SCHOOL TEACHER-SUBJECT ALLOCATION MANAGEMENT SYSTEM

DENNIS YAP JIAN YUAN

A project report submitted in partial fulfilment of the requirements for the award of Bachelor of Software Engineering with Honours

Lee Kong Chian Faculty of Engineering and Science
Universiti Tunku Abdul Rahman

April 2024

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature :

Name : Dennis Yap Jian Yuan

ID No. : 20UEB03334

Date : 25/04/2024

APPROVAL FOR SUBMISSION

I certify that this project report entitled **“SCHOOL TEACHER-SUBJECT ALLOCATION MANAGEMENT SYSTEM”** was prepared by **DENNIS YAP JIAN YUAN** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Software Engineering with Honours at Universiti Tunku Abdul Rahman.

Approved by,

Signature :

Supervisor : Too Chian Wen

Date : 25/04/2024

Signature : -

Co-Supervisor : -

Date : -

ACKNOWLEDGEMENTS

I would like to thank everyone who had contributed to the successful completion of this project. I would like to express my gratitude to my research supervisor, Dr. Too Chian Wen for her invaluable advice, guidance, and her enormous patience throughout the development of the research.

In addition, I would also like to express my gratitude to my loving parents and friends who had helped and given me encouragement.

ABSTRACT

The timetable systems for allocating teachers to subjects have improved from only digitalizing the management of timetable data to fully automating the timetabling process. However, the timetabling solutions available in the market are still having some shortcomings, leading to various issues that require solutions. So, the primary objective of this project is to identify the existing problems in the similarly School Teacher Subject Allocation Management Systems (STSAMS) used in public schools. Then, the project aims to propose optimal solutions to enhance these systems, solving any weaknesses or issues they may currently face. This project will build STSAMS as a web-based application in React.js that is equipped with API server, database, and the allotment algorithm with the Scrum methodology. The algorithm that will be used is Genetic Algorithm which has NP level of difficulty. So, requirement gathering using multiple methods like interviews and system analysis will be performed the narrow down the constraints used in the algorithm, while also lowering the complexity of the Genetic Algorithm to reduce the time complexity. Not just that, Genetic Algorithm, a meta-heuristic is also used to solve the clashing problem faced by existing system using deterministic algorithm. The STSAMS in this project is evaluated by using user acceptance test and unit test. The UAT results showed that STSAMS reached 75% user satisfaction level when the users are using the application.

TABLE OF CONTENTS

CHAPTER

CHAPTER 1

INTRODUCTION

1.0 Introduction

Living in a Fast-Paced World, the human population is increasingly adapting to the automation methodology of solving problems optimally and sparking innovations. By taking an example from the 4th industrial revolution, it utilizes robots to automatically produce goods. This approach can be proven to improve productivity and provide economic benefits (Trauth-Goik, 2021). Not just that, big data analytics are also one of the major components of Industry 4.0. It allows businesses to view trends and patterns from unstructured data. From this processed information, companies can prepare in advance to prevent risk and take the initiative to solve problems that are going to happen (Minellim, 2013). To further illustrate, problems can be as simple as dropping sales. The companies will be aware of the issues so that they are able to strategize and make decisions to mitigate or completely avoid loss of profit with these trends. Despite the benefits that automation provides, there is still a minority who prefer the old ways of completing tasks such as manual document filing, although it takes a lot of time to retrieve desired information (MESHDS, 2021).

By referring to the previously mentioned statement, there are still a lot of management systems that adopt semi-automated models. For example, the system that makes use of machinery by integrating with human labor. Its purpose is to act as a catalyst to speed up the work of an employee (Oosthuizen, 2022; jacqueline, 2022). Considering the School Teacher Subject Allocation Management System (STSAMS), its traditional form of allocation is dependent on the system administrators. This causes an issue to arise in which the teacher doesn't have a role in affecting the teaching of subjects (Jain, 2019). For example, the system administrator still has to manage the allotment manually as there is still a lack of automation even though the traditional form of STSAMS provides digitalization for its user. However, this is proven wrong by the rise of embedding algorithms into the STSAMS. For example, the most widely used algorithm for timetabling issues is a genetic algorithm, a meta-heuristic algorithm. The genetic algorithm is used due to its flexible nature as its design depends on its genetic encoding which is the smallest unit of the algorithm.

In a nutshell, the main direction of this project is to scout out the problems of existing STSAMS in the market and in use by public schools. Then, an optimal solution can be proposed to improve the systems in this study, while also solving its weaknesses or issues.

1.1 Background of The Problem

1.1.1 Literature Review

The problems are defined by studying similar STSAMS on the internet as there will be a possibility in which there is not a solution yet to be introduced as a resolution specifically for the allocation between teachers and subjects. For instance, by comparing the ideality and reality, weaknesses of similar systems can be analyzed to determine the issues that will lower the workflow efficiency and cause inconvenience to the users. Following that, solutions can be implemented to solve the problems identified from the existing systems.

1.1.2 School System Analysis

To get an insight into the real trend, the STSAMS of a school is studied and analyzed. As an illustration, questionnaires can be distributed to the teachers to come out with a statically proven problem faced by the teachers when using the school's STSAMS. Other than that, an interview can be done with the representative of the school to acquire the workflow on how the school manages the allotment of teachers and subjects. From that, the workflow then can be analyzed to find out the flaws so that it can be enhanced.

1.2 Problem Statements

1.2.1 Problem Statement 1

1.2.1.1 Lack of Teacher’s Preferences

Around the world today, people seek to have the freedom to choose; the power to make decisions from multiple courses of action before them (Averill, 1973; von Neumann & Morgenstern, 1944; Botti et al., 2023). The whole purpose of choice freedom is to let someone be in control, to express free will, and without the influence of outsiders to decide their options (Deci & Ryan, 1985; Wertenbroch et al., 2020). Despite that, the current existing School Teacher Subject Allocation Management System (STSAMS) such as Free Evolutionary Timetabling (FET) **does not consider the role of teachers to affect the outcome of the generated timetables**. In the aforementioned systems, the system administrators are only the ones that can decide teachers’ teaching subjects based on their expertise. So, this problem statement considers what if the teachers have a more preferred subject to teach because their teaching performance is affected by dissatisfaction and doubt about their ability to teach the allocated subjects (Ayeni & Amanekwe, 2018).

1.2.1.2 Proposed Solution

As an illustration, a teacher can teach English, Mathematics, and History subjects, but they have an ordered preference list of History, English, and Mathematics in descending order. So, a new constraint regarding the consideration of teacher’s subject preferences will be added to the implementation of the STSAMS managing algorithm, as a means to allow the teachers to have a higher chance to teach their favored subjects. Instead of only allowing the teacher to check for their allotment, functionality to decide their preferred subjects can be implemented in the teacher’s view in the system.

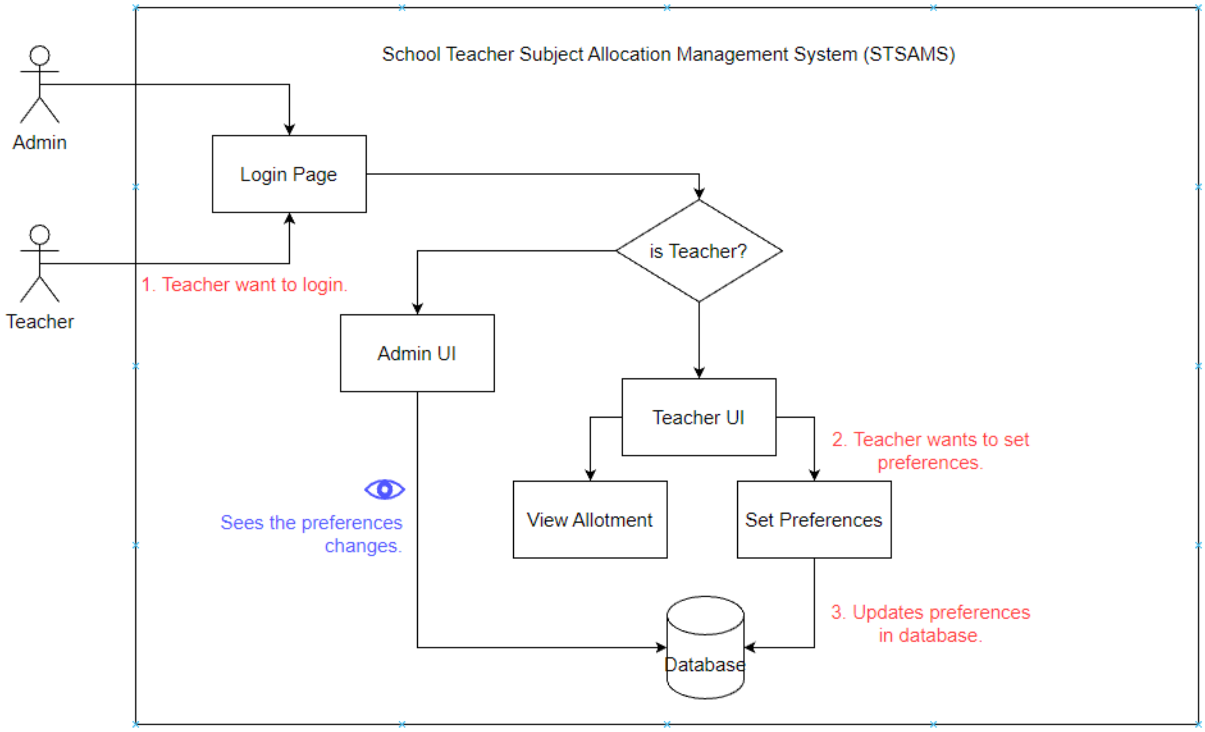

School Teacher Subject Allocation Management System (STSAMS)
Admin
Teacher
Login Page
is Teacher?
1. Teacher want to login.
Admin UI
Teacher UI
2. Teacher wants to set preferences.
View Allotment
Set Preferences
Sees the preferences changes.
3. Updates preferences in database.
Database

Figure 1.2.1.2.1: Process of setting preferences.

As shown in Figure 1.2.1.1, firstly, the STSAMS is designed to enable the system to get accessed from two sides such as the administrator's login and teacher's login. The users will have an account to log into the system. After the users log into the system as a teacher, the system will display the teacher's user interface instead of the administrators. Then, the user interface will include the options to choose between viewing allotment results or setting the teaching preferences. Lastly, the saved preferences will be updated in the database, so that the preference of each teacher can be seen and considered by the system administrator during the allotment process.

1.2.2 Problem Statement 2

1.2.2.1 Not Developed into Web Application

Furthermore, it is becoming more common that organizations to make use of web applications to run their business operations. This is due to the alluring benefits that it can provide. For example, web applications allow their users to access the system concurrently as it supports multi-user. Not just that, users can access the applications with different platforms such as desktops, laptops, and mobiles which improves flexibility and mobility; the users can access the system anytime and anywhere (SearchSoftwareQuality, n.d). In spite of that, **there are still a lot of existing systems such as FET that are running as local applications.** For STSAMS, this problem will affect the accessibility of its users including the teachers and system administrators. This is because local applications require the system administrators to save their changes on their computer as project files which causes them to be reliant and dependent on the file saved on the device. Transfer of files is required when the system administrators want to do their allocation work on other devices. Not to mention, sharing the allocated subject with the teacher is difficult as the teachers need to get access to the device that is installed with the system in order to view the subjects allocated to them. This is due to the fact that the data is not stored on a central server.

1.2.2.2 Proposed Solution

A web application can be developed for the STSAMS using the client-server architecture to support multi-user accessing, so that information can be shared between the system administrator and teachers. This is actually because of the client-server model that allows multiple users to get access to the same database simultaneously (Oluwatosin, 2014).

As can be seen in Figure 1.2.2.1 (Hamid, Abdulrahman & Khamees, 2020), there are a total of 3 layers in the client-server architecture. For further illustration, the client layer will be the devices that are being used by the teachers and system administrators to get access to the application layer. This layer allows multiple users to access the STSAMS at the same time. In addition, the application server layer of the architecture enables the teacher and system administrator to do CRUD operations on the central server and the changes will be reflected on each user’s side so that the data and information being accessed by users are the same and unified. However, permission must be granted to the user wisely to ensure data integrity. For example, teachers are only allowed to view the allotment result but are not allowed to modify data in the database.

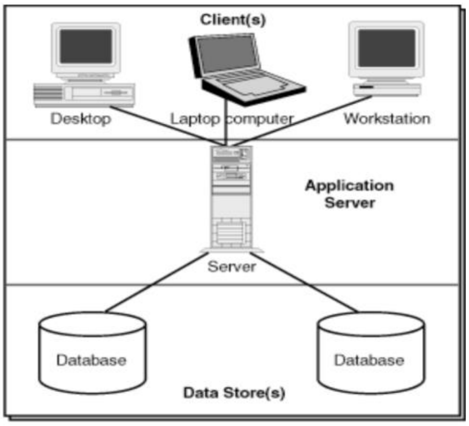


Figure 1.2.2.1: Client-server architecture

1.2.3 Problem Statement 3

1.2.3.1 Vague Functionality of Existing System

The quality and the feasibility of a system are of the utmost importance to secure customer satisfaction. The reason is that the quality of a system is determined by the requirements that will satisfy them (Hussain et al, 2015). Nonetheless, **the existing systems such as both FET and aSc TimeTables have implemented a lot of functionalities into the management systems.** To give an example, the functionalities are including class-subject allocation and teacher-subject allocation. For STSAMS, the class-subject allocation part of the system is not necessary as it only focuses on the management between the teachers and subjects. This is due to the fact that the class-subject allocation module can be said to be a reverse requirement: the requirements that are preferred not to be added to the STSAMS by the users (Mkpojiogu et al, 2016). As both Hussain et al. (2015) and Matzler et al. (1996) stated that these requirements will cause user satisfaction to decrease. For instance, the extra functionality in the system is not suitable for the school's workflow, at the same time increasing the learning curve of STSAMS's users as there are more functionalities required to be understood. Because of that, it will indirectly worsen the memory load of the users.

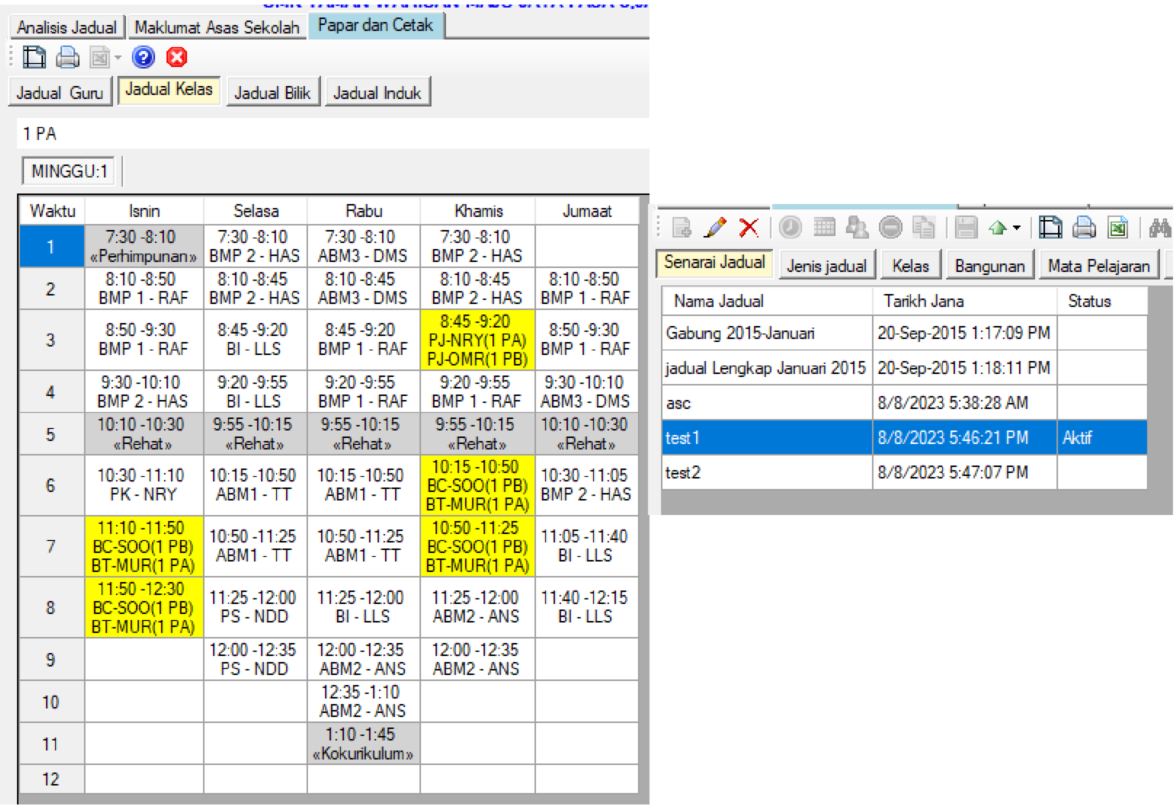
1.2.3.2 Proposed Solution

Therefore, interviews, questionnaires, and surveys can be prepared and done in public schools to uncover the school policy and guidelines on the allocation of teacher-subject and their workflow. Not just that, a study can be done to analyze the system that is currently being used by the school to acquire its implemented modules and set both hard constraints and soft constraints. The reason is that user requirements and system requirements can be designed using the findings from the schools; they are vital to understanding the user's needs. This is because every school has a big difference in terms of timetable scheduling rules. For example, some schools may allow two subjects to happen in the class at the same time. To further illustrate, in some schools, the non-muslim students will usually attend the Moral Education subject, while the Muslim students are taking the Islamic Studies subject. Covering users' needs can be also defined as satisfying the client's use cases or making sure that the developed system is compatible with the workflow of the client. In other words, the documents are to be used as a scope or control to ensure that the developed STSAMS is not unfinished or overdone.

