the tutorial include add assignment for the tutorial. Lastly save the tutorial if done.</p>

	 <p>The screenshot shows a 'New Post' form. At the top right, there is a 'Private' toggle switch. Below it is a 'Post Title' field with the placeholder text 'Enter Tutorial Title'. Underneath is a 'Description' field with a rich text editor toolbar containing icons for undo, redo, bold, italic, highlight, link, unlink, list, ordered list, link, unlink, and image. A red box highlights the 'Add Assignment' button at the bottom left of the form, and another red box highlights the 'Save Tutorial' button at the bottom right.</p>
<p>3. If user click for add assignment, user need to fill in the assignment details include assignment question, assignment programming language and assignment code template. Lastly, user can add test case for the system to automatically mark for the submission.</p>	



Assignment 1

Assignment Question

1

Code Template

Python

1

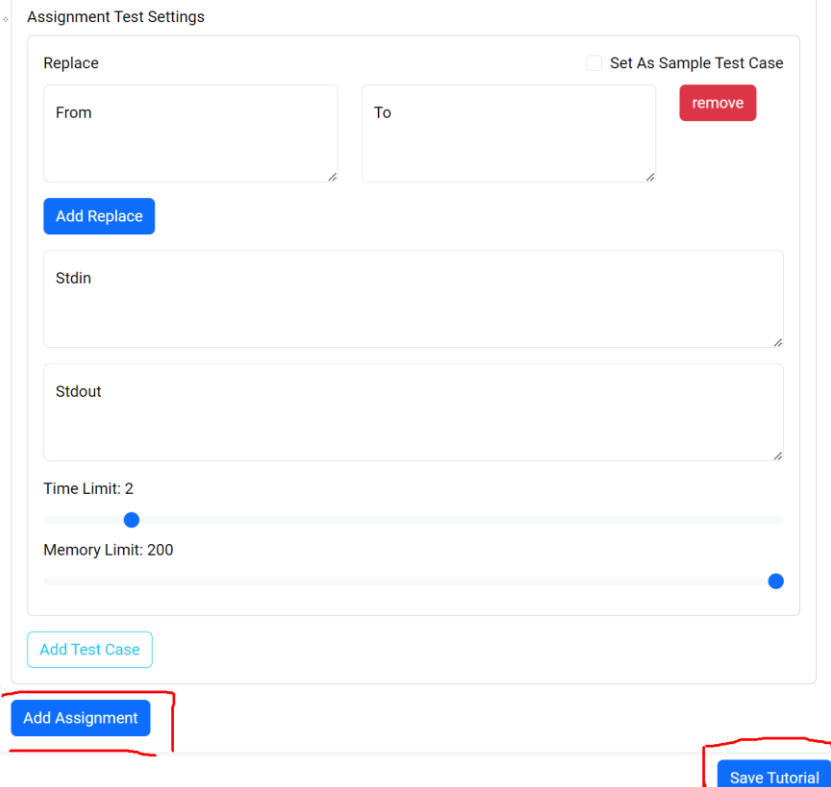
Assignment Test Settings

Add Test Case

Add Assignment

Save Tutorial

4. If user decide to add test setting, user need to fill in the test case details include the code inject, stdin, stdout, execution environment configuration. User can decide either set the test case as sample test case which only run for test run or test case that can run in test run or final submission. After fill in, user can either add another assignment or save the tutorial.



Assignment Test Settings

Replace Set As Sample Test Case

From To remove

Add Replace

Stdin

Stdout

Time Limit: 2


Memory Limit: 200

Add Test Case

Add Assignment

Save Tutorial

5. After saved the tutorial, the page will show the saved tutorial indicate the tutorial has been saved.

	 <p>The screenshot shows a web page with the following content:</p> <ul style="list-style-type: none"> Header: 'ystem Home' and a search bar. Title: 'List Comprehension (Python)' with a count of '0'. Text: 'List comprehension is a concise way of creating a new list in Python. It allows you to create a new list by applying an expression to each element of an existing list (or any iterable) that satisfies a certain condition.' Syntax: 'Here is the basic syntax for list comprehension in Python: new_list = [expression for item in iterable if condition]'. Below this, a list explains the parts: 'new_list' (name of new list), 'expression' (applied to each item), 'item' (variable for each item), 'iterable' (object returning elements), and 'condition' (optional filter). Examples: Three code snippets showing list creation for even numbers, squares, and names starting with 'S'. Text: 'List comprehension is a very powerful tool in Python and can help you write more concise and readable code. However, it's important to use it judiciously, as complex or nested list comprehensions can be difficult to read and debug.' Assignment 1: 'Assignment Question: Given a list of integers, write a Python program to create a new list containing only the even numbers from the original list using list comprehension.' Your Answer: A dropdown menu set to 'python' and a code editor containing: <pre>1 if __name__ == "__main__": 2 arr = list(map(int, input().split(" "))) 3 output = [] 4 print(output)</pre> Result: An empty text box for the output. Buttons: 'Test Run' and 'Submit'.
<p>User can test run assignment</p>	<p>1. This will be the assignment with be tested for this criteria. As shown in below figure, there will be two test case for the assignment, one test case is sample test case while the other is not. In other word,</p>

Chapter 5

with sample test cases and make submission with all test case	for test run, it will only involve one test case, while for submission, it will involve all test cases.
---	---

The screenshot displays the 'Automatic Marking System' interface. At the top, there is a navigation bar with 'Home', a search bar, and a user profile icon. The main content area is titled 'Edit Post' and includes a 'Public' toggle. The post title is 'Struct data type (CPP)'. The description section contains a rich text editor with a toolbar and a list of 13 numbered items. Item 1 is 'Struct CPP', item 2 is 'struct is a way to combine multiple fields to represent a composite data structure, which further lays the four', item 3 is 'struct can be represented as', item 4 is '...', item 5 is '...', item 6 is '...', item 7 is '...', item 8 is 'struct NewType {', item 9 is 'type1 value1;', item 10 is 'type2 value2;', item 11 is '...', item 12 is '...', and item 13 is '...'. Below the description is an 'Assignment Question' section with a rich text editor containing 13 numbered items. Item 1 is 'You have to create a struct, named Student, representing the student's details, as mentioned above, and', item 2 is '...', item 3 is '**Input Format**', item 4 is '...', item 5 is 'Input will consist of four lines.', item 6 is 'The first line will contain an integer, representing age.', item 7 is 'The second line will contain a string, consisting of lower-case Latin characters (a-z), representing the f', item 8 is 'The third line will contain another string, consisting of lower-case Latin characters (a-z), representing t', item 9 is 'The fourth line will contain an integer, representing the standard of student.', item 10 is '...', item 11 is 'Note: The number of characters in first_name and last_name will not exceed 50.', item 12 is '...', and item 13 is '**Output Format**'. Below the assignment question is a 'Code Template' section with a dropdown menu set to 'C++' and a code editor containing C++ code for a struct and a main function. The code includes headers for <math>, <stdio.h>, <vector>, <iostream>, and <algorithm>, and uses the <math> namespace. The main function reads four lines of input and prints the output. Below the code template is an 'Assignment Test Settings' section with two test case configurations. The first configuration has 'Set As Sample Test Case' checked and shows input '15 john carmack 10' and output '15 john carmack 10'. The second configuration has 'Set As Sample Test Case' unchecked and shows a long, complex input string and a corresponding output string. Both configurations have a 'Time Limit' of 2s and a 'Memory Limit' of 86mb (for the first) or 68mb (for the second). At the bottom right, there is an 'Update Tutorial' button.

This is sample test case

This is not sample test case

2. This is the execution result from test run. It only involve one test case, since there is one sample test case provide for test run.

Your Answer

```

cpp
1 #include <cmath>
2 #include <cstdio>
3 #include <vector>
4 #include <iostream>
5 #include <algorithm>
6 using namespace std;
7
8 struct Student {
9     int age;
10    string first_name;
11    string last_name;
12    int standard;
13 };
14
15 int main() {
16     Student st;
17
18     cin >> st.age >> st.first_name >> st.last_name >> st.standard;
19     cout << st.age << " " << st.first_name << " " << st.last_name << " " << st.standard;
20
21     return 0;
22 }

