School of Electronics and Computer Science

FACULTY OF ENGINEERING AND PHYSICAL SCIENCES

UNIVERSITY OF SOUTHAMPTON

Designing and Implementation of Online AI Experiment Platform Based on Serverless Architecture

Supervisor: Abdolbaghi Rezazadeh

Second Examiner: BooJoong Kang

Submitted in accordance with the requirements for

the degree of MSc Cyber Security

2023

Table of Contents

Statement of Originality

- I have read and understood the ECS Academic Integrity information and the University's Academic Integrity Guidance for Students.
- I am aware that failure to act in accordance with the Regulations Governing Academic Integrity may lead to the imposition of penalties which, for the most serious cases, may include termination of programme.
- I consent to the University copying and distributing any or all of my work in any form and using third parties (who may be based outside the EU/EEA) to verify whether my work contains plagiarised material, and for quality assurance purposes.

You must change the statements in the boxes if you do not agree with them.

We expect you to acknowledge all sources of information (e.g. ideas, algorithms, data) using citations. You must also put quotation marks around any sections of text that you have copied without paraphrasing. If any figures or tables have been taken or modified from another source, you must explain this in the caption and cite the original source.

I have acknowledged all sources, and identified any content taken from elsewhere.

If you have used any code (e.g. open-source code), reference designs, or similar resources that have been produced by anyone else, you must list them in the box below. In the report, you must explain what was used and how it relates to the work you have done.

I have not used any resources produced by anyone else.

You can consult with module teaching staff/demonstrators, but you should not show anyone else your work (this includes uploading your work to publicly-accessible repositories e.g. GitHub, unless expressly permitted by the module leader), or help them to do theirs. For individual assignments, we expect you to work on your own. For group assignments, we expect that you work only with your allocated group. You must get permission in writing from the module teaching staff before you seek outside assistance, e.g. a proofreading service, and declare it here.

I did all the work myself, or with my allocated group, and have not helped anyone else.

We expect that you have not fabricated, modified or distorted any data, evidence, references, experimental results, or other material used or presented in the report. You must clearly describe your experiments and how the results were obtained, and include all data, source code and/or designs (either in the report, or submitted as a separate file) so that your results could be reproduced.

The material in the report is genuine, and I have included all my data/code/designs.

We expect that you have not previously submitted any part of this work for another assessment. You must get permission in writing from the module teaching staff before re-using any of your previously submitted work for this assessment.

I have not submitted any part of this work for another assessment.

If your work involved research/studies (including surveys) on human participants, their cells or data, or animals, you must have been granted ethical approval before the work was carried out, and any experiments must have followed these requirements. You must give details of this in the report, and list the ethical approval reference number(s) in the box below.

My work did not involve human participants, their cells or data, or animals.

Abstract

Based on features of online AI experiments, for example, complex experiment environment, heavy demand for data, and high concurrency of experiment users, this essay design and implement an online AI experiment platform based on Serverless architecture. It uses Docker to provide a consistent and isolated experiment environment for every user. The Docker image the platform use integrates all libraries of Tensorflow and Jupyter Notebook which can provide users with massive basic AI algorithms and public data sets. To easily and efficiently monitor and manage containers, Kubernetes, an open-source application for managing containers on multiple hosts in the cloud platform, is applied in our Serverless architecture. It achieves the dynamic expansion and contraction of experiment containers and effectively improves the multi-user concurrency performance of the platform. In addition, the platform provides 2 machine-learning experiments to validate its useability. The test results show that the platform can stably provide AI experiment services for multiple users.

Chapter 1
Introduction

1.1 Problem

At the end of 2019, the COVID-19 pandemic broke out globally, impacting the traditional education model worldwide. In recent years, online theoretical teaching has been carried out successfully, but engineering experimental teaching still faces significant challenges. Taking the example of artificial intelligence laboratory courses, they require high-performance software and hardware, complex experimental environment configurations, large amounts of data for experimentation, and high algorithm implementation difficulty. These factors limit the development of online artificial intelligence experiments.

Besides this, In the past, if universities wanted to purchase cloud services, they usually had to rent servers. However, since servers were not used 24 hours a day, such as during the night, it would lead to higher rental costs, energy wastage, and maintenance expenses. With the development of cloud computing, a new cloud computing model, Serverless computing [1], also known as serverless architecture or Function as a Service (FaaS), where the cloud provider manages the infrastructure for running and scaling applications, is widely applied in cloud services. In a serverless environment, developers can focus solely on writing code for individual functions or microservices without having to worry about managing servers, virtual machines, or the underlying infrastructure.

1.2 Goals

Focusing on the aforementioned practical needs, this paper combines Serverless computing and online AI experiment education to design and implement an artificial intelligence experiment platform based on the Serverless architecture. The platform aims to offer sustainable, stable, and effective cloud services to those universities that want to conduct online AI experiments education. To validate the platform's functionality and usability, it provides a default machine learning experiment. The platform is supposed to be a new solution for the online implementation of various engineering experimental teaching in the future.

1.3 In-Scope

The online AI experiment platform will focus on providing basic online AI experiment services to university educators and students in terms of creating and deleting experiments, conducting experiments, personal information management, and platform resource management subsystems. The following will be included:

- Machine Learning experiment using open-resources dataset
- Easy to use, responsive, and straightforward UI.
- Provided platform accounts
- A web-based application that provides available functionalities
- Docker Image which Integrates everything necessary for AI experiments
- Serverless architecture which provides microservices via API
- Database and file system

1.4 Out-of-Scope

The following will not be included:

- The timing system. Students must complete the experiment within a specified time limit. If they exceed the time limit, it will be considered a failure. Alternatively, they can pause and save the current state to continue next time.
- New account registration.

Chapter 2

Literature Review

This section focuses on the analysis of the current global AI education status and similar products. It also has a detailed explanation of all tools and environments planned to be used during the development phase of the platform.

2.1 Background Research

Due to the rapid and widespread COVID-19, schools have to set up web-based courses to keep social distance. Undoubtedly this promotes the development of online education. Such kind a web-based education can maximize the use of learning resources. It breaks the limitations of time and space through the Internet. While making full use of resources, it also promotes academic communication among schools in the whole world. Under the background of globalization, this concept undoubtedly meets the needs of social development. At the same time, a web-based platform has the functions of automatic management and remote interaction which greatly facilitates teachers' management of teaching such an idea of combining the Internet and education provides a new method for traditional education.

Serverless technology is a kind of architecture and service model developed with the revolution of cloud service. It is considered to be the most potential development direction of cloud computing, and it must be the trend of cloud computing in the future.

Serverless products are fully automated elastic expansion and contraction. At the peak of business, in this paper, when a student user is conducting online AI experiments, the computing power and capacity of the product will be automatically expanded to carry more user requests. When the experiment is finished, the resources used will shrink to avoid resource waste.

2.2 Current Global AI Experiment Education Status

At present, artificial intelligence technology has become the next core of computer science development. The rise of deep learning and big data and the resulting personalized learning have contributed to another outbreak of the application of artificial intelligence in the field of education. On the other hand, only a few worldwide governments attach great importance to the development of artificial intelligence.

According to the current AI education status report published by UNESCO in 2022 [2], only 11 countries and 1 region have developed and implemented 14 artificial intelligence courses. The following sections will take MOOC [4] and courses at Stanford University as examples to introduce the popular products and courses of AI education and propose probable future directions.

2.2.1 AI Education Products

In recent years, artificial intelligence has become a significant part of education. In some higher education online platforms such as MOOCs, students can carry out personalized learning with the help of big data and artificial intelligence, improve learning efficiency, and promote the relative balance of high-quality resources in basic education. The continuous development of artificial intelligence teaching and experiment platforms plays an important role in promoting artificial intelligence education. As an artificial intelligence teaching management system integrating the functions of students, teachers, and administrators, it can support multiple students to attend classes at the same time and backend resources management of teachers and administrators and support teachers to participate in the personalized customization of artificial intelligence experiment courses and projects.

Therefore, designing and implementing an online education platform is the most popular and effective way to provide AI education.

2.2.2 AI Courses at Stanford University

Take the AI course CS221 [5] at Stanford University as an example. This course summarizes specialized artificial intelligence technologies such as web search, speech recognition, machine translation, and autonomous driving as complex problems in the real world. It defines artificial intelligence as the utilization of rigorous mathematical tools to solve these complex real-world issues. Centered around the three categories of machine learning, natural language processing, and computer vision, the course is designed to teach students methods for addressing artificial intelligence problems they may encounter in real-life situations.

In 2018, Stanford University offered the course "CS231N: Convolutional Neural Networks for Visual Recognition" [6]. The students were required to complete assignments using Jupyter Notebook [7]. Jupyter Notebook is an excellent data analysis software that facilitates interactive programming, making it very suitable for beginners in the field of artificial intelligence. This platform also integrated Jupyter

Notebook into the experimental environment for the convenience of both students and teachers.

2.3 Conclusion of Current AI Education Status

To sum up, due to the relatively developed computer technology, universities in European and American countries have carried out different forms of AI courses for a long time, but there is still no unified standard on how to reform the existing education methods for artificial intelligence. In 2020, mankind suffered the outbreak of COVID-19, which prompted the increasing demand for online education. It is foreseeable that in the future, AI online experiment courses will be widely applied. The AI online experiment platform designed in this paper effectively solves the problems that AI online education faces. It not only meets the needs of education but also provides a method for online experiments of other engineering courses. This paper will refer to Stanford's AI course on integrating Jupyter Notebook into the platform and will design a machine learning experiment to verify the platform's usability.

2.4 Tools and Environment

This subsection focuses on the analysis of all environments and techniques that will be used to develop the platform.

2.4.1 Programming Language: Python

Python [8] is a versatile and widely used programming language known for its simplicity and readability. Created by Guido van Rossum and first released in 1991, Python has since gained immense popularity due to its ease of use and strong community support. The language's extensive standard library and a vast ecosystem of third-party packages (through the package manager called "pip") make it suitable for a wide range of applications, such as web development, data analysis, scientific computing, artificial intelligence, automation, and more. In this paper, it is used to build the back-end part of the platform to connect to Serverless services via API. Besides this, it is also used to design Machine Learning experiments.

2.4.2 Development Environment

PyCharm [9] will be used as the development environment. PyCharm is an integrated development environment (IDE) specifically designed for Python development. It was

developed by JetBrains, a company known for creating powerful IDEs for various programming languages. PyCharm provides a comprehensive set of tools and features to assist developers in writing, debugging, and maintaining Python code efficiently. It also supports writing HTML + CSS + JS language to build web pages which helps the developer to develop web-based applications.

2.4.3 Storage: SQLite and NFS (Network File System)

SQLite [10] is a lightweight and self-contained relational database management system (RDBMS) that is widely used in various applications and platforms. Unlike traditional client-server databases, SQLite is serverless, which means it operates as an embedded database within the application itself, without requiring a separate database server to function. SQLite database is used to store user information and experiment information. Network File System (NFS) [11] is a distributed file system protocol that allows a client system to access files and directories over a network as if they were part of the client's local file system. NFS enables file sharing and centralizes file storage across multiple machines connected to a network, creating a unified file system view for all clients. NFS is used to share and mount files and directories among Kubernetes clusters.

2.4.4 Docker

Docker [12] is a platform and toolset that enables developers to build, distribute, and run applications inside lightweight, portable containers. It uses containerization technology to package an application along with its dependencies and runtime environment into a single, isolated unit known as a Docker container. These containers can be deployed consistently across different environments, ensuring that the application behaves the same way everywhere. In this paper, Docker is responsible for providing an isolated and specific experiment environment.

2.4.5 Kubernetes

Kubernetes (often abbreviated as K8s) [13] is an open-source container orchestration platform developed by Google. It automates the deployment, scaling, and management of containerized applications. Kubernetes is designed to handle the complexities of running applications in a distributed and dynamic environment, making it easier to manage containerized workloads at scale. Besides this, the system will also

build the platform management subsystem using Kubernetes-Dashboard.

2.4.6 Server Environment

The server and Serverless services will be implemented in Ubuntu [14] and VMware Workstation [15]. Ubuntu is a popular open-source Linux-based operating system developed by Canonical Ltd. It is one of the most widely used Linux distributions and is known for its user-friendly interface, ease of installation, and strong community support. VMware Workstation is a desktop virtualization application that allows users to create and run multiple virtual machines on their local computers

2.4.7 Test Tool: Apache Bench

Apache Bench [16] is a command-line tool and a part of the Apache HTTP Server project. It is a simple yet powerful utility used for benchmarking web servers and testing the performance of HTTP servers by sending a large number of requests to a web server and measuring its response times.

2.4.8 Experiment Environment

The experiment will be designed in Python and run in Jupyter Notebook. As mentioned in Chapter 2.2.2, Jupyter Notebook is an open-source web application that allows users to create and share interactive documents that combine live code, visualizations, and explanatory text. It also provides an interactive environment where code can be written and executed in cells. Each cell can contain code, Markdown text, equations, or other media so that educators can write guides in cells.

2.4.9 Version Control: GitHub

GitHub [17] is a web-based hosting service for version control using Git, a distributed version control system. It provides a platform for developers and teams to collaborate on software projects, track changes, manage code revisions, and share their work with others. GitHub is widely used by developers and open-source communities to host and contribute to a vast array of projects.

Chapter 3
System Analysis

This section focuses on the system analysis of the application which involves the identification of Stakeholders, User Stories, and Risk Analysis.

3.1 Stakeholders

The platform is primarily intended for universities that conduct AI education. Therefore, the primary stakeholders are educators and student users. Besides this, the university administrator is responsible for building a business relationship with the platform and does not directly interact with the system but has a reasonable influence on it. To keep long-term operation and maintenance, it is necessary to have a platform administrator to manage the platform resources. Therefore, there are 5 stakeholders in the system which is shown in table 3-1.

Stakeholder	Type	Role / Description
Educator	Primary	Educators are responsible for creating and uploading experiments to the platform by logging in with educator accounts.
Student	Primary	Student users conduct the experiment uploaded by educators by logging in with student accounts
Platform Administrator	Primary	PAs are responsible for checking system status and managing platform resources
University Administrator	Secondary	University administrator does not directly interact with the system, but provide a list of educators' and students' information.
As a developer	Facilitating	I will develop and maintain the system.

Table 3-1: Stakeholders in the System

3.2 User Stories

The tables below describe User Stories, a concise and informal description of a feature or functionality from the perspective of an end user [18], or stakeholders mentioned in section 3.1. The template of a user story used here is: **As a <role>, I want to <capability> so that <receive benefit>.**

ID	Role	Goal		Benefit
US01	**As an Educator, I want to**	be able to view the list of my students	**So that,**	I can manage my class
US02		be able to manage experiments (view, add, delete)		I can control and achieve expected education aims
US03		securely login to my account and logout		information can only be accessed by me
US04		manage my personal information (profile, portrait)		I can identify my information
US05		be able to check the results uploaded by students		I can check whether students have a good understanding of it
US06		be able to change my password		I can keep my account safe
US07		be able to check comments from students		I can get feedback from student

Table 3-2-1: User Stories of Educator

ID	Role	Goal		Benefit
US08	**As a Student, I want to**	be able to view the list of available experiments	**So that,**	I can choose the experiment I want to do
US09		be able to securely log into the system and logout		my information can only be accessed by me
US10		be able to conduct experiments on the platform (view instructions, do experiments, upload results)		I can finish my AI courses
US11		be able to change my password		I can keep my account safe
US12		be able to comment on the experiment		I can give personal feedback about it

Table 3-2-2: User Stories of Student

ID	Role	Goal		Benefit
US13	**As a Platform Administrator, I want to**	be able to supervise Kubernetes cluster status	**So that,**	I can check if Serverless services are running correctly
US14		be able to manage Educator and student accounts (view, add, delete)		I can check the current Educators' and students' information
US15		be able to securely log into the system and logout		platform resources can only be accessed by me
US16	**As a University Administrator, I want to**	Provide a table of course information, student information, and educator information	**So that,**	I can offer platform accounts to educators and students

Tabel 3-2-2: User Stories of PA and UA

3.3 Risk Analysis

The possible things and potential risks that might happen during the development phase of the application will be described below:

Risk	P	R	RE	Mitigation
Change project idea	1	4	4	I will follow the current idea and direction
Illness during the development	3	2	6	If I get unexpectedly ill, I will reschedule the plan and development to finish the project on time
Underestimated the FR size	2	2	4	Resize the requirements and deal with the requirements with higher priority first
Insufficient development time	2	2	4	I can work harder to make sure the submission date will not delay
Loss of data	2	4	8	I am currently using GitHub to back up all the data.

Table 3-3: Risk Management P = Probability (1-to-5), S = Severity (1-to-5), RE = Risk

Exposure (E = P x S, 1-to-25)

3.4 Functional and Non-Functional Requirements

Research methodologies are techniques used to identify, gather, and analyze the needs and preferences of users or stakeholders for a particular product, service, or system. These methods help me as a developer understand what users expect and desire. Some common research methods include Interviews, Surveys Questionnaires, and Case studies.

The requirements of the system are gathered by doing a case study on existing AI experiment courses at Stanford University and products and interviewing university educators and students.

The requirements are classified into functional requirements and non-functional requirements. Functional Requirements (FR) [19] define the specific functionalities or features that a software system must possess to meet the needs of its users. Non-Functional Requirements (NFR) [20] on the other hand, describe how the system should behave in terms of performance, usability, security, scalability, reliability, and other aspects that are essential for the system's effectiveness and user experience.

ID	Prio.	Related USs	Description
FR01	1	US03, US09, US15	The system must provide secure login and logout for all users
FR02	1	US16	The system must not allow users to register
FR03	1	US06, US11	The system must allow users to change passwords
FR04	2	US04	The system should allow users to manage personal information
FR05	1	US01	The system must show a list of current students
FR06	1	US02, US08	The system must show a list of current experiments
FR07	1	US02	The system must allow educators to manage new experiments (view, add, delete)
FR08	1	US05	The system must allow educators to check the results uploaded by students
FR09	2	US07	The system should allow educators to check comments on experiments uploaded by students

Table 3-4-1: Functional Requirements Table 1

ID	Prio.	Related USs	Description
FR10	1	US10, US08	The system must allow the student to select the experiment he/she wants to do
FR11	1	US10	The system must have experiment guidance and an introduction
FR12	1	US10	The system must allow students to upload the results of the experiment
FR13	2	US12	The system should allow students to comment on the experiment
FR14	1	US14	The system must allow the platform administrator to manage Educators' and students' accounts
FR15	1	US16	The system must allow the platform administrator to check the Kubernetes cluster status

Table 3-4-2: Functional Requirements Table 2

ID	Prio.	Description
NFR01	1	The system must react to user interactions within 500ms.
NFR02	1	The system must encrypt the passwords before storing them in the database.
NFR03	1	The system must ensure the data is protected from unauthorized access.
NFR04	1	The system must avoid overprivileged functions
NFR05	2	The system should be easy to understand so that users can use the system without any training.
NFR06	2	The system should provide good scalability.
NFR07	1	The system must have a persistent storage system

Table 3-4-3: Non-Functional Requirements

3.5 Use Case Diagram

This section displays three use case diagrams to demonstrate how primary stakeholders interact with the system. The system is supposed to provide secure login and logout, and proper personal information for all of the users. The use case diagram of educators is shown in Figure 3-5-1. They can manage experiments (view, add, delete) and view the list of students in the class. Figure 3-5-2 shows the use case diagram of student users. Their core requirement is to conduct AI experiments on the platform including viewing instructions, doing experiments, and uploading results. Besides this, they can comment on the experiment to give feedback to educators. The Platform Administrator, who is responsible for managing platform users (view, add, delete) and monitoring the status of the Kubernetes cluster, will be described in Figure 3-5-3.