1.2.4 Problem Statement 4

1.2.4.1 Clashing Problem during Allocation Process

In the 4th industrial revolution, humans utilize robots to automatically produce goods. This approach can be proven to improve productivity and provide economic benefits (Trauth-Goik, 2021). Big data analytics is one of the major components of Industry 4.0. It allows businesses to view trends and patterns from unstructured data. From this processed information, companies can prepare in advance to prevent risk and take the initiative to solve the problem that is going to happen (Minellim, 2013). Nonetheless, a preliminary study done at one of the public secondary schools in Malaysia which is the Sekolah Menengah Kebangsaan Taman Ehsan (SMKTE) concludes that the system being used by the school; WinJaws5 incorporated a type of deterministic algorithm. As stated by Rouse (2019) and Black (2009), the deterministic algorithm does not include randomness and its solutions are always similar when the inputs do not change. For example, using Winjaws5, two different generated timetables have the same outcome with the same dataset as shown in Figure 1.2.3.1 and Figure 1.2.3.2 below.



1 PA

MINGGU:1

Waktu	Isnin	Selasa	Rabu	Khamis	Jumaat
1	7:30 -8:10 «Perhimpunan»	7:30 -8:10 BMP 2 - HAS	7:30 -8:10 ABM3 - DMS	7:30 -8:10 BMP 2 - HAS	
2	8:10 -8:50 BMP 1 - RAF	8:10 -8:45 BMP 2 - HAS	8:10 -8:45 ABM3 - DMS	8:10 -8:45 BMP 2 - HAS	8:10 -8:50 BMP 1 - RAF

3	8:50 -9:30
	BMP 1 - RAF
		8:45 -9:20
		BI - LLS
			8:45 -9:20
			BMP 1 - RAF
				8:45 -9:20
				PJ-NRY(1 PA)
				PJ-OMR(1 PB)
					8:50 -9:30
					BMP 1 - RAF

4	9:30 -10:10 BMP 2 - HAS	9:20 -9:55 BI - LLS	9:20 -9:55 BMP 1 - RAF	9:20 -9:55 BMP 1 - RAF	9:30 -10:10 ABM3 - DMS
5	10:10 -10:30 «Rehat»	9:55 -10:15 «Rehat»	9:55 -10:15 «Rehat»	9:55 -10:15 «Rehat»	10:10 -10:30 «Rehat»

6	10:30 -11:10
	PK - NRY
		10:15 -10:50
		ABM1 - TT
			10:15 -10:50
			ABM1 - TT
				10:15 -10:50
				BC-SOO(1 PB)
				BT-MUR(1 PA)
					10:30 -11:05
					BMP 2 - HAS

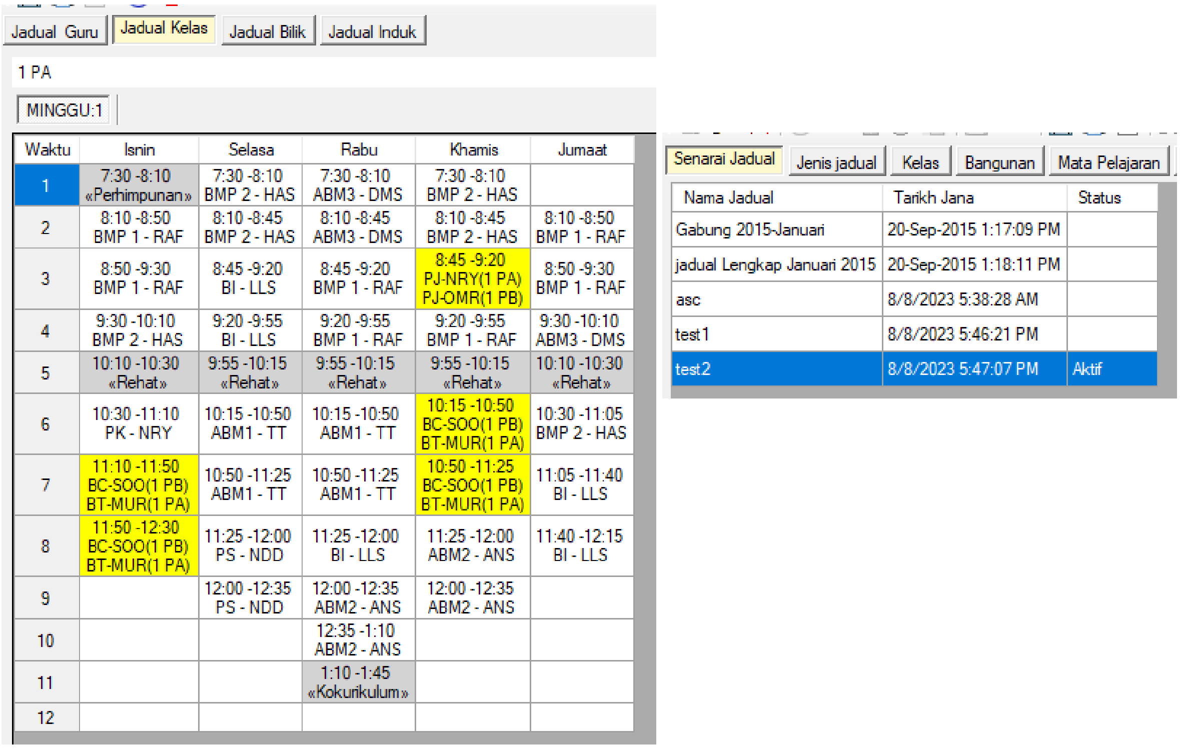
7	11:10 -11:50
	BC-SOO(1 PB)
	BT-MUR(1 PA)
		10:50 -11:25
		ABM1 - TT
			10:50 -11:25
			ABM1 - TT
				10:50 -11:25
				BC-SOO(1 PB)
				BT-MUR(1 PA)
					11:05 -11:40
					BI - LLS

8	11:50 -12:30
	BC-SOO(1 PB)
	BT-MUR(1 PA)
		11:25 -12:00
		PS - NDD
			11:25 -12:00
			BI - LLS
				11:25 -12:00
				ABM2 - ANS
					11:40 -12:15
					BI - LLS

9		12:00 -12:35 PS - NDD	12:00 -12:35 ABM2 - ANS	12:00 -12:35 ABM2 - ANS	
10			12:35 -1:10 ABM2 - ANS		
11			1:10 -1:45 «Kokurikulum»		
12					

Nama Jadual	Tarikh Jana	Status
Gabung 2015-Januari	20-Sep-2015 1:17:09 PM	
jadual Lengkap Januari 2015	20-Sep-2015 1:18:11 PM	
asc	8/8/2023 5:38:28 AM	
test1	8/8/2023 5:46:21 PM	Aktif
test2	8/8/2023 5:47:07 PM	

Figure 1.2.3.1: First-Generated Timetable for Class 1PA.



1 PA

MINGGU:1

Waktu	Isnin	Selasa	Rabu	Khamis	Jumaat
1	7:30 -8:10 «Perhimpunan»	7:30 -8:10 BMP 2 - HAS	7:30 -8:10 ABM3 - DMS	7:30 -8:10 BMP 2 - HAS	
2	8:10 -8:50 BMP 1 - RAF	8:10 -8:45 BMP 2 - HAS	8:10 -8:45 ABM3 - DMS	8:10 -8:45 BMP 2 - HAS	8:10 -8:50 BMP 1 - RAF

3	8:50 -9:30
	BMP 1 - RAF
		8:45 -9:20
		BI - LLS
			8:45 -9:20
			BMP 1 - RAF
				8:45 -9:20
				PJ-NRY(1 PA)
				PJ-OMR(1 PB)
					8:50 -9:30
					BMP 1 - RAF

4	9:30 -10:10 BMP 2 - HAS	9:20 -9:55 BI - LLS	9:20 -9:55 BMP 1 - RAF	9:20 -9:55 BMP 1 - RAF	9:30 -10:10 ABM3 - DMS
5	10:10 -10:30 «Rehat»	9:55 -10:15 «Rehat»	9:55 -10:15 «Rehat»	9:55 -10:15 «Rehat»	10:10 -10:30 «Rehat»

6	10:30 -11:10
	PK - NRY
		10:15 -10:50
		ABM1 - TT
			10:15 -10:50
			ABM1 - TT
				10:15 -10:50
				BC-SOO(1 PB)
				BT-MUR(1 PA)
					10:30 -11:05
					BMP 2 - HAS

7	11:10 -11:50
	BC-SOO(1 PB)
	BT-MUR(1 PA)
		10:50 -11:25
		ABM1 - TT
			10:50 -11:25
			ABM1 - TT
				10:50 -11:25
				BC-SOO(1 PB)
				BT-MUR(1 PA)
					11:05 -11:40
					BI - LLS

8	11:50 -12:30
	BC-SOO(1 PB)
	BT-MUR(1 PA)
		11:25 -12:00
		PS - NDD
			11:25 -12:00
			BI - LLS
				11:25 -12:00
				ABM2 - ANS
					11:40 -12:15
					BI - LLS

9		12:00 -12:35 PS - NDD	12:00 -12:35 ABM2 - ANS	12:00 -12:35 ABM2 - ANS	
10			12:35 -1:10 ABM2 - ANS		
11			1:10 -1:45 «Kokurikulum»		
12					

Nama Jadual	Tarikh Jana	Status
Gabung 2015-Januari	20-Sep-2015 1:17:09 PM	
jadual Lengkap Januari 2015	20-Sep-2015 1:18:11 PM	
asc	8/8/2023 5:38:28 AM	
test1	8/8/2023 5:46:21 PM	
test2	8/8/2023 5:47:07 PM	Aktif

Figure 1.2.3.2: Second-Generated Timetable for Class 1PA.

Hence, **a deterministic approach may or may not be able to address timetabling problems effectively.** If a deterministic method is applied to solve a problem but the solution causes clashing, it could happen again if the algorithm is run again. This is because the system administrator in SMKTE responsible to generate the timetables argued that the final result is always needed to be modified to solve clashing problems. Because of that, the school has decided to migrate to utilizing aSc Timetables.

1.2.4.2 Proposed Solution

1. **A digitalized manual approach**

The CRUD operations in STSAMS are not enough to improve users' task completion efficiency and effectiveness for the STSAMS. As for task completion efficiency and effectiveness, CRUD provides better efficiency in completing tasks compared to distributing teaching by paper. This is because doing addition, deletion, and modification to STSAMS is much faster. Not to mention, the system administrator can view the information in a more structured and cleaner way. However, it is not practical in scenarios where there are a lot of subjects and teachers in the school. This is because the allocation process will become a lot more complex as there are more combinations of subjects and teachers to be managed.

2. Inclusion of metaheuristic algorithm

As an improvement, a type of metaheuristic algorithm can be built in the STSAMS to do the scheduling for Teacher-Subject allocation. This is because, the truth is, creating an optimal solution to the time management problem is a challenging endeavor (Osorio & Esquivel, 2020). Similarly, Brownlee (2005) also mentioned that scheduling problems can be classified as NP-hard, restrictive, and combinatorial optimization issues. So, it is difficult to solve the timetabling problem if a determinist algorithm is used. As for solutions proposed by people, Brownlee (2005), and Awad et al. (2022) used a metaheuristic algorithm to solve the problem of scheduling, while Rossi-doria et al. (2003) compared the performance of multiple metaheuristic algorithms.

1.3 Project Objectives

The project objectives are separated into two parts which are Part 1 and Part 2. Part 2 cannot start if Part 1 is still not completed. Then, the two parts have different starting times as Part 2 starts after Part 1 ends. This project is directed to realize the following objective stated below: -

Part 1:

1. Design and implement a prototype that shows the overall system modules stated on the project scope in 12 weeks.

2. Compare different allotment algorithms to select the optimal algorithm for the allocation process in STSAMS by 8 September 2023.

3. Examine systems that are similar to STSAMS to distinguish the problems that can be enhanced by 8 September 2023.

4. Interview a public school to get an insight into the teacher-subject allocation in the school’s context by 8 September 2023.

5. Assess different development methodologies to decide the suitable approach to development for this project by 8 September 2023.

Part 2:

1. Design the important components of STSAMS such as the modules and user interfaces by using UML diagrams, storyboards, and flowlines; used to show the navigation of different pages on the website within 12 weeks.

2. Develop the STSAMS with the completed proposed modules both in teachers' and admins’ views within 12 weeks.

3. Test the STSAMS using unit tests, and user acceptance tests within 12 weeks.

1.4 Proposed Approach

This project will be using the Agile software development methodology specifically the Scrum method to develop the STSAMS.

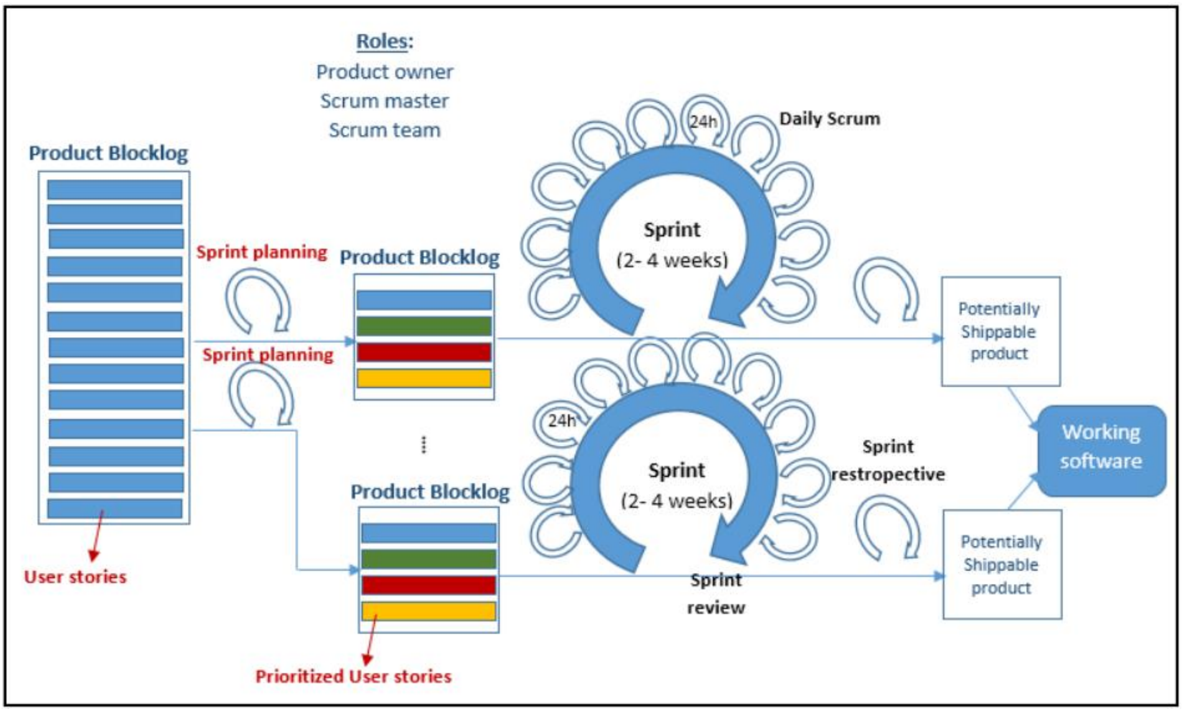


Figure 1.4.1: Scrum method life cycle

Figure 1.4.1 (Alsaqqa et al., 2020) illustrates the entire life cycle of the Scrum strategy. The Scrum method is the optimal development methodology for this project because it is compatible with the Work Breakdown Structure of this project. To explain why, Alutbi (2020) has mentioned in the guidelines the Work Breakdown Structure is involving the process of breaking down project activities or sub-activities into simpler, easier-to-handle tasks until the activities are described in enough depth to facilitate project management and development. Similarly, Mnkandla et al. (2004), and Cohen et al. (2003) stated that one of the advantages of Scrum is that it can dismantle the software product tasks into smaller components to be brought into the product backlog. For instance, STSAMS has a login module that can be broken down into more details parts such as the login page for system administrators and teachers. Even further, they can be divided into user interface design and backend which can include database design and data validation for password inputs. As for the bad side of Scrum, it does not provide any guidelines, practices, or application methods (Williams, 2010; Boehm and Turner, 2004). However, it can be turned into an opportunity because it can be tailored and modified to fits the practitioner’s needs.

1.5 Project Scope

The project targets to realize a School Teacher-Subject Allocation Management System (STSAMS) that encompasses a few modules in a web-based environment. Based on the existing system on the market and analysis done on the Malaysian public schools' timetabling software, enhancement can be made to further improve the efficiency and effectiveness of STSAM; the algorithm introduced is the core module that decides the quality of STSAM. This system will allow the users to get access using the internet browser as it was mentioned STSAMS will be developed in a web-based environment. The user that will be using the system is the teachers and the system administrator.

1.5.1 Teacher's Modules

The teachers can log in to the system through a teacher account. After logging in, the teachers are able to update their information such as name, date of birth, majors, and so on in their profile. These changes then will be reflected in the database for the system administrators to manage and analyze teacher's data. Other than that, teachers can also set their teaching preferences in the system to ensure a higher chance of teaching the selected subjects ordered by priority.

1.5.2 System Administrators' Modules

As for the system administrators, they can log into the system using an administrator's account which is a superuser account. They are to access the system to view the list of teachers, subjects, and classes. Then, they are responsible to create a teacher's account for the new teachers to log in with. After creation, the teacher's information will be left empty for the teachers to fill. If the teacher is logging in for the first time, they are required to set a new password to ensure security. Furthermore, the system administrators can change the content of this system. For example, they can add, update, and delete teachers, subjects, and classes. On top of that, the system will also provide filtering and sorting of the information for the system administrator to do analysis and searching. As for the most important part of the system; the scheduling, the system administrator can generate allocations by setting the hard constraints and software constraints for the algorithm in the system to consider while generating the allocation. For instance, constraints can be the maximum of subjects that a teacher can teach per day or the number of consecutive teaching subjects at the time. After the generation, the system administrator can export the results and also confirm to display the results to all teachers using the system in the teacher's user interface.

CHAPTER 2

LITERATURE REVIEW

2.1 Assessing the Determinants of Quality Allotment

Before diving into the details of scheduling methodologies, functionalities of the system, as well as how scheduling systems are enhanced to increase the productivity of its users, it is a must to first consider the theory and the factors behind the workload of teachers that affects teaching's performance, in which that is also directly related to the quality of the arranged schedule of teachers. To further illustrate my point, Ayeni et al (2018) stated that there are six components of related workload determinants as shown in Figure 2.1.1 below.