```

Result

Test Case 1: Pass

Test Run
Submit

3. This is the execution result from submission. It involve of 2 test case include sample test case and non-sample test case.

```

cpp
1 #include <cmath>
2 #include <cstdio>
3 #include <vector>
4 #include <iostream>
5 #include <algorithm>
6 using namespace std;
7
8 struct Student {
9     int age;
10    string first_name;
11    string last_name;
12    int standard;
13 };
14
15 int main() {
16     Student st;
17
18     cin >> st.age >> st.first_name >> st.last_name >> st.standard;
19     cout << st.age << " " << st.first_name << " " << st.last_name << " " << st.standard;
20
21     return 0;
22 }

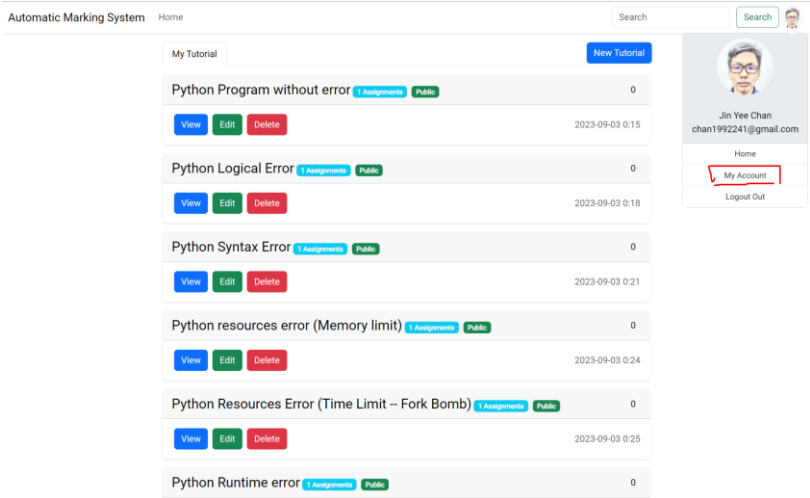
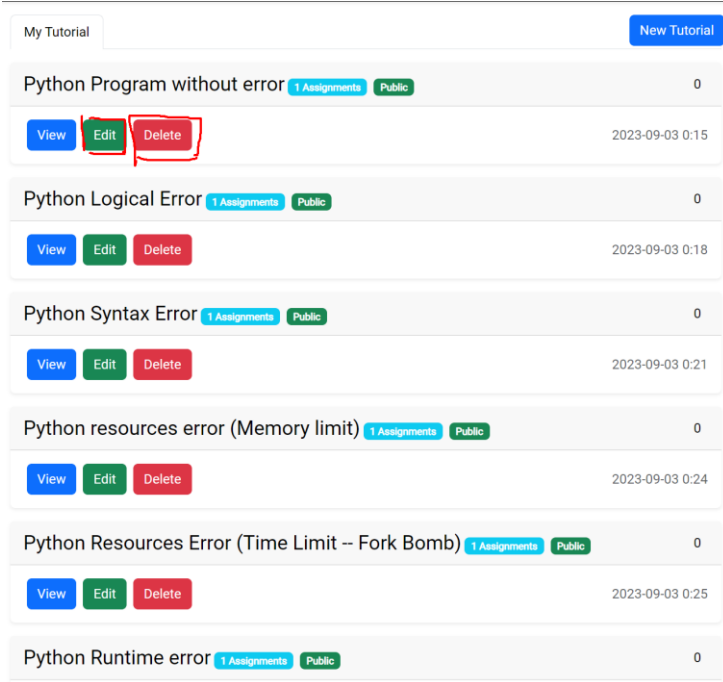
```

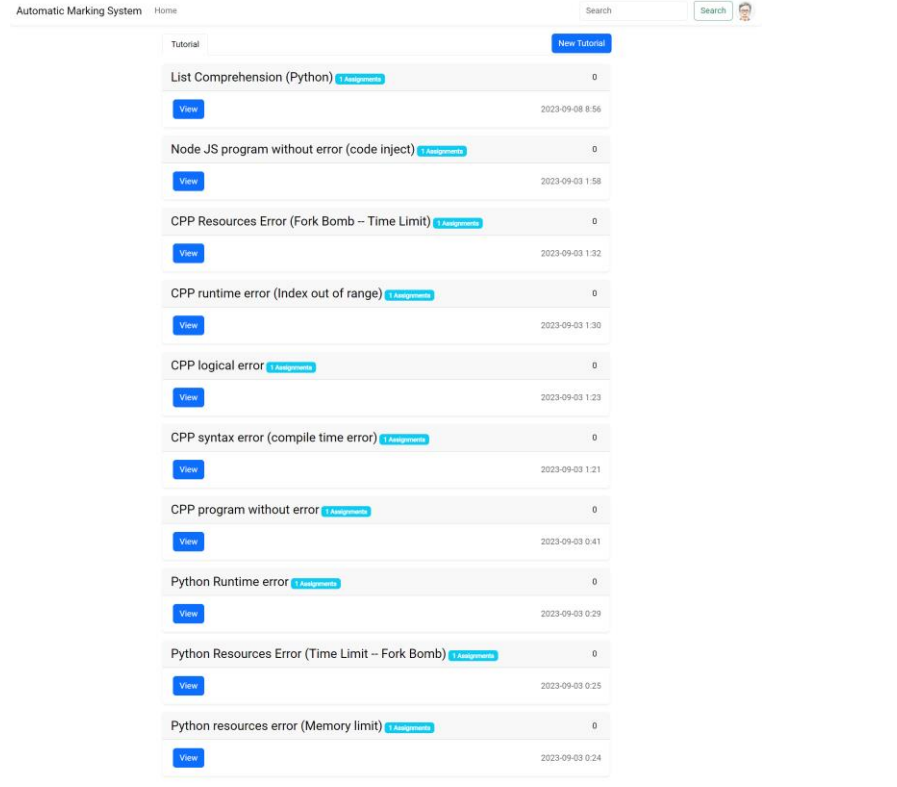
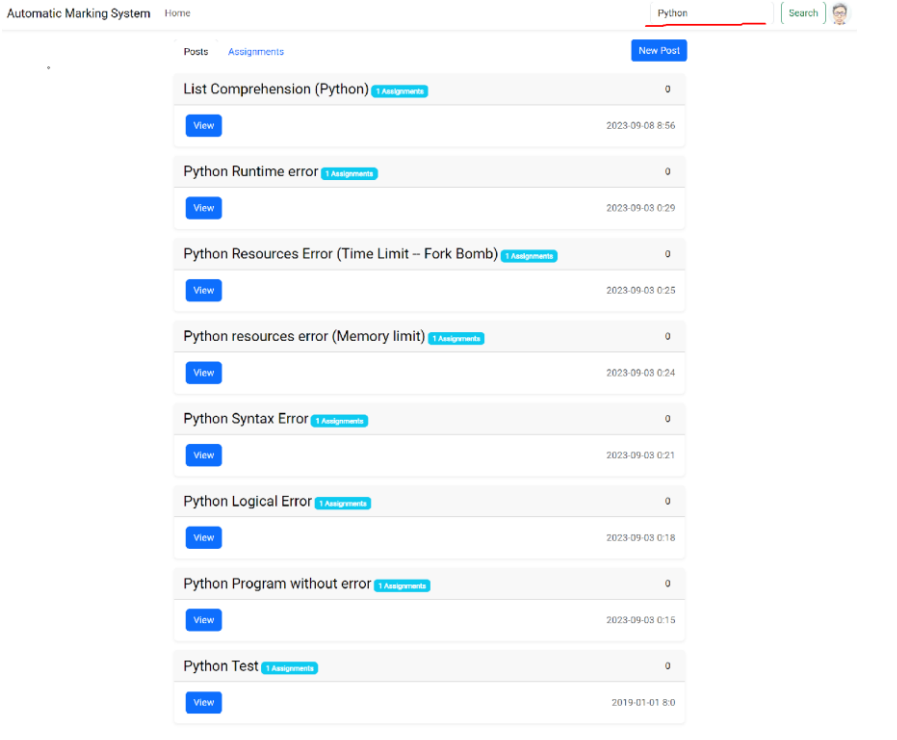
Result

Test Case 1: Pass

Test Case 2: Pass

Test Run
Submit

<p>User can edit and delete submission which make by itself.</p>	<p>1. If users want to delete or delete the tutorial they made, they can go to “my account” section.</p>  <p>2. Users can choose to edit or delete the tutorial they made by clicking the edit or delete button.</p> 
<p>User can view all tutorial made by other users</p>	<p>1. All tutorial posted by every user with public visibility will be available in main page.</p>

	
<p>User can search for tutorial publicly available.</p>	<p>1. User can search for tutorial at the search bar with keyword of the tutorial title. The display panel will only display tutorial with</p> 

5.3 Load Testing of CEE

This testing mainly involve of testing the scalability of the CEE and its performance. This evaluation mainly involve of checking either the object “To design scalable microservices architecture which ensure the submission can be done execute in shorter amount of time” is achieved. There are certain important criteria need to be considered for the tests details before the evaluation process being start. The criteria included

1. What is the number of test case for each assignment submission.
2. How many virtual users with concurrent submissions.
3. What is the memory limit and time limit for each submission.
4. What programming program with test cases to be executed.
5. What is the duration of stimulation test duration.

Then, the evaluation of the test process needs to be design carefully to represent testing process. The proposed evaluation methodology is to design benchmark equations. Below will be the proposed equations.

$$\left(\frac{\text{Total successful request} * \text{average response time}^2}{\text{Total successful request}} \right) + (\text{failed request count} * 30)$$

This equation was reference from research paper [5] where some amendment had been made. Instead of running request in batch, each request will run as standalone for certain interval. In short, the equation will punish longer response time and fail request. **The smaller the value, the lesser the response time.**

Besides, to simplify the evaluation process, there will be another endpoint purposely design to make submission for evaluation. The endpoint will not be exposed in production since the endpoint isn't required for authentication to access. The endpoint will query assignment data required by the test from application database (MongoDB) and pass it to router (submission handler) service, then the endpoint will continuously fetch submissions result from router service for 20 times with 1 seconds interval. In short, if there is no submission result after 20 seconds, the endpoint will return status code of 400 to represent execution failure. The reason behind this is the assumption that the users experience will drastically drop when wait more than 20 seconds.