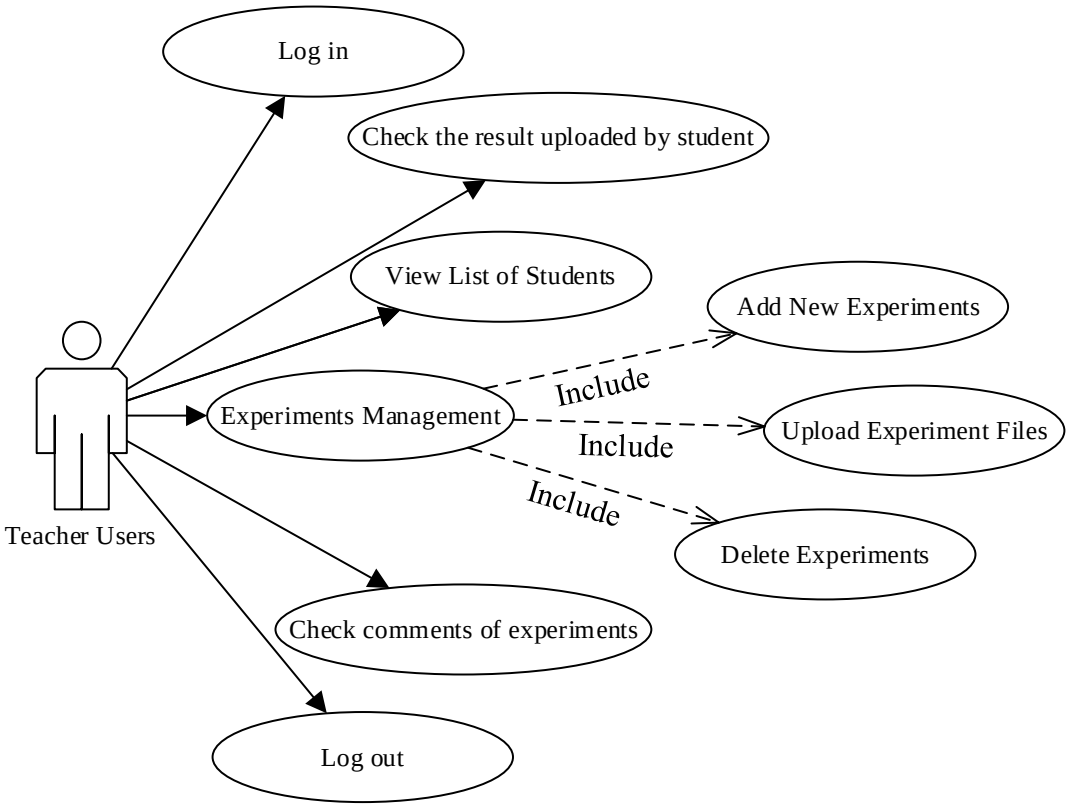

Figure 3-5-1: Educator User Use Case Diagram

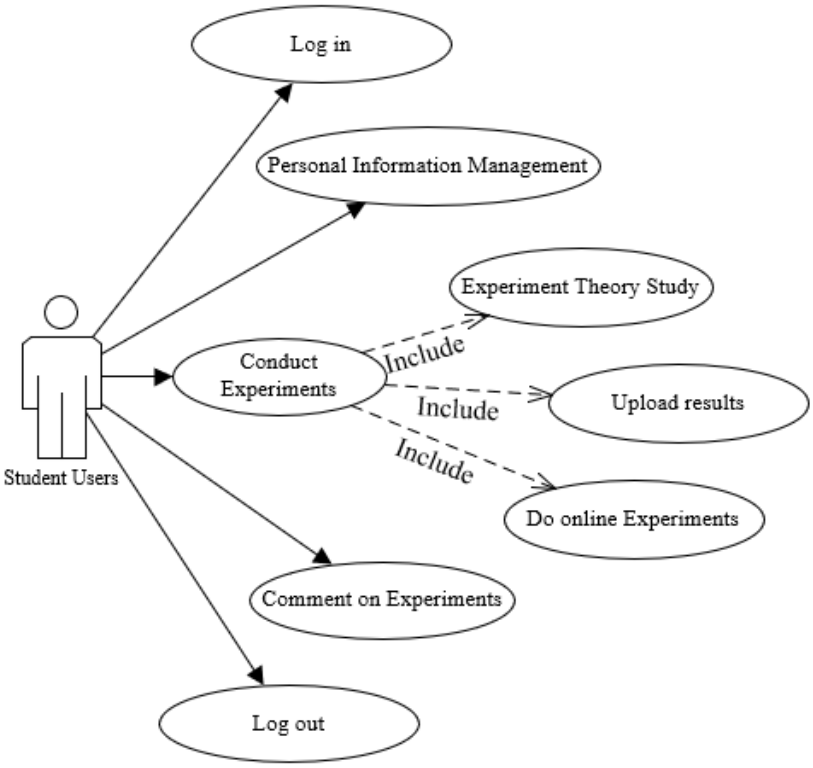

Figure 3-5-2: Student User Use Case Diagram

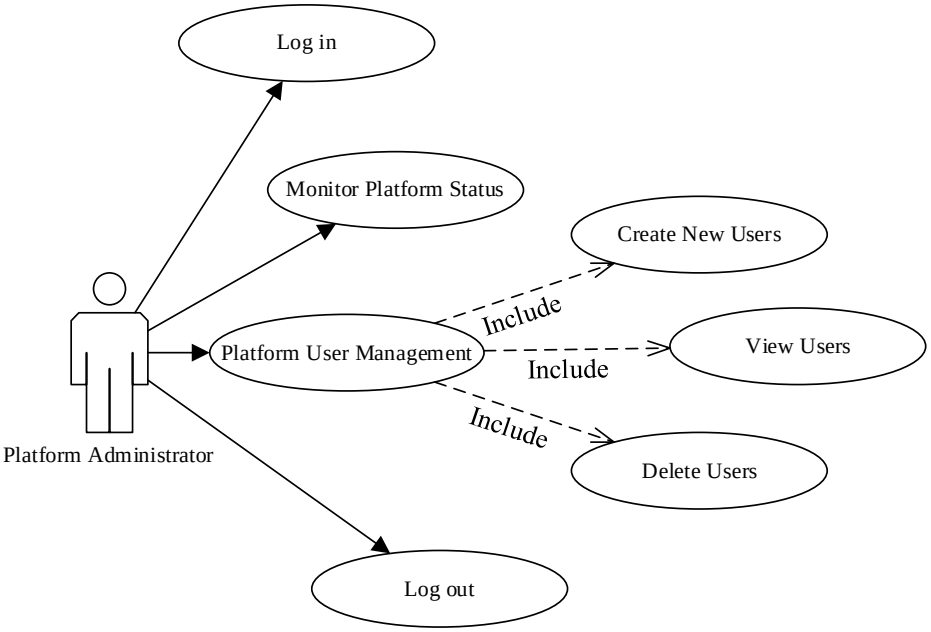

Figure 3-5-3: Platform Administrator Use Case Diagram

Chapter 4

Project Management

Project management is an important part of software engineering. Given the project constraints, it helps determine how the project is completed. The chapter includes the related constraints of the project, the software development model, and the project schedule

4.1 Project Constraints

The major related constraints that influence the development process of the project include time limits, knowledge gaps, and difficulties in gathering user requirements.

4.1.1 Time Limit

The expected time limit of this project is two and a half months. However, due to the late start of the project and heavy workload, the time spent on each phase should be seriously considered and the plan should be carried out strictly. A Gantt chart is provided in section 4.3 to make sure the project is well-structured.

4.1.2 Knowledge Gap

Developing such a well-structured web-based and cloud-based application requires good knowledge of cloud computing and software development. As the developer, I am supposed to research and learn advanced tools and technologies to provide integrated and stable services and meet user requirements. In addition, because of the frequent and various updates of tools and technologies, it is costly to configure the system and fix potential errors and bugs.

4.1.3 Difficulties in Gathering User Requirements

As the system aims to provide a new solution to solving the challenging current situation of online AI experiment education, requirements are mainly gathered by interviewing relative educators and students and studying existing AI courses and products. On the other hand, due to time limit constraints, it is challenging to produce a perfect system that meets all expected requirements. In addition, the lack of professional knowledge as a student also leads to Incomplete functionalities.

4.2 Software Development Model

The Agile model [21] is a software development approach that emphasizes iterative and incremental development, collaboration, flexibility, and customer feedback. It has gained significant popularity in the software development industry due to its ability to foster adaptability, customer-centricity, and efficient project delivery. Agile development offers numerous benefits to me. It places a strong importance on getting feedback and collaborations from customers and stakeholders, so that I, as a developer, can better understand their needs, and preferences, and get feedback from my supervisor. The Agile framework applied in this paper is Scrum.

Scrum [22] is one of the most popular Agile frameworks used in software development. The core concept is to iteratively finish the work in short time frames which is also known as Sprints [23].

To apply the Scrum framework, the first step is to create a Product Backlog [24] for the project. The product backlog is a prioritized and dynamic list of all the work items, features, user stories, and bug fixes required to complete a project or develop a product. The Product Backlog can be found in System Analysis.

4.2.1 Sprint Planning

When the product backlog is finished, developers normally start planning the sprint. Sprints can help us pay more attention to individual requirements rather than the whole project. In this paper, we split the development into 4 sprints according to functional requirements and non-functional requirements. Sprint 1 is to achieve the requirements of educators. After that, in sprint 2, the system is supposed to provide complete functionalities for student users. Platform Administrators can take part in the system when sprint 3 is finished. At the last, our team will meet all non-functional requirements to guarantee the integrity and usability of the system.

4.3 Gantt Chart

A Gantt chart is represented in this section (Figure 4-3) to demonstrate the planning and schedule of this project. It is week-based and describes each stage of the project including software development and essay writing. The allocation of each task may change and therefore, the Gantt Chart will be updated accordingly

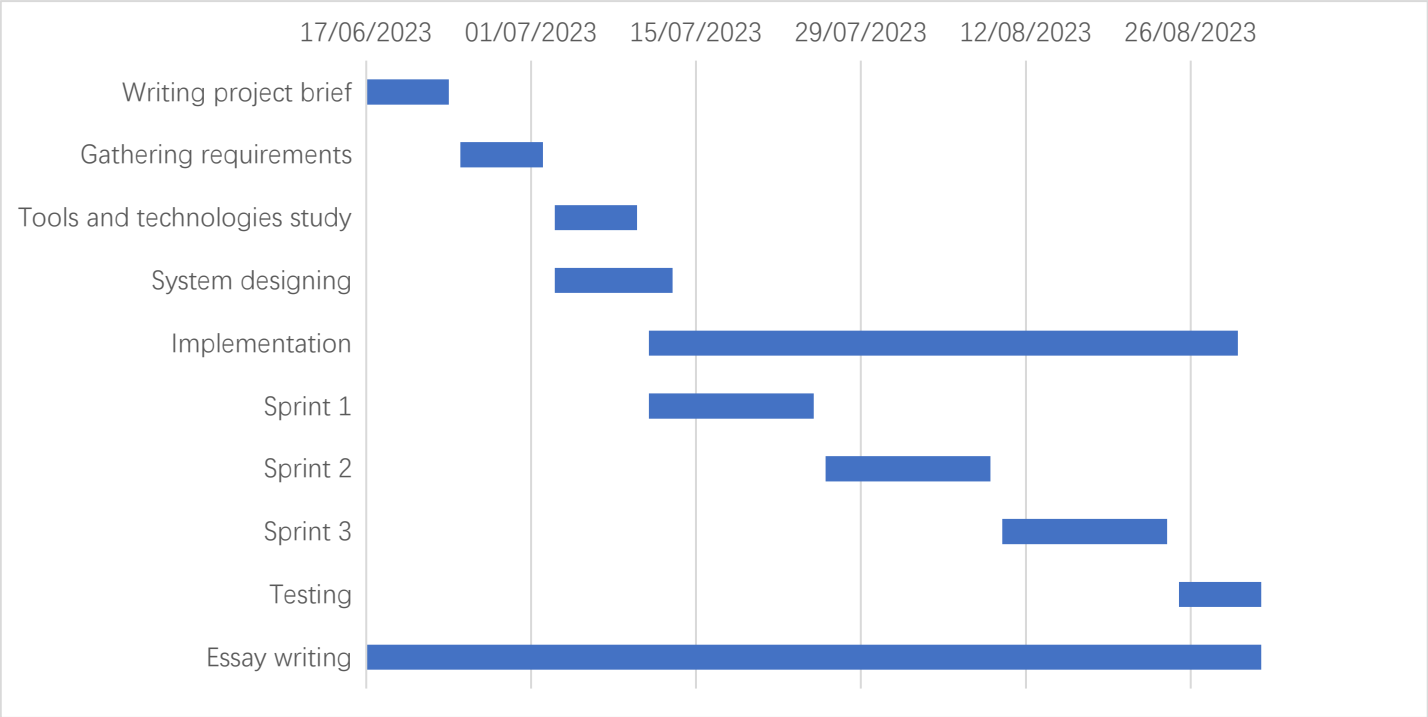

Figure 4-3: Gantt Chart of Project

Chapter 5
System Design

The chapter provides an overview of the system structure, UI design, Serverless services design, database design, and tests that are going to be implemented to meet user requirements.

5.1 System Structure

The system is an AI online experimental platform based on a Serverless framework, which adopts a three-tier structure. In this paper, the system needs to consider the requirements of scalability and flexible deployment and the features of Function as a Service and Backend, provide strictly controlled access to resources, standard authentication, and management interface, and microservices via APIs that between the platform and Kubernetes Cluster. The system structure is shown in Figure 5-1.

1) Client Tier

The client provides Web pages to users and communicates with the server through the HTTP protocol. The client should have 3 subsystems (Educator subsystem, Student subsystem, and Platform Administrator subsystem) to meet various functional requirements from stakeholders. In addition, the user interface is supposed to be clear, simple, beautiful, and easy to use.

2) Server Tier

The server is based on the Flask [25] framework. It is designed to provide APIs for the client to transfer data and send requests to the Kubernetes cluster and storage. There should be strict authentication and various routes for different users to achieve access control to the system services and resources.

3) Storage Tier

This system applies SQLite database to store basic resource information of the platform and the Network File System is used to store experiment files, configuration files and share other files between nodes.

4) Kubernetes Cluster

It aims to offer AI experiment services by container building, workload management, and automatic event triggering.

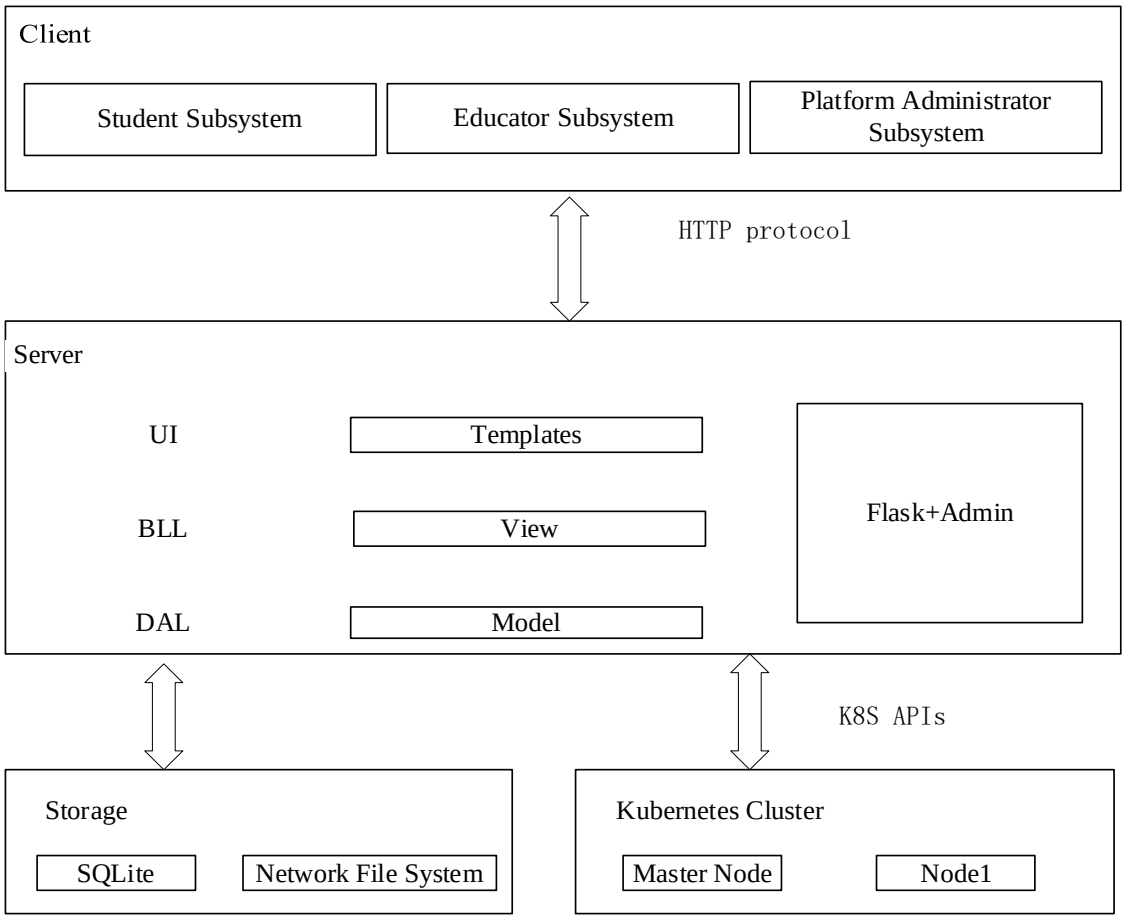

Figure 5-1: System Structure

5.2 User Interface

This subsection will describe the system's user interface. Figure 5-2-1 illustrates the home (login) page of the platform. Users are requested to log in with their username and password. As mentioned in Chapter 3, the system does not provide a register function therefore there is no register button.

When an educator/student/PA successfully logs in, the home page jumps to the educator/student/PA subsystem. The student subsystem is shown in Figure 5-2-2. In this subsystem, students can view the experiment list and start or comment on the chosen experiment by clicking the corresponding button. On the top of the page, there is a navigation bar that allows student to change their password and log out. The basic structure of educator subsystem (is the same as the student subsystem: a navigation bar, a list of functionalities on the left, and the main body (Figure 5-2-3). The difference is that the functionality is displayed on the left side and the main body.

In the PA subsystem (Figure 5-2-4), PA can check the list of current platform users and enter the Kubernetes cluster management system which is built with Kubernetes Dashboard [26]. Kubernetes Dashboard is a web-based user interface that provides a graphical representation of various aspects of a Kubernetes cluster. It allows users to manage and monitor their Kubernetes resources easily, providing an intuitive interface for performing common tasks and gaining insights into the cluster's health and performance.

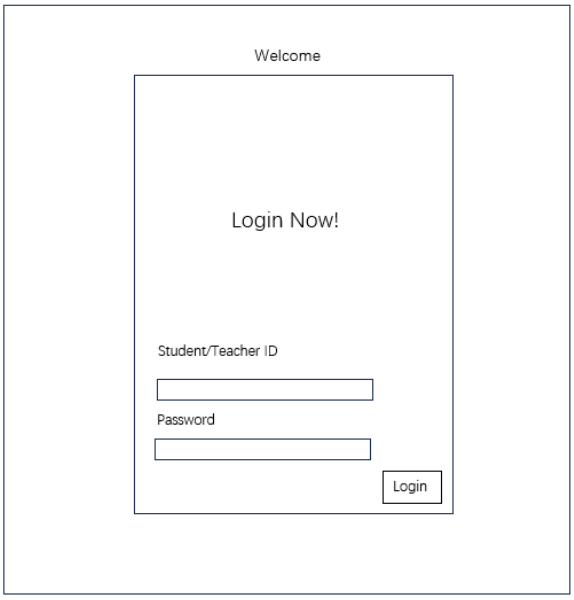

Figure 5-2-1: Home Page

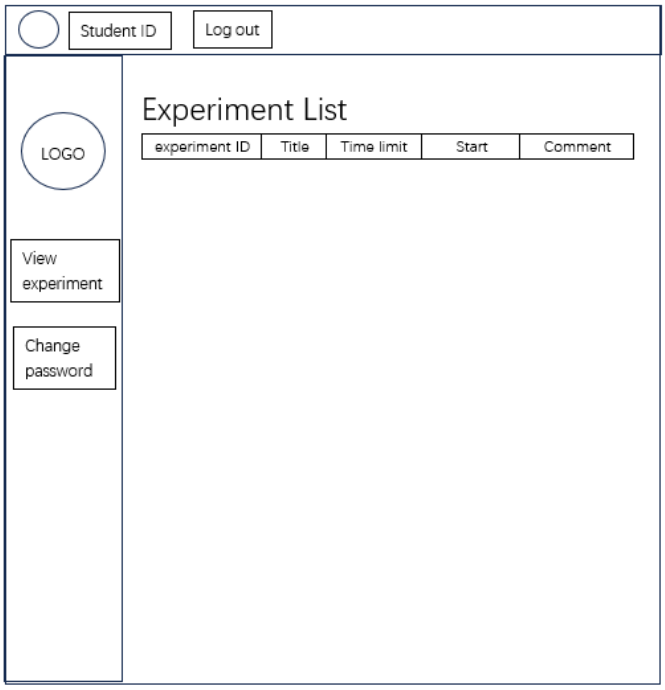

Figure 5-2-2: Student Subsystem

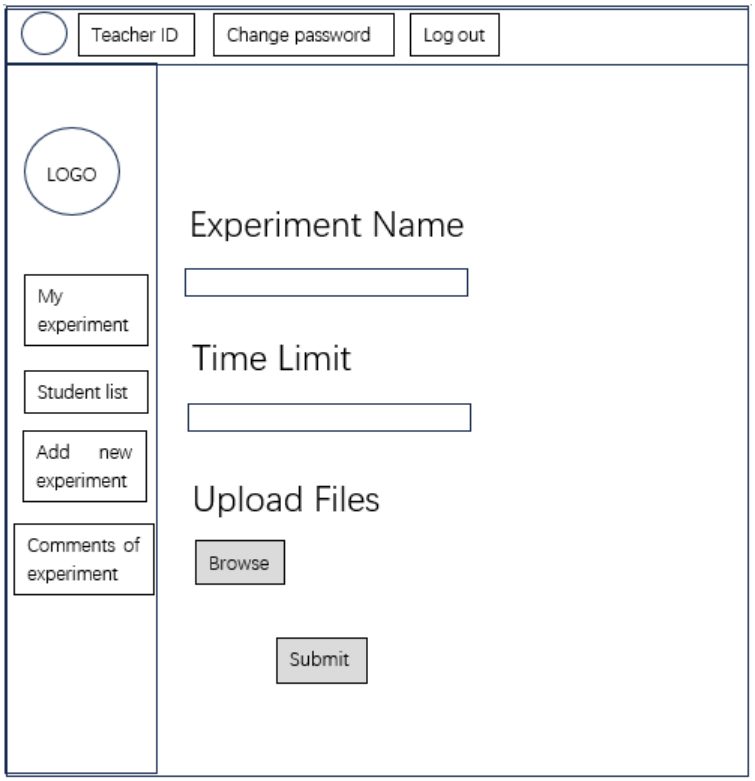

Figure 5-2-3: Educator Subsystem “Add new experiment” page

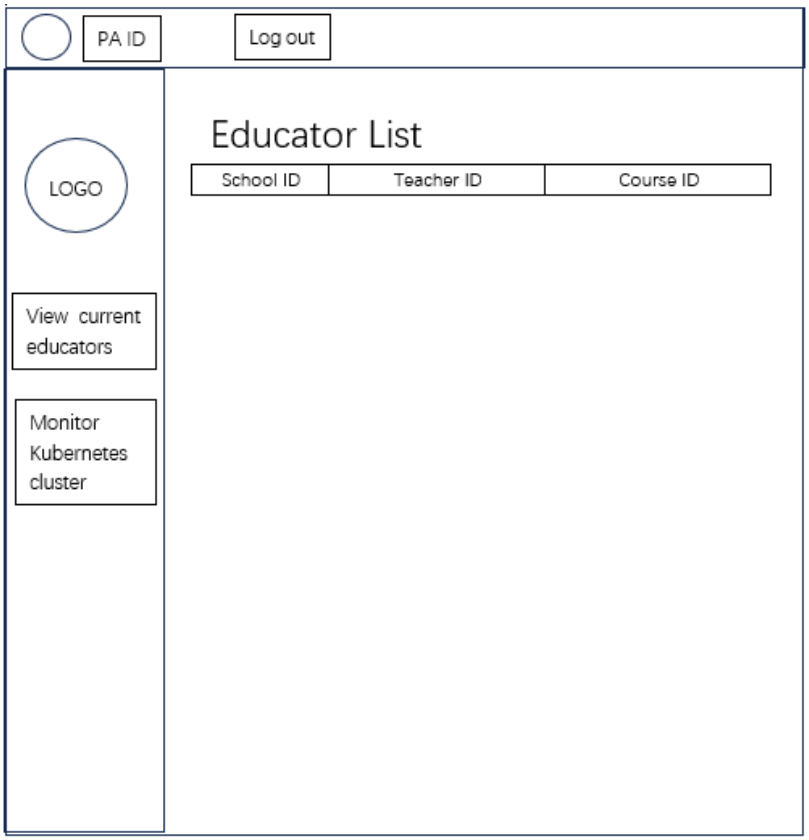

Figure 5-2-4: Platform Administrator Subsystem

5.3 Serverless Framework

5.3.1 Serverless Structure

This paper researches two Serverless architecture models that account for the highest proportion in the market: AWS Lambda and Microsoft Azure Function. In “Building Serverless Applications with Python” [27], AWS Lambda proposed a new concept: a function is a container. A function is a single code file or a code file compression package (contains the core code functions and the resources that functions will use). AWS Lambda users upload functions to the AWS platform, which implements Serverless by dispatching each user’s functions running in a specific container. Based on the above research, this paper designs a Serverless architecture by integrating container building, workload management (Dynamic expansion and contraction), and automatic event triggering. The Serverless architecture design is shown in Figure 5-3-1.