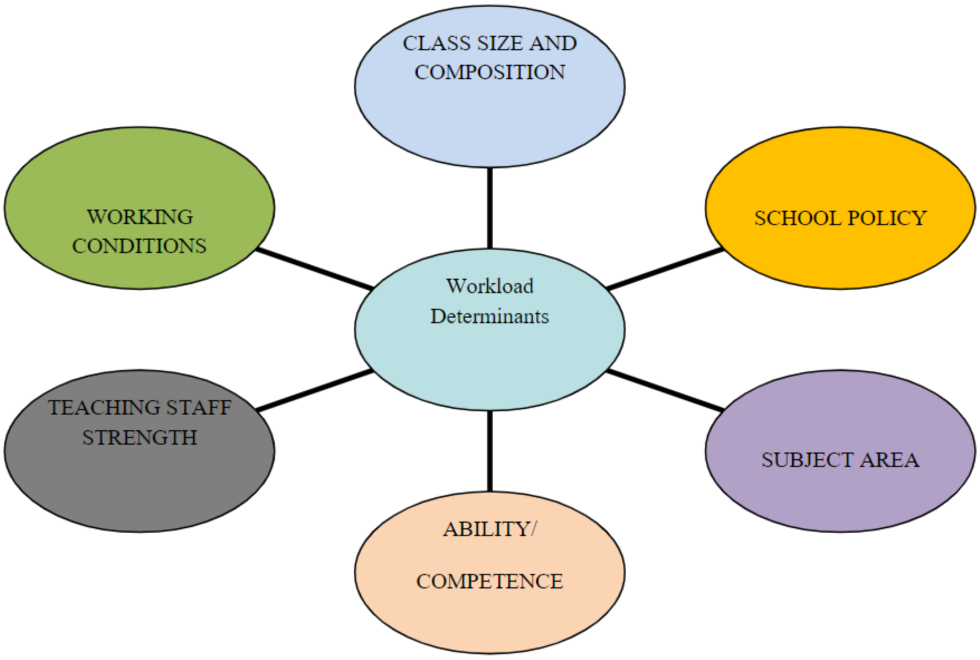


Figure 2.1.1: Conceptual framework of workload determinants.

To ensure the quality of teachers' subject scheduling, it is of utmost importance that the assigned tasks to the teacher must be both efficient and effective to the teachers. In this project, there are four factors that we will put our scope on, which are the class size and composition, school policy, subject area, and abilities. Why the other two factors are not considered as the factor in the study due to reasons firstly, STSAMS's main aim is not to increase or evaluate scheduling quality based on the working condition and teaching staff strength. This is due to the fact that working conditions vary from school to school in different locations such as in rural or urban areas, while also teaching staff strength is dependent on the quality of teachers hired and trained by different schools. Henceforth, STSAMS will only take into consideration the constraint that may affect the quality of scheduling outcomes significantly.

2.1.1 Class Size and Composition

First and foremost, class sizes are vital to the success of a quality allotment. This is because Ayeni et al (2018) argued that a large ratio between teachers and students may lead to poor performance in both students and teachers. Other than that, it will cause an ineffective workflow in the school. As to why teachers will have poor performance, Peter and Ligembe (2022) analyzed that in a total of 30 teachers, 83% of them voiced out that their performance is affected by larger class sizes as shown in Figure 2.1.2.

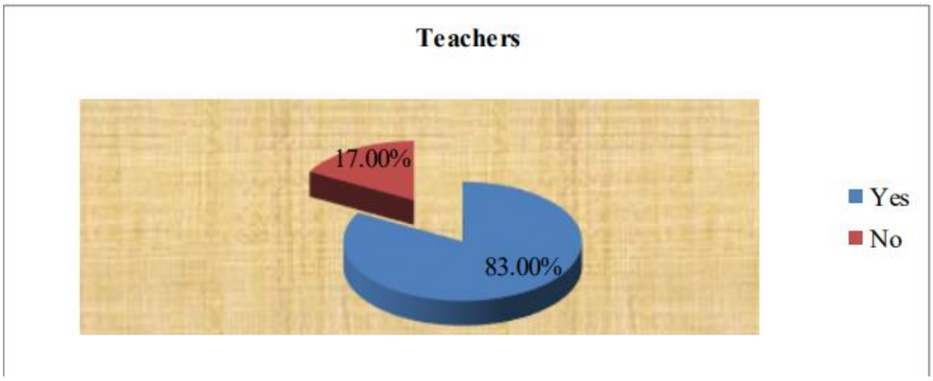


Figure 2.1.2: Perception of Teachers on the Effects of Large Class Size

This is because they found out that teachers face difficulty managing and teaching effectively in large classes, as they are unable to focus on every student in the class. For instance, teachers are hardly able to remember the students' names in a large class (Ayeni et al., 2018). Directly, the students are not properly guided in all classes as there is also a limited number of times each class period, leading to bad academic achievements. In another study, Nakamura and Dev (2022) similarly stated it is known to teachers and parents that smaller classes benefit students in terms of attention. Furthermore, it is discovered that the parents are more likely to have a one-on-one meeting with the teachers to solve student curriculum problems in a less crowded classroom, while also improving the bond between teachers and parents (Bascia et al., 2010).

2.1.2 School Policy

The school policy also plays a crucial role in determining the quality of the teacher-subject allotment. To give a few examples, the school officials are responsible for setting the policies such as the following: -

- Minimum or maximum teaching hours per week.
- Duration of each class period.
- Student study hours or teacher teaching hours per day.

It is necessary to take these school policies into account in the process of allotment, as a result of a study done by Stacey et al. (2023) has shown in Figure 2.1.3 that 89 out of 100 teachers agreed and strongly agreed that teaching and learning are affected by the heavy workload. This is because school policies of teachers or teachers' job scope can also include administrative jobs, and different positions in school such as the treasurer, secretary, subject coordinator, and teacher-in-charge for co-curricular societies and clubs.

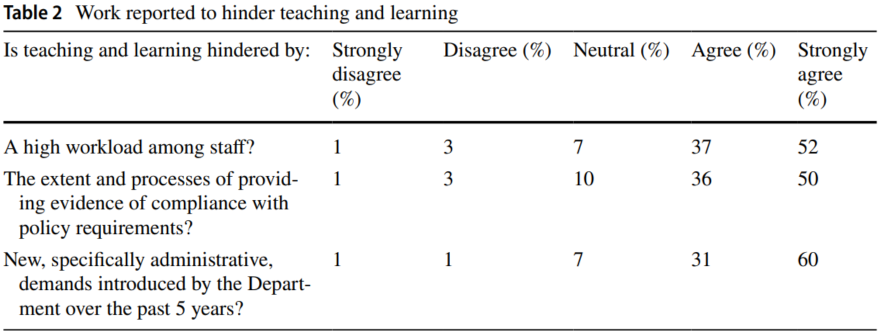
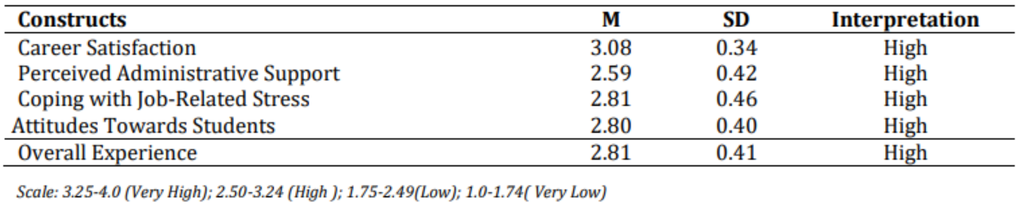


Table 2 Work reported to hinder teaching and learning

Is teaching and learning hindered by:	Strongly disagree (%)	Disagree (%)	Neutral (%)	Agree (%)	Strongly agree (%)
A high workload among staff?	1	3	7	37	52
The extent and processes of providing evidence of compliance with policy requirements?	1	3	10	36	50
New, specifically administrative, demands introduced by the Department over the past 5 years?	1	1	7	31	60

Figure 2.1.3: Work reported to hinder teaching and learning.

Hence, over-assignment will lead to career dissatisfaction, and eventually teachers will lose interest to focus on teaching the student. To support my point, Jomuad et al. (2021) concluded that teachers will face burnout by working more than the expected time, mostly affecting their career satisfaction in the result shown in Figure 2.1.4. Nonetheless, the teachers still tried their best to finish their assignments, producing a good performance. As a suggestion, a balanced workload should be given to the teachers by the school authorities to ensure work-life balance even though the teachers are able to cope with the intense working environment.



Constructs	M	SD	Interpretation
Career Satisfaction	3.08	0.34	High
Perceived Administrative Support	2.59	0.42	High
Coping with Job-Related Stress	2.81	0.46	High
Attitudes Towards Students	2.80	0.40	High
Overall Experience	2.81	0.41	High

Scale: 3.25-4.0 (Very High); 2.50-3.24 (High); 1.75-2.49(Low); 1.0-1.74(Very Low)

Figure 2.1.4: Level of Burnout Experience of the Teachers

2.1.3 Subject Area

To better understand the subject area, it can be defined as the level of cognition that a certain subject requires. Because of the high cognition, putting gap periods in between these subjects could increase the quality of a schedule and also be less tiring to teach. Ayeni et al. (2018) claimed that subjects such as Science, Mathematics, and English are exhausting for teachers to undertake as it is very complex and requires higher cognitive power compared to subjects such as Physical Education. This is because these highly cognitive subjects are based on Bloom's Taxonomy as shown in Figure 2.1.5 sketched by Adams (2015). For example, Howley-Rouse (2021) showed that Science and Mathematics subjects require an understanding of certain concepts. By understanding the basics, the learners are able to integrate different small solutions to solve complex problems that will vary in context.

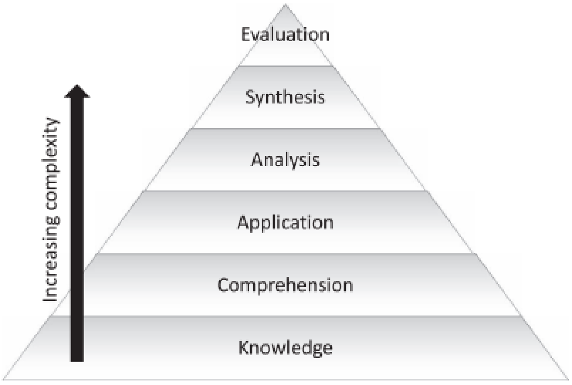


Figure 2.1.5: Bloom's Taxonomy

Other than the three subjects, Lavrijisen, et al. (2021) also classified more subjects such as Engineering, History, Economics, and many more in their research. As to why these subjects are exhausting to be taught, teachers are required to prepare more learning materials such as visualizations, mind-mapping, and remembering techniques for students to ease learning (Ayeni et al., 2018). Lazarides et al. (2019) said that teachers that are passionate about their subjects are likely to increase their pupils' interest in these subjects. Long story short, adopting resting times between cognitive-heavy subjects benefits both teachers' teaching and students' learning, thus improving the standard of the Teacher-Subject allocation.

2.1.4 Abilities

Lastly, assigning the right person to the right post is the most vital item to further increase schedule quality. As it ensures tasks are completed effectively and in a fast manner.

For example, Guerriero (2017) argued that a teacher's knowledge of the subject matter and instructional material plays a crucial role in ensuring that the students receive excellent guidance and information; hence achieving excellent academic performance. From this, we can know that Science teachers, for instance, are unable to teach Science subjects if they do not have the fundamentals of it or specialized in it. Similarly, Jeschke et al (2021) also stated that Mathematic teachers have a specialized understanding of the subject, encompassing both teaching and content knowledge. Their comprehensive knowledge of the topic enables them to adapt training to each student's requirements and successfully teach difficult mathematical concepts. Not just that, Math instructors have training and experience in teaching mathematics as they need to pass teaching school, which includes an understanding of successful teaching practices, evaluation processes, and subject-specific classroom management procedures. They can design exciting and successful learning experiences for pupils thanks to their knowledge. To realize the aforementioned factor, a module to sort teachers into compatible teaching subjects should be designed according to their major in their studies.

2.1.5 Summary

After diving into the necessary factors that define a quality allotment, a few constraints that can be included to the timetabling algorithm design of STSAMS can be came out with. Not just that, the system administrators are also allowed to make some value adjustments to the constraints. The possible constraints are shown in Table 2.1.1 below:

Table 2.1.1: Possible constraints that can be derived from the factors.

Factors	**Constraints**
Class Size & Composition	- Number of students per class.
School Policy	- Minimum or maximum teaching hours per week. - Duration of each class period. - Student study hours per day. - School teaching starting time & ending time. - School Non-Teaching Time (e.g., Recess Time, Rollcall) - Teacher’s empty time for activities outside of teaching. - Maximum number of consecutive class periods.
Subject Area	- Gaps between class periods for certain subjects.
Abilities	- Subjects that can be taught by each teacher.

2.2 Allocation and Timetabling Algorithms

Although the adoption of digitalization of teacher-subject allocation workflow has bring convenience to the schools, but it is still not efficient enough to support scalability. To give an illustration, using the 2 to the power of N, if there were 5 subjects in a school, the total number of timetable combinations would be 32. However, what if there will be 100 subjects in a school to be taught? It is impossible for the school to allocate all of them. Hence, to make the system more efficient, algorithms can be implemented in the process of subject allocation to find the best allocation. For example, genetic algorithms are very popular in solving scheduling problems.

2.2.1 Existing Design Solutions

Before making decisions on the selection of algorithms, it is fair that existing algorithm design solutions must be first thoroughly studied and compared to find out the resolution that best fits STSAMS. At the same time, clever designs can be also extracted from the studied solutions to create great ideas for STSAMS.

2.2.1.1 Brute Force Approach

Brute Force is a popular and commonly used algorithm to solve many kinds of problems including timetable scheduling as it is very simple to be developed. For example, brute force technique is also used in password cracking by assessing all combinations of passwords until the solution is found. Generate all possible combinations of allocating the teacher to different subjects. Not an optimal solution to find the best combination as brute force algorithms scale significantly more than genetic algorithms with the population size, in this case total classes assigned with teachers.

2.2.1.2 Particle Swarm Optimization

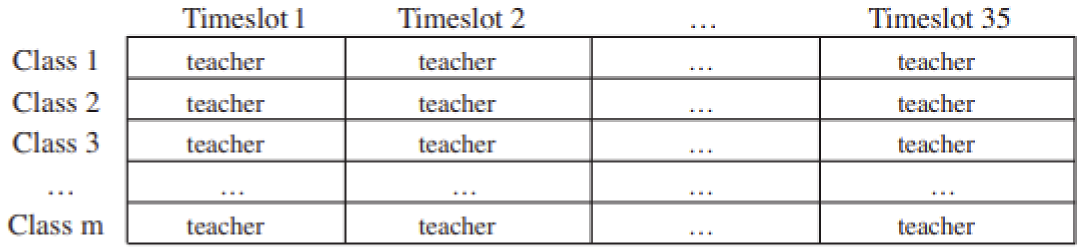
Particle Swarm Optimization is a type of meta-heuristic algorithm that is simulating the flocking of birds and schooling of fishes. Tassopoulos, et al. (2012) has designed and showed their implementation of PSO in school timetabling system for teachers and subjects. The implementation of the algorithm is accomplished with the help of hard constraints & soft constraints to achieve optimal scheduling. In their context, the school will be changing their timetable every week by using the timetable scheduling. It defines “time period” as each teaching hour in a week. For an instance, if school time is starting at 9 a.m.to 5 p.m., that means there are a total of 8 time periods. In another study, Chu, et al. (2012) have shown results to prove that PSO has significantly reduced the computational time needed to find the solution than the commonly used Genetic Algorithm. This is due to the fact that it can produce almost the optimal solution in lesser evolution compared to other heuristic algorithms. However, there will not be an algorithm that is without its cons, as the main concern of the system that embedding PSO is the business rules. Tassopoulos, et al (2012) categorized constraints; also, can be seen as business rules into two components which are the hard constraints and soft constraints. The constraints are segregated as below in Table 2.2.1.2.1:

Table 2.2.1.2.1: Constraints used.

Type of Constraint	Constraints
Hard Constraint	- Teachers’ day availability. - Teachers’ hour availability. - Classes’ hour availability. - Classes’ empty hours. - Teachers-classes-lesson assignment. - Co-teaching cases. - Subclasses cases.
Soft Constraint	- Teachers’ teaching hours’ dispersion - Teachers’ empty hours - Teachers’ empty hours’ dispersion - Classes-lessons dispersion

To keep track of all record defining the relations between lessons, teachers, and classes, they have utilized multidimensional arrays by storing binary values. The integer 1 or Boolean of true will be used to represent hitting the constraints, otherwise integer 0 or Boolean of false shows off-target. Other than that, costing system also been used to calculate the seriousness of constraints violations. These costing systems usually are only applied to the soft constraints, as soft constraints decide the quality of a schedule, while hard constraints adherence is a must to display the schedule feasibility.

2.2.1.2.1 Algorithm Design



	Timeslot 1	Timeslot 2	…	Timeslot 35
Class 1	teacher	teacher	…	teacher
Class 2	teacher	teacher	…	teacher
Class 3	teacher	teacher	…	teacher
…	…	…	…	…
Class m	teacher	teacher	…	teacher

Figure 2.2.1.2.1: Particle encoding.

1. Initializing the population of particles. In this case, the system designed the particle encoding to be a complete timetable allocation in a matrix form as shown in Figure 2.2.1.2.1 (Tassopoulos, 2012). For example, X is an array of particles, P is the number of currently accessing particle, C is the classroom number, and finally T representing the timeslot number in the timetable. To access the 3rd timetable slot of class number 10 and the 5th particle, the array is called as "X [5][10][3]". Then, value of the array is the teacher assigned to the timeslot of a particular class.

2. Deciding stop condition such as number of generations need to be iterated.

3. Evaluating the particle's fitness. Then, eliminate "weak" particles using a defined tolerance factor.

   Tolerance = xx1 + xx2 * global_best_fitness

   xx1 = ((current generation*10 + 1) *10.0)/ (10000 − current generation)

   xx2 = xx2 + 0.002375

   If particle's fitness is more than tolerance, then it will be deactivated. A deactivation of a particle means that the particle is eliminated. This approach is documented by the authors that it is the fastest way to get rid of the non-fit particles.