Chapter 5

For simplicity, the program that will be executed in evaluation process is simple program that print “hello world” with 3 test cases in Python programming language with 2 second time limits and 200mb memory limit. Before discuss about number of virtual users with concurrent submission in the evaluation, there are some assumption need to be made included:

- There will be 1400 monthly active users.
- 10% will use the system daily.
- Each users run 3 assignment per day.
- Each users have 1 hour session.
- Each user will make submission every 2 to 4 seconds.

The evaluation mainly involve of different type of Load Testing. [43] Load testing is a type of performance test that evaluates a software system or application’s capability to handle normal and high workloads. There will be several type of load testing being conducted in this project include baseline load testing, spike testing and stress testing.

1. Baseline load testing is a type of testing focuses on evaluating the system's performance under normal and expected load conditions.
2. Spike testing tests the system’s capability to handle abrupt and substantial user load surges.
3. Stress testing will evaluate the system’s capability to handle higher traffic environments.

Then, the evaluation will involve of different number of virtual users to simulate real life traffic include regular traffic, spike traffic, and low traffic.

5.3.1 Performance of CEE in high traffic before scaled

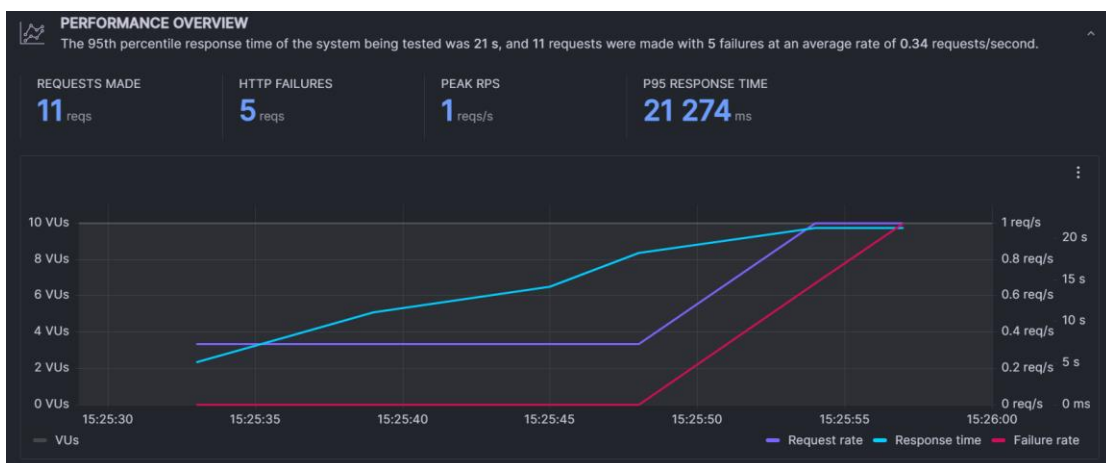


Figure 5.3.1.1 CEE metric before scaled

Before load testing toward production ready architecture, load test will be performed toward architecture which hadn't been scaled to visualize the difference and tell the problem for current architecture which without being scaled. This result from above figure is from load testing from 20 virtual users in 10 seconds duration, each virtual users will sent an request with maximum of 2 requests in 2 seconds intervals. As shown in the figure above, there will be 11 requests made and 5 requests is failures requests. The graph shown that the each request response time keep increase from 5 seconds until 20 seconds, the reason behind this is the CEE need to finish the code execution of first user's submission before proceed to another user's submission. If first user's submission take 5 seconds to execute, then the second user's submission need to wait 5 seconds to be executed. In short, this test show that it is not practical to execute the submission synchronously. The result also shows that the important of scale effectively based on the traffic or tasks to achieve better performance.

Using the benchmarking formular discuss in previous sections, we can calculate the benchmark for this evaluation has value of 591 which is consider bad, since the average response time is 21 seconds which over the expected response time which is lesser than 20 seconds.

$$\left(\frac{6 * 21^2}{6}\right) + (5 * 30) = 591$$

With this benchmark result, this performance can be compare to performance after scaled strategy had been implemented.

5.3.2 Performance of CEE after scaled

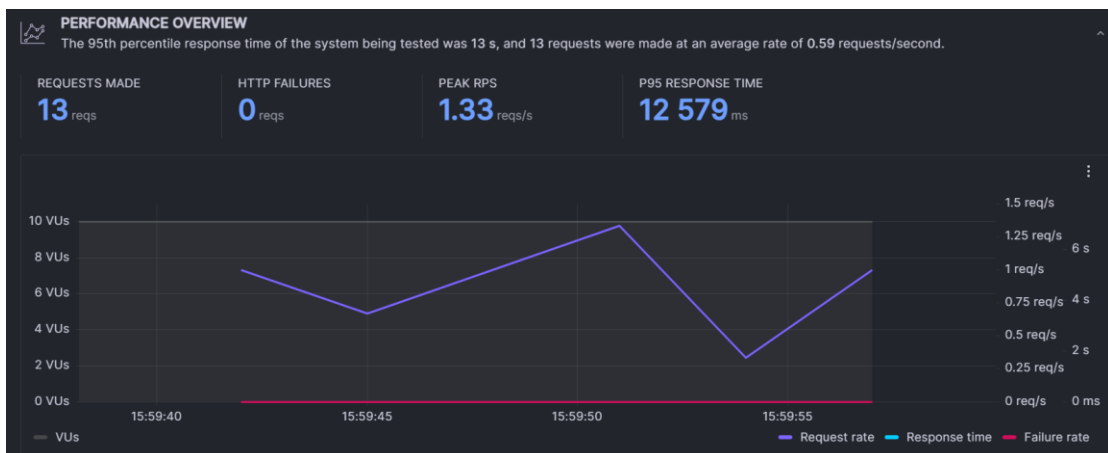


Figure 5.3.1.2 CEE metric after scaled

Above figure is the execution time for with scaled CEE from 20 virtual users in 10 seconds duration, each virtual users will sent an request with maximum of 2 requests in 2 seconds intervals. As shown in the figure above, there will be 13 requests made with 0 failures requests. The average response time is 10 seconds, the min response time is 6 seconds while the max response time is 13 seconds. The benchmark for this running evaluation is 100, which is better than without scaled CEE performance.

$$\left(\frac{13 * 10^2}{13} \right) + (0 * 30) = 100$$

Compare to benchmark result from CEE before scaled, this benchmark result had been improved.

Baseline Load Testing

Baseline load testing will stimulate real life normal traffic. In this case, there are some assumptions made including there will be 100 virtual users in each day. From this assumption, the virtual users will divided into different phases to simulate usage in different time of the day. Below is the table from the assumption made.

Time	Active Virtual Users
0000 – 8000	10 VUS
8000 – 1200	20 VUS
1200 – 1600	30 VUS

1600 – 2359	80 VUS
Total	140 VUS

Then the active virtual users will be scaled down to fit the load test duration. For example, for total of 3 minutes (180 seconds) load test, the load will be distributed according to the ratio from table above. Below is the number of virtual users in each phases.

Evaluation seconds	Active Virtual Users
0 – 54 seconds	2 vus
54 – 84 seconds	4 vus
84 – 114 seconds	7 vus
114 – 180 seconds	10 vus
Total	23 VUS (300 seconds, 3 mins)

Besides, there will be several scaling configuration will be evaluated and collected the performance result. The scaling configuration included minimum number of code execution environment (minimum Pod Count), number of instance increase according to queue length (Pod increase per task) and max code execution environment count (Max Pod Count). These configurations will be divided into different categories including “normal”, “optimized” and “saving”. The configuration of each category is shown as below table:

Scaling Category	Configuration
Normal	Max Pod Count: 5 Minimum Pod count: 2 Pod Increase per task: 1
Optimized	Max Pod Count: 8 Minimum Pod count: 2 Pod Increase per task: 2
Saving	Max Pod Count: 3 Minimum Pod count: 1 Pod Increase per task: 1

These configuration will be evaluated using the number of virtual users specified from above table to fulfill the different requirements. For example, if the requirements is to

have highly performance center system, then “optimized” configuration is recommended, if the requirement to have a less performance center but less cloud resources usage, then “saving” configuration is recommended else if the requirement to have median between performance and cloud resources usage, then “normal” configuration is recommended. Table below is the result from the load testing for different configurations.

Table 5.3.2.1 Baseline Load Testing Result

Scaling Strategy	No of failure response	No of successful response	Average Response time (seconds)	Min Response Time / Max Time (seconds)
Normal	0	68	9	4 / 17
Optimized	1	73	9	3 / 18
Saving	1	62	11	4 / 17

Description

According to result from above table, it shows that “normal” and “optimized” configurations have identical result. However, “optimized” configurations can handle more request under same time constraints. For “saving” configuration, the test shows it has less perform compare to other configurations. In short, the performance is acceptable under baseline traffic.

Spike Testing

In this testing, it involves of test the system’s capability to handle abrupt and substantial user load surges. This testing will ramp up the number of virtual users to 25 under 1 minutes, then the load ramp down to 0 to observe the side effect.