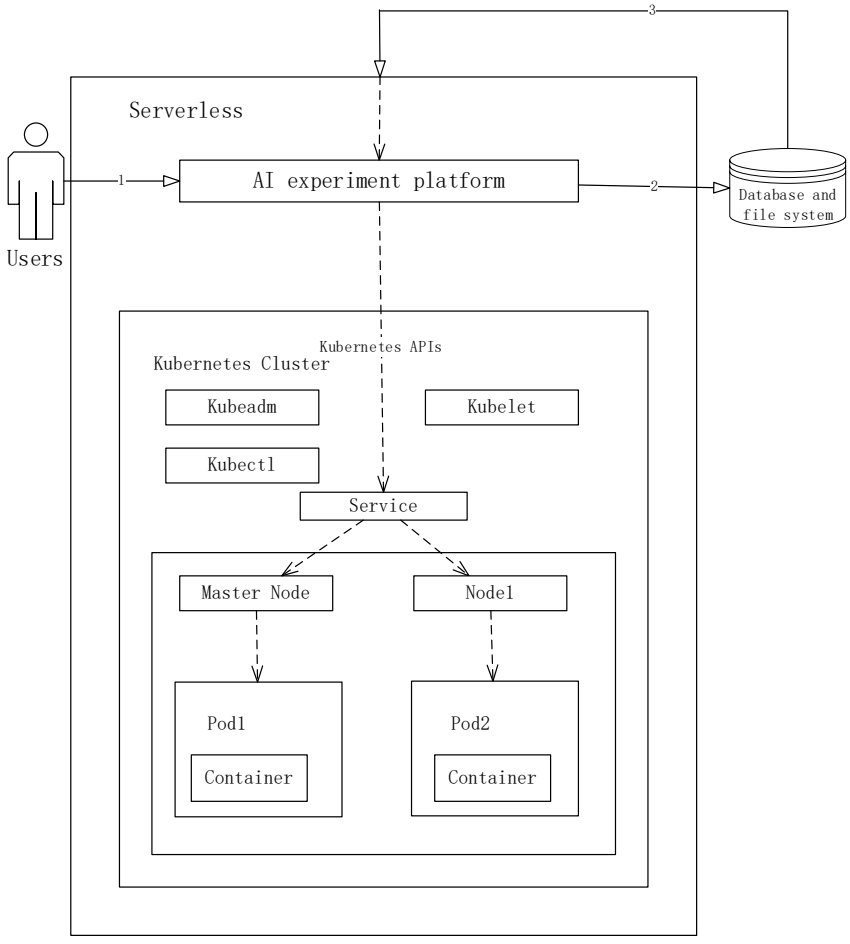

Figure 5-3-1: Serverless Structure

1) **Containers Building**: This platform uses Docker to provide containers for users. It could create a consistent and isolated AI experiment environment for users. The docker image of this platform should integrate a large number of basic artificial intelligence algorithms, which can meet the needs of most AI experiments. In addition, platform administrators need to regularly maintain Docker images and integrate the latest algorithm library into the image for users to use

2) **Workload Management (Dynamic expansion and contraction)**: Kubernetes is implemented to manage containers and achieve dynamic expansion and contraction of them. When users request to do experiments, the platform communicates with the Kubernetes cluster through Kubernetes API, then automatically starts containers in Pods of Nodes. After finishing experiments, Kubernetes will close containers to achieve automatic contraction.

3) **Automatic Event Triggering**: Users start containers from the Kubernetes cluster by clicking events on the platform. Under this architecture, teacher users upload experiment code files to the platform in advance, then student users click events to trigger the platform to automatically run "functions" in specific containers.

5.3.2 Kubernetes Cluster

A Kubernetes cluster [28] is a collection of nodes that work together to run containerized applications and manage them using the Kubernetes orchestration system. The cluster consists of two main components:

1) **Master Node**: The master node is responsible for managing and coordinating the entire cluster. It runs several essential components, including the Kubernetes control plane components like the API server, scheduler, controller manager, etcd (a distributed key-value store that stores the cluster's configuration data). The API server acts as the central communication hub for all cluster operations.

2) **Worker Nodes**: Worker nodes are the machines where the containerized applications are deployed and run. Each worker node runs a container runtime (like Docker) to manage containers. These nodes communicate with the master node to receive instructions and updates on the desired state of the cluster.

In this paper, there is going to be **1 master node** and **1 worker node** in the cluster. All the resources (experiment files, database) are only stored in the Master node when applications (e.g., containers) provided to users will only run in the worker node. These

nodes will be implemented and initiated with kubeadm, kubectl, and kubelet. And they will also be given static IP addresses.

5.3.3 Kubeadm, Kubectl, Kubelet

Kubeadm, kubectl, and kubelet [29] are three essential command-line tools used in Kubernetes for different purposes:

1) **Kubeadm**: Kubeadm is a command-line tool used for setting up a Kubernetes cluster. It simplifies the process of creating a cluster by handling the necessary steps to initialize the control plane, join worker nodes, and manage the cluster's configuration.

2) **Kubectl**: Kubectl is the command-line interface (CLI) tool used to interact with the Kubernetes cluster. It serves as the primary tool for deploying and managing applications, inspecting the cluster's resources, and managing the cluster itself. Kubectl allows developers to create, modify, and delete resources using YAML by running specific commands.

3 **Kubelet**: Kubelet is responsible for maintaining the communication between the control plane and the node, and it ensures that containers are running as expected on the node. The kubelet monitors the state of the containers and their corresponding Pod specifications It also communicates with the container runtime to manage the containers' lifecycle and resource usage on the node.

5.3.4 Flannel

Flannel [30] is a network solution that provides a virtual network overlay spanning the entire Kubernetes cluster for containers. When containers are created on different nodes, Flannel assigns them unique virtual IP addresses, enabling transparent communication across nodes. This allows containers to communicate as if they were on the same local network, regardless of their actual location on different nodes.

With Flannel, we can easily deploy and manage applications across the entire Kubernetes cluster, ensuring communication between containers. The communication process is illustrated in Figure 5-3-4.

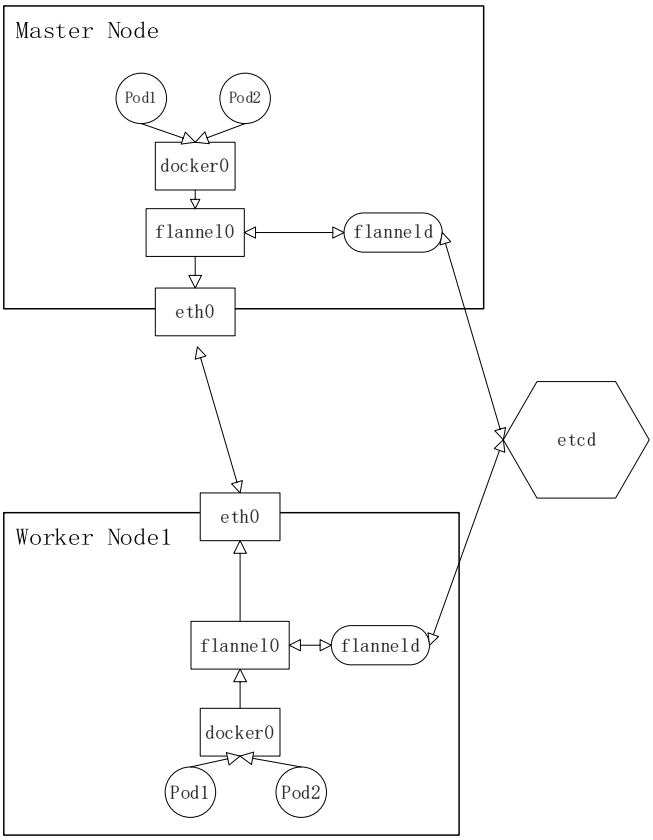

Figure 5-3-4: Flannel Communication Process

5.4 Database and Network File System

5.4.1 Database

This paper applies SQLite database to store basic resources' information such as users' information, experiments' information, etc. It is noticeable that the platform has no register function. Educator and student accounts are created by platform administrators. In the platform, one course is taught only by one teacher in one school in a university. The course code might be different in different schools. Experiments uploaded by an educator are only available to students of that educator. To get educator accounts, the university has to offer an application with teachers', courses, and students' information. After platform administrators check the application, users' accounts will be distributed to universities. To distinguish teachers' and students' account IDs from various schools, teacher users' account IDs will be in the form of "University name + numbers" e.g., UoS001. Default passwords are teachers' telephone numbers. The student user's account ID will be "University name + student

ID in university" e.g., UoS34180761. Default passwords are the student's student ID. The database diagram is shown in Figure 5-4-1.

There are 8 tables in the database:

1) Student: Store student users' information

2) platform_Manager: store platform administrators' information

3) School: Store different universities' information

4) Experiment: Store platform experiment information

5) Course: Store courses information

6) Class: Store teaching classes' information

These 7 tables above are all used to store basic information. The following Table 8 is to represent the "many to many" relation between courses and experiments.

8) Exp_Course: represents the "many to many" relation between Table 4 and 5

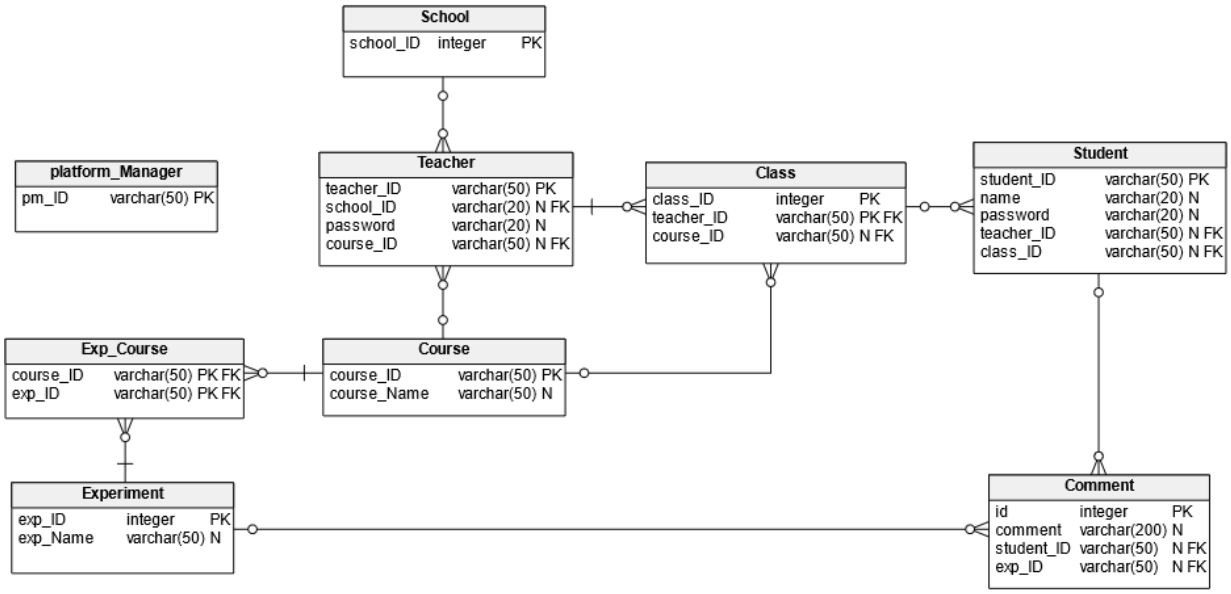

Figure 5-4-1 Database Diagram

5.4.2 Network File System (NFS)

When a container running in pod crashes, Kubelet restarts the container with a clean state which will cause all of the files that were created or modified during the lifetime of the container to be lost. On the other hand, multiple containers running in a pod need to share files so that they can get access to the same experiment files. To ensure that pods can use the same persistent storage data on any node, we need to use a network storage solution to provide data volumes [31] for the pods. A common network storage solution is NFS. With NFS, Nodes in the cluster can share files and synchronize data with each other by mounting directories (Figure 5-4-2-1). NFS is also the storage backend of Persistent Volumes (PV), PersistentVolumeClaim (PVC) which is shown in Figure 5-4-2-2:

1) **Persistent Volumes (PV):** A Persistent Volume is a storage resource in a Kubernetes cluster that has a lifecycle independent of any individual pod using it. It represents a physical or networked storage resource such as a physical disk, network-attached storage which is NFS in this paper, or cloud storage. PVs have properties like capacity, access modes, and storage class.

2) **Persistent Volumes Claim (PVC):** A Persistent Volume Claim is a request for a specific amount of storage with particular characteristics. It's made by a pod to request storage for its data. PVCs are bound to PVs by Kubernetes, which means a PVC will bind to a suitable PV based on the requested storage class, capacity, and access mode. This allows pods to use storage without needing to know the specific details of where or how the storage is provisioned.

To dynamically provision a PV and PVC, the cluster needs the support of provisioning and storage classes:

1) **Provisioning:** Provisioning in Kubernetes refers to the process of allocating storage resources to fulfill the storage requirements of applications. When an application or pod needs storage, a PVC is created to request the desired amount of storage with specific characteristics, and then the cluster provisions the actual storage to satisfy this claim. To achieve this process, the cluster must create a service account for it,

2) **Storage Classes (SC):** A Storage Class is a Kubernetes resource that defines the provisioning and configuration details for dynamically created Persistent Volumes. It acts as a blueprint for dynamically provisioning storage when PVCs are created. A Storage Class specifies attributes such as the storage provisioner, reclaim policy (what

happens to the storage when the associated PVC is deleted), access modes (how the storage can be accessed), and other parameters.

With provisioning and SC, we can dynamically create a PV for the PVC. PVCs will only bind to the PV which has the same storage class as PVC. The advantage of doing this is to avoid situations where some PVCs are allocated to PVs that greatly exceed their resource requirements.

This paper is going to start NFS between nodes, after that, we will create one PV to provide a NFS storage resource for PVCs in various pods for different students.

To let students get access to experiment files which are stored in the Master node and persistently store their results from containers running the Worker node, pods have to mount directories in the platform by using NFS, PV, and PVC so that pods can share files between Master node and Worker node.

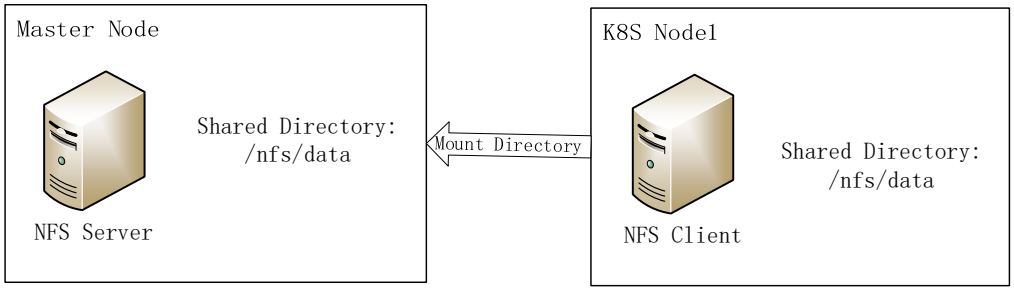

Figure 5-4-2-1: Network File System

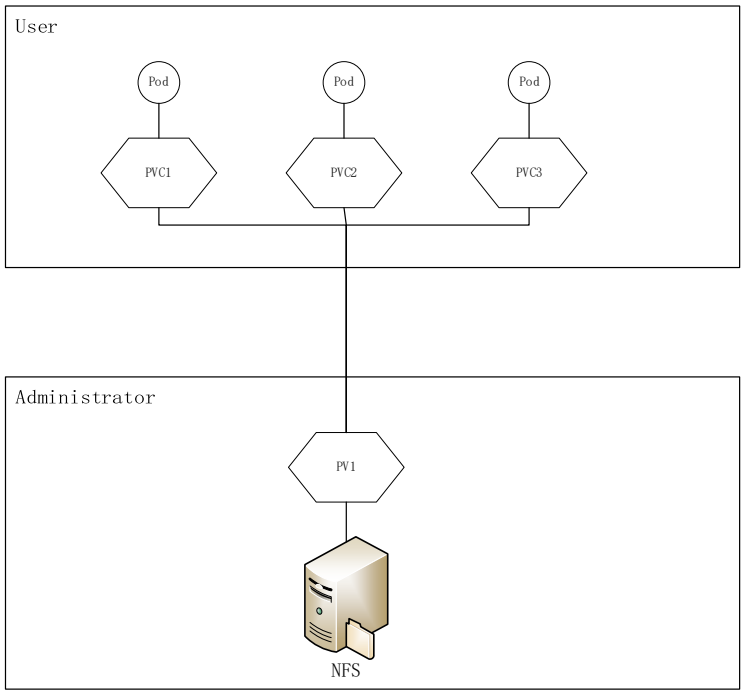

Figure 5-4-2-2: PV and PVC

5.5 Experiment Design

Machine learning is a field of artificial intelligence that involves the development of algorithms and models that enable computers to learn from and make predictions or decisions based on data. Instead of being explicitly programmed to perform certain tasks, machine learning systems use patterns and insights from data to improve their performance over time. There are several types of machine learning techniques which are shown in Figure 5-5.

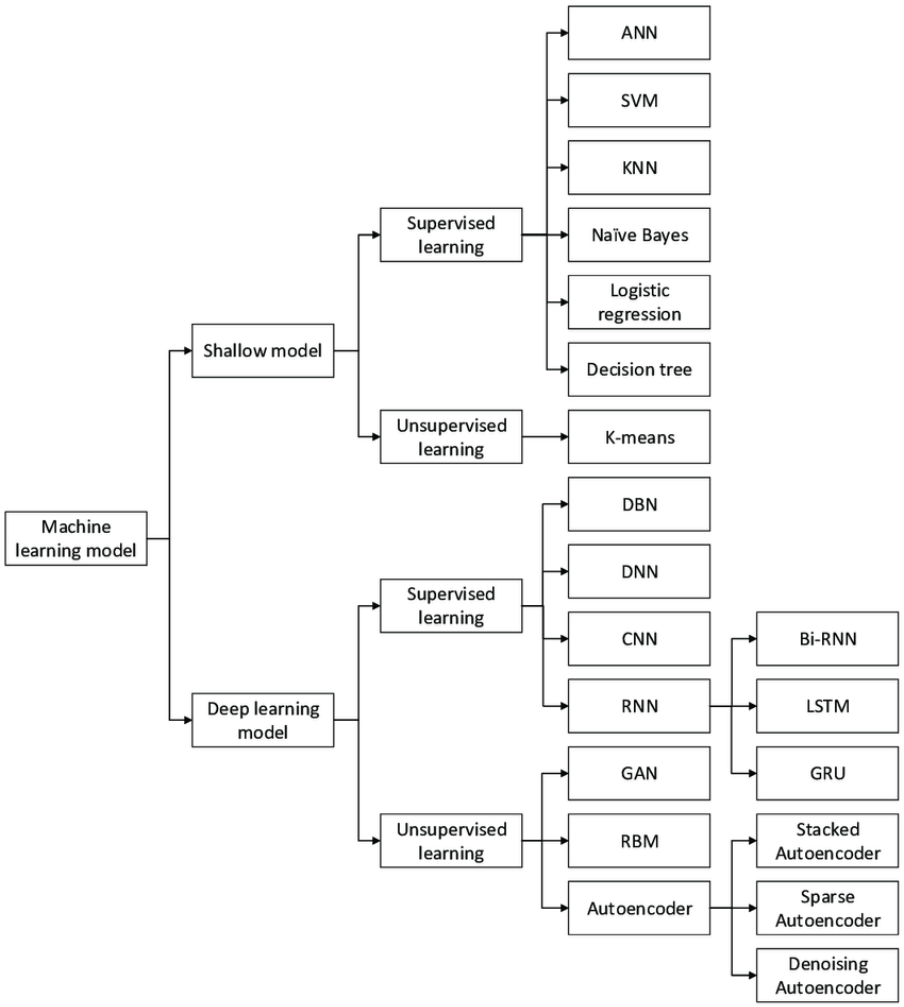

Figure 5-5: Taxonomy of Machine Learning [32]

Among these types of machine learning, supervised learning is more widely used in AI experiment education. In supervised learning, the algorithm is trained on labeled data, meaning that the input data is paired with the correct output. The algorithm learns to map inputs to outputs and can then make predictions on new, unseen data. The input and output are structured, which is simple and clear for students to understand and learn the basic knowledge of machine learning.