4. Updating personal best matrix, personal best fitness of current active particles if the fitness is better than previously set fitness. Then, if the fitness is better than the global fitness, update it as the best global matrix and fitness. As to how the better fitness particles are measured, the fitness particles in lower value are seen as the better ones.

5. Applying transformation to the active particles to produce a potentially better particle.

- S = Transform the particles by swapping two randomly chosen timetable slots for current active particle.
- W = Transform S with its own personal best matrix of current active particle.
- Perform other necessary transformations with the global best matrix to produce the new global best.

6. Evaluating fitness of the particle. Then, update global fitness value if it is better than the previous best global fitness.
7. Repeating 3rd-6th process until stop condition.

Long story short, it is applying the main concept of PSO by implementing the encodings, then deciding the methodology to evaluate the fitness of the encodings. Iterating through all of the encodings to find the best among them, while also implementing variation with mutations, swapping, and transformations to increase the chance of producing best new solutions. Finally, it is decided by the algorithm creator as when to stop the algorithm.

2.2.1.2.2 Algorithm Application

In relation to the use case in Malaysia, classes' empty hours which also means time period that a classes does not have lessons can be applied with the rollcall time every first period of school on Monday, recess time provided by the school; usually happening in between lessons, and Muslims prayer time on Friday that will mandate finishing morning classes early at 12:30p.m as well as starting afternoon classes late at 2:30p.m. In Malaysian education culture, it is not common to have a teacher co-teaching a class after preschool. So, it is hardly applicable in Malaysian public-school settings. However, there are cases in Malaysian public school separating Islam education and Moral education and P.E. to boys and girls to incorporate the subclasses cases. As for the classes-lessons dispersion, it is true that lessons need to have same teaching hours for each class, but different classes can have varying number of subjects taken. So, some classes may end their day sooner than others. For example, classes in Malaysia public school will be focusing on different specialties such Science classes, Computer Science classes, Art Classes as well as the Accounting, Business and Economics during Form 4 and Form 5 which are 11th grader and 12th grader in the United States.

2.2.1.3 Maximum Flow with Edmunds-Karp Algorithm

The next assessing technique will be on the concept of maximum flow, simulating water travelling from the source through the pipeline with limited flow to the sink. A study has been done by Asano, et al. (2000) on the algorithms that has been used to pair with the maximum flow concept. For example, the list includes two commonly used deterministic algorithms such as the Edmunds-Karp algorithm and Ford-Fulkerson algorithm. Other that that, they also explained how they were implemented to suits the implementation of maximum flow. According to Jain (2019), he has proposed a subject-teachers allocation algorithm that is similar to Hospital-Patient distribution based on maximum flow that involves Edmunds-Karp algorithm and Bipartite Matching with visualization of Bipartite graph shown by Angelidakis (2020) in Figure 2.2.1.3.1.

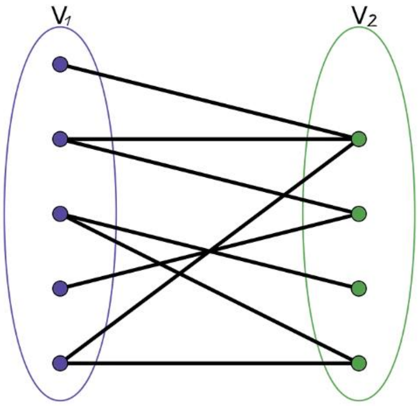


Figure 2.2.1.3.1: Bipartite Graph

The main objective of the resolution given by Jain (2019) is in the traditional method, teachers are given subjects to teach from the choices provided by the institution, but some subjects will be left not getting picked up. Hence, this solution is to satisfy as much teachers as possible while also prevent subject vacancies by automation.

2.2.1.3.1 Algorithm Design

1. Prepare a list of available subjects and teachers.
   - ➢ For example, available subjects list is referring to the subjects that can be chosen by the teachers to teach. Each subjects have capacities that can be defined. If a subject has capacity of 2, that means two teachers can teach the subjects.

- ➢ The list of available teachers can have their preferred subject to teach. So, the teachers can be allocated to the preferred subjects in the bipartite matching process.

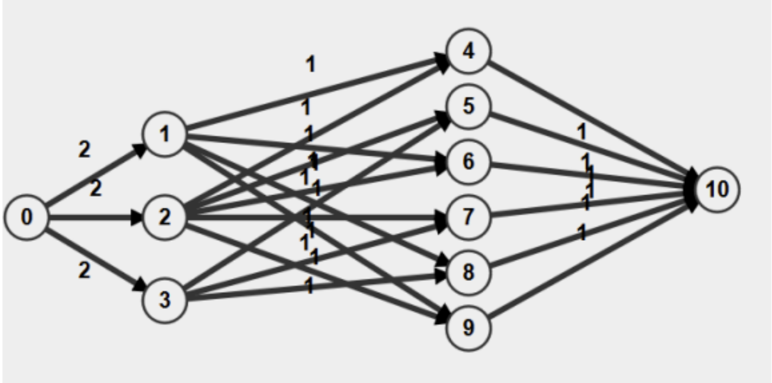


Figure 2.2.1.3.2: Bipartite Graph

2. Use the bipartite matching technique to form a bipartite graph as shown in Figure 2.2.1.3.2 prepared by Jain (2019) that maps the teachers to the at least one preferred subject.
3. Then, Edmunds-Karp algorithm is used to allocate the teachers to their preferred subjects at the same time producing the max-flow based on the maximum capacity of subjects can be chose, producing possible result like in Figure 2.2.1.3.3, showing input flow of 6 in total and also 6 to output.

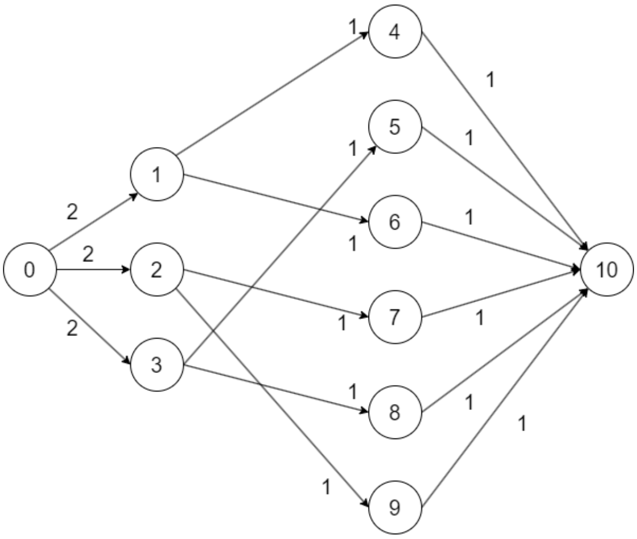


Figure 2.2.1.3.3: Allocation Result after Applying Edmunds-Karp algorithm.

4. When iterating through the subjects, if there are no teachers left to be assigned, brute-force algorithm is used to backtrack and do necessary reassigning to optimize the scheduling.
5. Then, print the max flow paths for visualization.

2.2.1.3.2 Algorithm Application

Table 2.2.1.3.1: Benefits and Problems of the Implementation.

Benefits	**Problems**
Prevention of unallocated teachers and subject vacancies	The matching is not efficient in terms of preference treatment. For example, teacher can't decide what subjects are more preferred than others. The algorithm treats of preferences equally and is not weighted.
Autonomous and efficient	Teachers are required to choose the preferences of teaching subjects from an available list of subjects. As a solution, default preferences values can be set to teachers.
Teachers/Lecturer has more freedom as they are allowed to give their preferred subjects to teach.	Consideration of time clashing between subjects is not done.

2.2.1.4 EA, Genetic Algorithm

Implementation of Genetic Algorithm is very similar to PSO as they share the same traits in terms of processes and techniques. For example, both includes the encoding but in different wordings which is chromosomes and mutations. In terms of techniques, Genetic Algorithm also simulates the behavior of biological lifeforms. This is because the process of natural selection is simulated in order to identify the best answers to optimization and search issues. It is a kind of evolutionary algorithm that is founded on the ideas of natural selection and genetics. In this section, we will be discussing and summarizing two existing solutions for timetabling using Genetic Algorithms.

2.2.1.4.1 Research 1 (Chan, et al., 2006)

Table 2.2.1.4.1: Analysis of Genetic Algorithm for Research 1.

Attributes	**Descriptions**
Settings	NTU university – student & course
Problem with Existing System	- Tedious - Students are required to plan ahead for the registration. - Planning which lecture group, they want to enroll, then the tutorial and practical classes. - Some courses are very popular, while some are not due to undesirable time periods.
Proposed EA System	It can automatically allocate timetable according to student's preferences. Descriptions: - a) Hard Constraint & Soft Constraints b) Hard Constraints (timetable clashes & total credit hour), Soft Constraints (preferences) c) Timetable clash issues are resolved at the data entry level. d) System tries to satisfy student's first choice & second choice. e) System will not provide alternative for allocations as usually students will reject it.

Design	1) Generate population (decided by first & second choices given by the students) 2) Sort the population by the fitness. 3) Crossover – higher fitness value = higher chance to be picked 4) Choose 2 randomly from the population. 5) For each population, higher weight / fitness means higher chance to be chosen. 6) 1, 2, 3, 4, 5 ,6 – 5 and 6 have more chance to be chosen 7) Use 2 chosen numbers to do crossover. 8) For example, 101110 & 110110 9) Randomly generate where to cut, a. 1 0 | 1 1 1 0 & 1 1 | 0 1 1 0 Result: b. 1 1 | 1 1 1 0 & 1 0 | 0 1 1 0 10) Mutation 11) Pick new generation. For example, if there is a population of 10 combinations. The system will randomly pick 2 combinations from the population and do crossover. After the crossover, mutation will happen to the combination, where a binary can be flipped in the combination. We will pick the two newly generated combination and the top 8 of the previous generation as population for the next generation. Then, repeat until finding the best solution.

2.2.1.4.2 Research 2 (Pambudi, et al., 2019)

Table 2.2.1.4.2: Analysis of Genetic Algorithm for Research 2.

<table>
<tr><th>Attributes</th><th>Descriptions</th></tr>
<tr><td>Settings</td><td>Lecturer & Course/Subject</td></tr>
<tr><td>Goals</td><td>- To improve the quality and academic abilities of its students due to the lecturer’s high performance and self-confident teaching style.
- To allow lecturers to teach certain subject according to their interests. For example, a lecturer who is experienced in networking teaches TCP/IP.
- To find the optional subject allocation so lecturer/teacher’s performance in teaching is as high as possible, while satisfying their interests.</td></tr>
<tr><td>Design</td><td>Assigning lecturer/teacher according to their preferences.
1. Prepare information about course details including course code, course name, credit & number of classes needed to be taught.
2. Collect the preferences of teacher of each course.
3. Propose design of the biological representation/encoding. For example, there is 10 classes in total for a total of 5 courses and 5 lecturers.
<table>
<tr><td>1</td><td>3</td><td>2</td><td>4</td><td>4</td><td>5</td><td>2</td><td>5</td><td>3</td><td>1</td></tr>
<tr><td colspan="3">Course 1</td><td colspan="2">Course 2</td><td>Course 3</td><td colspan="2">Course 4</td><td colspan="2">Course 5</td></tr>
</table>
Note: The encodings are generated before the simulation. The number of encodings can be different.
4. Fitness function. It is used to evaluate how good as a solution an encoding is. Its design depends on the business model. For example, a weight system can be implemented. For courses that is preferred by the lecturer will contribute lower weight to the total weight. So, lower weight means higher fitness, which is the better solution.
5. Crossover. This phase is to select 2 random parent encodings. Then, decide the cut points to do crossover. 1|1 &amp; 2|2 &gt;&gt;&gt; 1|2 2|1
6. Mutation. Same as the biological theory, each crossover has the chance to cause mutation. In this case, swap mutation can be used to swap two</td></tr>
</table>

	teachers from different class. (Not necessary in the different classes, swapping only lecturer from same course code could happen.) 7. Selection. Sort all possibilities according to the fitness. Eliminate few encodings that has the lowest fitness. Repeat 5th step until break condition, for example maximum of 100 generations (loops).

2.2.2 Analysis and Comparison between Algorithms

Table 2.2.2.1: Comparison between studied algorithms.

Researching Algorithm	**Speed**	**Advantages**	**Disadvantages**	**Combination with other techniques**	**Model**
Brute Force Algorithm (Baturu et al., 2020 and Mohammad et al., 2006)	Slow	1. High problem-solving range. 2. Low complexity and simple for understanding. 3. Good at solving searching, sorting, string matching, or matrix multiplication problem. 4. Can be implemented in any programming languages.	1. Most algorithms using brute force algorithm are inefficient. 2. Brute force algorithms are clumsy and slow. 3. Lack of creativity compared to other algorithms at problem solving.	-	-
Particle Swarm Optimization (Palupi et al., 2011)	Fast	1. Adaptable to suit teachers' preferences. 2. Able to please a variety of niche constraints, to better	1. Impossible to work out the scattering and optimization problem.	Timetable slot swapping.	Simulates bird flocking and fish shoaling.

		match the workflow of specific schools.	2. Unable to be applied non-coordinate system problems.		
Bipartite Matching	Slow	1. Prevention of unallocated teachers and subject vacancies. 2. Prevention of clashes in timetable scheduling.	1. The matching is not efficient in terms of preference treatment. 2. Usage of brute force algorithm for backtracking.	Edmunds Karp Algorithm & Brute Force Algorithm.	Using bipartite graph, with source and sink.
Genetic Algorithm - Research 1 (Chan et al., 2006)	Moderate	1. Mostly used for complex timetable scheduling problems. 2. Problem can be solved in a parallel manner. 3. Support for multiple parameters.	1. High time and computational complexity. 2. Requires the need to convert real-world problems into genetic encodings.	Crossover & Mutation using Bit Flip Mutation.	Simulates biological evolution.
Genetic Algorithm - Research 2 (Pambudi at al., 2019)				Crossover & Mutation using Swap Mutation.	

Based on Table 2.2.2.1, Genetic Algorithm will be the perfect candidate for the timetabling algorithm for STSAMS as it is widely used for complex timetable scheduling problem. Genetic algorithms will be slower compared to PSO, but it is easier to be come out with using genetic algorithms as it allows the STSAMS to solve timetable problems for multiple teachers concurrently. So, it is worth while to sacrifice some of the generation speed for the shorter implementation time. As why to not choosing Brute Force Algorithm and Bipartite Matching, because complex problems like timetable problems has too many parameters and constraints to be consider. If there were to be implemented in STSAMS, it will take a very long time to finish generating timetables of many teachers.

2.3 Similar Existing Systems

2.3.1 FCVAC Course Allocation System

2.3.1.1 System Description

Rauf, et al. (2018) has only written about their concept about developing the system including the system design, but unfortunately there is no evidence of the developed system to be tested and played around with. The main objective of this study is to come out with a basic system design to act as a guidance for future works.

2.3.1.2 System Design

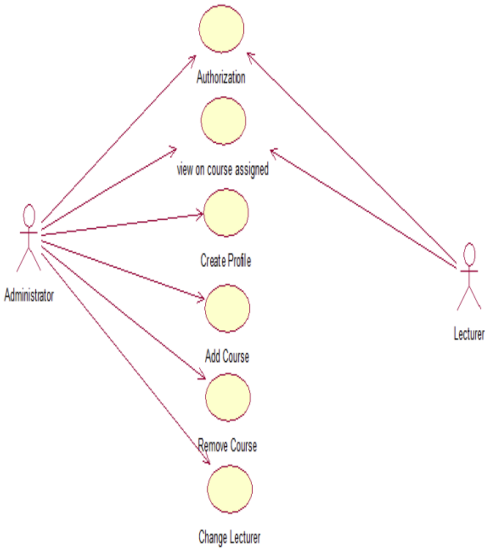


Figure 2.3.1.2.1: Use Case Diagram (Rauf, et al., 2018).

It is proposed that Administrator should be having the ultimate control on the entire system. So, A superuser account should be assigned to the system administrator to do operation on the allocation system (Rauf, et al., 2018). For example, by referring to the use case diagram in Figure 2.3.1.2.1, the system administrator is allowed to log in to the system, view on course assigned of all lecturers, create lecturer's profile, adding course, while removing it and changing the lecturer in which the course is assigned to. Subsequently, the lecturers are only able to log in using lecturer account and view courses that are assigned to them.

2.3.1.3 Application

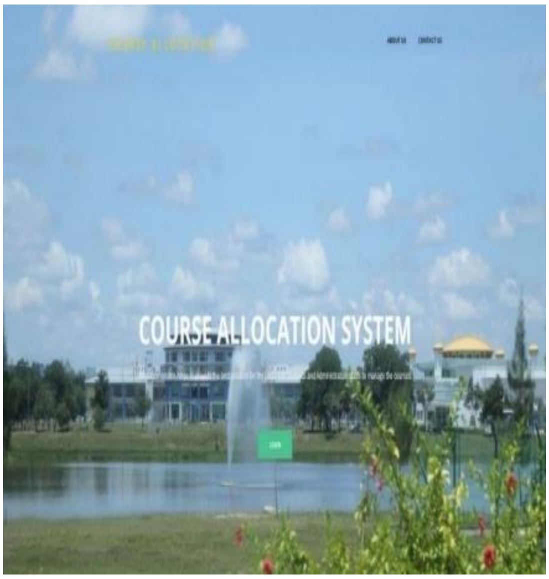
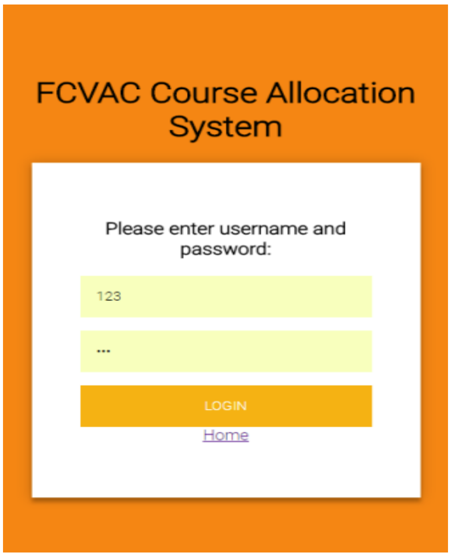


Figure 2.3.1.2.2: Home Page (Rauf, et al., 2018).