Table 5.3.2.2 Spike Testing Results

Scaling Strategy	No of failure response	No of successful response	Average Response time (seconds)	Min Response Time / Max Time (seconds)
Normal	60	12	13	4 / 22
Optimized	62	15	12	4 / 23
Saving	71	8	12	4 / 21

Description

According to result from above table, it seem the CEE require time more than 20 seconds to process each submission because the failure response is relative high compare to successful response. Recall from previous section, the evaluation endpoint will have 20 seconds of maximum to test the execution duration of submission, if the CEE take more than 20 seconds, then the evaluation endpoint will return failure status. In this test result, the “optimized” configuration able to handle the submission better compare to others configurations. “Saving” configuration perform most poorly.

Stress Testing

Stress testing will evaluate the system's capability to handle higher traffic environments. According to the mean of stress testing, the load testing is designed to fulfil the meaning of stress testing. This test will load twice of the baseline load testing's traffic, below is the number of virtual users being tested.

Evaluation seconds	Active Virtual Users
0 – 54 seconds	4 vus
54 – 84 seconds	8 vus
84 – 114 seconds	14 vus
114 – 180 seconds	20 vus
Total	46 VUS (300 seconds, 3 mins)

Table below is the result collected.

Table 5.3.2.3 Stress Testing Result

Scaling Strategy	No of failure response	No of successful response	Average Response time (seconds)	Min Response Time / Max Response Time (seconds)
Normal	50	49	9	4 / 22
Optimized	43	60	10	4 / 21
Saving	57	34	13	4 / 22

Description

From the result collected, it shows that the failure response count is higher compare to base line load testing. From this observation, it clearly shows that the almost half of the submission response time is longer that exceed 20 seconds. This result shows that “optimized” configuration able to handle more stress load well compared to other configurations. However, “saving” configuration perform the worst.

5.4 Evaluation Discussion

Recall from previous section, the evaluation include unit test for CEE, platform testing and load testing had been conducted. This section will discuss the results of evaluation from each testing. First, unit test for CEE is primarily to test the capability of CEE handle different scenarios for submission and check either the CEE able to execute with test case correctly. Second, there is platform testing conducted. This test will if the users able to use the platform like adding assignment, adding test case, submission execution etc. Lastly, load testing is conducted to examine the CEE performance under different traffic scenarios. These test well covered from system functionality to system performance and all test able to fulfill every project object include:

1. To design automatic marking system which able to execute submission code securely.

Unit test for CEE has included the test result to test the CEE under malicious code submission. Besides, the test show that the CEE will limit the resources include memory and time limit when handle each submission.

2. To develop platform which allow users to submit their source code for marking.

Platform test has included all functionality that make up this objective. Users can interact with system like adding assignment, set up test case, update assignment etc.

3. To design scalable microservices architecture which ensure the submission can be done execute in shorter amount of time

CEE has high scalability with its architecture. CEE can talk to its submission database and message broker service. With the load testing, it proves that CEE able to handle multiple submissions. Under normal circumstances, “normal” configurations perform well enough to handle each submission. If in a circumstances where cloud resources is

critical, “saving” configurations is recommended. Lastly, if in a situation where performance is critical, “optimized” configurations is recommended.

5.5 Innovate Features Testing – Code Inject

This feature is the innovate functionality added to easier lecturer process in designing programming questions. In normal circumstances, lecturer will pass data to the submission program through stdin. However, in the process of developing a input format is time consuming. This features provide another way for lecturers to pass input toward the submission program. Below is the evaluation of this feature.

This will be the submission program used to this evaluation.

```
#include <iostream>
#include <cstdio>
using namespace std;

/*
Add `int max_of_four(int a, int b, int c, int d)` here.
*/
int max_of_four(int a, int b , int c, int d) {
    int result = -9999;
    if (a > result){
        result = a;
    }
    if (b > result ){
        result = b;
    }
    if (c > result){
        result = c;
    }
    if (d > result) {
        result = d;
    }
    return result;
}

int main() {
    int ans = max_of_four(<replace-a>, <replace-b>, 4, 5);
    printf("%d", ans);
    return 0;
}
```

Chapter 5

Code inject mean replace some part of the code with certain value specified. From the above code, there will be two part be replace which is <replace-a> and <replace-B>. The value will be specified in assignment test setting. Below is example of achieve this features in test setting.

The screenshot displays the 'Assignment Test Settings' interface. At the top, there is a 'Replace' section with a checked checkbox labeled 'Set As Sample Test Case'. Below this, there are two rows of configuration. The first row shows 'From' as '<replace-a>' and 'To' as '2', with a red 'remove' button to the right. The second row shows 'From' as '<replace-b>' and 'To' as '3', also with a red 'remove' button. A blue 'Add Replace' button is positioned below these rows. Further down, there are two text input fields: 'Stdin' and 'Stdout', with the value '5' entered in the 'Stdout' field. At the bottom, there are two sliders: 'Time Limit: 2s' and 'Memory Limit: 200mb', both with blue circular markers indicating their current values.

Figure 5.5.1 Code Inject Test Testing

This test settings result in the part of the code being replace with another value in execution. From above example, <replace-a> will be replace with value 2 while <replace-b> will be replace with value 3 during execution. Below is the execution result from this configuration.

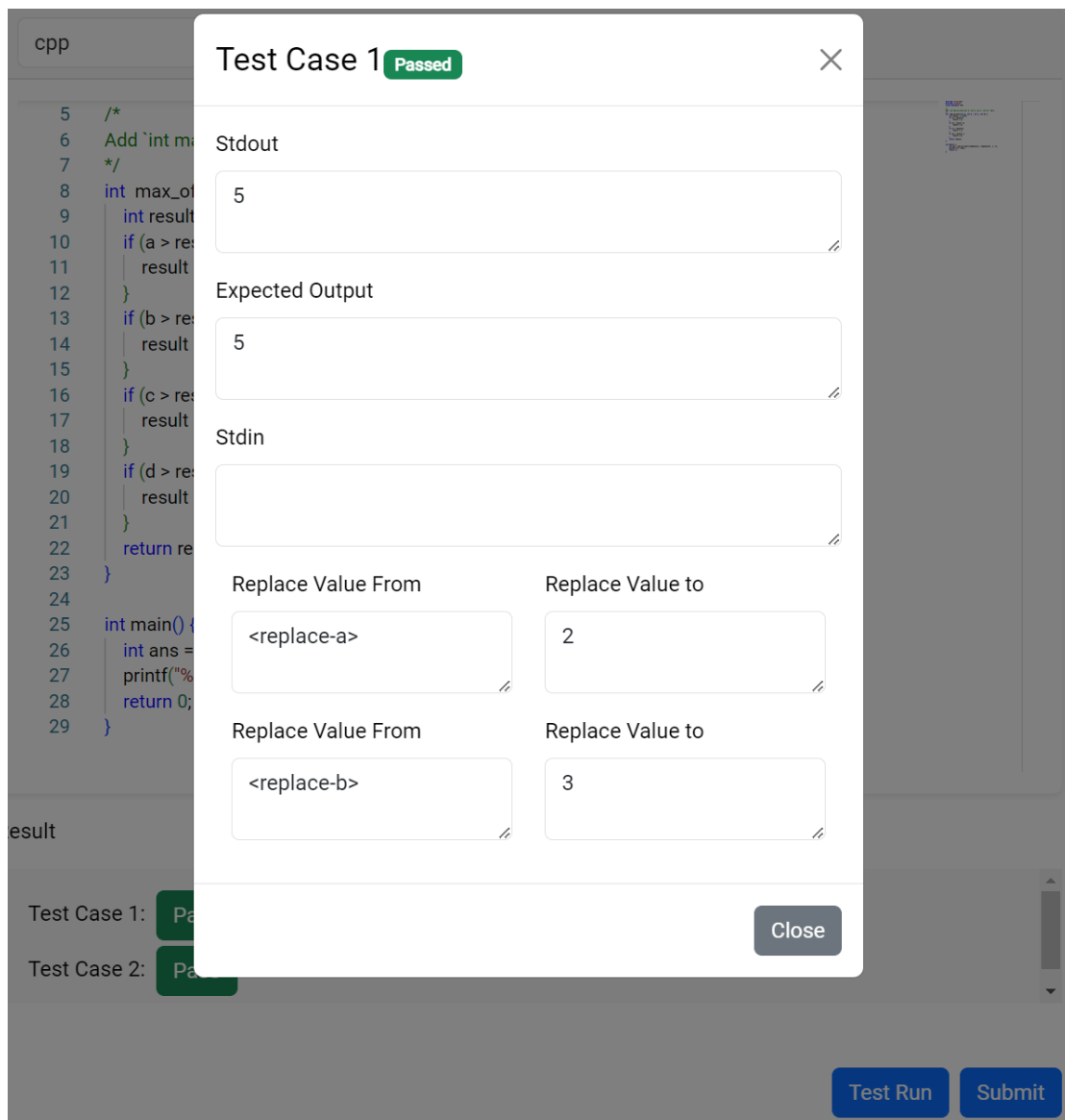


Figure 5.5.2 Code Inject Submission Result

5.6 Project challenges

This project involve of several challenge which may take longer time to solve including:

1. Architecture. Before start this project, there will be monolith and microservices architecture had been discuss. Lastly, microservice had been choose since this architecture is easier to scale and able to fit the project objective well.
2. How to handle submission. Users will submit the source to backend side for execution. Process and execution procedure of this process flow need to be solve to support its efficiency and maintainability. It is important that the flow support multiple submission. Besides, after the submission is ready to be executed, how

should the submission be executed. This challenges is discuss in CEE implementation section where the submission is handle by process initially. Later, this implementation is proved that is not secure when handle malicious code. Lastly, DinD technology is used to solve this problem.