To better validate the platform's usability and integrity, as well as to verify its capability to provide online AI experimentation services successfully, this article presents two supervised machine learning experiments. One is using a logistic regression classifier to classify the type of Iris in the iris dataset by displaying the decision boundary. In this experiment, students are required to find the optimal value of hyperparameter (C). Another one is to use a confusion matrix to evaluate the performance of the decision tree on classifying the iris dataset and find the best training/testing split. In this experiment, students will learn how the percentage of data used for training and testing the data has an impact on the performance of the classifier and how to evaluate a model.

5.5.1 Supervised Learning: Logistic Regression Classifier Experiment

Logistic Regression (LR) is a statistical algorithm that is widely used in supervised learning and for binary classification tasks, where the goal is to predict the probability that an input belongs to one of two possible classes.

In logistic regression, the algorithm models the relationship between the input features and the probability of belonging to a specific class using the logistic function. This experiment aims to request students to find the best value of C and the corresponding decision boundary.

5.5.1.1 Parameter C

In LR, the parameter C represents the inverse of regularization strength where regularization is a technique used to prevent overfitting in the model. A smaller value of C increases the regularization strength, making the model simpler and potentially reducing the chances of overfitting. A larger value of C reduces the strength of regularization, allowing the model to fit the training data more closely which might lead to potentially overfitting.

C must be a float number that is bigger than 0. The default value is 1.0 (Figure 5-5-1-1), which means that the ratio of the regularization to the loss function is 1 to 1. In this experiment, students are asked to change the value of C and check how the decision boundary changes.

```
LogisticRegression = LogisticRegression(C=1)
```

Figure 5-5-1-1: C in LR

5.5.1.2 Sklearn

Scikit-learn [33], often abbreviated as sklearn, is an open-source machine-learning library for the Python programming language. This library is chosen in this paper due to a variety of algorithms that can be easily implemented for different applications and various datasets that can be applied in machine learning, data science, and other AI areas. Scikit-learn supports a consistent API for different algorithms, making it easier to conduct experiments with different models and techniques.

5.5.1.3 Iris Dataset

The iris flower has various varieties, and flowers of different varieties have their distinct characteristics. These can be briefly summarized by observing the following four aspects: sepal length, sepal width, petal length, and petal width. We hope that the computer can identify patterns from known sample data so that it can recognize the variety of the flower based on these characteristics.

By printing this dataset (Figure 5-5-1-3), we can see that there are 'data', 'target', 'target_names', and 'feature_names', as well as some descriptive information.

'data' is a two-dimensional array with 150 rows and 4 columns. Each row represents a data sample, totaling 150 samples, and the four columns correspond to the four feature names in 'feature_names': 'sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)'.

'target' is a one-dimensional array with 150 elements, corresponding sequentially to the 150 samples in 'data'. The values 0, 1, and 2 represent the three classes in 'target_names': 'setose', 'versicolor', and 'virginica', which are the three varieties of iris flowers.

In this experiment, the logistic regression classifier is supposed to be trained with the first two columns of data (sepal length, sepal width) and all the rows from the target. After training, it will be used to classify and determine which target (0,1,2) each row belongs to.

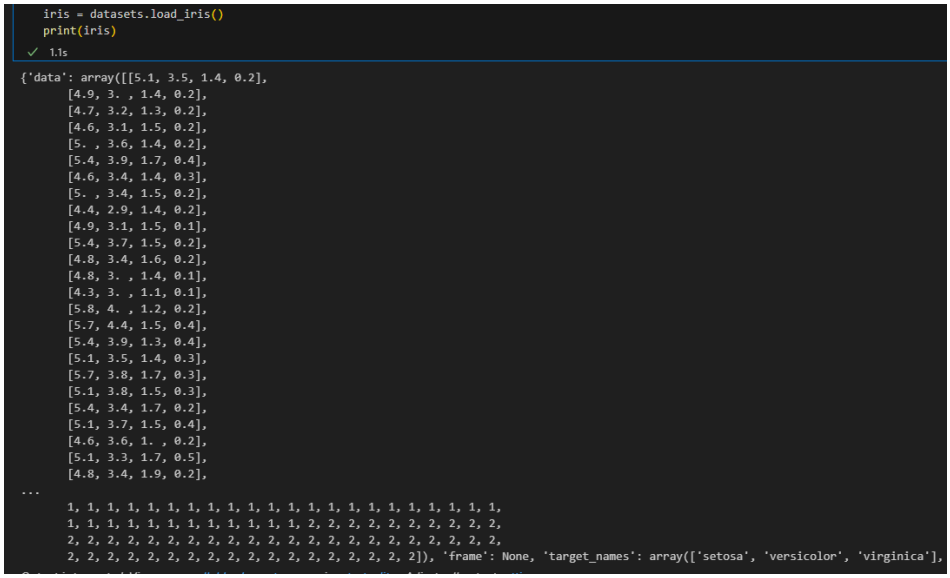

Figure 5-5-2-1: Iris dataset

5.5.1.4 Decision Boundary Display

A decision boundary is a conceptual boundary that separates different classes or categories in a dataset. It represents the region where the algorithm makes a decision about which class a particular data point belongs to.

In the context of this logistic regression experiment, adjusting the parameter C influences the position and shape of the decision boundary. A larger value of C might result in a more flexible decision boundary that closely fits the training data, potentially leading to overfitting, while a smaller value might result in a simpler decision boundary that is less influenced by individual data points, potentially leading to underfitting. The goal is to find the right balance that generalizes well to unseen data.

To allow users to observe the changes in the decision boundary more intuitively, this project will use the DecisionBoundaryDisplay function of sklearn-inspection to visualize the model results as shown in the following figure.

5.5.1.5 Educational Goal

This experiment involves using the logistic regression algorithm to classify the open-source iris dataset from the sklearn library. By adjusting the parameter C, we will examine its effect on the model's decision boundary. During this process, students are supposed to find the best model of LRC (e.g. find the optimal value of C and the corresponding decision boundary that the one which is the closest to the actual Iris

data). This experiment can help students have a basic understanding of how to use and load a dataset, use a classifier in Scikit, plot decision boundaries, and change hyperparameters to find the optimal classifier. The sample of the decision boundary is shown in Figure 5-5-1-5.

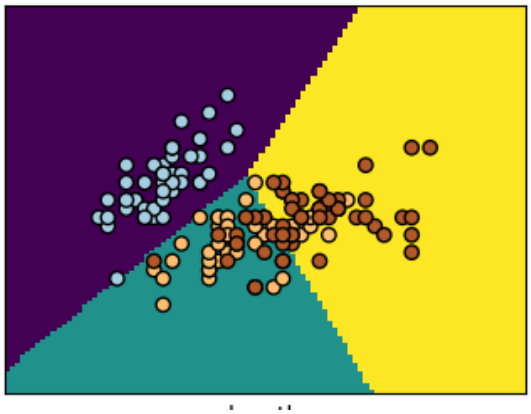

Figure 5-5-1-5: Sample of Decision Boundary

5.5.1.6 LR Experiment Procedure

To help developers get a clear and better understanding of how this experiment works, this section presents a workflow chart in Figure 5-5-1-6 to illustrate the structure of the code file and how the experiment works.

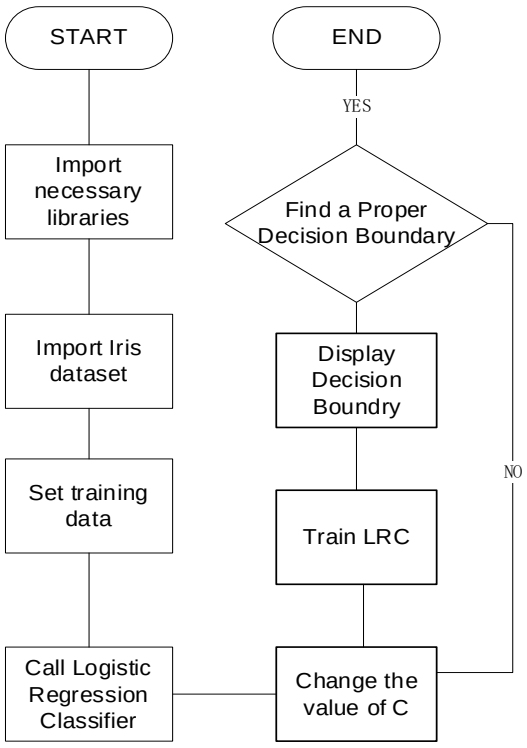

Figure 5-5-1-6: Experiment Work Flow Chart

5.5.2 Supervised Learning: Decision Tree Experiment

A decision tree is a supervised machine-learning algorithm used for both classification and regression tasks. It's a graphical representation of a decision-making process that involves making a series of decisions based on the features of a given input. Each decision in the tree leads to a set of possible outcomes or actions.

This experiment is going to request students to use a decision tree algorithm to classify Iris data with different ratios of training/testing data splitting, then find the best ratio by checking the performance of the model with the confusion matrix.

5.5.2.1 Training/Testing Data Split

Training and testing data splitting is a fundamental practice in machine learning and data analysis. It involves dividing a dataset into two separate subsets: one for training a model and the other for evaluating its performance. The ratio of the splitting can lead to a huge impact on the model performance.

5.5.2.2 Confusion Matrix

A confusion matrix is a tool used in the field of machine learning and classification to visualize the performance of a classification algorithm. It provides a detailed summary of the predicted and actual class labels for a set of data instances. The confusion matrix is especially useful for understanding the performance of a model in terms of its true positive, true negative, false positive, and false negative predictions.

- True Positives (TP):

These are the instances that are correctly predicted as positive by the model.

- True Negatives (TN):

These are the instances that are correctly predicted as negative by the model.

- False Positives (FP):

These are instances that the model predicts as positive but are negative in reality.

- False Negatives (FN):

These are instances that the model predicts as negative but are positive in reality.

A confusion matrix is typically presented in a table format, where rows represent the actual class labels, and columns represent the predicted class labels. It helps you understand how well your model is performing for each class and which types of errors it's making.

Table 5-5-2-2 is an example of how a confusion matrix might look:

	Predicted Positive	Predicted Negative
Actual Positive	TP	FN
Actual Negative	FP	TN

Table 5-5-2-2: Confusion Matrix

From this confusion matrix, we can calculate various evaluation metrics, such as:

- Accuracy: (TP + TN) / (TP + TN + FP + FN)
- Precision: TP / (TP + FP)
- Recall (Sensitivity or True Positive Rate): TP / (TP + FN)
- F1-Score: 2 * (Precision * Recall) / (Precision + Recall)

5.5.2.3 Educational Goal

This experiment involves the use of a decision tree to classify the Iris dataset and vary the ratio of testing/training data splitting to find the optimal performance of the decision tree mode. There are 3 main educational goals of this experiment:

1) Understand how different data splitting ratios affect the decision tree model's performance.
2) Identify the optimal ratio that balances model accuracy and generalization.
3) Gain insights into potential overfitting or underfitting tendencies of the decision tree algorithm

5.5.2.4 DT Experiment Procedure

To help developers get a clear and better understanding of how this experiment works, this section presents a workflow chart in Figure 5-5-2-4 to illustrate the structure of the code file and how the experiment works.

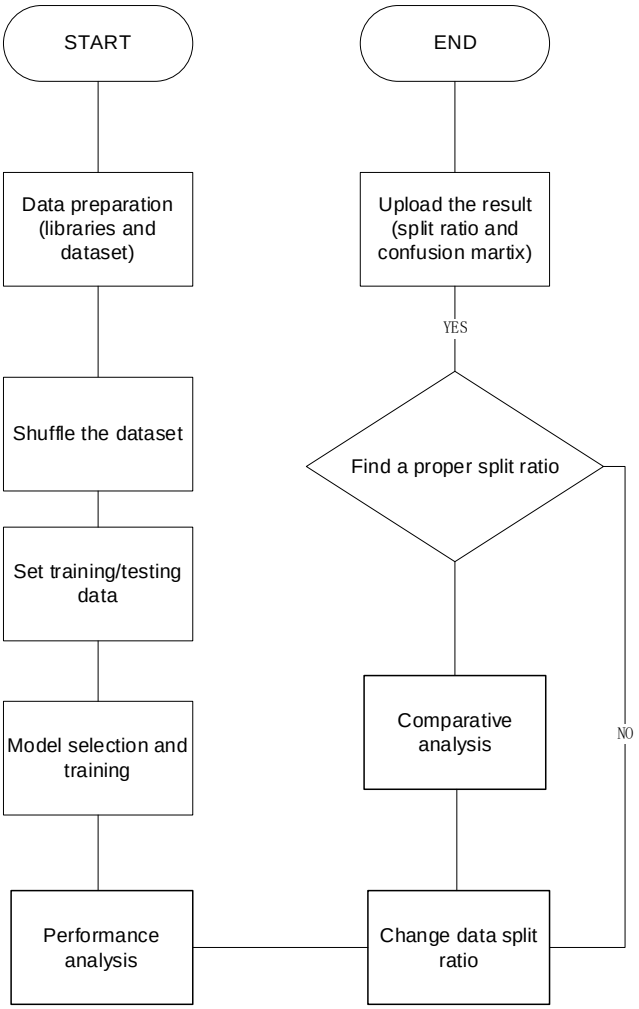

Figure 5-5-2-4: Decision Tree Experiment Work Flow Chart

5.6 Stakeholder Function Work Flow

This section will describe the main function design of the platform administrator, educator user, and student user and illustrate their core function workflow with flow charts.

5.6.1 Platform Administrator Function Design

Platform administrator function module: educator user management, Serverless service monitoring (core function). Platform administrator workflow is shown in Figure 5-6-1. Before entering the platform administrator system, platform administrators should log in to the platform administrator login page. If login is successful, platform administrators can manage platform backend resources and monitor the Kubernetes cluster.

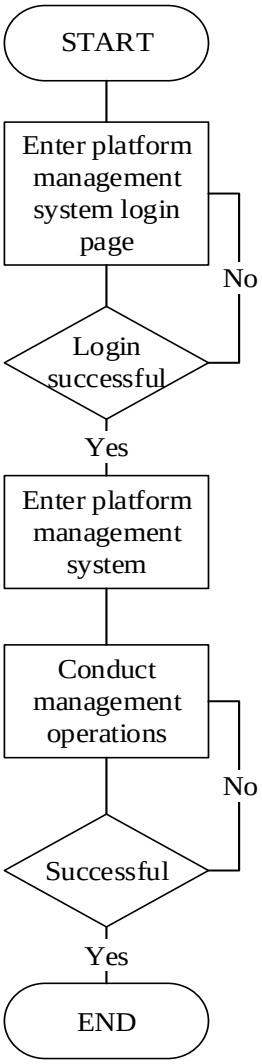

Figure 5-6-1: Platform Administrator Core Function Work Flow Chart

5.6.2 Educator Function Design

Educator function module: student user management (view, add, delete), experiments management including add, delete, and view (main function), and personal information management. The educator user core function workflow is shown in Figure 5-6-2. The most significant part of the teacher user function module is uploading experiment files. Files parameters and their description are shown in Table 5-6-2.

Parameter	Description
experiment_Name.ipynb	Experiment code file
result.ipynb	Empty experiment result file
.png, .jpg, .text, etc.	Picture and text resources of the experiment

Table 5-6-2: Experiment Files uploaded by teacher user

1) “experiment_Name.ipynb”

Educators use JupyterNotebook to call basic algorithms or self-designed algorithms to complete the design of AI experiments. Reserve the core codes of experiments in the form of a blank code block, and use the markdown function to insert the description of this step above the blank code block and give it to the students to complete. For the actual teaching needs, educators can insert the experiment flow chart by using the markdown function in the " experiment_Name.ipynb "

2) “result.ipynb”

Educators can design a variety of AI experiments, and experiment results are different. Therefore, this paper specially designs a blank result file. After completing the experiment on the experiment page, students return to the upper level, copy the experiment results to the " result. ipynb" file, and store them permanently on the platform.

3) “.png, .jpg, .text, etc.”

When teacher users upload the experiment file package, it should contain pictures, texts, and other resource files that are used to introduce the experiment.

It is noticeable that the files uploaded by educators will be stored in the “shared” folder in the platform. At the beginning of the experiment, students need to copy code files to their directories which will be described in the next section.

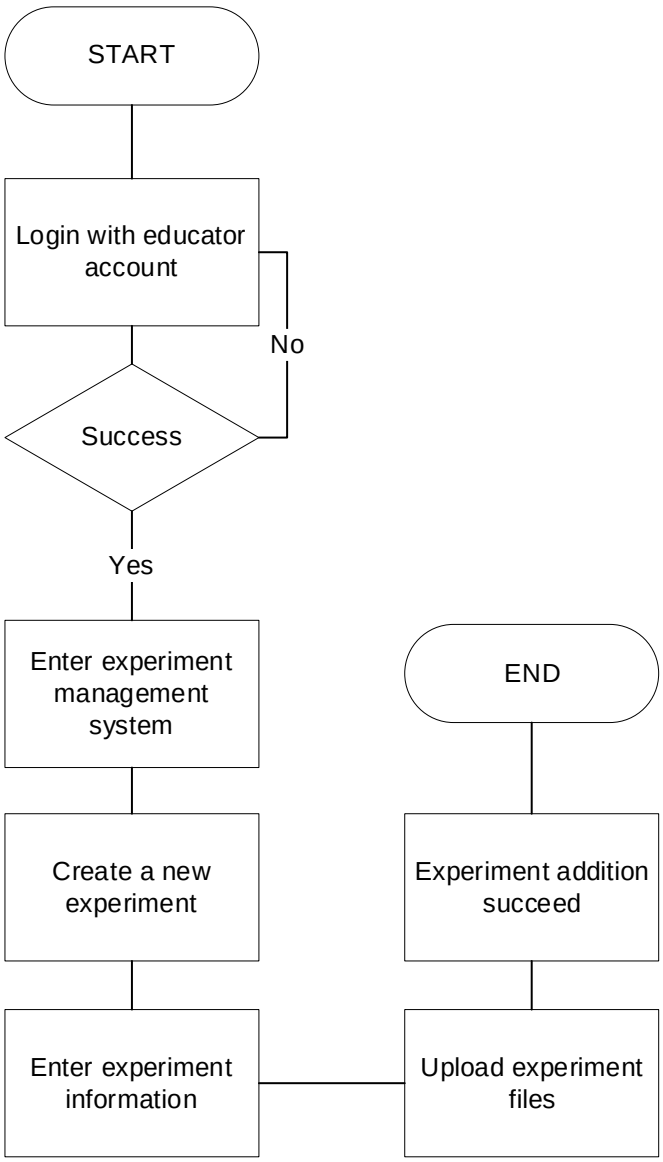

Figure 5-6-2: Educator Core Function Work Flow Chart

5.6.3 Student Function Design

Student users function module: personal information management, conduct experiment (core function), comment on experiments. The core function is doing AI experiments function in the platform. The workflow of doing an online AI experiment is illustrated in Figure 5-6-3-1.

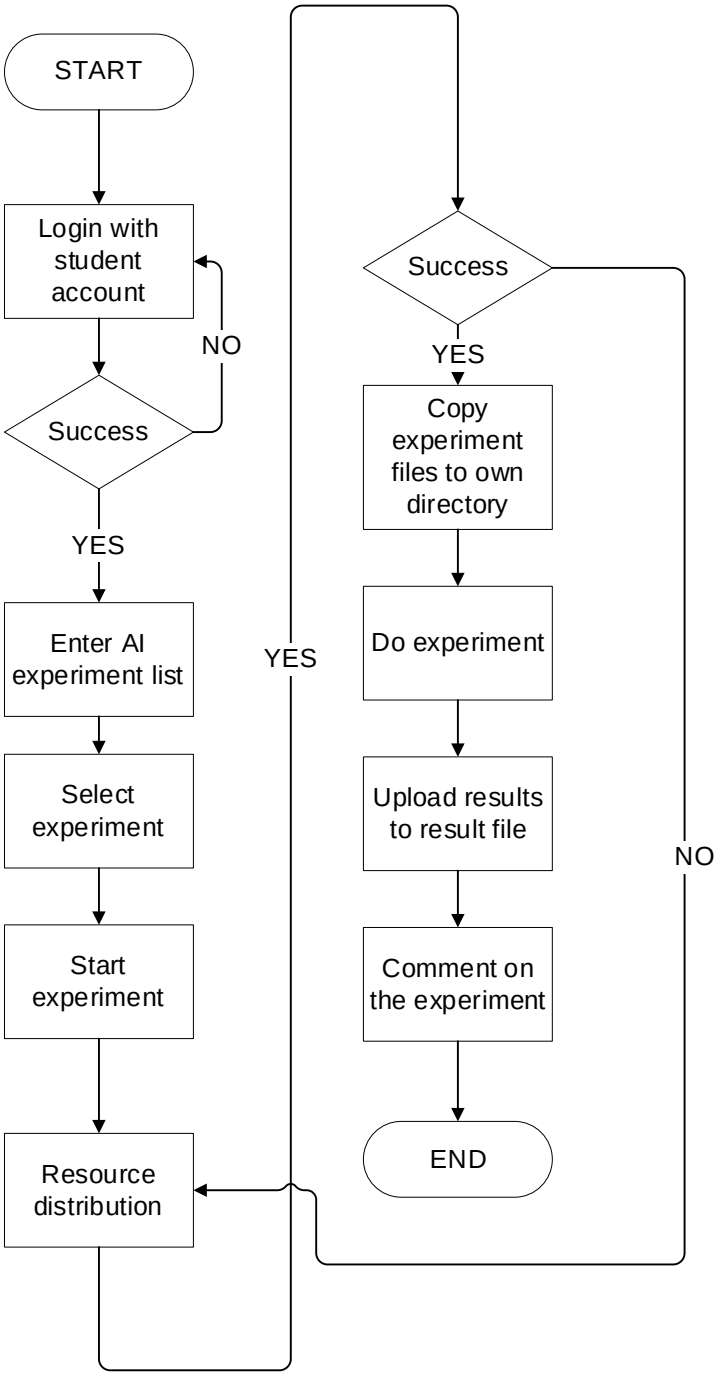

Figure 5-6-3: Student Core Function Work Flow Chart

5.6.3.1 Kubernetes Cluster Resource Distribution

This subsection detailly describes how the Serverless services work in the Kubernetes cluster and distribute resources for students. The procedure is also displayed in Figure 5-6-3-1-1.