In terms of usability, some textual contents in the home page shown in Figure 2.3.1.2.2 are hardly readable as there is no clear color contrast between the texts in yellow and white with the background. However, the black texts are acceptable as it is noticeable from the bright background image. As a solution, black background with opacity but not completely transparent or without opacity can be added behind those text to show the color contrast to make it legible. Other than that, different text colors and a plain background instead of image background also used to make the texts readable.


FCVAC Course Allocation System

Please enter username and password:

123

•••

LOGIN

Home

Figure 2.3.1.2.3: Administrator Login Page (Rauf, et al., 2018).

As for the administrator login page in Figure 2.3.1.2.3, the white box has shown a clear grouping of text input fields. Then, a clear contrast of color is used to differentiate the login buttons and the other components, and also it is the most important button to initiate the login functionality after the home hyperlink. Nonetheless, the labels for the login credentials should be placed on top of the data input boxes, to clearly define the type of value to be entered in the text boxes instead of just specifying “Please enter username and password:” on top.

2.3.2 WinJaws5

2.3.2.1 System Description

WinJaws5 is developed by DMH Technology, and the current system version of assessment is Version 5.0.4.7. WinJaw5 is specifically developed from public schools in Malaysia as one of the interviewing schools which is SMK Taman Ehsan in this project is also using WinJaw5 to manage their school. Before getting into details, it is vital to first get an overview of the system. The main concept the system is to input necessary information, set the generation constraints, generate the timetables, finally print them out. Nonetheless, there are some limitations and unnecessary step are included the system, making the school workflow slower. So, we will address them in the later section.

2.3.2.2 System Functionalities

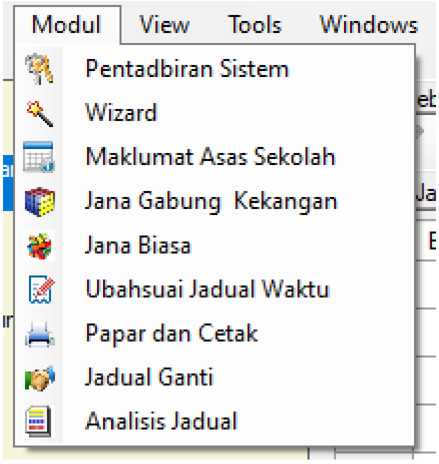


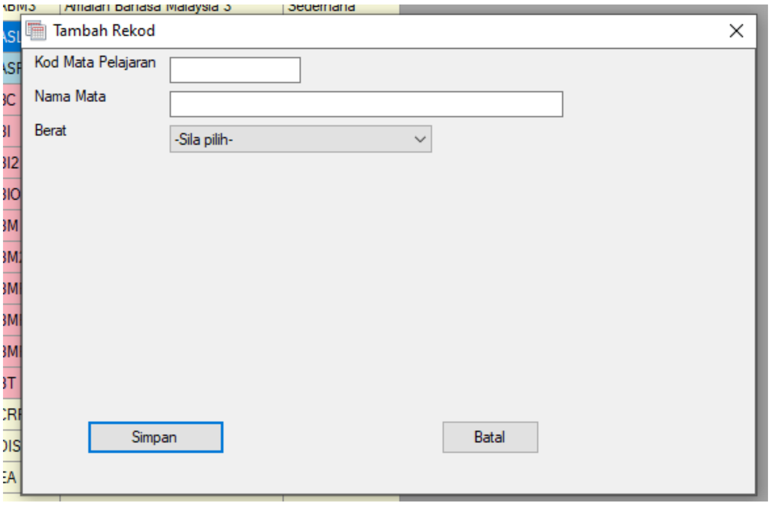
Figure 2.3.2.2.1: Screenshot of the sub-menus under the Module selection

Under the sub-menu of the Module section shown in Figure 2.3.2.2.1, the system administrators are able to do operations on system managements, basic school information management, two different timetabling generations such as basic generation and merged generation, timetable modification, displaying and printing, replacement timetable, and timetable analysis.

First and foremost, the system administrators are allowed to import and export their current WinJaws5 project using the Microsoft Database (.mdb) file format under system management. This is useful for the system administrators when they want to save their progress on the project to save it for another day or transfer their work to another system administrator.

To make it secure, WinJaw5 actually allows password to be set on the project just in case the project is gotten into the wrong hands as there are personal data can be directly mapped to the teachers, and also school information regarding the subjects, classes, and student's data. Then, type of institution can be also selected along with the school badge to give a clear definition of a school as there are some cases in Malaysia that different schools such as primary and secondary school are sharing the same top management.

Other than that, all the information about the generated timetables, classes, subjects, teachers, classroom, and timetabling constraints. The basic CRUD functions can be done on the aforementioned information. To give an illustration, Figure 2.3.2.2.2 is showing the addition of new subjects to the list with data field of subject code and name, as well as the weight of the subjects as some subjects require higher cognition. Then, Figures 2.3.2.2.3 and 2.3.2.2.4 visualize the system interface for the modification and deletion functionalities, while also providing a confirmation dialog to the user before making any changes. As for the timetabling constraints, WinJaws5 is designed in such a way that classes must be attached to the created constraints. For example, the teaching hours of classes are first to be set as shown in Figure 2.3.2.2.5. It can also be defined under the status column to decide if the time period has classes. In this case, the unchecked box states that the first period and sixth period have morning rollcall and recess. After that, classes such as 4FA, 4NE, 5FA, and 5NE can be attached to the set constraints as shown in Figure 2.3.2.2.6.



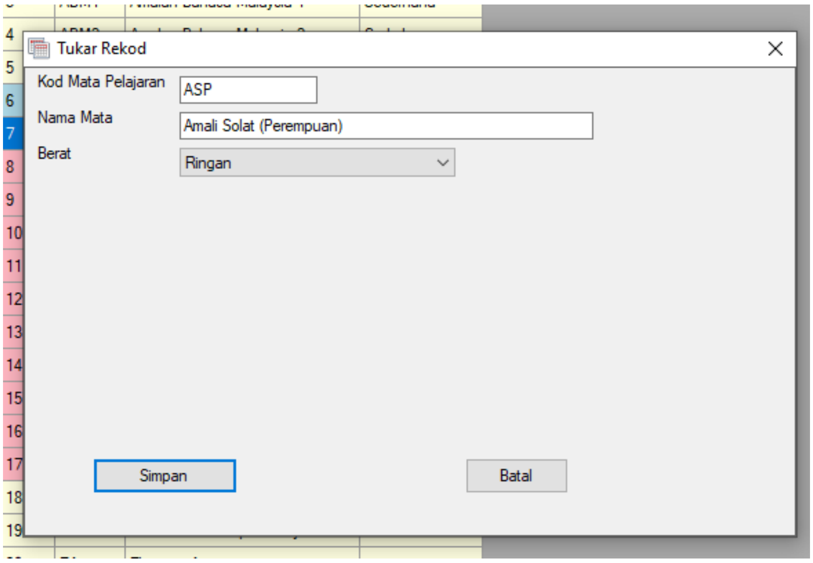
Tambah Rekod

Kod Mata Pelajaran

Nama Mata

Berat: -Sila pilih-

Figure 2.3.2.2.2: Screenshot of new subject creation.



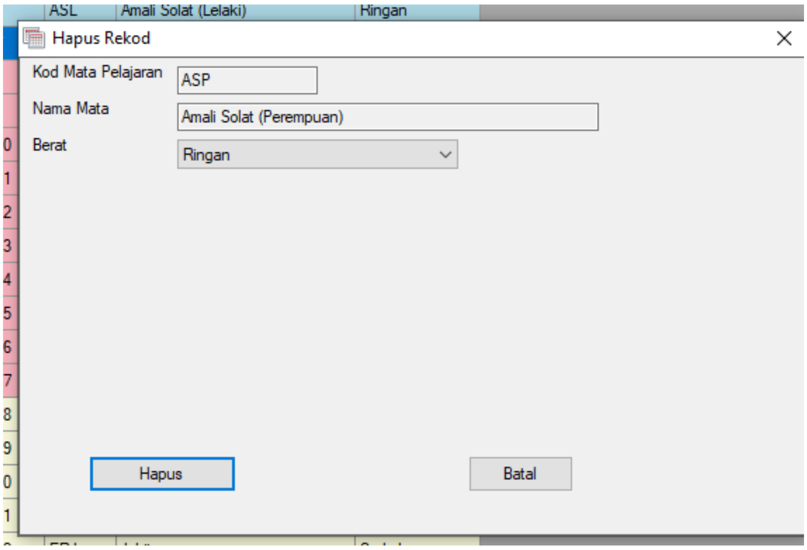
Tukar Rekod

Kod Mata Pelajaran: ASP

Nama Mata: Amali Solat (Perempuan)

Berat: Ringan

Figure 2.3.2.2.3: Screenshot of existing subject data modification.



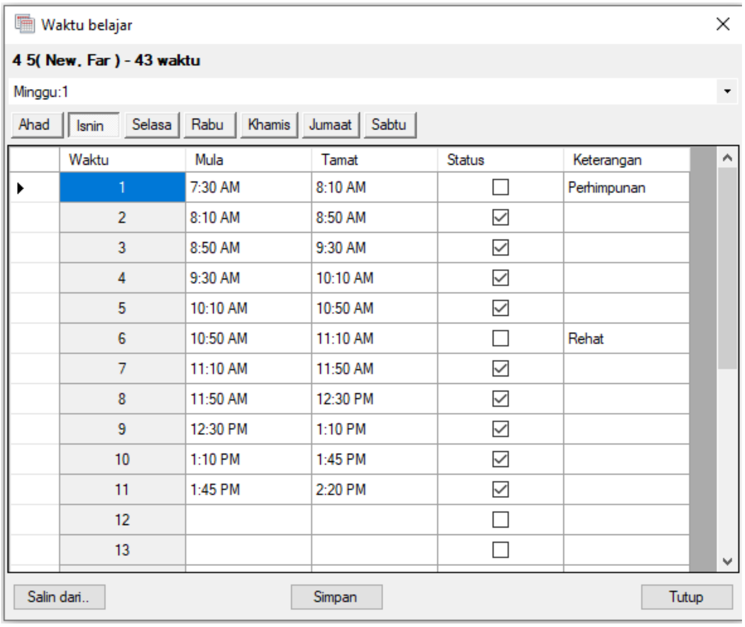
Hapus Rekod

Kod Mata Pelajaran: ASP

Nama Mata: Amali Solat (Perempuan)

Berat: Ringan

Figure 2.3.2.2.4: Screenshot of existing subject data deletion.



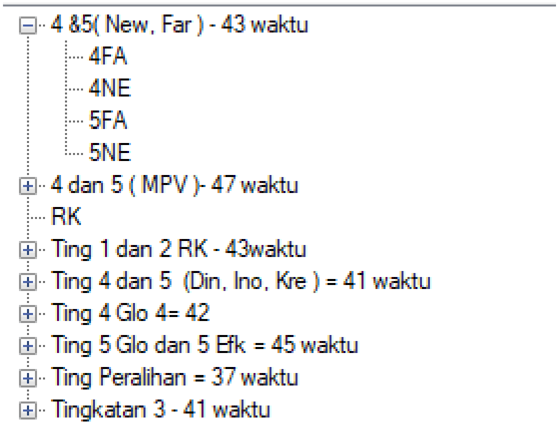
Waktu belajar

4 5(New, Far) - 43 waktu

Minggu:1

Waktu	Mula	Tamat	Status	Keterangan
1	7:30 AM	8:10 AM	☐	Perhimpunan
2	8:10 AM	8:50 AM	☑	
3	8:50 AM	9:30 AM	☑	
4	9:30 AM	10:10 AM	☑	
5	10:10 AM	10:50 AM	☑	
6	10:50 AM	11:10 AM	☐	Rehat
7	11:10 AM	11:50 AM	☑	
8	11:50 AM	12:30 PM	☑	
9	12:30 PM	1:10 PM	☑	
10	1:10 PM	1:45 PM	☑	
11	1:45 PM	2:20 PM	☑	
12			☐	
13			☐	

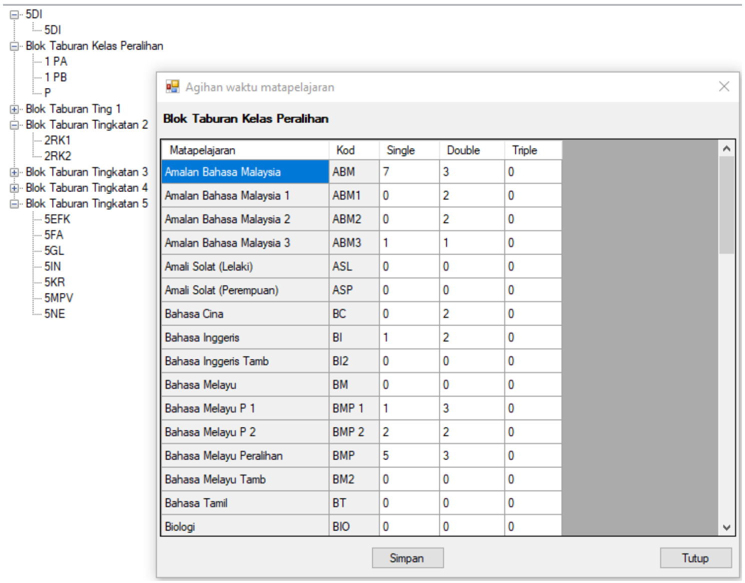
Figure 2.3.2.2.5: Screenshot of teaching hours of classes.



- 4 &5(New, Far) - 43 waktu
  - 4FA
  - 4NE
  - 5FA
  - 5NE
- 4 dan 5 (MPV)- 47 waktu
- RK
- Ting 1 dan 2 RK - 43waktu
- Ting 4 dan 5 (Din, Ino, Kre) = 41 waktu
- Ting 4 Glo 4= 42
- Ting 5 Glo dan 5 Efk = 45 waktu
- Ting Peralihan = 37 waktu
- Tingkatan 3 - 41 waktu

Figure 2.3.2.2.6: Screenshot of attaching classes to a constraint.

Using the same concept, the distribution of the subjects can be also defined as shown in Figure 2.3.2.2.7. To ensure better understanding, time periods can be seen as blocks. Then, the time periods for subjects can be determined if the system mandate the subjects' periods to be combined as double blocks or triple blocks for consecutive classes when timetable generation happens. So, the subject's periods would not be scattered all over the place. As for the subclasses, WinJaws5 allows some classes to be merged into one class or separating a subject to be held in two classes.



5DI
5DI
Blok Taburan Kelas Peralihan
1 PA
1 PB
P
Blok Taburan Ting 1
Blok Taburan Tingkatan 2
2RK1
2RK2
Blok Taburan Tingkatan 3
Blok Taburan Tingkatan 4
Blok Taburan Tingkatan 5
5EFK
5FA
5GL
5IN
5KR
5MPV
5NE

Agihan waktu matapelajaran

Blok Taburan Kelas Peralihan

Matapelajaran	Kod	Single	Double	Triple
Amalan Bahasa Malaysia	ABM	7	3	0
Amalan Bahasa Malaysia 1	ABM1	0	2	0
Amalan Bahasa Malaysia 2	ABM2	0	2	0
Amalan Bahasa Malaysia 3	ABM3	1	1	0
Amali Solat (Lelaki)	ASL	0	0	0
Amali Solat (Perempuan)	ASP	0	0	0
Bahasa Cina	BC	0	2	0
Bahasa Inggeris	BI	1	2	0
Bahasa Inggeris Tamb	BI2	0	0	0
Bahasa Melayu	BM	0	0	0
Bahasa Melayu P 1	BMP 1	1	3	0
Bahasa Melayu P 2	BMP 2	2	2	0
Bahasa Melayu Peralihan	BMP	5	3	0
Bahasa Melayu Tamb	BM2	0	0	0
Bahasa Tamil	BT	0	0	0
Biologi	BIO	0	0	0

Figure 2.3.2.2.7: Screenshot of subject distribution constraint.

Furthermore, the system administrators can then generate the timetables automatically by using generation functionality in the system. However, it is separated into two modules such as basic generation and merged generation. They are actually not different kind of generation methods for the timetables, but actually basic generation is just generation without constraint and merged generation is with the consideration of the constraints. It is a rule set by the developers that the timetables must be first generated using the basic generation, followed by the merging generation after that. Then, if the system administrators feel that the generated timetables are unsatisfactory, they can choose to manually edit it themselves while using the timetabling analysis function provided by WinJaws5. Then, they can view specific teacher's timetables then print them out. Finally, WinJaw5 also implemented an isolated replacement timetable from the main timetable to assign replacement teachers to replace absent teachers.

2.3.2.3 Application

Table 2.3.2.2.1: Strength and Weakness of WinJaw5 system.

Strengths of WinJaws5	**Weaknesses of WinJaw5**
It allows modification on generated timetables for final touches.	It does not support multiple language and it is only in the Malay language
It allows progress to be saved in .mdb file format.	It does technically support multiusers with project importing and exporting but does not allow them to work concurrently.
It can generate timetables for the system administrator automatically.	It does not allow teachers to view the scheduled timetable themselves unless the system administrator upload the timetable to WinJaw5 web.
Object-oriented approach on the design of constraints by using the attaching technique of classes to the created constraints.	Longer workflow or time taken for the system administrator to print out the timetables to distribute among teachers.
It provides a structured data of the generated timetables for the system administrators to do analysis.	It is also stated in the project problem statement that WinJaw5 uses deterministic algorithm which is not effective to solve clashing problems.

CHAPTER 3

DEVELOPMENT METHODOLOGY & TOOLS

3.1 Introduction

It is utmost important to plan and implement an organized methodology to develop a system. By following the properly designed development guidelines, it ensures what the things that need to be done in each of the development phases. Not just that, every progress in development process will be recorded, so that version control can be done on the developing system to trace back on what and when of components that has been added to the system feature. This is because we will not remember everything that we added to the system deep into the system development, especially when the system is near completion, and it is already very complex. Other than that, tasks must be broken down into as small as possible to clearly understand what needs to be done in each module in the system. This is due to the fact that the task will be unachievable and too large in scale if tasks are decided by only modules, but not each small components in the modules. Then, there are modules that has higher priority to finish earlier than other modules, else the dependent modules cannot be developed as it requires some function from other modules. Therefore, a schedule will be created for the project as well as the development methodology with application in each phase of production and supporting tools for the project.