3. **Scaling and multi-threading.** To implement a system with high scalability, horizontal scaling is recommended instead of vertical scaling. However, due to the involvement socket connection in CEE implementation, it is hard to scale due to the nature of socket connection. To solve this problem, event driven architecture is adopted. To scale the CEE, event base auto-scaling is used along with the architecture. Lastly, to optimize the execution process, multi-threading is used to handle each test case since each test case is independent to each other. However, the environment has limit resources, so if the concurrent execution take over the resources, it might cause the delay of the execution which cause the time limit of execution exceed. For example, a submission has 3 test case, each test case used up to 100 mb of memory, if CEE execute these 3 test cases at the same time, it might cause up to 300 mb of memory, but if the resources limit is 200mb, then some execution may delay and result in time limit exceeding. To solve this, there will be limited number of threading is allowed for parallel execution.
4. **Technology choice and learning curve.** Each technology used in this project requires some learning curve. Besides, not every technology is useful and require some exploration. For example, initially docker desktop is primarily development tool for Kubernetes cluster. However, due to certain bug with docker desktop for implement monitoring service, it force to migrate to Kind Kubernetes development cluster. To fully migrate, it required to learn how to setup load balancer, ingress etc to accommodate the need of new development environment.
5. **Load Testing.** Load testing not just involve of running the script. However, it involve of many decision making, like how many virtual users should be run concurrently, how many interval of each user to make submission, how many of the load is suitable of each type of testing etc. Besides, load testing also involve of analyze the result collected.

CHAPTER 6 Conclusion and Recommendations

6.1 Conclusion

In conclusion, this project will automate the process of marking programming exercise with test case provided with 3 objectives include to design automatic marking system which able to execute submission code securely, to develop platform which allow users to submit their source code for marking and to design scalable microservices architecture which ensure the submission can be done execute in shorter amount of time. The project aims to reduce the lecturer workload in marking programming exercise yet provide opportunity for students to practice programming. This project had proposed solution by create a scalable, manageable and resource efficiency automatic marking system.

This system contain of two main component includes front end and backend. Backend will contain component includes router (submission handler), message broker, submission database, judge service and Code Execution Environment (CEE). Router will act as interface for front end to connect to CEE where it will generate token that represent for that submission, the submission detail will save to submission database and the token will enqueue to message broker. CEE will act as secure environment to execute submission code which may be malicious. Later, the result from the execution will send to judge service to judge the output base on the test case provided. After the judge, the result will be saved to submission database. Router will provide endpoint to retrieve the submission result using the token provided.

To ensure the system not being affected by malicious code. CEE had been carefully designed with two components include worker and sandbox. Worker will handle for preparation of source code, input and output. Sandbox will provide secure environment to execute anonymous code. Once worker receive submission detail from submission queue, worker will prepare every file needed, then worker will launch new sandbox which contain interpreter or compiler for particular programming language and execute the code in that sandbox. The sandbox will be clean up once the execution is finished, and the worker will remove all the file involve of that submission.

Chapter 6

Every time the system needs to support new programming language, developer just need to reapply new CEE with new environment variable with the language details. Besides, CEE will handle each submission in first in first out sequence to ensure fairly. This approach is to simplify the management process of the CEE where developer can add or remove supported language by specifying in environment variable. Besides, multithreading is used to optimize the execution of submission. Lastly, to scale the CEE, event base auto-scaling had been implemented.

The front end will involve of web server which serve React application and handle for application resources. Besides, web server will act as gateway for user to access restricted resources and use of backend system. To achieve this, web server will connect to application database to store, retrieve, update and delete data related. The ideology behind of front-end web page design is user can upload tutorial with assignment that act as programming exercise to allow other users to practice what had been teach in the tutorial. Test Case for the assignment can be set to allow system to automatically mark for the assignment. After other users had attempt the assignment and make submission, web server will prepare data required by CEE and sent to backend for the execution result. Web server will return submission token toward web page where the web page will continuously fetch the submission result base on the token given.

In production, CEE require of server to execute code, this would cause certain amount of expenditure toward the server. Due to this issue, authentication is required to use the CEE to avoid CEE being exploited by anonymous users. To perform authentication and authorization in front end, Google Identity Service had been used. Once user login in web page, Google Authorization Server will issue “code” where the code will be sent to web server in exchange of user details. To manage the login context, Redux had been used together with React. When user successful login, Redux will store the JSON web token and username. In contrast, Redux will remove the JSON web token when user decided to logout.

To ensure every component involved able to integrate with each other and work as whole, Kubernetes will be used to deploy every component in local environment. Each component will expose certain port which allow other components to connect to that component. Besides, to replicate the deployment in local environment, infrastructure as

code tool, Terraform will be used to provision cloud resources required by the system and deploy the system using Kubernetes like configuration file.

6.2 Recommendation

Lastly, there are several recommendations to future research to explore include implement or research about Machine Learning-based Auto-Grading. Future research can explore the use of machine learning algorithms to automatically grade programming assignments. This would reduce the manual workload of instructors and provide more objective grading criteria for students. Besides, the current project is designed to handle several programming languages, but it would be interesting to analyse the performance of different programming languages in terms of execution time, memory usage, and scalability. This analysis can be used to optimize the system and enhance its performance.

REFERENCES

- [1] N. Yaacob*, A. L. Yeon, and R. A. Rahman, “Workload Of Academic Staff In Malaysian Public University: Perspectives Of Top Management,” presented at the ILC 2017 - 9th UUM International Legal Conference, Dec. 2018, pp. 873–883. doi: 10.15405/epsbs.2018.12.03.89.
- [2] W. Zhou, Y. Pan, Y. Zhou, and G. Sun, “The framework of a new online judge system for programming education,” in *Proceedings of ACM Turing Celebration Conference - China*, Shanghai China: ACM, May 2018, pp. 9–14. doi: 10.1145/3210713.3210721.
- [3] H. Z. Dosilovic and I. Mekterovic, “Robust and Scalable Online Code Execution System,” in *2020 43rd International Convention on Information, Communication and Electronic Technology (MIPRO)*, Opatija, Croatia: IEEE, Sep. 2020, pp. 1627–1632. doi: 10.23919/MIPRO48935.2020.9245310.
- [4] baeldung, “Layered Architecture | Baeldung on Computer Science,” Nov. 02, 2021. <https://www.baeldung.com/cs/layered-architecture> (accessed Apr. 24, 2023).
- [5] A. Hafiz and K. Jin, “Architecture of a Flexible and Cost-Effective Remote Code Execution Engine.” arXiv, May 03, 2021. Accessed: Dec. 19, 2022. [Online]. Available: <http://arxiv.org/abs/2105.01266>
- [6] “What are microservices?,” *microservices.io*. <http://microservices.io/index.html> (accessed Apr. 24, 2023).
- [7] “KEDA,” *KEDA*. <https://keda.sh/> (accessed Sep. 05, 2023).
- [8] H. Zhang, M. Zhang, F. Meng, X. Zhou, and D. Chu, “Application of the Online Judge Technology in Programming Experimental Teaching,” vol. 480, doi: 10.2991/assehr.k.201023.059.
- [9] “HackerRank - Online Coding Tests and Technical Interviews,” *HackerRank*. <https://www.hackerrank.com/> (accessed Apr. 19, 2023).
- [10] “What is agile methodology? Modern software development explained,” *InfoWorld*. <https://www.infoworld.com/article/3237508/what-is-agile-methodology-modern-software-development-explained.html>
- [11] “Load balancer types - Amazon Elastic Container Service.” <https://docs.aws.amazon.com/AmazonECS/latest/developerguide/load-balancer-types.html> (accessed Apr. 23, 2023).

References

- [12] “What is AMQP Protocol? All you need to know.” <https://www.wallarm.com/what/what-is-amqp> (accessed Apr. 23, 2023).
- [13] bunny.net, “What is Base64 Encoding & Decoding? | bunny.net.” <https://bunny.net/academy/http/what-is-base64-encoding-and-decoding/> (accessed Mar. 29, 2023).
- [14] “What is Terraform | Terraform | HashiCorp Developer,” *What is Terraform / Terraform / HashiCorp Developer.* <https://developer.hashicorp.com/terraform/intro> (accessed Apr. 25, 2023).
- [15] “Express - Node.js web application framework.” <https://expressjs.com/> (accessed Mar. 29, 2023).
- [16] “React.” <https://react.dev/> (accessed Mar. 29, 2023).
- [17] “Redux - A predictable state container for JavaScript apps. | Redux.” <https://redux.js.org/> (accessed Mar. 29, 2023).
- [18] “Production-Grade Container Orchestration,” *Kubernetes.* <https://kubernetes.io/> (accessed Apr. 18, 2023).
- [19] tomkerkhove, “Kubernetes Event-driven Autoscaling (KEDA) (Preview) - Azure Kubernetes Service,” Jun. 06, 2023. <https://learn.microsoft.com/en-us/azure/aks/keda-about> (accessed Sep. 05, 2023).
- [20] A. Kariya, “Answer to ‘Kubernetes node-exporter container is not working it shows this error message,’” *Stack Overflow*, Mar. 16, 2023. <https://stackoverflow.com/a/75756981> (accessed Jun. 30, 2023).
- [21] “kind.” <https://kind.sigs.k8s.io/> (accessed Jun. 30, 2023).
- [22] “Command Line Interface - AWS CLI - AWS,” *Amazon Web Services, Inc.* <https://aws.amazon.com/cli/> (accessed Apr. 23, 2023).
- [23] “Terraform CLI Documentation | Terraform | HashiCorp Developer,” *Terraform CLI Documentation | Terraform | HashiCorp Developer.* <https://developer.hashicorp.com/terraform/cli> (accessed Apr. 23, 2023).
- [24] “Welcome to Python.org,” *Python.org*, Apr. 13, 2023. <https://www.python.org/> (accessed Apr. 23, 2023).
- [25] “Ingress,” *Kubernetes.* <https://kubernetes.io/docs/concepts/services-networking/ingress/> (accessed Sep. 06, 2023).
- [26] “MetalLB, bare metal load-balancer for Kubernetes.” <https://metallb.universe.tf/> (accessed Sep. 06, 2023).