Student users send the experiment request to the platform by clicking the event. The platform reads the code file uploaded in advance by teacher users from the storage, finds the corresponding image name from the database, and then automatically starts a corresponding Docker container instance in the pod created by the Kubernetes cluster according to the experiment name.

After a series of tasks are completed, it is regarded as the successful allocation of experiment resources. After successful allocation, students will see the code file corresponding to the experiment stored in the “shared” directory (Figure 5-6-3-1-2), copy the code file to their directories, and then click the file to complete the experiment. In the end, copy the experiment results to the " result.ipynb" file stored in the “shared” directory. Completing the above tasks within the specified time will be regarded as finishing the experiment successfully, otherwise, it will be regarded as failing the experiment.

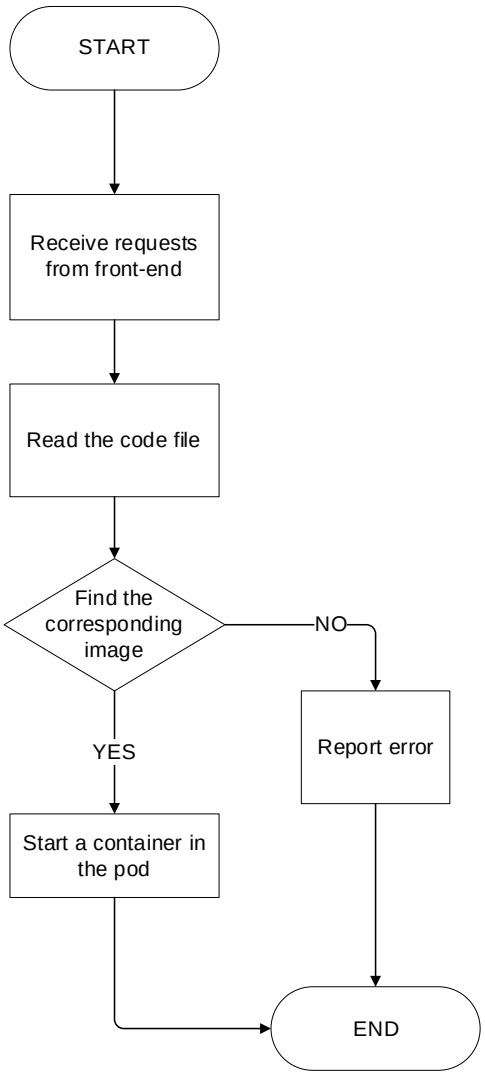

Figure 5-6-3-1-1: Resource Distribution Work Flow Chart

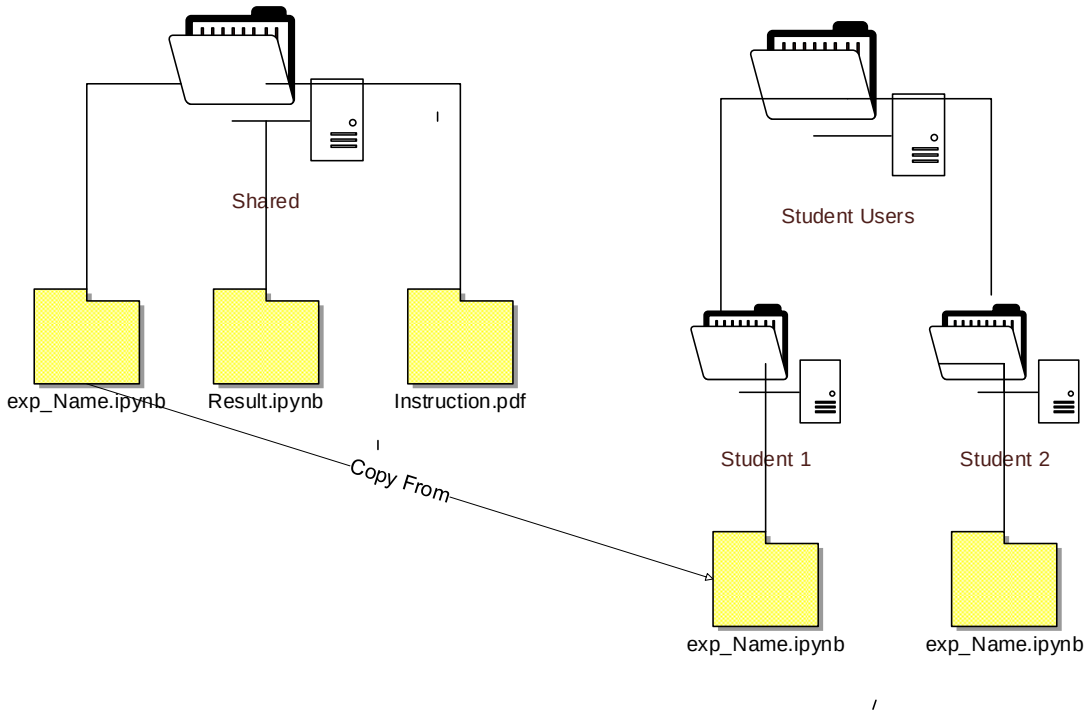

Figure 5-6-3-1-2: Platform File Storage

Chapter 6
System Implementation

This chapter describes the platform implementation, which includes front-end web pages (educator subsystem, student subsystem, platform administrator subsystem), back-end server (educator services, student services, platform administrator services), and Kubernetes cluster. In addition, the supervised learning experiment will also be implemented based on the design.

6.1 Environment Setup

Before the implementation, we need to set up the necessary environment and tools for the platform. The precise information of tools and steps are listed below:

- Installing VMware Workstation. The whole project will run on virtual machines.
- Creating two virtual machines as nodes with Ubuntu images and assigning them static IP addresses.
    1) Master Node: OS: Ubuntu 20.04 IP address: 192.168.100.100
    2) Worker Node1: OS: Ubuntu 20.04 IP address: 192.168.100.101
- Installing SQLite DB Browser, VScode, Pycharm
- Installing Python 3.9 and creating a Flask project in Pycharm
- Installing Docker and pulling images is essential for building a Kubernetes cluster. The image list is shown in Figure 6-1.
- Installing kubeadm, kubectl, kubelet version:1.22.4 on both nodes

After finishing all these steps, it is regarded as successfully setting up the basic environment of the platform. And now we can start developing the project.

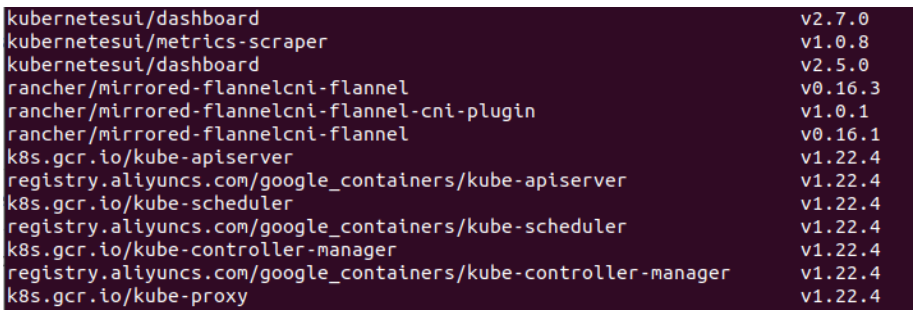

Figure 6-1: Docker Images of Kubernetes Cluster

6.2 Project Code Structure

This section introduces the front-end code structure and the back-end code structure which is automatically created by Pycharm. Figure 6-2-1 displays the code structure of the project. The “app” folder contains the core code files of the platform, including templates (front-end web pages), platform static resources (CSS, JS, images), back-end routes, and APIs are written in “views.py” and the database is built in “model.py”. Other folders and Python files present the basic configuration and initiation codes of the Flask project.

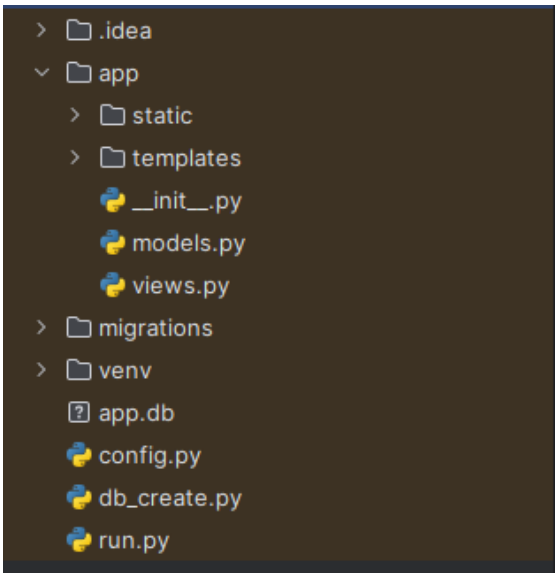

Figure 6-2-1: Code Structure of Project

As the educator subsystem, student subsystem, and platform administrator subsystem have similar internal structures, we take the educator subsystem as an example to show the front-end package and structure of subsystems in Figure 6-2-2.

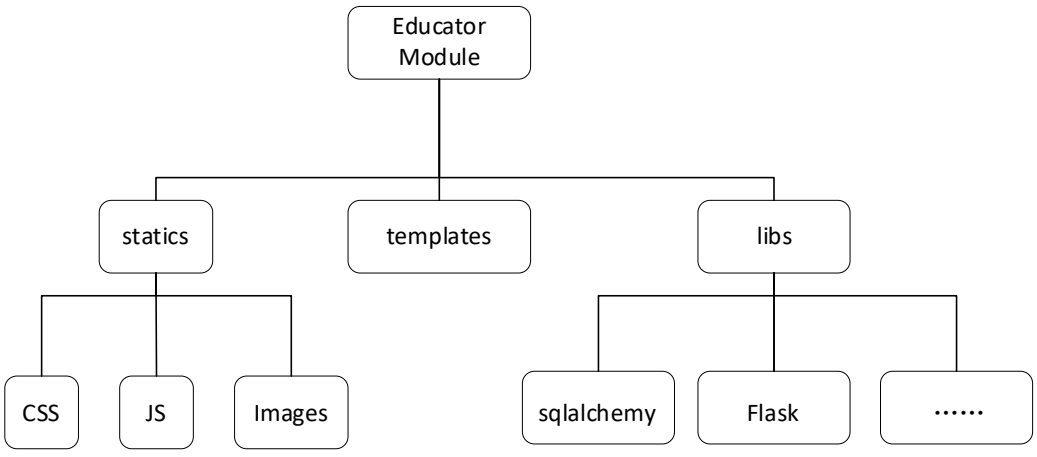

Figure 6-2-2: Educator Module Structure

6.3 Implementation of Educator Module

This section describes how to implement educator subsystems and functions including login, managing experiments, viewing students, and other functions.

6.3.1 Login

Figure 6-3-1-1 shows the login interface. Users are required to enter the username and password they were given before. The system employs a one-click login procedure. When clicking the login button, the server will query the database to check if the combo of username and password is available (Figure 6-3-1-2). If it is, users can successfully log in to the system and set the session username, otherwise, it will return to the login page and alert “username or password is not available”. There is an animation when switching to the platform administrator login page which is applied by using Animate.css [33], an open-source CSS project on GitHub.

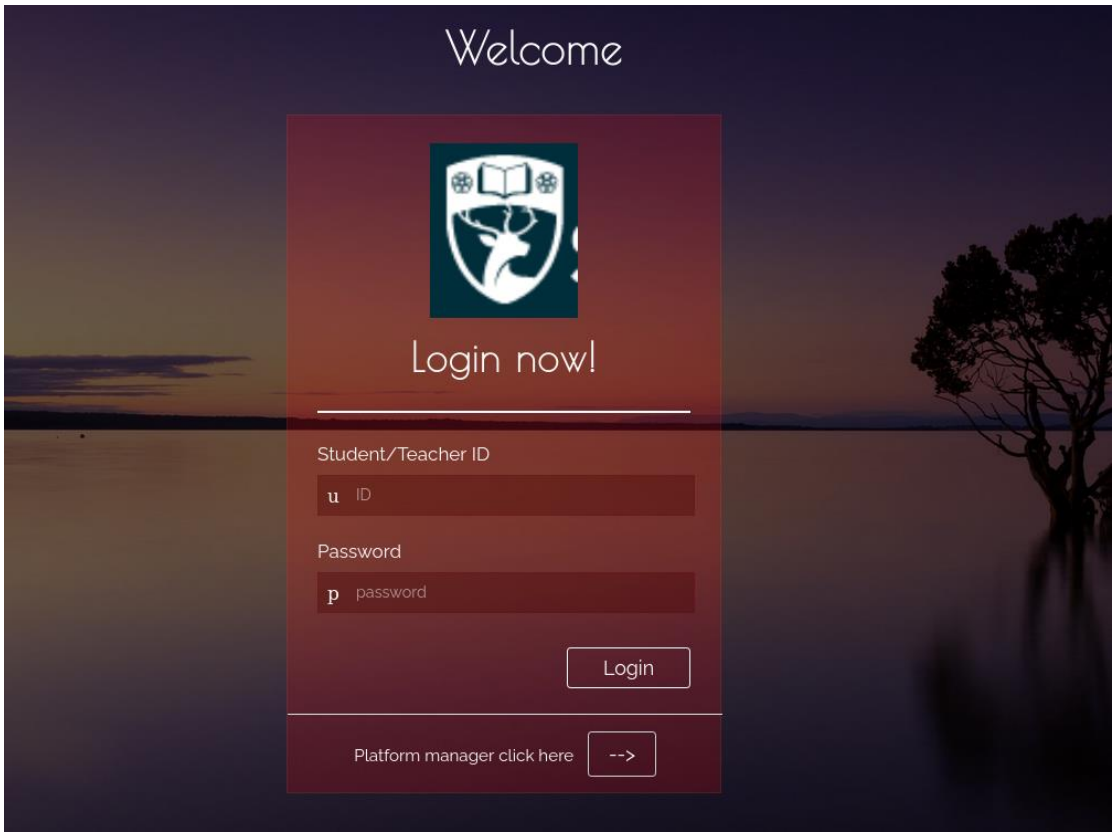

Figure 6-3-1-1: Login Page

After successfully authenticating the user, the system will store the username in session and cookie. In the following user’s operations, the session will be used to query

the database to get corresponding information and resources. For example, view the student list of this educator and manage experiments belonging to the user.

```
if t_check:
    if t_check.password == pwd:
            session['username'] = ID
            session['log'] = 'login'
            course_ID = Teacher.query.filter_by(teacher_ID=session['username']).first().course_ID
            course = Course.query.filter_by(course_ID=course_ID).first()
            exps = course.Exps
            response = make_response(render_template('teacher_system.html', exps=exps, flag=0, username=ID))
            response.set_cookie('username', ID)
            return response
    else:
        return render_template('index.html', flag=1)
```

Figure 6-3-1-1: Login Function Back-end Code

6.3.2 Experiment Management

After logging in as an educator, the platform jumps to the experiment management submodule first. This section explains this module in the following order: view, add, and delete.

6.3.2.1 View Experiments

Figure 6-3-2-1 displays the interface and result of the viewing experiment function. The table contains the name, time limit, experiment ID, and delete button of each experiment.

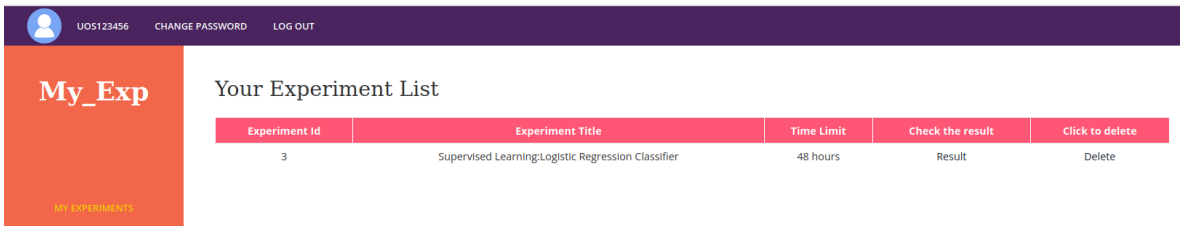

Figure 6-3-2-1-1: Viewing Experiment Page

In this project, tables that are used to display information from the database are built in a single HTML file, and then the target HTML page as a template as shown in Figure 6-3-2-2.

```
<div class="contact">
  <h3 class="tittle">Your Experiment List</h3>
   {% extends "teacher_exps.html" %}
</div>
```

(a)

```
<tr>
    <th>Experiment Id </th>
    <th>Experiment Title</th>
    <th>Time Limit</th>
    <th>Click to delete</th>
</tr>

    {% for exp in exps %}
<tr>
    <form  method="post">
    <td >{{exp.exp_ID}}</td>
      <td >{{exp.exp_Name}} </td>
    <td>{{exp.timeLimit}}</td>
    <td><a href="{{url_for('delete_Exp',exp_ID=exp.exp_ID)}}" name="delete_Exp" onclick="{if (confirm('Are you sure to delete this expriment?'))
{this.document.formname.submit();return true;}return false;}"> Delete</a></td>
      </form>
```

(b)

Figure 6-3-2-1-2: HTML Code of Viewing Experiment

As mentioned above, the back end uses a username as a key to query the experiment information uploaded by the educator and display the list of it. It queries the course ID according to the teacher username first, then finds the course with the course ID and returns the experiment of that course to the experiment table template.

```
if session['log'] == 'login':
    course_ID = Teacher.query.filter_by(teacher_ID=session['username']).first().course_ID
    course = Course.query.filter_by(course_ID=course_ID).first()
    exps = course.Exps
    return render_template('teacher_system.html', exps=exps, username=session['username'])
else:
    return render_template('index.html', flag=5)
```

Figure 6-3-2-1-3: Back-end Code of Querying Experiments

6.3.2.2 Add a New Experiment

Adding a new experiment is the core function of the educator module. The steps are illustrated in Section 5.6.2. Figure 6-3-2-2-1 shows the UI of adding the experiment page. On this page, the educator is required to enter the experiment name, and time limit and upload experiment files including code files, result files, and other files. After clicking the submit button, the system will add the information about this experiment to the database. Then, it creates folders for this experiment in the Master Node in the format of Figure 5-6-3-1-2. One “exp_name/shared” folder is used to store files

uploaded by the educator and the user's students' directories. The back-end code is shown in Figure 6-3-2-2-2. The created directories in the Master node are displayed in Figure 6-3-2-2-3.

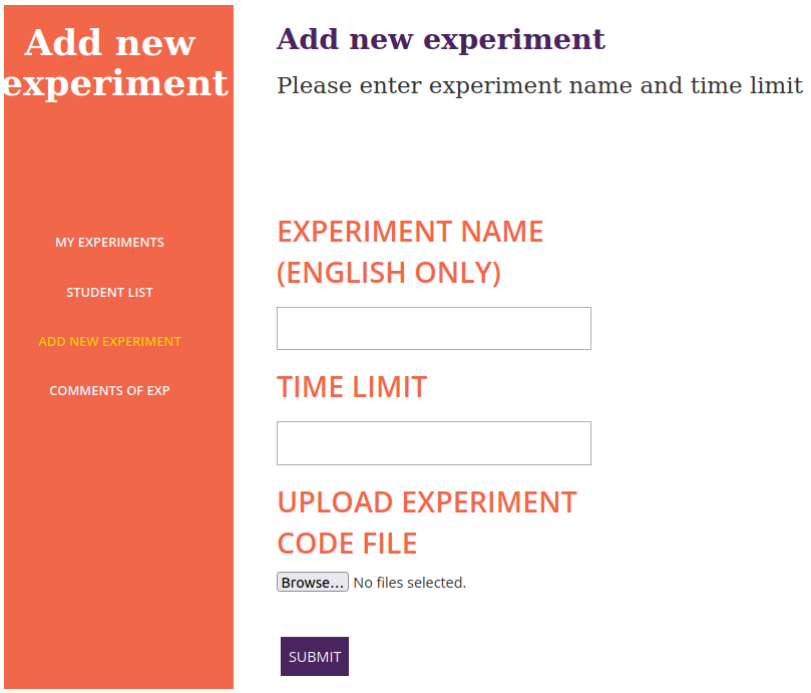

Figure 6-3-2-2-1: Adding a New Experiment Page

```
time_Limit = request.form['time_Limit']
basepath = '/home/kyokoz/data/' + str(exp_Name) +'/shared/'
if not os.path.exists(basepath):
    os.makedirs(basepath)
students = Student.query.filter_by(teacher_ID=session['username']).all()
for student in students:
    stu_path = '/home/kyokoz/data/' + str(exp_Name) +'/' + student.student_ID
    if not os.path.exists(stu_path):
        os.makedirs(stu_path)
new_exp = Experiment(exp_Name = exp_Name, timeLimit = time_Limit)
upload_path = os.path.join(basepath,secure_filename(f.filename))
f.save(upload_path)
db.session.add(new_exp)
```

Figure 6-3-2-2-2: Back-end Code of Adding a New Experiment Function

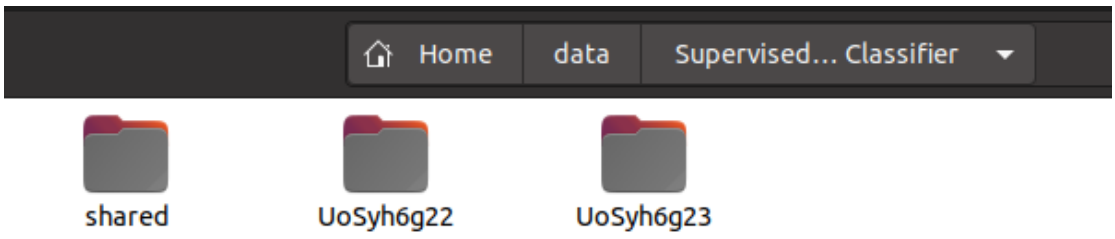

Figure 6-3-2-2-3: Created Folders in Master Node

6.3.2.3 Delete an Experiment

This function is designed as a one-click trigger event. While the educator is viewing the experiment list, if he clicks the delete button, the system will find the corresponding experiment information in the database, and then safely remove it. The UI and HTML code of this function can be found in Section 6.3.2.1. Figure 6-3-2-3 shows the back-end code of deleting an experiment.

```
def delete_Exp(exp_ID):
    course_ID = Teacher.query.filter_by(teacher_ID=session['username']).first().course_ID
    course = Course.query.filter_by(course_ID=course_ID).first()
    exps = course.Exps
    for exp in exps:
        if str(exp.exp_ID) == str(exp_ID):
            exps.remove(exp)
            db.session.delete(exp)
            db.session.commit()
```

Figure 6-3-2-3: Back-end Code of Deleting an Experiment

6.3.3 View Comments from Students

In the student subsystem, students are allowed to comment on experiments to provide some feedback to the educator. As an educator, getting feedback from students can help us improve the quality of education and have better communication with students. Therefore, the educator module also provides such a function for educators. Figure 6-3-3-1 shows the UI of the checking comments function page. On this page, educators can see the content of comments, and experiment ID. Student's information is hidden to protect their personal information. The HTML code of this table is displayed in Figure 6-3-3-2 and the Back-end Code is in Figure 6-3-3-3.