3.2 Project Methodology

This project will be using the Scrum methodology to develop STSAMS. However, the project team as well as the development team are clueless when it comes to tailor a software process according to Xu, et al. (2008) as there are no proper guidelines on the practice of the Scrum methodology. So, Akbar (2019) has proposed his framework for customizing software development processes that is specifically agile based. In his work, the process framework is formulated by him mentioned that the sub-processes of the three key processes such as resource management, communication, and requirement management can be tailored to fit each project as revealed in Figure 3.2.1. In this case, the main development phases of the project will be planning, design, implementation, and deployment.

First and foremost, the **planning phase** of this project can be including the three key processes mentioned by Akbar (2019). For resource management, a schedule will be drafted along with the work breakdown structure of the entire project. This is because the schedule will have the information regarding the resources required such as time and cost for each project tasks. The sub-processes in the schedule with the generally adopted project life cycles only can be modified to fit this project's needs and requirements. To further illustrate, web application developments and system migrations may have the same project life cycle such as initiation, planning, execution, and closure but each smaller tasks that need to be done can be completely varies as their system requirements are different. Not just that, it is important to involve the users in the planning phase of the project which is the earliest stage as this is the phase that requires minimum cost for each system requirement modifications (Saif, et al., 2021). So, the system requirements can be formulated by studying the system manuals, testing the existing systems, holding surveys and virtual interview on the teachers and system administrators of a public school in Malaysia. To ease the communication of the project, the gathered requirements can be organized in a requirements traceability matrix.

Furthermore, the **implementation and deployment phases** of the project can be combined with each iteration of project using the Scrum methodology. For example, Schwaber (2004) has shown the detail Scrum flow for software development using Figure 3.2.2. The product backlog in the Scrum methodology can be seen as the system requirements for the STSAMS which is defined in the planning phase of the project. However, planning phase can also be done or repeated at every time in the project development as emerging or new system requirements can be added into the product backlog. When there are new requirements added into the product backlog, the requirement modelling for STSAMS needs to be updated during the **designing phase**, so it reflects the latest update of the system. Not just that, version control must be done on the models to show the changes made to each model updates.

Last but not least, before each iteration of Scrum, components of the system that needs to be done can be selected from the product backlog into the sprint backlog. Then, sprints will be initiated to complete each requirement and user interface for STSAM that are stated in the sprint backlog. After each sprint are completed, each newly developed functionalities are to be demonstrated after deployment to the project supervisor to ensure the quality of the system. There are solution two outcomes of each iteration. Firstly, the next iteration will be start instantly if the previous sprint is completed as there will be less coordination because of the one-man team. Next, if there are items that is not completed in time in previous iterations, it can be nominated as a candidate for the next sprint. These processes will be done iteratively until the STSAMS is completed development and ready for full deployment.

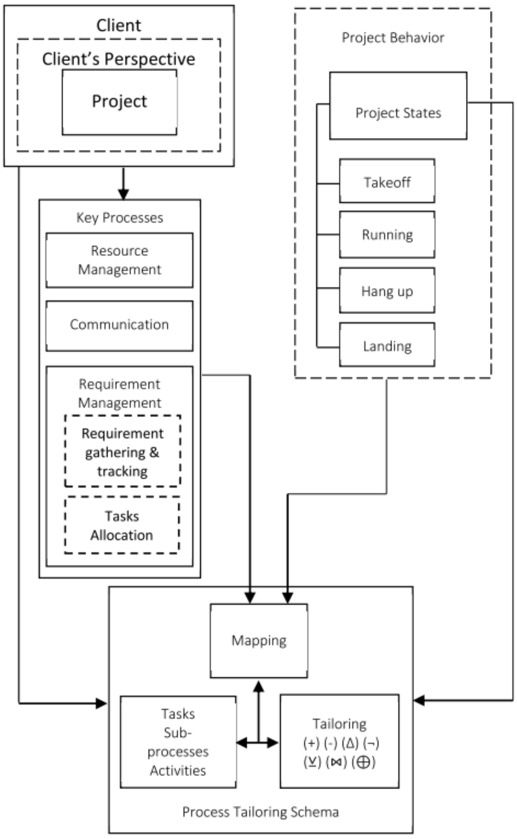


Figure 3.2.1: Theoretical process tailoring framework.

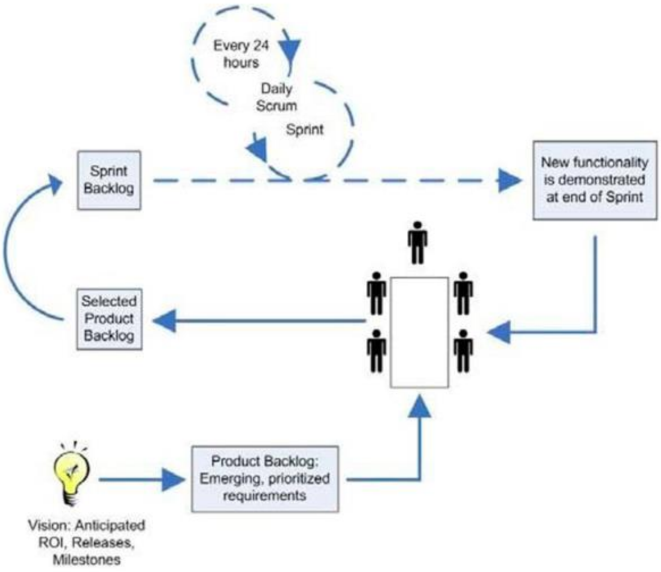
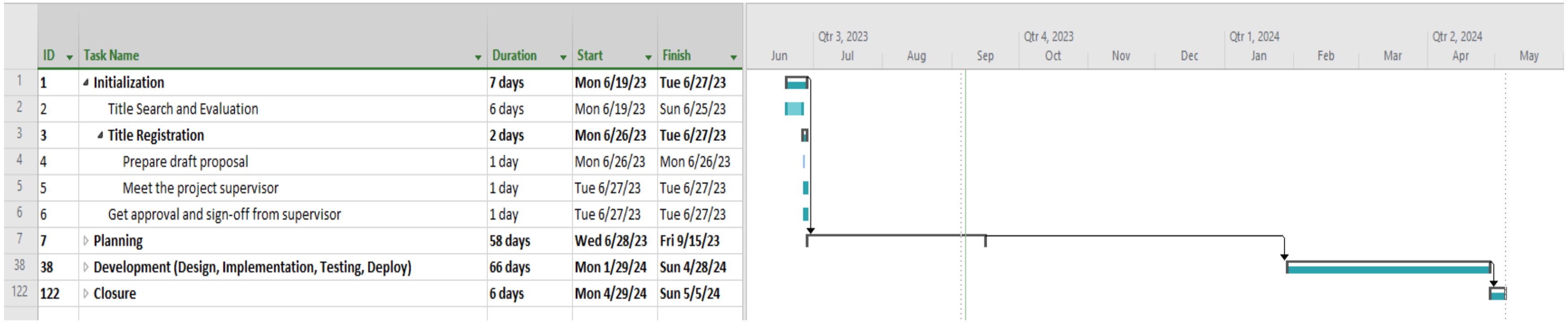

Every 24 hours
Daily Scrum
Sprint
Sprint Backlog
New functionality is demonstrated at end of Sprint
Selected Product Backlog
Product Backlog: Emerging, prioritized requirements
Vision: Anticipated ROI, Releases, Milestones

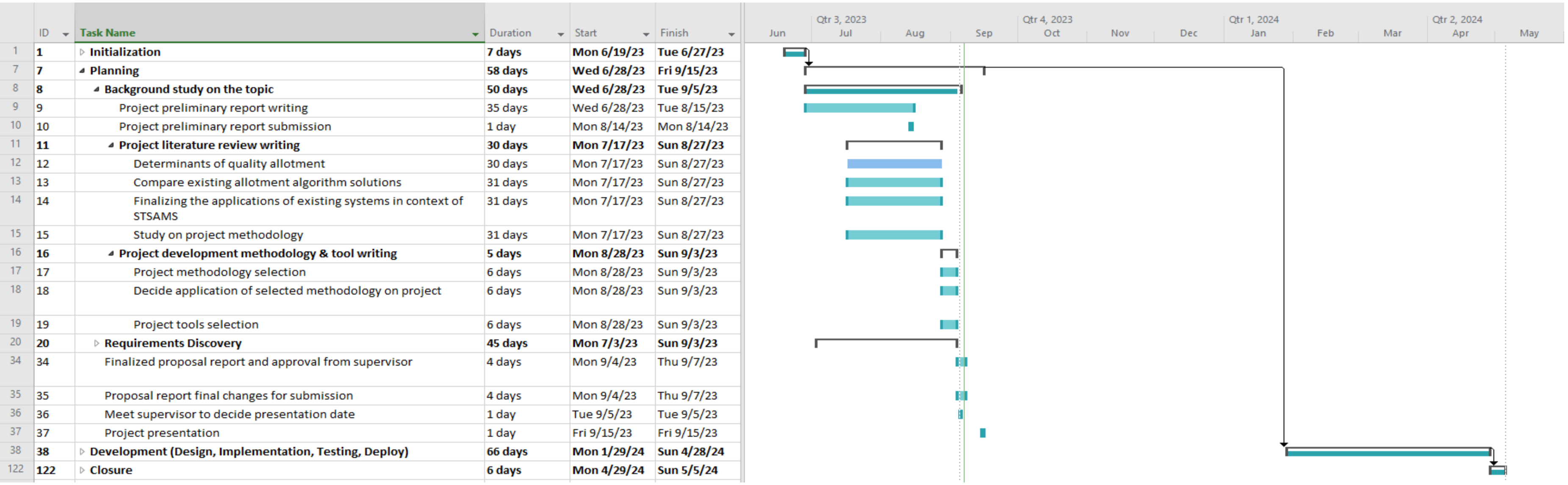
Figure 3.2.2: Scrum flow for software development.

3.3 Project Management Plan & Schedule



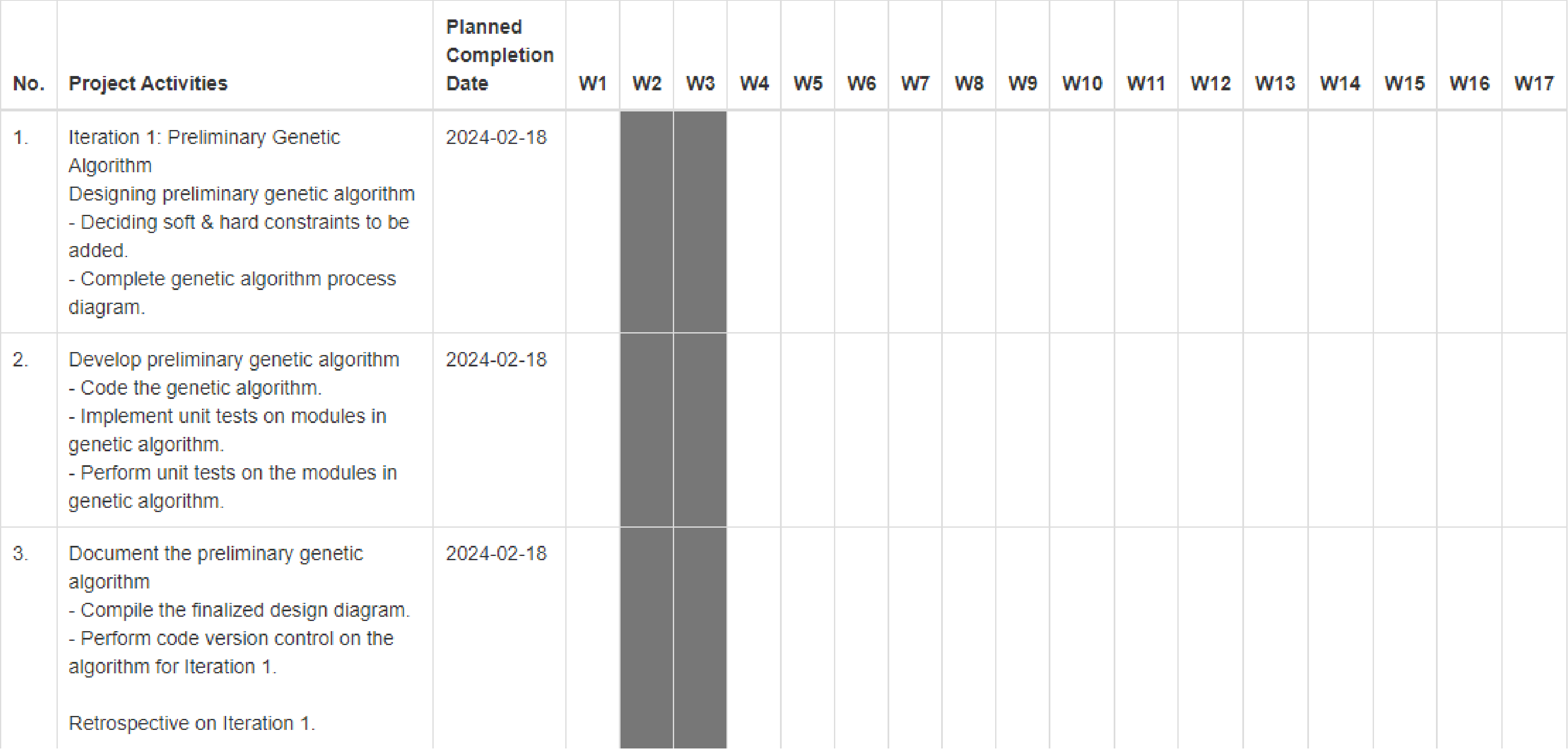
	ID	Task Name	Duration	Start	Finish
1	1	◢ Initialization	7 days	Mon 6/19/23	Tue 6/27/23
2	2	Title Search and Evaluation	6 days	Mon 6/19/23	Sun 6/25/23
3	3	◢ Title Registration	2 days	Mon 6/26/23	Tue 6/27/23
4	4	Prepare draft proposal	1 day	Mon 6/26/23	Mon 6/26/23
5	5	Meet the project supervisor	1 day	Tue 6/27/23	Tue 6/27/23
6	6	Get approval and sign-off from supervisor	1 day	Tue 6/27/23	Tue 6/27/23
7	7	▷ Planning	58 days	Wed 6/28/23	Fri 9/15/23
38	38	▷ Development (Design, Implementation, Testing, Deploy)	66 days	Mon 1/29/24	Sun 4/28/24
122	122	▷ Closure	6 days	Mon 4/29/24	Sun 5/5/24

Figure 3.3.1: Schedule focusing on Initialization Phase.



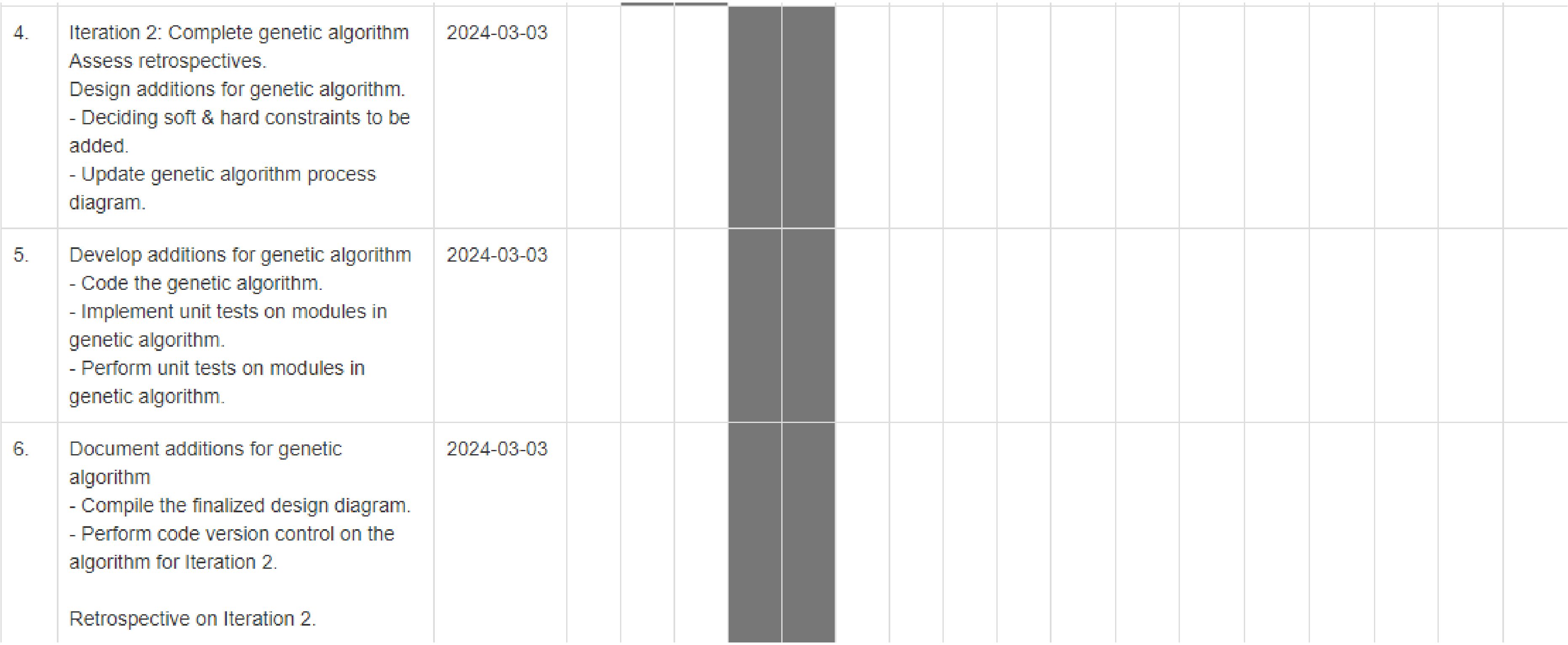
	ID	Task Name	Duration	Start	Finish
1	1	▷ Initialization	7 days	Mon 6/19/23	Tue 6/27/23
7	7	◢ Planning	58 days	Wed 6/28/23	Fri 9/15/23
8	8	◢ Background study on the topic	50 days	Wed 6/28/23	Tue 9/5/23
9	9	Project preliminary report writing	35 days	Wed 6/28/23	Tue 8/15/23
10	10	Project preliminary report submission	1 day	Mon 8/14/23	Mon 8/14/23
11	11	◢ Project literature review writing	30 days	Mon 7/17/23	Sun 8/27/23
12	12	Determinants of quality allotment	30 days	Mon 7/17/23	Sun 8/27/23
13	13	Compare existing allotment algorithm solutions	31 days	Mon 7/17/23	Sun 8/27/23
14	14	Finalizing the applications of existing systems in context of STSAMS	31 days	Mon 7/17/23	Sun 8/27/23
15	15	Study on project methodology	31 days	Mon 7/17/23	Sun 8/27/23
16	16	◢ Project development methodology & tool writing	5 days	Mon 8/28/23	Sun 9/3/23
17	17	Project methodology selection	6 days	Mon 8/28/23	Sun 9/3/23
18	18	Decide application of selected methodology on project	6 days	Mon 8/28/23	Sun 9/3/23
19	19	Project tools selection	6 days	Mon 8/28/23	Sun 9/3/23
20	20	▷ Requirements Discovery	45 days	Mon 7/3/23	Sun 9/3/23
34	34	Finalized proposal report and approval from supervisor	4 days	Mon 9/4/23	Thu 9/7/23
35	35	Proposal report final changes for submission	4 days	Mon 9/4/23	Thu 9/7/23
36	36	Meet supervisor to decide presentation date	1 day	Tue 9/5/23	Tue 9/5/23
37	37	Project presentation	1 day	Fri 9/15/23	Fri 9/15/23
38	38	▷ Development (Design, Implementation, Testing, Deploy)	66 days	Mon 1/29/24	Sun 4/28/24
122	122	▷ Closure	6 days	Mon 4/29/24	Sun 5/5/24

Figure 3.3.2: Schedule focusing on Planning Phase.