References

- [27] “index | Alpine Linux.” <https://www.alpinelinux.org/> (accessed Apr. 23, 2023).
- [28] “resource — Resource usage information,” *Python documentation*. <https://docs.python.org/3/library/resource.html> (accessed Feb. 24, 2023).
- [29] “Docker: Accelerated, Containerized Application Development,” May 10, 2022. <https://www.docker.com/> (accessed Feb. 24, 2023).
- [30] packagecloud, “3 Methods to Run Docker in Docker Containers | Packagecloud Blog.” <https://blog.packagecloud.io/3-methods-to-run-docker-in-docker-containers/> (accessed Feb. 25, 2023).
- [31] “Docker SDK for Python — Docker SDK for Python 6.0.1 documentation.” <https://docker-py.readthedocs.io/en/stable/index.html> (accessed Feb. 25, 2023).
- [32] “@monaco-editor/react,” *npm*, Aug. 23, 2023. <https://www.npmjs.com/package/@monaco-editor/react> (accessed Sep. 10, 2023).
- [33] “uiwjs/react-markdown-editor.” uiw, Sep. 08, 2023. Accessed: Sep. 12, 2023. [Online]. Available: <https://github.com/uiwjs/react-markdown-editor>
- [34] M. Sherif, “React OAuth2 | Google.” Apr. 07, 2023. Accessed: Apr. 07, 2023. [Online]. Available: <https://github.com/MomenSherif/react-oauth>
- [35] “Microservices Monitoring: Challenges, Metrics & Tips for Success,” *Lumigo*. <https://lumigo.io/microservices-monitoring/> (accessed Sep. 09, 2023).
- [36] “prometheus-operator/kube-prometheus: Use Prometheus to monitor Kubernetes and applications running on Kubernetes.” <https://github.com/prometheus-operator/kube-prometheus> (accessed Sep. 10, 2023).
- [37] “k6 Documentation.” <https://k6.io/docs> (accessed Sep. 10, 2023).
- [38] O. Toby, “The 7 Most Common Types of Errors in Programming and How to Avoid Them.” <https://textexpander.com/blog/the-7-most-common-types-of-errors-in-programming-and-how-to-avoid-them>
- [39] “python - Official Image | Docker Hub.” https://hub.docker.com/_/python (accessed Apr. 11, 2023).
- [40] “node - Official Image | Docker Hub.” https://hub.docker.com/_/node (accessed Apr. 18, 2023).
- [41] “frolvlad/alpine-gxx - Docker Image | Docker Hub.” <https://hub.docker.com/r/frolvlad/alpine-gxx/> (accessed Apr. 11, 2023).

References

- [42] “frolvlad/alpine-rust - Docker Image | Docker Hub.”
<https://hub.docker.com/r/frolvlad/alpine-rust> (accessed Apr. 11, 2023).
- [43] “What is Load Testing: Process, Tools, & Best Practices,” *BrowserStack*.
<https://browserstack.wpengine.com/guide/load-testing/> (accessed Aug. 23, 2023).

Weekly Log

FINAL YEAR PROJECT WEEKLY REPORT

(Project 2)

Trimester, Year: T1 Y4	Study week no.: 2
Student Name & ID: 1901409	
Supervisor: Ms Lai Siew Cheng	
Project Title: Automatic Marking System for Programming Subject	

1. WORK DONE

2. WORK TO BE DONE

- Re-architect CEE architecture
- setup monitoring services

3. PROBLEMS ENCOUNTERED


- Scaling with socket file seem very tricky

4. SELF EVALUATION OF THE PROGRESS

- No development progress



Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project 2)

Trimester, Year: T1 Y4	Study week no.: 4
Student Name & ID: Chan Jin Yee 1901409	
Supervisor: Ms Lai Siew Cheng	
Project Title: Automatic Marking System	

1. WORK DONE

- Monitoring services

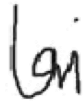
2. WORK TO BE DONE

- rearchitect CEE

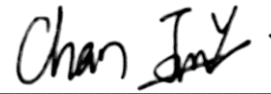
3. PROBLEMS ENCOUNTERED

- asynchronous execution of submission.

4. SELF EVALUATION OF THE PROGRESS



Supervisor's signature



Student's signatures

FINAL YEAR PROJECT WEEKLY REPORT

(Project 2)

Trimester, Year: T3 Y3	Study week no.: 7
Student Name & ID: Chan Jin Yee 1901409	
Supervisor: Ms Lai Siew Cheng	
Project Title: Automatic Marking System	

1. WORK DONE

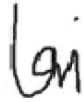
1. New scalable CEE architecture.

2. WORK TO BE DONE

1. Configure event auto scaling
2. Load testing

3. PROBLEMS ENCOUNTERED

4. SELF EVALUATION OF THE PROGRESS



Supervisor's signature



Student's signatures

FINAL YEAR PROJECT WEEKLY REPORT

(Project 2)

Trimester, Year: T3 Y3	Study week no.: 10
Student Name & ID: Chan Jin Yee 1901409	
Supervisor: Ms Lai Siew Cheng	
Project Title: Automatic Marking System For Programming Subject	

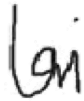
1. WORK DONE

1. Parallel submission execution
2. Load testing

2. WORK TO BE DONE

3. PROBLEMS ENCOUNTERED

4. SELF EVALUATION OF THE PROGRESS



Supervisor's signature



Student's signatures

APPENDIX

A-1. Python Program used in CEE unit testing

Python Program without error

```
if __name__ == '__main__':
    n = int(input())
    arr = map(int, input().split())
    arr = list(set(arr))
    arr.sort()
    print(arr[-2])
```

Test Case 1 (Stdin)

4
57 57 -57 57

Test Case 1 (Stdout)

-57

Test Case 2 (Stdin)

10
5 5 5 5 5 5 5 5 5 6

Test Case 2 (Stdout)

5

Python Program with syntax error

```
if __name__ == '__main__':
    n = int(input())
    arr = map(int, input().split())_
    arr = list(set(arr))
    arr.sort()
    print(arr[-2])
```

Test Case 1 (Stdin)

4
57 57 -57 57

Test Case 1 (Stdout)

-57

Test Case 2 (Stdin)

10
5 5 5 5 5 5 5 5 5 6

Test Case 2 (Stdout)

5

Python With Logical Error

Python Program with syntax error

```
if __name__ == '__main__':
    n = int(input())
```



```
arr = map(int, input().split())
arr = list(set(arr))
print(arr[-2])
```

Test Case 1 (Stdin)

4
57 57 -57 57

Test Case 1 (Stdout)

-57

Test Case 2 (Stdin)

10
5 5 5 5 5 5 5 5 5 6

Test Case 2 (Stdout)

5

Python Resources error. (Time Limit)

```
[o.fork() for (o, i) in [(__import__('os'), __import__('itertools'))]
for x in i.repeat(0)]
```

Test Case 1 (Stdin)

Test Case 1 (Stdout)

Test Case 1 Memory Limit: 50mb

Test Case 1 Time Limit: 2 seconds

Python Resources error. (Memory Limit)

```
repeatable_string = ""
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur vitae leo ac lacus iaculis semper. Praesent tincidunt vestibulum augue non egestas. In eu malesuada neque, non dictum lectus. Morbi ut efficitur arcu. Vestibulum non orci vel sapien feugiat cursus nec et magna. Aenean id sem aliquet, pulvinar est id, dignissim nibh. Vivamus lacinia mauris ut elit pulvinar gravida.

Vestibulum lacinia, est nec imperdiet sagittis, dui dui vulputate libero, non ultricies urna mauris et erat. Nunc venenatis eros sed nisl pretium, sed euismod neque maximus. Sed tincidunt scelerisque ipsum nec vulputate. Donec non lorem arcu. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. In sagittis turpis non elementum fringilla. Phasellus pretium euismod lorem, ac sodales tortor consequat in.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque mattis dictum lorem at scelerisque. Praesent eu vestibulum mi. Curabitur tincidunt tortor facilisis elit fringilla, quis lobortis dui euismod. Praesent nec molestie orci, eget faucibus augue. Mauris quis felis dapibus, vestibulum eros id, dignissim ante. Fusce venenatis elit sit amet ipsum euismod, ac molestie magna ullamcorper. Duis tempor arcu id maximus pretium. Cras hendrerit ante convallis tortor interdum consequat. Cras at risus quis mauris vehicula dapibus.

Mauris sed commodo eros. Quisque ac elit est. Curabitur maximus consequat laoreet. Fusce facilisis, felis non cursus egestas, nibh sapien consectetur nisi, cursus ultricies arcu nisi vitae justo. Maecenas sed eleifend odio. Praesent consectetur aliquet velit, vitae pretium dolor finibus et. Nulla facilisi. Fusce iaculis in dolor eu sodales. Mauris sodales ac tortor eget euismod. Sed pretium scelerisque ipsum, in aliquet turpis. In hac habitasse platea dictumst.

Morbi tincidunt eleifend urna sed dignissim. Nunc id tincidunt metus, nec consectetur nunc. Curabitur sed placerat magna. Morbi in tincidunt elit. Sed imperdiet malesuada mi, sit amet dignissim neque feugiat et. In tincidunt laoreet neque, vel consequat velit interdum eget. Morbi in erat euismod, vulputate nulla id, fringilla nisi. Morbi eget tortor id urna condimentum fermentum sit amet a orci.