Figure 6-3-3-1: Viewing Comments Page

```
    {% for comment in comments %}
<tr>
    <form  >
    <td >{{comment.comment}}</td>
    <td>{{comment.exp_ID}}</td>
      </form>
</tr>
 {%endfor%}
```

Figure 6-3-3-2: HTML Code of Comments Table

```
@app.route('/check_comment')
def check_comment():
    if session['log'] == 'login':
        comments = []
        students = Student.query.filter_by(teacher_ID=session['username']).all()
        temps = Comment.query.all()
        for temp in temps:
            for student in students:
                if temp.student_ID == student.student_ID:
                    comments.append(temp)
        return render_template('check_comments.html', comments = comments,username=session['username'])
    return render_template('index.html', flag=5)
```

Figure 6-3-3-3: Back-end Code of Viewing Comments

6.3.4 Change Password

The default passwords of educators are their phone numbers. To keep the account safe, the educator is allowed to change the password to what they want. It is designed as a one-click event. When the user clicks the “change the password” button in the nav bar, the platform jumps to the target page. On that page, the user is required to enter the old password and new password, after clicking submit, the server will verify whether or not the old password is correct as shown in Figure 6-3-4-1, if it is, the password is updated and the platform will be automatically logged out and have to login again with the new password. The front-end page of this function is displayed in Figure 6-3-4-2.

```
if t_check:
    if t_check.password == pwd:
        t_check.password = Npwd
        db.session.commit()
        session['log'] = 'logout'

        return redirect(url_for('login'))
    else:
        return render_template('change_pwd.html', flag=1)
```

Figure 6-3-4-1: Back-end Code of Changing Password

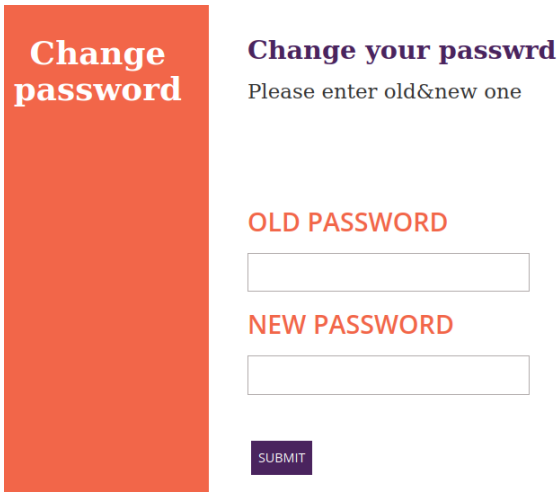

Figure 6-3-4-2: Changing Password Page

6.3.5 View Student List

By clicking the “student list” button on the left side, educators can view the list of students. A course of an educator might have more than one class, therefore, the table shows the student ID, name, and class of students taught by this educator as shown in Figure 6-3-5-1.

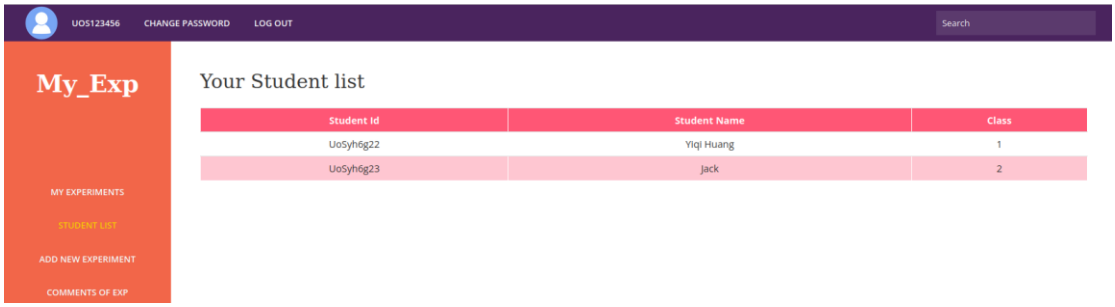

Figure 6-3-5-1: Viewing Student List Page

The server uses Educator ID as a key to query students taught to this user and return the information of them to the front-end template like viewing experiments and comments. Figure 6-3-5-1-2 shows the back-end code of this function.

```
if session['log'] == 'login':
    student_list = Student.query.filter_by(teacher_ID=session['username']).all()
    return render_template('check_students.html', students=student_list,username=session['username'])
return render_template('index.html', flag=5)
```

Figure 6-3-5-2: Viewing Student List Page

6.3.6 View Uploaded Results

To be able to view real-time results of various experiments uploaded by students, upon sending a request from the platform to the Kubernetes cluster by clicking the “result” button on the experiment list, the cluster will, based on the selected experiment, return the IP address of corresponding container and NFS shared directory for the teacher to access. Additionally, educators can download the result file in Jupyter Notebook by clicking the download button (Figure 6-3-6).

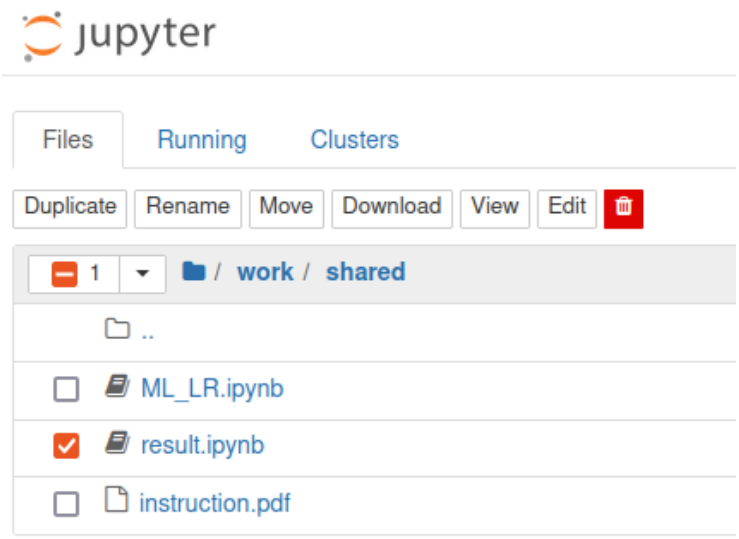

Figure 6-3-6: Check the Result File

6.4 Implementation of Student Module

This section describes how to implement student subsystems and functions including comments on experiments and conducting online AI experiments. The login function and change password function are the same as the educator’s; therefore, this section will no longer describe them again.

6.4.1 View Experiments

After login to the student subsystem, the platform will list available experiments for the student as shown in Figure 6-4-1-1. This table shows the experiment ID, name, time limit, and two buttons (start, comment) corresponding to another two one-click events that start this experiment and comment on the experiment.

Figure 6-4-1-1: Experiment List Page for Student

When the server receives the request from the front-end, it uses student ID as a key to query his educator first, after that, the server finds the course taught by that educator and eventually returns the information of the experiment of that course to the front-end template to show it. Figure 6-4-1-2 shows the back-end code of it.

```
@app.route('/stu_exps', methods=['GET', 'POST'])
def stu_exps():
        teacher_ID = Student.query.filter_by(student_ID=session['username']).first().teacher_ID
        course_ID = Teacher.query.filter_by(teacher_ID=teacher_ID).first().course_ID
        course = Course.query.filter_by(course_ID=course_ID).first()
        exps = course.Exps
        return render_template('student_system.html', exps=exps, username=session['username'])
```

Figure 6-4-1-2: Back-end Code of Viewing Experiments for Student

6.4.2 Conduct Online AI Experiments

Conducting online AI experiments on the platform is the core function of student users. This submodule consists of three parts: theory study, doing experiments, and uploading results.

6.4.2.1 Theory Study

When the user clicks the “start” button on the experiment list, the back-end will send a request via Kubernetes RESTful API to ask for resource distribution. the Kubernetes cluster will create a container inside the pod running on worker node 1 using the installed Docker image jupyter/tensorflow-notebook: Ubuntu:20.04 [35]. Once the container is successfully created, the cluster will return the address it and open it on the front end as shown in Figure 6-4-2-1-1.

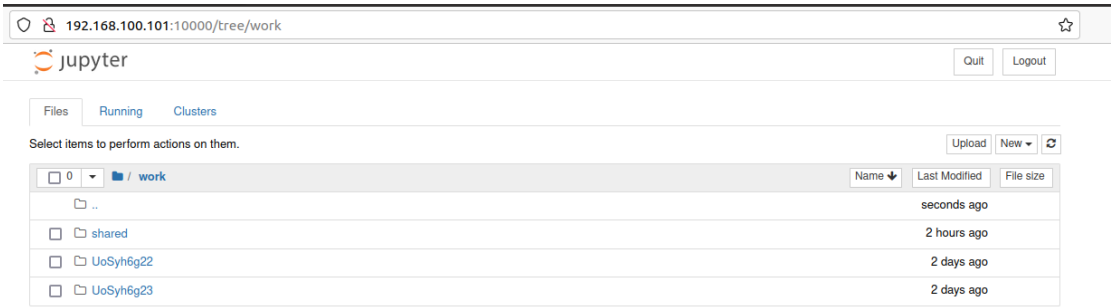

Figure 6-4-2-1-1: Jupyter Notebook Page

As Jupyter Notebook requires a token to open it in the container, therefore, the server will use Kubernetes API to check the token of that container, then join the token to the address in the format of “https://192.168.100.101:10000/token=xxxxxx”, so that the front-end can automatically open the corresponding container and provide the experiment environment for the student. The back-end code is shown in Figure 6-4-2-1-2.

```
req = requests.get('http://localhost:8080/api/v1/namespaces/ai/pods/ml/log').text
temp = json.dumps(req)
token = temp[(temp.find('http://ml:8888/?token=') + 22):temp.find('http://ml:8888/?token=') + 70]
return redirect("http://192.168.100.101:10000/?token="+token)
```

Figure 6-4-2-1-2: Back-end Code of Opening the Container

After entering the Jupyter notebook, the student is supposed to read the instructions stored in the “shared” folder which is shown in Figure 6-4-2-1-3. The instruction describes the steps of how to do this experiment and explains the principle and parameters of relative functions and algorithms. With theory study, students can have a deeper understanding of what they are going to do next.

Figure 6-4-2-1-3: “Shared” Folder

6.4.2.2 Do Experiment

When the student has a clear understanding of the experiment, he is supposed to copy the experiment code file to his directory (e.g., UoSyh6g22) and double-click it to open it on Jupiter's notebook to start the experiment. The opened code file is shown in Figure 6-4-2-2-1.

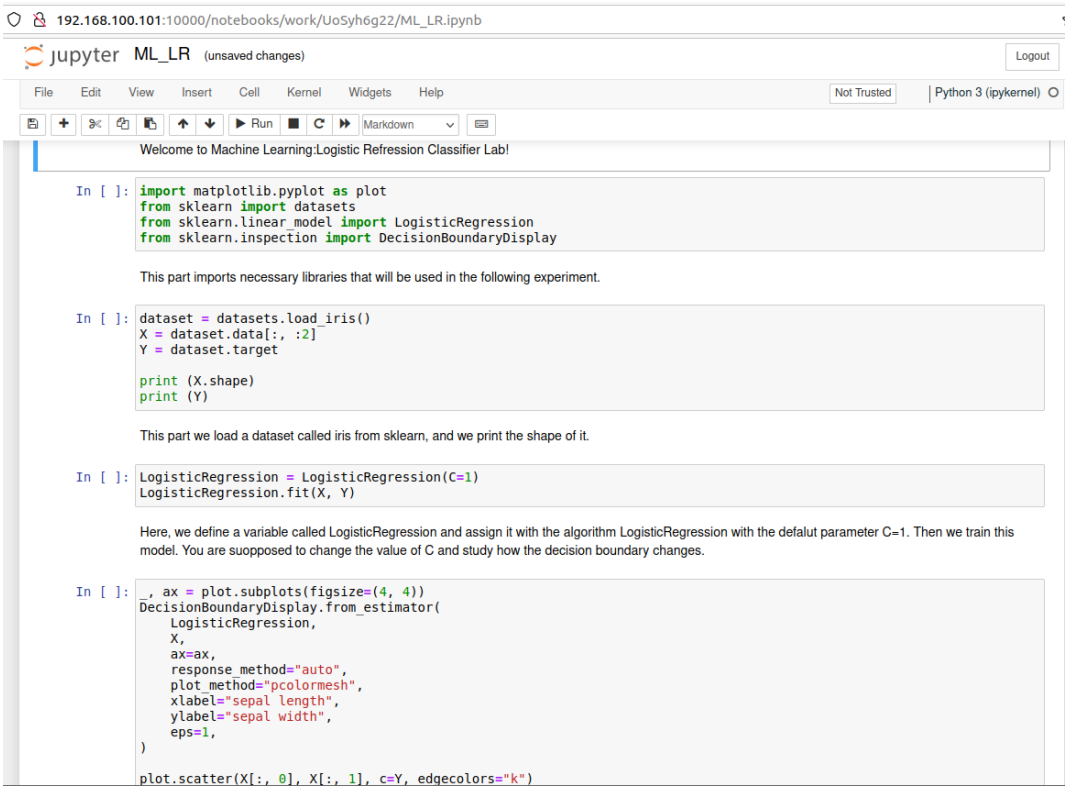

Figure 6-4-2-2-1: ML_LR.ipynb in Jupyter Notebook

In this file, the student is requested to change the value of C then click the “Run” button to watch the decision boundary of this experiment and find the optimal value of C and the result of the decision boundary. The result after clicking the “Run” button is shown in Figure 6-4-2-2-2.

```
In [16]: _, ax = plot.subplots(figsize=(4, 4))
DecisionBoundaryDisplay.from_estimator(
    LogisticRegression,
    X,
    ax=ax,
    shading='auto',
    response_method="auto",
    plot_method="pcolormesh",
    xlabel="sepal length",
    ylabel="sepal width",
    eps=1,
)

plot.scatter(X[:, 0], X[:, 1], c=Y, edgecolors="k")
plot.show()
```

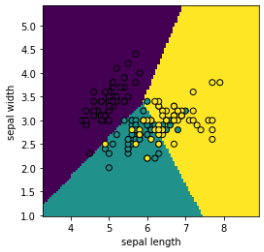

Figure 6-4-2-2-2: Result of Decision Boundary (when C=1)

6.4.2.3 Upload Result

As described in the last section, the experiment code file will finally return an image of the decision boundary to the student. Therefore, the student should save the image with different values of C, and then copy them to the “result. ipynb” file stored in the “shared” folder. The student should also mark his student ID and the corresponding value of C of each image as shown in Figure 6-4-2-3.

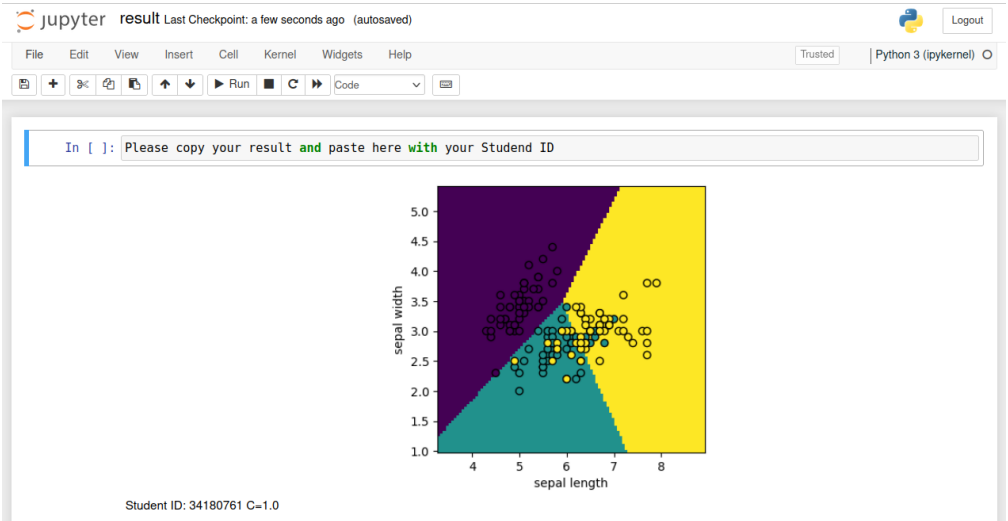

Figure 6-4-2-3: Copy the Result to “result.ipynb”

6.4.3 Comment on Experiments

In the experiment list, students can click the “comment” button to comment on the experiment. The front-end page is shown in Figure 6-4-3-1.

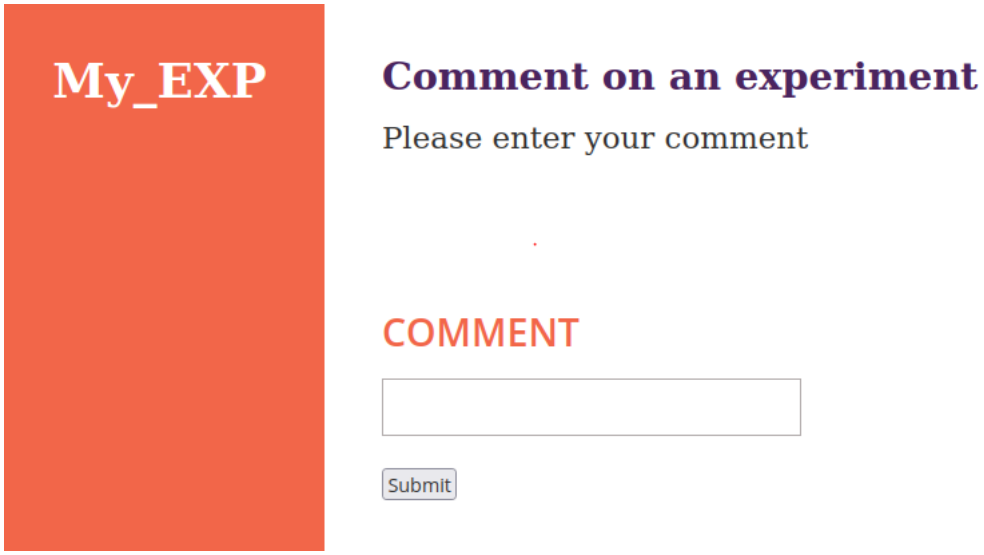

Figure 6-4-3-1: Commenting Page

After clicking submit, the back-end will receive the text and store it in the database with the content and experiment ID, so that the educator can distinguish the comment about which experiment. Figure 6-4-3-2 displays the back-end code of this function.

```
stu_comment = request.form['comment']
new_comment = Comment(comment = stu_comment, student_ID= session['username'], exp_ID=exp_ID)
db.session.add(new_comment)
db.session.commit()
return redirect(url_for('stu_exps'))
```

Figure 6-4-3-2: Back-end Code of Commenting

6.5 Implementation of Platform Administrator Module

This section explains how to implement platform administrator service features such as login, The platform administrator module is mainly used to manage platform users and monitor Kubernetes cluster health.

6.5.1 View Platform Users

As a platform administrator, it is necessary to be able to check if there is any suspicious account in the system. Although all of the accounts can only be created by platform

administrators, it is still possible that the attacker injects unknown accounts into the system. Figure 6-5-1 shows the front-end page of the viewing educator list.

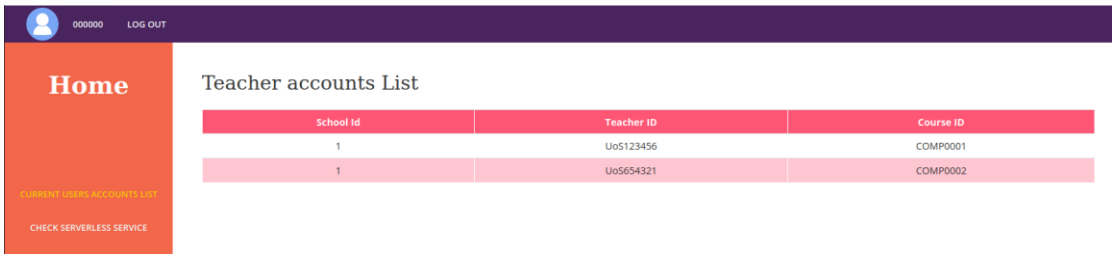

Figure 6-4-2-3: Viewing Educator Account Page

6.5.2 Monitor Kubernetes Cluster

The platform installed Kubernetes Dashboard to help platform administrators better monitor the Kubernetes cluster. Kubernetes Dashboard is installed as an application in the worker node 1 and has been bound to port 30001. When clicking the “check Serverless service” button, the back end will visit this port (192.168.100.101:30001) and jump to the login page. (Figure 6-5-2-1). On this page, the platform administrator is required to enter the unique token that is only held by him to enter to Kubernetes Dashboard which is shown in Figure 6-5-2-2. The token is generated by the cluster when installing Kubernetes Dashboard. It can be found in **Appendix A**.