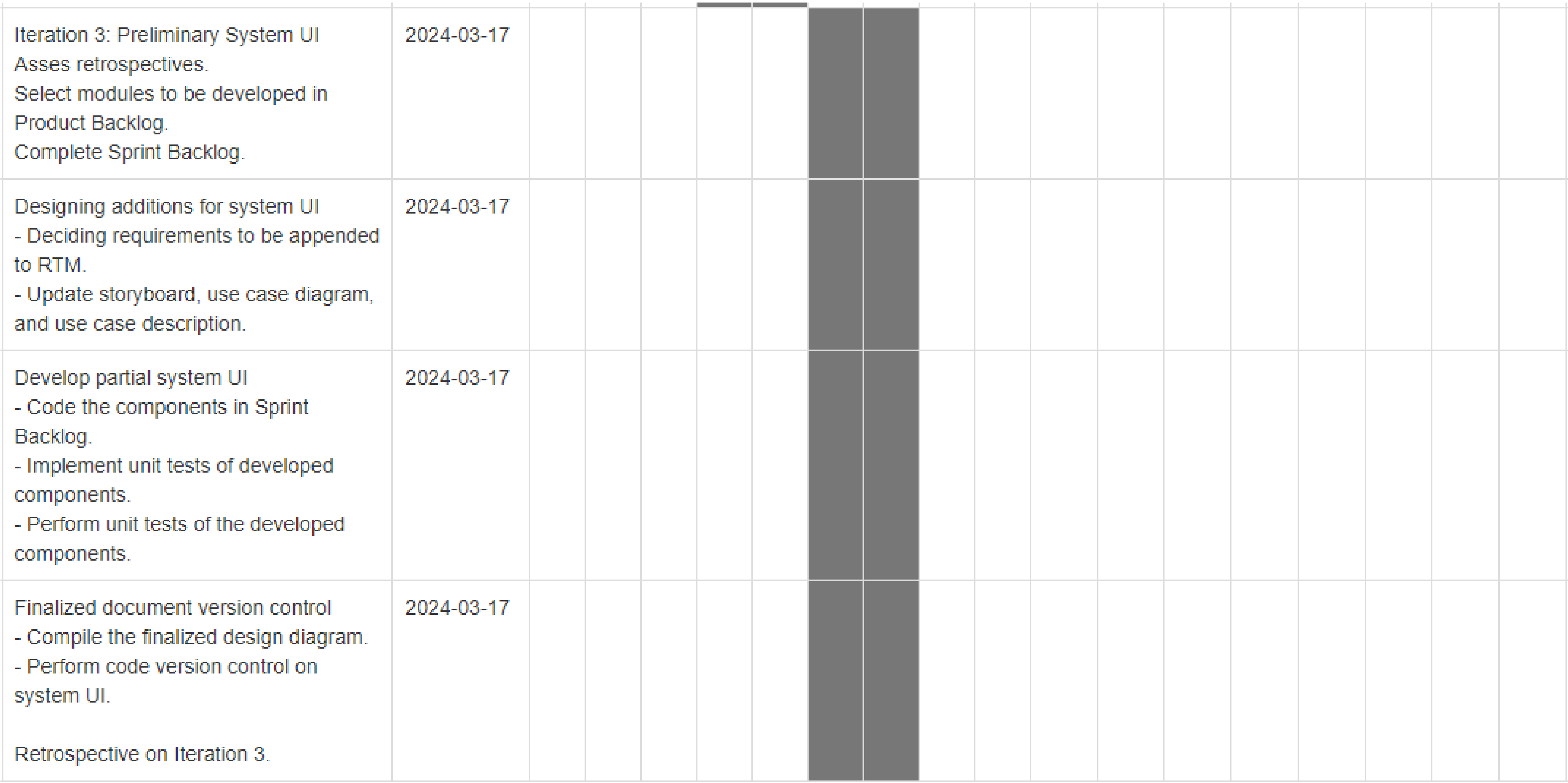
No.	Project Activities	Planned Completion Date	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14	W15	W16	W17
1.	Iteration 1: Preliminary Genetic Algorithm Designing preliminary genetic algorithm - Deciding soft & hard constraints to be added. - Complete genetic algorithm process diagram.	2024-02-18		■	■														
2.	Develop preliminary genetic algorithm - Code the genetic algorithm. - Implement unit tests on modules in genetic algorithm. - Perform unit tests on the modules in genetic algorithm.	2024-02-18		■	■														
3.	Document the preliminary genetic algorithm - Compile the finalized design diagram. - Perform code version control on the algorithm for Iteration 1. Retrospective on Iteration 1.	2024-02-18		■	■														

Figure 3.3.3: Schedule focusing on Iteration 1 in Development Phase.



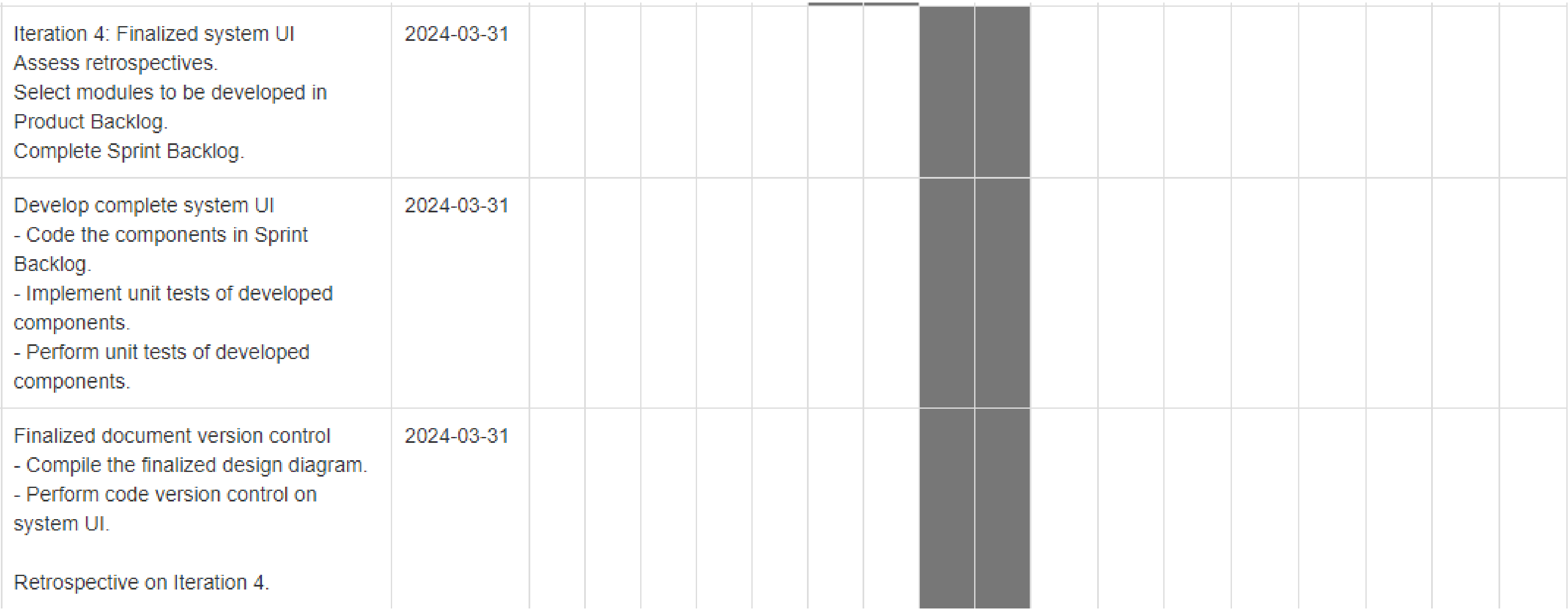
4.	Iteration 2: Complete genetic algorithm Assess retrospectives. Design additions for genetic algorithm. - Deciding soft & hard constraints to be added. - Update genetic algorithm process diagram.	2024-03-03				■	■												
5.	Develop additions for genetic algorithm - Code the genetic algorithm. - Implement unit tests on modules in genetic algorithm. - Perform unit tests on modules in genetic algorithm.	2024-03-03				■	■												
6.	Document additions for genetic algorithm - Compile the finalized design diagram. - Perform code version control on the algorithm for Iteration 2. Retrospective on Iteration 2.	2024-03-03				■	■												

Figure 3.3.4: Schedule focusing on Iteration 2 in Development Phase.



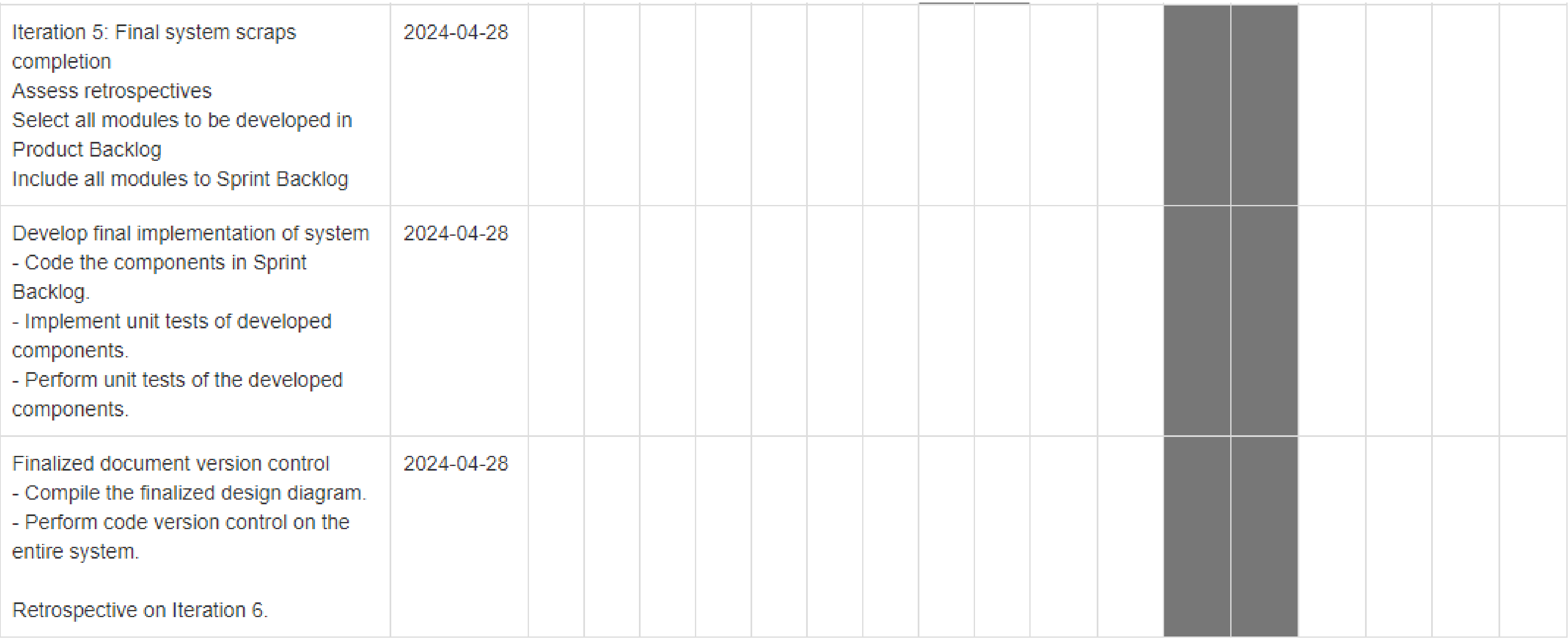
Iteration 3: Preliminary System UI Asses retrospectives. Select modules to be developed in Product Backlog. Complete Sprint Backlog.	2024-03-17						▨	▨										
Designing additions for system UI - Deciding requirements to be appended to RTM. - Update storyboard, use case diagram, and use case description.	2024-03-17						▨	▨										
Develop partial system UI - Code the components in Sprint Backlog. - Implement unit tests of developed components. - Perform unit tests of the developed components.	2024-03-17						▨	▨										
Finalized document version control - Compile the finalized design diagram. - Perform code version control on system UI. Retrospective on Iteration 3.	2024-03-17						▨	▨										

Figure 3.3.5: Schedule focusing on Iteration 3 in Development Phase.



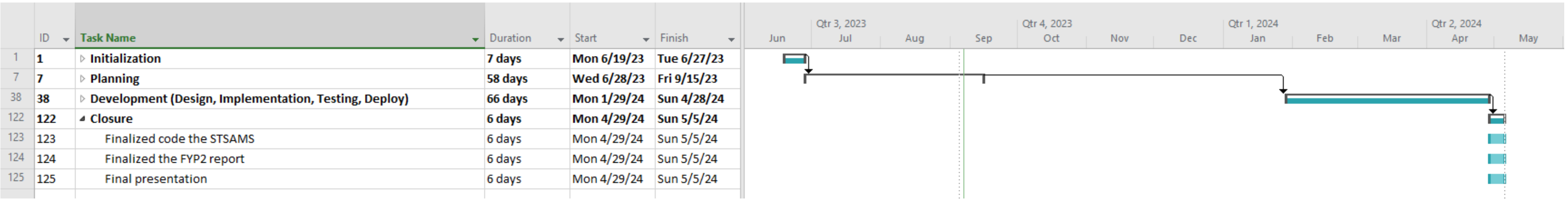
Iteration 4: Finalized system UI Assess retrospectives. Select modules to be developed in Product Backlog. Complete Sprint Backlog.	2024-03-31								▨	▨								
Develop complete system UI - Code the components in Sprint Backlog. - Implement unit tests of developed components. - Perform unit tests of developed components.	2024-03-31								▨	▨								
Finalized document version control - Compile the finalized design diagram. - Perform code version control on system UI. Retrospective on Iteration 4.	2024-03-31								▨	▨								

Figure 3.3.6: Schedule focusing on Iteration 4 in Development Phase.



Iteration 5: Final system scraps completion Assess retrospectives Select all modules to be developed in Product Backlog Include all modules to Sprint Backlog	2024-04-28												■	■				
Develop final implementation of system - Code the components in Sprint Backlog. - Implement unit tests of developed components. - Perform unit tests of the developed components.	2024-04-28												■	■				
Finalized document version control - Compile the finalized design diagram. - Perform code version control on the entire system. Retrospective on Iteration 6.	2024-04-28												■	■				

Figure 3.3.7: Schedule focusing on Iteration 5 in Development Phase.



	ID	Task Name	Duration	Start	Finish
1	**1**	▷ **Initialization**	**7 days**	**Mon 6/19/23**	**Tue 6/27/23**
7	**7**	▷ **Planning**	**58 days**	**Wed 6/28/23**	**Fri 9/15/23**
38	**38**	▷ **Development (Design, Implementation, Testing, Deploy)**	**66 days**	**Mon 1/29/24**	**Sun 4/28/24**
122	**122**	◢ **Closure**	**6 days**	**Mon 4/29/24**	**Sun 5/5/24**
123	123	Finalized code the STSAMS	6 days	Mon 4/29/24	Sun 5/5/24
124	124	Finalized the FYP2 report	6 days	Mon 4/29/24	Sun 5/5/24
125	125	Final presentation	6 days	Mon 4/29/24	Sun 5/5/24

Figure 3.3.8: Schedule focusing on Closure Phase.

3.4 Verification / Test Plan

3.4.1 Objectives

The objective of testing the application is to:

1. To reduce the risk arising from the bugs in the system under test.
2. To find the existence of the bugs in the application so that we can acquire information to fixing the bugs.
3. To build confidence in the level of quality of the application.
4. To ensure that the API routes are called properly in the STSAMS frontend.
5. To verify that the allocation algorithm for STSAMS is working with specified constraints.
6. To reduce the logic error by testing the logic of the functions in the API server.

3.4.2 Scope

The scope refers to the breadth and depth of testing that will perform on the application. Test scope in this application is:

1. Functionality in STSAMS frontend: Functionality refers to the specific functionality of the application to be tested including features and workflows.
   - Routing helper functions to communicate with the API server.
   - Utility functions: The other functionalities that do additional logic operations in the frontend.
2. Functionality in STSAMS backend.
   - Route functions: The functions that will receive requests from the frontend, do necessary operations on the database, and send a response back to the frontend as feedback.
3. Constraints in STSAMS allocation algorithm: Constraints refers to the chosen conditions that need to be considered during the timetable allocations to produce varied results.