```
"" * 100000000
```

```
print(repeatable_string)
```

Test Case 1 Memory Limit: 50mb

Test Case 1 Time Limit: 2 seconds

A-2 NodeJS Program used in CEE unit testing

NodeJS Program without error

```
let n;  
let arr = ["<replace-1>"];  
  
arr = [...new Set(arr)];  
arr.sort((a, b) => b - a);  
console.log(arr[1]);
```

Test Case 1 Replace From

"<replace-1>"

Test Case 1 Replace To

57, 57, -57, 57

Test Case 1 (Stdout)

-57

Test Case 2 Replace From

"<replace-1>"

Test Case 2 Replace To

5, 5, 5, 5, 5, 5, 5, 5, 5, 6

Test Case 2 (Stdout)

5

NodeJS Program with syntax error (Runtime Error)

```
let n;
let arr = ["<replace-1>"];
```

```
arr = [...new Set(arr)];
arr.sort((a, b) = b - a);
console.log(arr[1]);
```

Test Case 1 Replace From

"<replace-1>"

Test Case 1 Replace To

57, 57, -57, 57

Test Case 1 (Stdout)

-57

Test Case 2 Replace From

"<replace-1>"

Test Case 2 Replace To

5, 5, 5, 5, 5, 5, 5, 5, 5, 6

Test Case 2 (Stdout)

5

NodeJS With Logical Error

```
let n;
let arr = ["<replace-1>"];
```

```
arr = [...new Set(arr)];
arr.sort((a, b) => b - a);
console.log(arr[0]);
```

Test Case 1 Replace From

"<replace-1>"

Test Case 1 Replace To

57, 57, -57, 57

Test Case 1 (Stdout)

-57

Test Case 2 Replace From

"<replace-1>"

Test Case 2 Replace To

5, 5, 5, 5, 5, 5, 5, 5, 5, 6

Test Case 2 (Stdout)

5

NodeJS Resources error. (Time Limit)

```
(function f() {
  require('child_process').spawn(
    process.argv[0],
    ['-e', '(' + f.toString() + '());'])
  ;}()
);
```

Test Case 1 (Stdin)**Test Case 1 (Stdout)****Test Case 1 Memory Limit: 50mb****Test Case 1 Time Limit: 2 seconds****A-3 C++ Program used in CEE unit testing**

CPP Program without error

```
#include <iostream>
#include <string>
#include <vector>
using namespace std;

void solve(string str, int strlen, int noOftestCase){
  int counter = 1;
  cout << "Case #" << noOftestCase + 1 << ": ";
  for (int i = 0; i < strlen - 1; i++){
    cout << counter << " ";
    if (str[i] < str[i+1]){
      counter++;
    }else counter = 1;
  }
  cout << counter << endl;
}

int main(){
  int noOfTestCase, strlenHold;
  string strHold;
```

```

vector<int> strlength;
vector<string> str;
cin >> noOfTestCase;
cin.ignore();
for (int i =0 ;i<noOfTestCase; i++){
    cin >> strlengthHold;
    cin.ignore();
    strlength.push_back(strlengthHold);
    cin >> strHold;
    cin.ignore();
    str.push_back(strHold);
    solve(str[i], strlength[i], i);
}
}

```

Test Case 1 (Stdin)

2

4

ABBC

6

ABACDA

Test Case 1 (Stdout)

Case #1: 1 2 1 2

Case #2: 1 2 1 2 3 1

CPP with compile error (syntax error)

```

#include <iostream>
#include <string>
#include <vector>
using namespace std;

void solve(string str, int strlen, int noOftestCase){
    int counter = 1;
    cout << "Case #" << noOftestCase +1<< ": "
    for (int i = 0; i < strlen - 1; i++){
        cout << counter << " ";
        if (str[i] < str[i+1]){
            counter++;
        }else counter = 1;
    }
    cout << counter << endl;
}

int main(){
    int noOfTestCase, strlengthHold;
    string strHold;
    vector<int> strlength;
    vector<string> str;
}

```

```

cin >> noOfTestCase;
cin.ignore();
for (int i =0 ;i<noOfTestCase; i++){
    cin >> strlengthHold;
    cin.ignore();
    strlength.push_back(strlengthHold);
    cin >> strHold;
    cin.ignore();
    str.push_back(strHold);
    solve(str[i], strlength[i], i);
}
}

```

Test Case 1 (Stdin)

2

4

ABBC

6

ABACDA

Test Case 1 (Stdout)

Case #1: 1 2 1 2

Case #2: 1 2 1 2 3 1

CPP with Logical error

```

#include <iostream>
#include <string>
#include <vector>
using namespace std;

void solve(string str, int strlen, int noOftestCase){
    int counter = 1;
    cout << "Case #" << noOftestCase +1<< ": ";
    for (int i = 0; i < strlen - 1; i++){
        cout << counter << " ";
        if (str[i] < str[i+1]){
            counter++;
        }else counter = 1;
    }
    cout << counter + 1<< endl;
}

int main(){
    int noOfTestCase, strlengthHold;
    string strHold;
    vector<int> strlength;
    vector<string> str;
    cin >> noOfTestCase;
    cin.ignore();
}

```

```

    for (int i =0 ;i<noOfTestCase; i++){
        cin >> strlengthHold;
        cin.ignore();
        strlength.push_back(strlengthHold);
        cin >> strHold;
        cin.ignore();
        str.push_back(strHold);
        solve(str[i], strlength[i], i);
    }
}

```

Test Case 1 (Stdin)

2

4

ABBC

6

ABACDA

Test Case 1 (Stdout)

Case #1: 1 2 1 2

Case #2: 1 2 1 2 3 1

CPP with run time error

```

#include <iostream>
using namespace std;

int main(){
    int a[5];
    for (int i = 0; i < 5; i++){
        cout << a[i] << endl;
    }
    cout << a[6] << endl;
}

```

Test Case 1 (Stdin)***Test Case 1 (Stdout)*****CPP resources error (Fork Bomb)**

```

#include <cstdlib>
int main(int argc, char **argv){
    while (1) system(argv[0]);
    return 0;
}

```

Test Case 1 (Stdin)***Test Case 1 (Stdout)***

Test Case 1 Memory Limit: 50mb

Test Case 1 Time Limit: 2 seconds

A-4 Rust Program used in CEE unit testing

Rust Program without error

```
use std::io;

fn super_reduced_string(input: String) {
    if input.is_empty() {
        println!("Empty String");
        return;
    }

    let input_chars: Vec<char> = input.chars().collect();
    let input_size: usize = input.len();
    let mut found_adjacent_letters: bool = false;

    for i in 0..(input_size - 1) {
        if input_chars[i] == input_chars[i + 1] {
            let result: String = reduce_string(&input_chars,
input_size, i);

            super_reduced_string(result);
            found_adjacent_letters = true;
            break;
        }
    }

    if !found_adjacent_letters {
        println!("{}", input);
    }
}

fn reduce_string(input_chars: &Vec<char>, input_size: usize, index:
usize) -> String {
    let mut result = String::new();
    let left_input: String = substring(&input_chars, 0, index);
    let right_input: String = substring(&input_chars, index + 2,
input_size);

    result.push_str(&left_input);
    result.push_str(&right_input);
}
```



```

    result
}

fn substring(input: &Vec<char>, start_index: usize, end_index: usize)
-> String {
    let mut result = String::new();

    for i in start_index..end_index {
        result.push(input[i]);
    }

    result
}

fn main() {
    let input: String = read_line();
    super_reduced_string(input);
}

fn read_line() -> String {
    let mut buffer = String::new();
    let reader: io::Stdin = io::stdin();

    reader
        .read_line(&mut buffer)
        .expect("Could not read stdin!");

    buffer.trim().to_string()
}

```

Test Case 1 (Stdin)

aaabccddd

Test Case 1 (Stdout)

abd

Test Case 2 (Stdin)

aa

Test Case 2 (Stdout)

Empty String

Test Case 3 (Stdin)

baab

Test Case 3 (Stdout)

Empty String

Rust with compile error (syntax error)