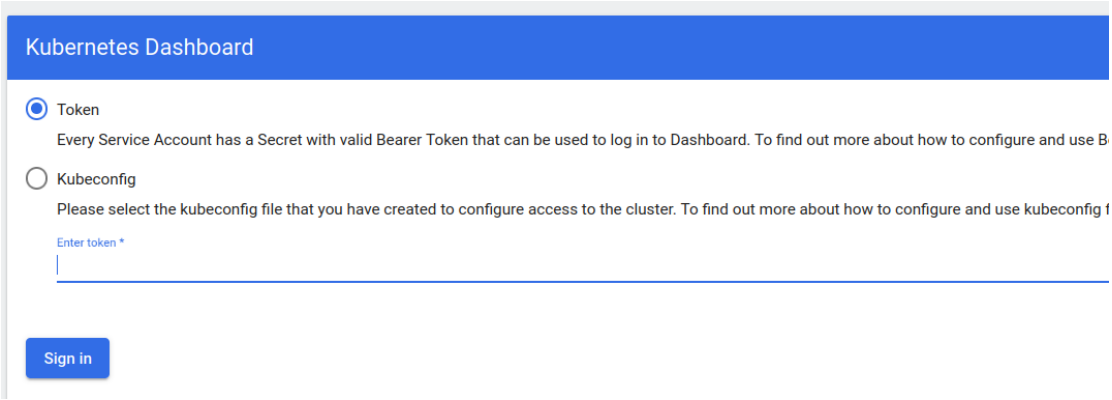

Figure 6-5-2-1: Kubernetes Dashboard Login Page

In the Kubernetes Dashboard, the cluster resources are grouped based on their namespace. Platform administrators can check which pods are running in a specific namespace and more information about those namespaces including workload, service, and storage.

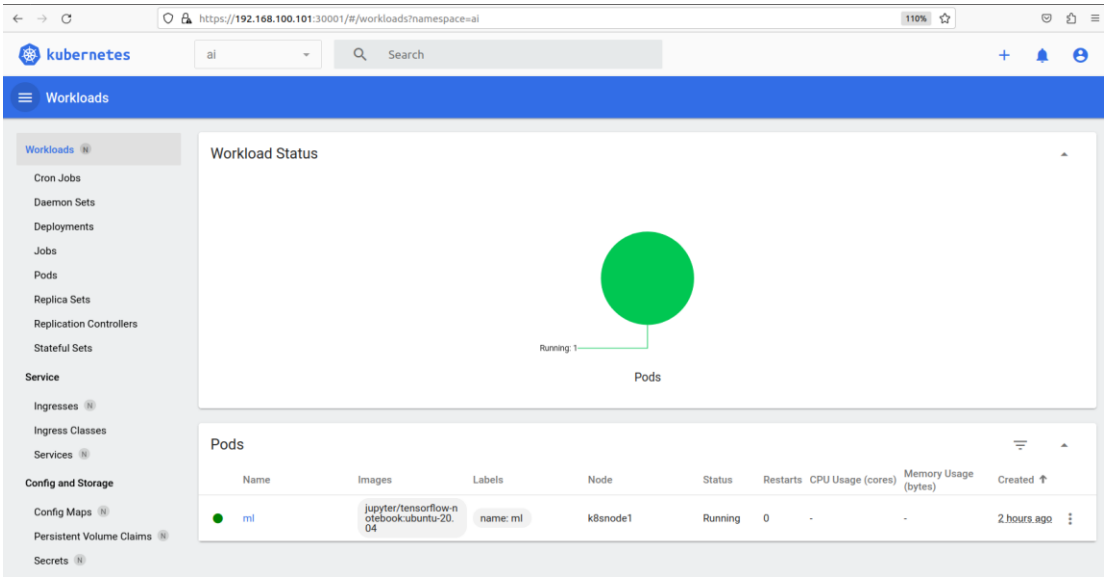

Figure 6-5-2-2: Kubernetes Dashboard Home Page

6.6 Experiment Implementation

This section will explain the codes of experiments in the platform by splitting them into the cell, introducing how codes work and what is the purpose. The complete codes can be found in **Appendix B**.

6.6.1 Logistic Regression Classification Experiment Implementation

The first step in building the experiment is to choose and import libraries. The following libraries are included:

1) matplotlib.pyplot: it is a module within the larger Matplotlib library, which is a widely used plotting library in Python. Matplotlib is designed to create static, interactive, and animated visualizations in Python. The pyplot module provides a simple and convenient interface for creating various types of plots and visualizations.

2) Datasets from sklearn: It provides a structured collection of data used for training, testing, and evaluating machine learning algorithms and models. In this paper, the Iris dataset from this library is chosen as the training and testing dataset.

3) LogisticRegression from sklearn.linear_model: It provides the logistic regression algorithm simply and conveniently. The user can directly call it using the API provided by Sklean.

4) DecisionBoundaryDisplay from sklearn.inspection: The sklearn.inspection module is typically used for tools related to model inspection and validation. This paper applies it to display the visualized figure of the decision boundary.

Table 6-6-1-1 shows the code for importing necessary libraries in the experiment.

Table 6-6-1-1 Codes of importing libraries

Code description: Import necessary libraries
Core codes:
import matplotlib.pyplot as plot
from sklearn import datasets
from sklearn.linear_model import LogisticRegression
from sklearn.inspection import DecisionBoundaryDisplay

Then, we need to load the Iris dataset. In this experiment, we choose the first two features for training so that the algorithm can distinguish the iris flower into various categories based on sepal length and sepal width. Table 6-6-1-2 shows the code for loading the iris dataset and setting training data.

Table 6-6-1-2 Codes of loading Iris dataset

Code description: Load the Iris dataset and set training data
Core codes:
dataset = datasets.load_iris()
X = dataset.data[:, :2]
Y = dataset.target

After that, we can call the logistic regression algorithm and train it with the training data. In the experiment, the default value of C is 1, and students are required to change it. The core code of this stage is shown in Table 6-6-1-3

Table 6-6-1-3 Codes of training LR

Code description: Load LR and train it with training data
Core codes: LogisticRegression = LogisticRegression(C=1) LogisticRegression.fit(X, Y)

The final step is to test the LR classifier and illustrate a visualized figure of the decision boundary of LR. The core code is shown in Table 6-6-1-4.

Table 6-6-1-4: Codes of displaying decision boundary

Code description: Display decision boundary
Core codes: fig, ax = plot.subplots(figsize=(4, 4)) DecisionBoundaryDisplay.from_estimator(LogisticRegression, X, ax=ax, response_method="auto", plot_method="pcolormesh", xlabel="sepal length", ylabel="sepal width",) plot.scatter(X[: , 0], X[: , 1], c=Y, edgecolors="black") plot.show()

Firstly, we create a figure and axes using the subplot() function. Then DecisionBoundaryDisplay.from_estimator () is called to create the decision boundary image on the axes. The parameter description is shown in Table 6-6-5.

Table 6-6-1-5: Parameters of DecisionBoundaryDisplay.from_estimator ()

Parameter	Description
x: {array-like, sparse matrix, dataframe}	Input data that should be only 2-dimensional.
ax: Matplotlib axes, default=None	Axes object to plot on.
label: str, default=None	Default label to place on the x-axis
label: str, default=None	Default label to place on the y-axis
plot_method: {'contourf', 'contour', 'pcolormesh'}, default='contourf'	Plotting method to call when plotting the response.

6.6.2 Decision Tree Classification Experiment Implementation

This experiment also imports the sklearn library to call the decision tree algorithm and iris dataset. In addition, the sklearn.metrics library is applied here to quickly calculate cores including f1_score, accuracy_score,precision_score, and recall_score in the confusion matrix. The first step is to prepare data as shown in Table 6-6-1-1.

Table 6-6-2-1 Codes of preparing data

Code description: Prepare necessary data

Core codes:

```
from sklearn import datasets, tree
from sklearn.metrics import f1_score, accuracy_score, precision_score,
recall_score
import random
dataset = datasets.load_iris()
x = dataset.data[:, :2]
y = dataset.target
n_samples = len(x)
```

the data in the Iris dataset is arranged in a sequential order based on the target data (types of iris flower) with the order 0, 1, 2, It's necessary to shuffle the data before

splitting it into the testing and training sets. Table 6-6-2-2 displays how to shuffle the data.

Table 6-6-2-2 Codes of shuffling data

Code description: Shuffle Iris dataset

Core codes:
id = [i for i in range(0,n_samples)]
random.shuffle(id)
x_copy = x.copy()
y_copy = y.copy()
for j in range(0,n_samples):
    x[j] = x_copy[id[j]]
    y[j] = y_copy[id[j]]

After that, we need to split the testing data and training data for the model. Students are supposed to change the ratio here to watch the various changes in the performance of the decision tree model and then analyze it and find the optimal ratio. The code for splitting data is shown in Table 6-6-2-3.

Table 6-6-2-3 Codes of splitting testing/training data

Code description: Split testing and training data
Core codes: ratio = 0.5 (default is 0.5) x_train = x[: int(ratio * n_samples)] y_train = y[: int(ratio * n_samples)] x_test = x[int(ratio * n_samples) :] y_test = y[int(ratio * n_samples) :]

Then, the decision tree algorithm is trained with the training data and the predicted labels of testing data. Table 6-6-2-3 displays the process. The "max_depth" parameter controls the maximum depth or levels that the tree can grow to during the training process. One of the main reasons to use the "max_depth" parameter is to prevent the decision tree from becoming overly complex and overfitting the training data.

Table 6-6-2-3 Codes of training decision tree and predicting

Code description: Train the model and predict labels
Core codes: decisionTree = tree.DecisionTreeClassifier(max_depth=3) decisionTree.fit(x_train, y_train) predictions = decision tree.predict(x_test)

Finally, the performance of the model is displayed in the form of a confusion matrix as shown in Table 6-6-2-4.

Table 6-6-2-4 Codes of displaying confusion matrix

Code description: Display confusion matrix
Core codes: accuracy = accuracy_score(y_test, predictions) precision = precision_score(y_test, predictions, average='weighted') recall = recall_score(y_test, predictions, average='weighted') f1 = f1_score(y_test, predictions, average='macro') print("Classification Metrics:") print(f"Accuracy:", accuracy) print(f"Precision:", precision) print(f"Recall:", recall) print(f"F1 Score:", f1)

At the end of the experiment, students are required to upload the confusion matrix and explain the ratio of splitting he set to the result file.

6.7 Implementation of Kubernetes Cluster

This section will introduce how this paper builds a Kubernetes cluster on a local machine and deploys applications to the cluster. The fundamental environment setup can be found in Chapter 6.1.

6.7.1 Create the Cluster

First of all, on the designated master node, use kubeadm to initialize the cluster. This involves setting up the control plane components, such as the API server, etc (key-value store), and controller manager. The Kubernetes installed on the master and worker node is version 1.22.4. After initializing the master node, join the worker node to the cluster by using kubeadm on a worker node. The final cluster is shown in Figure 6-7-1.

```
NAME       STATUS   ROLES                  AGE   VERSION
k8snode1   Ready    <none>                 15d   v1.22.4
ubuntu     Ready    control-plane,master   15d   v1.22.4
```

Figure 6-7-1: Kubernetes Cluster

6.7.2 Config Flannel

In this step, we need to make sure that nodes in the cluster can communicate and transmit data to each other. Therefore, flannel is deployed with the official .yaml configuration file (Figure 6-7-2-1) on the master node. Then, check if it is successfully deployed on the cluster. The result is shown in Figure 6-7-2-2.

```
serviceAccountName: flannel
initContainers:
- name: install-cni-plugin
 #image: flannelcni/flannel-cni-plugin:v1.0.1 for ppc64le (dockerhub limitations may apply)
  image: rancher/mirrored-flannelcni-flannel-cni-plugin:v1.0.1
  command:
  - cp
  args:
  - -f
  - /flannel
  - /opt/cni/bin/flannel
  volumeMounts:
  - name: cni-plugin
    mountPath: /opt/cni/bin
- name: install-cni
 #image: flannelcni/flannel:v0.16.1 for ppc64le (dockerhub limitations may apply)
  image: rancher/mirrored-flannelcni-flannel:v0.16.1
  command:
```

Figure 6-7-2-1: Flannel Configuration File

```
flannel.1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1450
        inet 10.244.0.0  netmask 255.255.255.255  broadcast 10.244.0.0
        inet6 fe80::30f2:1dff:fecc:9128  prefixlen 64  scopeid 0x20<link>
        ether 32:f2:1d:cc:91:28  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 378 overruns 0  carrier 0  collisions 0
```

Figure 6-7-2-2: Flannel

6.7.3 Config NFS Storage

The cluster uses a provisioner and storage class to achieve dynamically provisioning persistent volume and persistent volume claims based on the network file system. This subsection will introduce how to implement and configure these services.

6.7.3.1 Implement NFS Server

The master node should install the NFS server and create the shared directory. In the cluster, the shared directory is /home/kyokoz/data. After starting the NFS service, worker node 1 should verify if NFS is working as shown in Figure 6-7-3-1.

```
Export list for 192.168.100.100:
/home/kyokoz/data 192.168.100.0/24
```

Figure 6-7-3-1: Verify NFS on Worker Node 1

6.7.3.2 Implement NFS Client Provisioner

The cluster uses a provisioner to provision PV and PVC. To deploy provisioner, we need to create a service account called provisioner and use Role Based Access Control to make sure it gets the authorization to access the specific namespace (storage class, PV, NFS, and PVC). “RBAC.yaml” file [36] is used to deploy the provisioner. Figure 6-7-3-2-1 displays the core code of “RBAC.yaml”. In this file, we need to change the default namespace to where it deployed in our cluster. After creating the provisioner, configure the storage class using “Storage Class.yaml” (Figure 6-7-3-2-2). The final step is to use “Deployment.yaml” to deploy and install the provisioner and storage class as a resource of the cluster. In this file, the default NFS shared directory should be modified to the cluster NFS server IP address and directory as shown in Figure 6-7-3-2-3. All of these .yaml configuration file is downloaded from

the Kubernetes-nfs GitHub repository. Figure 6-7-3-2-4 shows the deployment result of the NFS client provisioner.

```yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: nfs-client-provisioner
  # replace with namespace where provisioner is deployed
  namespace: kube-nfs
---
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: nfs-client-provisioner-runner
rules:
  - apiGroups: [""]
    resources: ["nodes"]
    verbs: ["get", "list", "watch"]
  - apiGroups: [""]
    resources: ["persistentvolumes"]
    verbs: ["get", "list", "watch", "create", "delete"]
  - apiGroups: [""]
    resources: ["persistentvolumeclaims"]
    verbs: ["get", "list", "watch", "update"]
  - apiGroups: ["storage.k8s.io"]
    resources: ["storageclasses"]
    verbs: ["get", "list", "watch"]
  - apiGroups: [""]
    resources: ["events"]
    verbs: ["create", "update", "patch"]
```

Figure 6-7-3-2-1: Code of RBAC.yaml

```yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: nfs-storage-class
  namespace: kube-nfs
provisioner: nfs-client-provisioner
parameters:
  archiveOnDelete: "false"
```

Figure 6-7-3-2-2: Code of Storage Class.yaml

```yaml
    env:
      - name: PROVISIONER_NAME
        value: nfs-client-provisioner
      - name: NFS_SERVER
        value: 192.168.100.100 #replace by the IP address of NFS server
      - name: NFS_PATH
        value: /home/kyokoz/data #replace by the path of NFS directory
volumes:
  - name: nfs-client-root
    nfs:
      server: 192.168.100.100 #replace by the IP address of NFS server
      path: /home/kyokoz/data #replace by the path of NFS directory
```

Figure 6-7-3-2-3: Core Code of Deployment.yaml

```
root@ubuntu:/home/kyokoz# kubectl get pods -n  kube-nfs
NAME                                     READY   STATUS    RESTARTS      AGE
nfs-client-provisioner-57f7557cd4-r8rks   1/1     Running   2 (42m ago)   4d2h
```

Figure 6-7-3-2-4: NFS-Client-Provisioner

6.7.3.3 Implement PV and PVC

After the provisioner is deployed, PV and PVC should be implemented to provide storage for containers. When writing a configuration file, we need to choose the same storage class and NFS for PV and PVC so that PVC can correctly be bound to the corresponding PVC. Figure 6-7-3-3-1 and Figure 6-7-3-3-2 show the configuration code of PV and PVC. The result of deploying PV and PVC is illustrated in Figure 6-7-3-3-2.

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: nfs-pv
  namespace: kube-nfs
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteMany
  storageClassName: nfs-storage-class
  nfs:
    path: /home/kyokoz/data
    server: 192.168.100.100
```

Figure 6-7-3-3-1: Code of PV.yaml

```yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: nfs-pvc
  namespace: kube-nfs
spec:
  storageClassName: nfs-storage-class
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 2Gi
```

Figure 6-7-3-3-1: Code of PVC.yaml

```
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS
nfs-pv                                     1Gi        RWX            Retain           Available
pvc-eef30738-9ba5-4d66-b74c-c1a7bff32911   2Gi        RWX            Delete           Bound
```

Figure 6-7-3-3-2: PV and PVC

6.7.4 Implement Pods

Creating a pod also involves defining a pod specification using a YAML configuration file and then using Kubectl to apply it to the cluster.

In the YAML file, it should indicate the image the pod uses, the name and namespace, and the PVC and NFS it uses as the storage. Besides this, to make it can be accessed from outside the cluster, the pod is bound to the port of the host machine so that when the back-end receives the request from a user, it can get access to the pod and return the IP address of it to the client to open the pod and provide the experiment environment for students. The configuration file is shown in Figure 6-7-4-1 and the implementation of the pod used for machine learning experiment is displayed in Figure 6-7-4-2.

```
spec:
  containers:
  - name: ml
    image: jupyter/tensorflow-notebook:ubuntu-20.04
    ports:
      - containerPort: 8888
        hostPort: 10000
        protocol: TCP
    volumeMounts:
      - name: nfs-pvc
        mountPath: /home/jovyan/work
  volumes:
    - name: nfs-pvc
      nfs:
          path: /home/kyokoz/data/Supervised Learning:Logistic Regression Classifier
          server: 192.168.100.100
```

Figure 6-7-4-1: Code of Pod Configuration File

```
root@ubuntu:/home/kyokoz/pods_yaml# kubectl describe pod ml -n ai
Name:         ml
Namespace:    ai
Priority:     0
Node:         k8snode1/192.168.100.101
Start Time:   Wed, 16 Aug 2023 06:47:33 -0700
Labels:       name=ml
Annotations:  <none>
Status:       Running
IP:           10.244.1.27
IPs:
  IP:  10.244.1.27
Containers:
  ml:
    Container ID:   docker://280192ebbc42c514ca76f90234ba64bc144d1142015bb
    Image:          jupyter/tensorflow-notebook:ubuntu-20.04
    Image ID:       docker-pullable://jupyter/tensorflow-notebook@sha256:2
    Port:           8888/TCP
    Host Port:      10000/TCP
    State:          Running
      Started:      Wed, 16 Aug 2023 06:47:34 -0700
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /home/jovyan/work from nfs-pvc (rw)
```

Figure 6-7-4-2: ml Pod

6.8 Implementation of Database

After creating the database using SQLite, to make it connect with the back-end and platform, Flask is required to register it in model.py by using a database library. One of the popular database libraries for Flask is SQLAlchemy. Figure 6-8 displays the code for defining tables as Python classes. Then, the database is available in routes.