```

use std::io;

fn super_reduced_string(input: String) {
    if input.is_empty() {
        println!("Empty String");
        return;
    }

    let input_chars: Vec<char> = input.chars().collect();
    let input_size: usize = input.len();
    let mut found_adjacent_letters: bool = false;

    for i in 0..(input_size - 1) {
        if input_chars[i] == input_chars[i + 1] {
            let result: String = reduce_string(&input_chars,
input_size, i);

            super_reduced_string(result);
            found_adjacent_letters = true;
            break;
        }
    }

    if !found_adjacent_letters {
        println!("{}", input);
    }
}

fn reduce_string(input_chars: &Vec<char>, input_size: usize, index:
usize) -> String {
    let mut result = String::new();
    let left_input: String = substring(&input_chars, 0, index);
    let right_input: String = substring(&input_chars, index + 2,
input_size);

    result.push_str(&left_input);
    result.push_str(&right_input);

    result
}

fn substring(input: &Vec<char>, start_index: usize, end_index: usize)
-> String {
    let mut result = String::new();

    for i in start_index..end_index {
        result.push(input[i]);
    }
}

```

```

    result
}

fn main() {
    let input: String = read_line();
    super_reduced_string(input);
}

fn read_line() -> String {
    let mut buffer = String::new();
    let reader: io::Stdin = io::stdin();

    reader
        .read_line(&mut buffer)
        .expect("Could not read stdin!");

    buffer.trim().to_string()
}

```

Test Case 1 (Stdin)

aaabccddd

Test Case 1 (Stdout)

abd

Test Case 2 (Stdin)

aa

Test Case 2 (Stdout)

Empty String

Test Case 3 (Stdin)

baab

Test Case 3 (Stdout)

Empty String

Rust With Logical Error

```

use std::io;

fn super_reduced_string(input: String) {
    if input.is_empty() {
        println!("Empty String");
        return;
    }

    let input_chars: Vec<char> = input.chars().collect();

```

```

let input_size: usize = input.len();
let mut found_adjacent_letters: bool = false;

for i in 0..(input_size - 1) {
    if input_chars[i] == input_chars[i + 1] {
        let result: String = reduce_string(&input_chars,
input_size, i);

        super_reduced_string(result);
        found_adjacent_letters = true;
        break;
    }
}

if !found_adjacent_letters {
    println!("{}", input);
}
}

fn reduce_string(input_chars: &Vec<char>, input_size: usize, index:
usize) -> String {
    let mut result = String::new();
    let left_input: String = substring(&input_chars, 1, index);
    let right_input: String = substring(&input_chars, index + 2,
input_size);

    result.push_str(&left_input);
    result.push_str(&right_input);

    result
}

fn substring(input: &Vec<char>, start_index: usize, end_index: usize)
-> String {
    let mut result = String::new();

    for i in start_index..end_index{
        result.push(input[i]);
    }

    result
}

fn main() {
    let input: String = read_line();
    super_reduced_string(input);
}

```

```
fn read_line() -> String {  
    let mut buffer = String::new();  
    let reader: io::Stdin = io::stdin();  
  
    reader  
        .read_line(&mut buffer)  
        .expect("Could not read stdin!");  
  
    buffer.trim().to_string()  
}
```

Test Case 1 (Stdin)

aaabccddd

Test Case 1 (Stdout)

abd

Test Case 2 (Stdin)

aa

Test Case 2 (Stdout)

Empty String

Test Case 3 (Stdin)

baab

Test Case 3 (Stdout)

Empty String

Rust with run time error

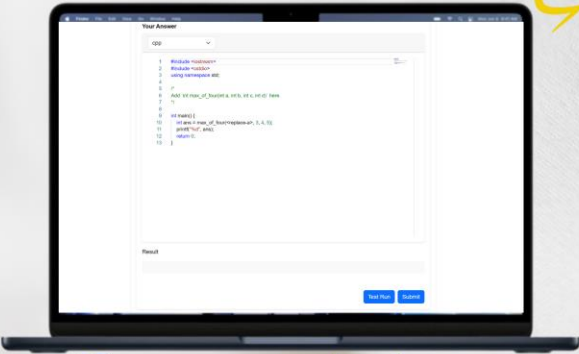
```
fn main(){  
    let x = vec![1,2,3];  
    let index = 4;  
    println!("The value at index {} is: {}", index, x[index]);  
}
```

Test Case 1 (Stdin)


Test Case 1 (Stdout)

POSTER

AUTOMATIC MARKING SYSTEM FOR PROGRAMMING SUBJECT



Saviour of Lecturer




Student: Chan Jin Yee
Supervisor: Ms Lai Siew Cheng

Objective

1. Reduce Workload of Lecturer
2. Provide Learning Space for student toward programming exercise
3. Help Lecturer mark for programming exercise.

Features

1. Secure Environment to execute anonymous code
2. Editor for users to write their code
3. Allow users to test run their program.
4. Execution and Judge output will be displayed.
5. Markdown editor to set exercise.



Wow!

PLAGIARISM CHECK RESULT

Automatic Marking System For Programming Subject

ORIGINALITY REPORT

1%

SIMILARITY INDEX

1%

INTERNET SOURCES

0%

PUBLICATIONS

0%

STUDENT PAPERS

PRIMARY SOURCES

- 1** Submitted to Dhirubhai Ambani Institute of Information and Communication <1%
Student Paper
- 2** Herman Zvonimir Dosilovic, Igor Mekterovic. "Robust and Scalable Online Code Execution System", 2020 43rd International Convention on Information, Communication and Electronic Technology (MIPRO), 2020 <1%
Publication
- 3** www.oopen.org <1%
Internet Source
- 4** Nasima Begum, Md Azim Hossain Akash, Sayma Rahman, Jungpil Shin, Md Rashedul Islam, Md Ezharul Islam. "User Authentication Based on Handwriting Analysis of Pen-Tablet Sensor Data Using Optimal Feature Selection Model", Future Internet, 2021 <1%
Publication
- 5** Ron C. L'Esteve. "Chapter 10 Owning a Portfolio of Agile Cloud Products", Springer Science and Business Media LLC, 2023 <1%

PLAGIARISM CHECK RESULT

Publication

6	dalspace.library.dal.ca Internet Source	<1 %
7	www.coursehero.com Internet Source	<1 %
8	www.science.gov Internet Source	<1 %
9	digitalcommons.njit.edu Internet Source	<1 %
10	www.vingle.net Internet Source	<1 %
11	www.rishabhsoft.com Internet Source	<1 %
12	third-bit.com Internet Source	<1 %

Exclude quotes On
Exclude bibliography Off

Exclude matches < 8 words

PLAGIARISM CHECK RESULT

Universiti Tunku Abdul Rahman			
Form Title : Supervisor's Comments on Originality Report Generated by Turnitin			
for Submission of Final Year Project Report (for Undergraduate Programmes)			
Form Number: FM-IAD-005	Rev No.: 0	Effective Date: 01/10/2013	Page No.: 1 of 1



FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Full Name(s) of Candidate(s)	Chan Jin Yee
ID Number(s)	1901409
Programme / Course	Information System Engineering
Title of Final Year Project	Automatic Marking System for Programming Subject

Similarity	Supervisor's Comments (Compulsory if parameters of originality exceeds the limits approved by UTAR)
Overall similarity index: <u> 1 </u> % Similarity by source Internet Sources: <u> 1 </u> % Publications: <u> 0 </u> % Student Papers: <u> 0 </u> %	
Number of individual sources listed of more than 3% similarity: <u> N.A. </u>	
Parameters of originality required and limits approved by UTAR are as Follows: (i) Overall similarity index is 20% and below, and (ii) Matching of individual sources listed must be less than 3% each, and (iii) Matching texts in continuous block must not exceed 8 words	

Note Supervisor/Candidate(s) is/are required to provide softcopy of full set of the originality report to Faculty/Institute

Based on the above results, I hereby declare that I am satisfied with the originality of the Final Year Project Report submitted by my student(s) as named above.

Ln

PLAGIARISM CHECK RESULT

Signature of Supervisor

Name: Ms Lai Siew Cheng

Date: 15 September 2023

Signature of Co-Supervisor

Name: _____

Date: _____



UNIVERSITI TUNKU ABDUL RAHMAN

FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY

(KAMPAR CAMPUS)

CHECKLIST FOR FYP2 THESIS SUBMISSION

Student Id	19ACB01409
Student Name	Chan Jin Yee
Supervisor Name	Lai Siew Cheng

TICK (√)	DOCUMENT ITEMS
	Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item.
√	Title Page
√	Signed Report Status Declaration Form
√	Signed FYP Thesis Submission Form
√	Signed form of the Declaration of Originality
√	Acknowledgement
√	Abstract
√	Table of Contents
√	List of Figures (if applicable)
√	List of Tables (if applicable)
	List of Symbols (if applicable)
√	List of Abbreviations (if applicable)
√	Chapters / Content
√	Bibliography (or References)
√	All references in bibliography are cited in the thesis, especially in the chapter of literature review
√	Appendices (if applicable)
√	Weekly Log

FYP 2 Checklist

√	Poster
√	Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005)
√	I agree 5 marks will be deducted due to incorrect format, declare wrongly the ticked of these items, and/or any dispute happening for these items in this report.

*Include this form (checklist) in the thesis (Bind together as the last page)

I, the author, have checked and confirmed all the items listed in the table are included in my report.



(Signature of Student)

Date: 15 September 2023