```
class Student(db.Model):
    __tablename__ = "Student"
    student_ID = db.Column(db.String(50),primary_key=True,unique=True)
    name = db.Column(db.String(20))
    password = db.Column(db.String(20))
    teacher_ID = db.Column(db.String(50), db.ForeignKey('Class.teacher_ID'))
    class_ID = db.Column(db.String(50), db.ForeignKey('Class.class_ID'))

class Teacher(db.Model):
    __tablename__ = "Teacher"
    teacher_ID = db.Column(db.String(50),primary_key=True,unique=True)
    school_ID = db.Column(db.String(20),db.ForeignKey('School.school_ID'))
    password = db.Column(db.String(20))
    course_ID = db.Column(db.String(50), db.ForeignKey('Course.course_ID'))
```

Figure 6-7-4-2: Define tables as Python Classes

Chapter 7
Testing and Evaluation

This section discusses the software testing and evaluation process. The discussion focuses on whether the system meets the user's requirements and achieves the expected functionality and performance.

7.1 Testing

The system consists of three tier parts (front-end, back-end, Serverless services) and has three functional modules including the educator module, student module, and platform administrator module. Each module will be tested to make sure it meets the requirements of the stakeholders. Then the non-functional requirements test is going to check the security, usability, and other aspects that are essential for the system's effectiveness and user experience.

7.1.1. Testing Environment

Tables 7.1 and 7.2 show the hardware and software environments used in this test for the system.

CPU	4 Cores
RAM	6 GB
Hard Disk	50 GB

Table 7-1: Hardware Environment

Operating System	Ubuntu:20.04
Python	3.9
Database	SQLite v3.12.99
Browser	Chrome v116.0.5845.97
Kubernetes	v1.22.4

Table 7-2: Software Environment

7.1.2 Functional Requirements Test

Table 7-1-2 shows the test case table of the functional requirements test of the platform.

Table 7-1-2: Functional Requirements Test Case

Platform Name	Online AI Experiment Platform Based on Serverless Architecture			
Aimed users	Platform Administrator, Educator, Student			
Test functions	Platform Administrator Module Functions, Educator Module Functions, Student Module Functions			
Test purpose	Validate whether the platform meets the user's requirements as a design			
Number	Test Name	Input	Operation Steps	Test Result
1	Login the system	Username and password	1. Enter a different username and password 2. Press “Login” button	PASS
2	Change Password	Old Password and New Password	1. Press “Change Password” button 2. Enter an old and new password 3. Press “Submit” button	PASS
3	View Student List	None	1. Press “View student list” button	PASS
4	View Experiment List	None	1. Press the “My Experiments” button	PASS
5	Delete an Experiment	Select an Experiment	1. Press the “delete” button in the experiment list	PASS
6	Check the Result Uploaded by the Student	Select an Experiment	1. Press the “ result” button in the experiment list	PASS
7	Create a New Experiment	Experiment Files and Information	1. Press the “Create a new experiment” button 2. Enter experiment information and select files 3. Press “submit” button	PASS

8	Check Comments	None	1. Press the “Check comments” button	PASS
9	Log out	None	1. Press the “Log out” button	PASS
10	Do an Experiment	Select an experiment	1. Login to the Student system 2. Start an experiment	PASS
11	Upload the Result	Experiment Result	1. Copy the result 2. Paste it to the “result. ipynb” file	PASS
12	View Educator List	None	1. Press the “View Educator List” button	PASS
13	Open Kubernetes Dashboard	None	1. Press the “Monitor Kubernetes Cluster” button	PASS
14	Comment on the Experiment	Select an Experiment	1. Press the “Comment” button in the experiment list 2. Enter the comment	PASS
15	Check If the Experiment Code File is Available	Experiment Code File	1. Open the experiment code file 2. Run it in IDE 3. Change parameters and run it again 4. Compare the results	PASS
16	Check the Communication between Nodes	None	1. Choose one node 2. Ping another node's IP address	PASS
17	Check If NFS is Working	A Test File	1. Put the test file in the shared directory on the worker node 2. Check the shared directory in the master node	PASS
18	Check the Cluster	Kubectl Commands	1. Open terminal 2. Enter “Kuberctl get nodes”	PASS
19	Check Whether Pods are Running	Kubectl Commands	1. Open terminal 2. Enter “Kubectl get pods -n ai” 3. Check the status of pods	PASS

20	Check the Validity of Database	User Data	1. Open the database 2. Add and delete some data	PASS

7.1.3 Non-Functional Requirements Test

This section will test the non-functional requirement including the user experience and the security and performance. Table 7-1-3 shows the test cases applied in this test. It focuses on the requirements and potential risks of the platform during its use. The test helps to guarantee the security, usability, scalability, and integrity of the system.

Table 7-1-3: Non-Functional Requirements Test Case

Platform Name	Online AI Experiment Platform Based on Serverless Architecture			
Aimed users	Platform Administrator, Educator, Student			
Test functions	UI, Security, User Experience			
Test purpose	Check the non-functional requirements of the system			
Number	Test Name	Input	Operation Steps	Test Result
1	Grey Test	Grey Test Code	1. Open the platform 2. Open the developer tool to enter the grey test code	PASS
2	Authorisation Bypass Test	Empty Session	1. Clean the session 2. Enter the system using the URL 3. Check if can successfully enter the system	PASS
3	Check Data Encryption	Database	1. Open the database 2. Check if the passwords are encrypted	PASS
4	Check If UI Can be Easily Understood	None	1. Open the platform 2. Manually operate some functions 3. Check if there are logistic errors or text display error	PASS
5	SQL Injection	SQL Injection Code	1. Open the Login page 2. Enter SQL injection code 3. Check if it can log in to the system	PASS

7.1.4 Performance Test

This paper uses Apache Bench to simulate 100 requests from 10 concurrent users to the platform server and Serverless service. Test results are as Figure 4-15 and Figure 4-16. The average response time of the platform server is 14.085ms, and that of the Serverless service is 40.923ms (Figure 7-1-4-1 and 7-1-4-2). According to test results, this paper confirms that the platform can provide stable Serverless services for multiple users.

```
Concurrency Level:      10
Time taken for tests:   0.141 seconds
Complete requests:      100
Failed requests:        0
Total transferred:      390300 bytes
HTML transferred:       374700 bytes
Requests per second:    709.98 [#/sec] (mean)
Time per request:       14.085 [ms] (mean)
Time per request:       1.408 [ms] (mean, across all concurrent requests)
Transfer rate:          2706.09 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.1      0       0
Processing:     3   13   6.8     11      36
Waiting:        3   13   6.8     11      36
Total:          4   13   6.8     11      36

Percentage of the requests served within a certain time (ms)
  50%     11
  66%     12
  75%     13
  80%     14
  90%     24
  95%     34
  98%     36
  99%     36
 100%     36 (longest request)
```

Figure 7-1-4-1: Average Response Time of Platform Server

```
Concurrency Level:      10
Time taken for tests:   0.409 seconds
Complete requests:      100
Failed requests:        0
Total transferred:      465200 bytes
HTML transferred:       448800 bytes
Requests per second:    244.36 [#/sec] (mean)
Time per request:       40.923 [ms] (mean)
Time per request:       4.092 [ms] (mean, across all concurrent requests)
Transfer rate:          1110.12 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.1      0       0
Processing:    12   30  13.0     29     123
Waiting:        4   20  12.6     19     117
Total:         13   30  13.1     29     123

Percentage of the requests served within a certain time (ms)
  50%     29
  66%     34
  75%     35
  80%     37
  90%     41
  95%     44
  98%     53
  99%    123
 100%    123 (longest request)
```

Figure 7-1-4-2: Average Response Time of Serverless Services

7.2 Evaluation

7.2.1 Result Evaluation

As described in Chapter 1, the purpose of this project is to design and deploy a web-based AI experimentation platform for universities engaged in AI education. This platform utilizes a serverless architecture to provide an experimental environment and other cloud services. It enables better interaction between teachers and students in the context of online education, facilitating AI experiment-based teaching and offering a convenient and cost-effective service for educational institutions.

Based on the testing results, the final system proposed can fulfill the majority of functional and non-functional requirements. The implemented features include all functionalities of the student module, educator module, and platform administrator module. Additionally, during the deployment and testing phases, this paper addresses non-functional requirements such as security, scalability, and usability. Measures such as load management, discussion of potential risks associated with cloud services, and testing for common web application attacks are carried out to experiment with the non-functional requirements.

Due to time constraints and gaps in knowledge, a few non-core requirements in this paper were not fully met. However, the platform can still be considered successfully deployed.

7.2.2 Process Evaluation

At the beginning of the project, the learning of Serverless technology and background investigation proceeded according to the plan. However, due to the initial exposure to Kubernetes and Docker, a significantly greater amount of time than anticipated was invested in system design, deployment, and debugging. Utilizing the Flask framework for development resulted in a shorter front-end development timeline than expected. Simultaneously, owing to the substantial volume of backend code and design, the backend development phase took longer than projected.

During the actual deployment of the system, testing and deployment occurred concurrently, resulting in a relatively shorter testing phase. In Figure 7-2-1, we compare the planning progress (blue) with the actual development progress (green) to visually illustrate the difference.

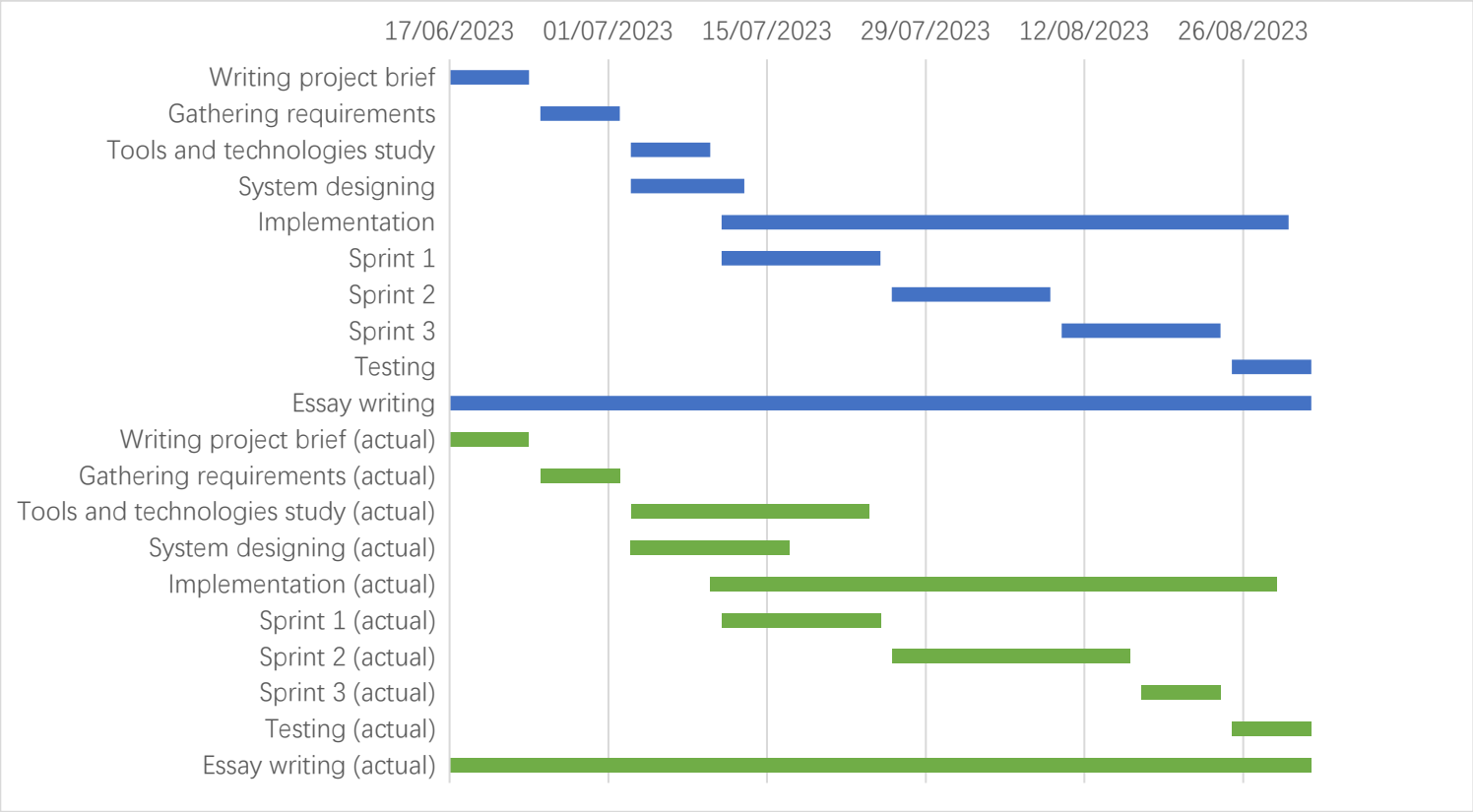

Figure 7-2-1: Gantt Chart Comparison

Chapter 8
Conclusion and Future Work

This chapter provides a conclusion for this project and discusses potential improvement and future work of it.

8.1 Conclusion

This paper is based on the research of AI education, and combines online education and Serverless, designing and implementing an online AI experiment platform based on Serverless architecture facing universities which provides a method to set online engineering experiment courses. The platform uses Docker container technology as an AI experiment environment. The Docker image integrates libraries of Tensorflow which meets most of the requirements of doing AI experiments. Kubernetes is also applied to achieve container management, building Kubernetes clusters to offer Serverless services. On top of that, this paper designs a database according to the practical education situation in universities, considering many practical applications to optimize platform details.

The platform provides two supervised learning experiments for users. According to the test, the platform and Kubernetes cluster offer the experiment environment and necessary libraries. Even though there are still several limitations in the platform, the test result shows that the platform can provide AI experiment services for multiple users.

8.2 Future Work

This section discusses future work including cyber security and implementation on a cloud platform that I am planning to implement after the submission.

8.2.1 Cyber Security

Service Level Agreement (SLA) is a contractual agreement between a service provider and a customer that outlines the specific terms, expectations, and quality metrics for the services being provided. SLAs are commonly used in cloud computing to define the level of service that the customer can expect to receive and to ensure that the service provider meets those expectations. In this paper, I, as the service provider am responsible for providing SLA to users. In SLA, we need to discuss the platform

performance, provided functionalities, and provided skills and should include a detailed description of the services.

8.2.2 Improved Persistent Storage

To achieve persistent storage, when the Kubernetes cluster starts a container, it mounts a local directory as the container working directory. Users' operation in the container is saved in local files. This design leads to a problem that if the local machine cannot work or managers delete those files accidentally, the stored platform resources will be destroyed. An effective method to improve the file system is to apply cloud storage service in the platform such as buying private cloud disk service from Google Drive to store files that are used in experiments.

8.2.2 External Access to the Platform

In this paper, containers are bound to ports of the local machine so that users can access them outside the Kubernetes cluster. But the number of ports is finite and it is impossible to dynamically and automatically distribute ports to containers. On the other hand, users can not access Serverless services from external machines by entering the IP addresses of containers in browsers. Implementing Serverless services on a Serverless cloud environment provided by cloud service companies like Amazon or Tencent instead of local machines can solve this issue. Once users want to start containers, the Serverless cloud environment will automatically distribute a unique IP address to each container, therefore users can access containers by entering the IP address of containers in browsers from an external machine.

List of References

[1] Taibi, Davide. Serverless Computing-Where Are We Now, and Where Are We Heading[J]. IEEE Software, 2021, 38(1): 25-31.

[2] UNESCO [66698] ED-2022/FLI-ICT/K-12. K-12 AI curricula: a mapping of government-endorsed AI curricula, 2022.

[3] State Council [2017] NO.35. the development plan for a new generation of artificial intelligence [Z].

[4] MOOC. https://www.mooc.org/

[5] Stanford. CS221: Artificial Intelligence: Principles and Techniques. [Online] Available https://stanford-cs221.github.io/summer2023/, Summer 2022-2023.

[6] Stanford. CS231n: Convolutional Neural Networks for Visual Recognition. [Online] Available http://cs231n.stanford.edu/2018/, Spring 2018.

[7] Jupyter Notebook. https://jupyter.org/

[8] Python. https://www.python.org/

[9] PyCharm. https://www.jetbrains.com/pycharm/

[10] SQLite. https://www.sqlite.org/index.html

[11] Microsoft. Network File system.2022. [Online] Available:

https://learn.microsoft.com/en-us/windows-server/storage/nfs/nfs-overview

[12] Docker. https://www.docker.com/

[13] Kubernetes. https://kubernetes.io/

[14] Ubuntu. https://ubuntu.com/

[15] VMware Workstation. https://www.vmware.com/uk/products/workstation-pro.html

[16] Apache Bench. https://httpd.apache.org/docs/2.4/programs/ab.html

[17] GitHub. https://github.com/

[18] Airfocus. What Is A User Story? Definition, History, Advantages, and Disadvantages. [Online] Available: https://airfocus.com/glossary/what-is-a-user-story/.

[19] Matthew Martin. What is a Functional Requirement in Software Engineering? Specification, Types, Examples, 2021. [Online] Available: https:// www.guru99.com/functional-requirement-specification-example. html.

[20] Dean Leffingwell. Nonfunctional Requirements, 2021.[Online] Available

https://scaledagileframework.com/nonfunctional-requirements/

[21] ATLASSIAN. What is the Agile methodology? [Online] Available: https://www.atlassian.com/agile

[22] Ajay Sarangam. 5 Important Types Of Agile Methodology, 2021. [Online] Available: https://www.jigsawacademy.com/blogs/product-management/types-of-agile-methodology

[23] Adobe Workfront. What is a Sprint in Agile? — Adobe Workfront. [Online] Available: https://www.workfront.com/project-management/ methodologies/scrum/sprints.

[24] Johan Karlsson. Agile Product Backlogs. [Online] Available: https://www.perforce.com/resources/hns/agile-product-backlog-basics

[25] Flask. [Online] Available: https://flask.palletsprojects.com/en/2.3.x/

[26] Kubernetes. Kubernetes Dashboard. [Online] Available: https://kubernetes.io/docs/tasks/access-application-cluster/web-ui-dashboard/

[27] Jalam Raj Rohit (India). Turing Programming Series, Serverless Architecture Application Development in Python Implementation Press, 2019:13-14.

[28] Kubernetes. Kubernetes Cluster. [Online] Available: https://kubernetes.io/docs/concepts/architecture

[29] Kubernetes. Kubeadm, Kubectl, Kubelet. [Online] Available: https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/

[30] Flannel. [Online] Available: https://github.com/flannel-io/flannel

[31] Kubernetes. Volumes. [Online] Available: https://kubernetes.io/docs/concepts/storage/volumes/

[32] Liu, Hongyu & Lang, Bo. (2019). Machine Learning and Deep Learning Methods for Intrusion Detection Systems: A Survey. Applied Sciences. 9. 4396. 10.3390/app9204396.

[33] Scikit-learn [Online] Available: https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

[34] Dan Eden.Animate.css [Online] Available:http://daneden.me/animate

[35] DockerHub. jupyter/tensorflow-notebook: Ubuntu:20.04 [Online] Available:https://hub.docker.com/r/jupyter/tensorflow-notebook

[36] Kubernetes-NFS GitHub. NFS Configuration Deploy file. [Online] Available: https://github.com/kubernetes-retired/external-storage/tree/master/nfs-client/deploy

Appendix A
Secret Key and Token

- Kubernetes Dashboard Token:
  eyJhbGciOiJSUzI1NiIsImtpZCI6IkVERDdfcndnUnBfVVpKM0h3QmEtNUt3ZFRV
  VmN5aE9YUG9BTEcxVEU1UlEifQ.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2Vh
  Y2NvdW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2
  UiOiJrdWJlLXN5c3RlbSIsImt1YmVybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc
  2VjcmV0Lm5hbWUiOiJkYXNoYm9hcmQtYWRtaW4tdG9rZW4tNWx4NjQiLCJrd
  WJlcm5ldGVzLmlvL3NlcnZpY2VhY2NvdW50L3NlcnZpY2UtYWNjb3VudC5uYW
  1lIjoiZGFzaGJvYXJkLWFkbWluIiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3V
  udC9zZXJ2aWNlLWFjY291bnQudWlkIjoiMjZkOWY5YTYtZjdlYy00MDI3LTk0ZjIt
  Nzk5ZDk5Nzg0MWY5Iiwic3ViIjoic3lzdGVtOnNlcnZpY2VhY2NvdW50Omt1YmUt
  c3lzdGVtOmRhc2hib2FyZC1hZG1pbiJ9.EjLb7lrsr6VDoQ5ZmlojuIyXnEpVLbI3U
  ReHebq4UPGhiZg08F8a8vhMSHtuiKCgh0BJGY60lCCmrQD7dt2rcAQ8wywgS
  Ppm_Qf8gjugLOQA07CWkjkukYo5FkglhwfOGG98Z4SC89QpNgQ5R1O0umdg
  KG82SgSf9X26DNLQI8CyvgYnXeUBUPavq5Gst_7jLR5p3euJfDOll8fkCZy5dvK
  1xGqK9Lsf38gbftCb0mcurjEP16jHfR1Br8_vN10y8ASIW3xY_HoXdjy1TJ8eHIPh
  PiPeBRnkxKj8JqGOt0RIY83yDrRLQVpxe5H2pSdQ_8hFSGF6YGgSidruFw

- LRC Container Token: 75c6ad80871f76847a41f06b8d6f2738a0bfa6f3c2bace35

- Decision Tree Container Token:
  57dfef6d124ccb8571362d32c5d876278860b895a39ff204

Appendix B
Experiment Codes

B.1 Logistic Regression Classifier Experiment Code

Table B.1 Codes of Logistic Regression Classifier Experiment

Code description: Logistic Regression Classifier Experiment Codes

Codes:

```
import matplotlib.pyplot as plot
from sklearn import datasets
from sklearn.linear_model import LogisticRegression
from sklearn.inspection import DecisionBoundaryDisplay
dataset = datasets.load_iris()
x = dataset.data[:, :2]
y = dataset.target
LogisticRegression = LogisticRegression(C=1)
LogisticRegression.fit(X, Y)
fig, ax = plot.subplots(figsize=(4, 4))
DecisionBoundaryDisplay.from_estimator(
    LogisticRegression,
    X,
    ax=ax,
    response_method="auto",
    plot_method="pcolormesh",
    xlabel="sepal length",
    ylabel="sepal width",
)
plot.scatter(X[: , 0], X[: , 1], c=Y, edgecolors="black")
    plot.show()
```

B.2 Decision Tree Experiment Code

Table B.2 Codes of Decision Tree Experiment

Code description: Decision Tree Experiment Codes

Codes:

```
from sklearn import datasets,  tree
from sklearn.metrics import  f1_score, accuracy_score, precision_score, recall_score
import random

dataset = datasets.load_iris()
x = dataset.data[:, :2]
y = dataset.target
n_samples = len(x)
id = [i for i in range(0,n_samples)]
random.shuffle(id)
x_copy = x.copy()
y_copy = y.copy()
for j in range(0,n_samples):
    x[j] = x_copy[id[j]]
y[j] = y_copy[id[j]]
ratio = 0.75
print ("samples:", n_samples)

#train data
x_train = x[: int(ratio * n_samples)]
y_train = y[: int(ratio * n_samples)]
print ("train data shape:", x_train.shape)
```

```
#test data
x_test = x[int(ratio * n_samples) :]
y_test = y[int(ratio * n_samples) :]
print ("test data shape:", x_test.shape)

decisionTree = tree.DecisionTreeClassifier(max_depth=3)
decisionTree.fit(x_train, y_train)
predictions = decisionTree.predict(x_test)

print ("predictions:", predictions)
print ("test data:", y_test)
accuracy = accuracy_score(y_test, predictions)
precision = precision_score(y_test, predictions, average='weighted')
recall = recall_score(y_test, predictions, average='weighted')
f1 = f1_score(y_test, predictions, average='macro')

print("Classification Metrics:")
print(f"Accuracy:", accuracy)
print(f"Precision:", precision)
print(f"Recall:",    recall)
print(f"F1 Score:",    f1)
```

Appendix C
Design Archive

The design archive submitted includes the flowing files:

- experiment_files: It contains the two experiment files (code file, result file)
- figures&tables: It contains all the figures and tables in the dissertation
- Kubernetes_config: It contains Kubernetes cluster configuration files including flannel, k8s dash board, pods yaml and pv, pvc, sc.
- Platform: It contains the front-end, back-end code files and database