3.4.3 Test Items

3.4.3.1 Frontend Web Pages

1. Allotment Sets Page
2. Announcements Page
3. Classes Page
4. Static Timeslot Page
5. Study Hours Page
6. Subjects Page
7. Subject Distributions Page
8. Teachers Page
9. Teacher Preferences Page
10. Timetable Generations Page
11. Venues Page
12. Venue Usages Page

3.4.3.2 Backend API Routes

1. Allotment Sets
2. Announcements
3. Classes
4. Dynamic Timetables
5. Static Timetables
6. Static Timetabling Algorithm
7. Study Hours
8. Subject Distributions
9. Subjects
10. Teachers
11. Teaching Classes
12. Venue Usages
13. Venues

3.4.4 Features to be tested.

3.4.4.1 Allotment Sets

- postAllotmentSet function which sends JSON data to the API server to create a new allotment set.

- getAllotmentSet function which sends ID to the API server to receive the allotment set details with the specified ID.

- getAllotmentSets function which receives all allotment sets from the API server.

- deleteAllotmentSet function which sends ID to the API server to delete allotment set with the specified ID.

3.4.4.2 Annoucements

- postAnnouncement function which sends JSON data to the API server to create a new announcement.

- getAnnouncements function which receives all announcements from the API server.

- getAnnouncement function which sends ID to the API server to receive the announcement details with the specified ID.

- putAnnouncement function which sends ID and JSON data to the API server to update existing announcement with the specified ID.

- deleteAnnouncement function which sends ID to the API server to delete announcement with the specified ID.

3.4.4.3 Classes

- postClass function which sends JSON data to the API server to create a new class.

- getClasses function which receives all classes from the API server.

- getClass function which sends ID to the API server to receive the class details with the specified ID.

- putClass function which sends ID and JSON data to the API server to update existing class with the specified ID.

- deleteClass function which sends ID to the API server to delete class with the specified ID.

- handleSorting function which switch the sorting modes for a specific table column name.

- handleFilterAndSort function which filters and sorts out classes according to the sorting mode.

3.4.4.4 Static Timeslots

- postStaticTimeslot function which sends JSON data to the API server to create a new static timeslot.

- getStaticTimeslots function which receives all static timeslots from the API server.

- getStaticTimeslot function which sends ID to the API server to receive the static timeslot details with the specified ID.

- putStaticTimeslot function which sends ID and JSON data to the API server to update existing static timeslot with the specified ID.

- deleteStaticTimeslot function which sends ID to the API server to delete static timeslot with the specified ID.

3.4.4.5 Study Hours

- postStudyHour function which sends JSON data to the API server to create a new study hour.

- getStudyHours function which receives all study hours from the API server.

- putStudyHour function which sends ID and JSON data to the API server to update existing study hour with the specified ID.

- deleteStudyHour function which sends ID to the API server to delete study hour with the specified ID.

3.4.4.6 Subjects

- postSubjects function which sends JSON data to the API server to create a new subject.

- getSubjects function which receives all subjects from the API server.

- getSubject function which sends ID to the API server to receive the subject details with the specified ID.

- putSubject function which sends ID and JSON data to the API server to update existing subject with the specified ID.

- deleteSubject function which sends ID to the API server to delete subject with the specified ID.

- handleSorting function which switch the sorting modes for a specific table column name.

- handleFilterAndSort function which filters and sorts out subjects according to the sorting mode.

3.4.4.7 Subject Distributions

- postDistribution function which sends JSON data to the API server to create a new subject distribution.

- getDistributions function which receives all subject distributions from the API server.

- putDistribution function which sends ID and JSON data to the API server to update existing subject distribution with the specified ID.

- deleteDistribution function which sends ID to the API server to delete subject distribution with the specified ID.

- validateInputData function which will if check name, subjects and classes are empty.

3.4.4.8 Teachers

- postTeacher function which sends JSON data to the API server to create a new teacher.

- getTeachers function which receives all teachers from the API server.

- getTeacherDetails function which sends ID to the API server to receive the teacher details with the specified ID.

- putTeacher function which sends ID and JSON data to the API server to update existing teacher with the specified ID.

- deleteTeacher function which sends ID to the API server to delete teacher with the specified ID.

- getTeachingSubjects function which sends ID to the API server to receive the teaching subjects of a teacher with the specified ID.

- putTeachingSubjects function which sends ID and JSON data to the API server to update existing teaching subjects of a teacher with the specified ID.

- postEmail function which sends JSON data to the API server to create to send an email.

- getTeacher function which sends access token to the API server to receive the first-time login teacher details.

- teacherLogin function which sends credentials to the API server to authenticate a teacher login.

3.4.4.9 Teacher Preferences

- getPreferences function which sends ID to the API server to receive the teacher preferences with the specified ID.

- putPreferences function which sends ID and JSON data to the API server to update existing teacher preferences with the specified ID.

- getTeachingClasses function which sends ID to the API server to receive the teaching classes of a teacher with the specified ID.

- getOrdinal function which receives an integer then return the ordinal of the integer.

3.4.4.10 Timetable Generations

- generateStaticTimetable function which sends the timetabling settings to the API server to generate and receive generated static timetable.

- getStaticProgress function which receives the current generation progress of the static timetable.

- postStaticTimetable function which sends the generated static timetable to the API server to save it.

- generateDynamicTimetable function which sends the timetabling settings to the API server to generate and receive generated dynamic timetable.

- getDynamicProgress function which receives the current generation progress of the dynamic timetable.

- getConsoleMessages function which receives the console messages of the generating dynamic timetable.

- postDynamicTimetable function which sends the generated dynamic timetable to the API server to save it.

3.4.4.11 Utilities

- generateDigitCode function which generate a randomized 6-digit code.

- generatePassword function which generate a randomized default password with inputted length of the required password.

- randomNumberInRange function which generate a random number between two inputted minimum and maximum values.

3.4.4.12 Venues

- postVenues function which sends JSON data to the API server to create a new venue.

- getVenues function which receives all venues from the API server.

- getVenue function which sends ID to the API server to receive the venue details with the specified ID.

- putVenue function which sends ID and JSON data to the API server to update existing venue with the specified ID.

- deleteVenue function which sends ID to the API server to delete venue with the specified ID.

- handleSorting function which switch the sorting modes for a specific table column name.

- handleFilterAndSort function which filters and sorts out venues according to the sorting mode.

3.4.4.13 Venue Settings

- getVenueUsages function which receives all venue usages from the API server.

- putVenueUsages function which sends ID and JSON data to the API server to update existing venue usages with the specified ID.

3.4.5 Features not to be tested.

- Components rendered by web pages.

- Utility functions for allocation genetic algorithm.

- Features other than specified features that are to be tested.

3.4.6 Entry Criteria

1. Test team readiness - The testing team at this point are prepared to perform software testing tasks. The member in the project is all equipped with the knowledge and skillsets to create test cases, use automation tools.

2. Completed code development – The software in question has been completed and is ready for testing.

3. Test environment readiness - The test environment for the software is properly set up which includes the installation of the test automation tools for faster test execution.

5. Test data availability - The test data such as the input data, expected results and the scenarios are made available and validated.

3.4.7 Exit Criteria

1. All the planned tests have been carried.

2. The software has reached the required level of quality as determined by the team member.

3. All the required documentation needed for the project has been completed, such as the test cases.

4. Reached the final phase of the project life cycle.

3.5 Project Tools

3.5.1 Hardware

A laptop computer will be used in developing STSAMS which includes the following hardware.

Table 3.5.1.1: Components of laptop computer.

Components	**Specifications**
Model	Dell Inspiron 15 3000
Processor	Intel® Core™ i5-1035G1
Operating System	Window 11 64-bit
Graphic	Intel® UHD Graphics
Memory	20GB RAM
Storage	512 GB SSD

3.5.2 Software

The main development software for STSAMS will be Visual Studio Code as it provides the correct format of indentations also built in with autocompletion for curly brackets as JavaScript will be used for this development. Other than that, VS Code also allows developers to add extension on language supports, local server hosting, and code snippets to autocomplete the lengthy syntaxes. Other software and tools used in this project are listed as below:

Table 3.5.2.1: Software and Tools to be used in the project.

Attributes	**Software/Tools**	**Justifications**
Document Preparations	MS Word	1. MS Word will be used in preparing the reports for the project including the proposal reports, and final report. 2. Other document related to STSAMS such requirement traceability matrix, use case description will be recorded in MS Word.
	Google Form	1. It is a free software to design questionnaires so that the survey for requirement discovery can be done virtually without printing papers.

		2. It also provides data analysis for the data collected from questionnaires using charts and spreadsheet.
	Draw.io	1. Draw.io is an open-source software that provide its users to do requirements modelling and draw storyboards. 2. It is also very easy to learn and beginner friendly as software such as Enterprise Architect requires practices to master it. 3. Reserve the time to learn modelling for developing the system.
	MS Project	1. It is used to create a schedule for this project. 2. It provides management of tasks with indentation and numbering. Other than that, information such as start and end date, completion percentage, and cost can also be added.
Frontend Languages	HTML	1. Add elements to the website. 2. Form functionality for login, searching, basically the user interfaces for CRUD operations on managing courses, subjects, teacher's information, and admin's profile. For example, multiple components in React project that can be used such as view, button, and text.
	CSS	1. Beautify the contents, strife for usability concept in designing the user interface. 2. Provide support for multiple devices such as desktop, laptop, tablet, mobile phone with different display resolution which is a responsive web design.
	JavaScript	1. JavaScript is main language that is being used in React projects. It can be used in most operations of the system such as data validation, SQLite database connections, SQL querying.

		2. Capture user events on for example buttons to do tasks, such as adding, updating new contents, and deleting existing contents. 3. JavaScript is also used to manage states in components using the setState, componentDidMount, componentWillUpdate, and componentWillUnmount methods.
Database	SQLite	1. SQLite is an embedded, server-less relational database. So, an external server is not required to host the database for the website as the local data storage is built into the application.
	MySQL	1. MySQL is the query language will used in pair with SQLite to do operations on the database such as defining table data fields and constraints, inserting, updating, and deleting data.
Libraries	React	1. React is an open-source library, so there are various custom-built components that can be used in the development of STSAMS. 2. Other than that, the React library is designed to be used in web application development, and it provides the easy way to convert a web application to mobile app just in case it needs to be built in mobile application.
	Jest	1. Jest is open source and suitable for this project as it is a framework for JavaScript unit testing.

CHAPTER 4

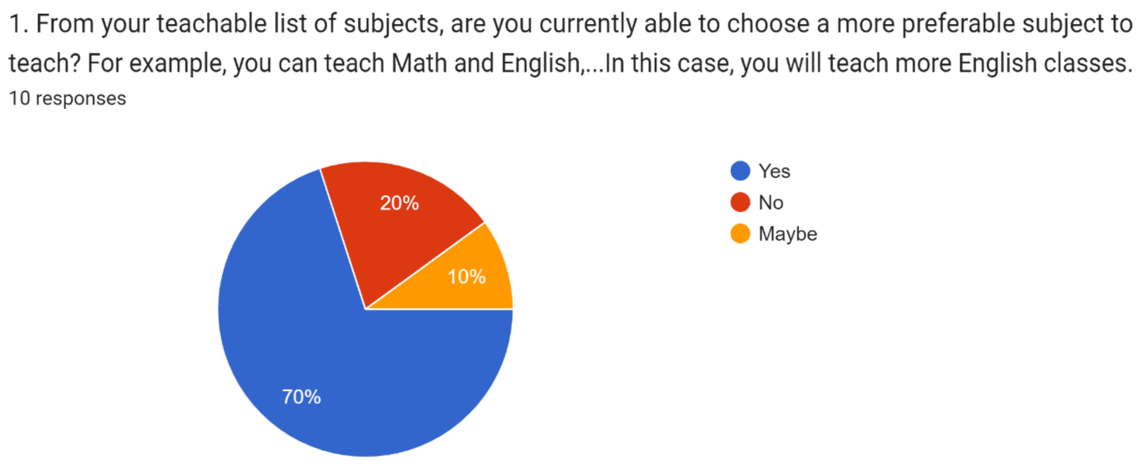
PROJECT SPECIFICATIONS

In this chapter, we will be discussing the requirement discovery techniques, proposed initial system design using requirement specifications and requirement modelling, as well as the system flow of the entire STSAMS.

4.1 Requirement Discovery

4.1.1 Questionnaires

To understand the preferences of teachers regarding the STSAMS, surveys have been distributed to 10 teachers in SMK Taman Ehsan to obtain their opinions on the school's current system and workflow. The following shows the questions asked, the answers provided by the teachers, as well as the conclusions that can be done on those responds: -



1. From your teachable list of subjects, are you currently able to choose a more preferable subject to teach? For example, you can teach Math and English,...In this case, you will teach more English classes.

10 responses

Figure 4.1.1.1: Survey Question 1.

From the results of this question, we can know that majority of teachers are able to select their preferences of teaching. However, the timetable system, WinJaws5 that being used by the school is not actually providing the section for the teachers to set the teaching subjects priorities to their likings. For example, this result is showing that teachers are able to choose few subjects from their teachable list of subjects, but not able to set the priorities between them. The priority will decide whether a subject will occur more often than other lower priority subjects in a timetable. Then, it is a responsibility of the system administrators to key in the teaching subjects given by the teacher to the system. Using this approach, it requires a lot of communication between teachers and system administrators, hence it is wiser to automate this process by allowing the teacher to directly set their teaching priorities to the constraint of the timetabling algorithm. By the time the system administrators are generating a new timetable, the priorities set by the teachers can be directly fetched from the database without any communications needed.

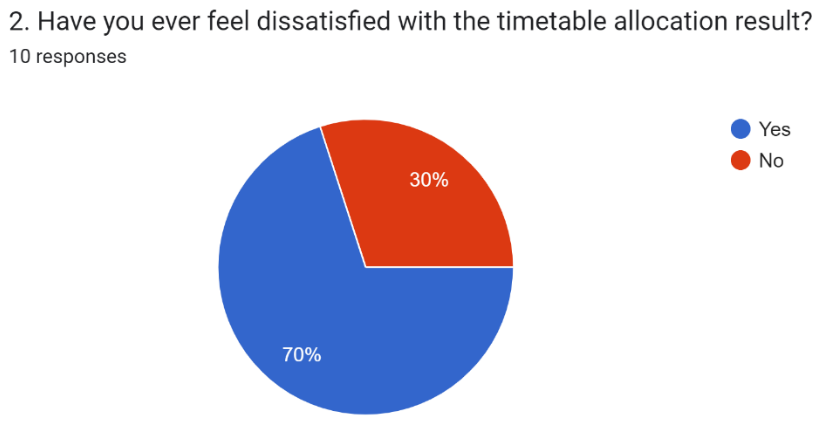


Figure 4.1.1.2: Survey Question 2.

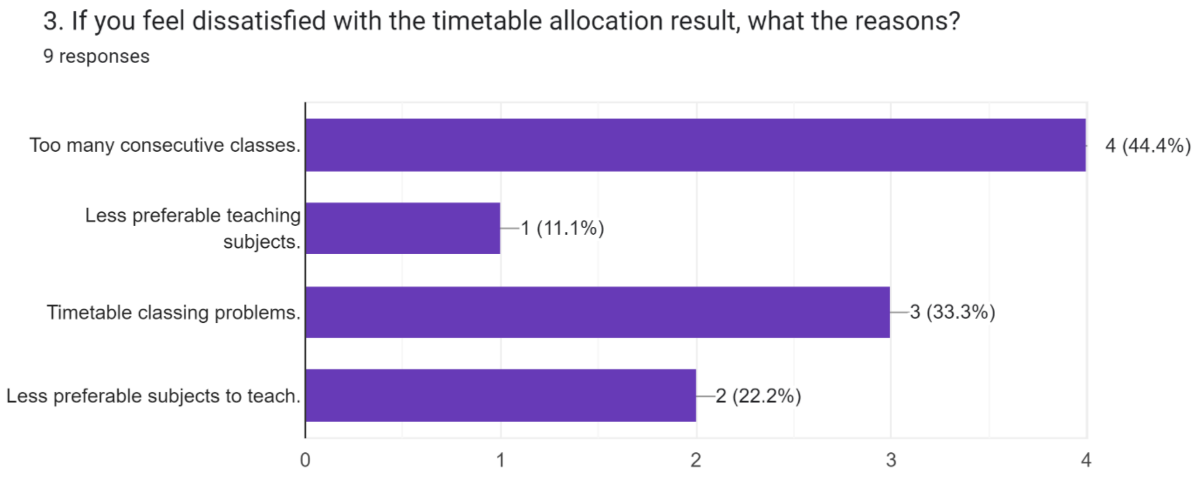


Figure 4.1.1.3: Survey Question 3.

From Figure 4.1.1.2, we found that 70% of teachers that participated in this survey ever felt dissatisfied with the timetable allocation result. To clarify as to why there are 10 answers in question 3 shown in Figure 4.1.1.3, because it allows the teachers to pick more than one answer. We can know that the most chosen reason by the teachers is having too many consecutive classes from Figure 4.1.1.3. This is because there are few gaps in between classes for the teachers to take a break. As mentioned in the literature review, it is tiring for the teachers to have high cognitive subjects consecutively as they require more preparation and thinking power. Then, there is an equal score of 3 teachers for both timetable clashing problems and less preferrable subjects to teach. The timetable clashing problems are the result of the usage of deterministic method for the timetabling algorithm as mentioned with details in Section 1.2.4. As for the problem with less preferrable subjects to teach, it is also a priority of this project to implement the constraint for the timetabling algorithm to give teachers more freedom to choose as mentioned in Section 1.2.1. The followings are the functional requirements included in STSAMS: -

Table 4.1.1.1: Problems converted as functional requirements of STSAMS.

Problems	**Functional Requirement ID**
Too many consecutive classes.	FR49
Timetable classing problems.	FR48
Less preferable subjects to teach.	FR19

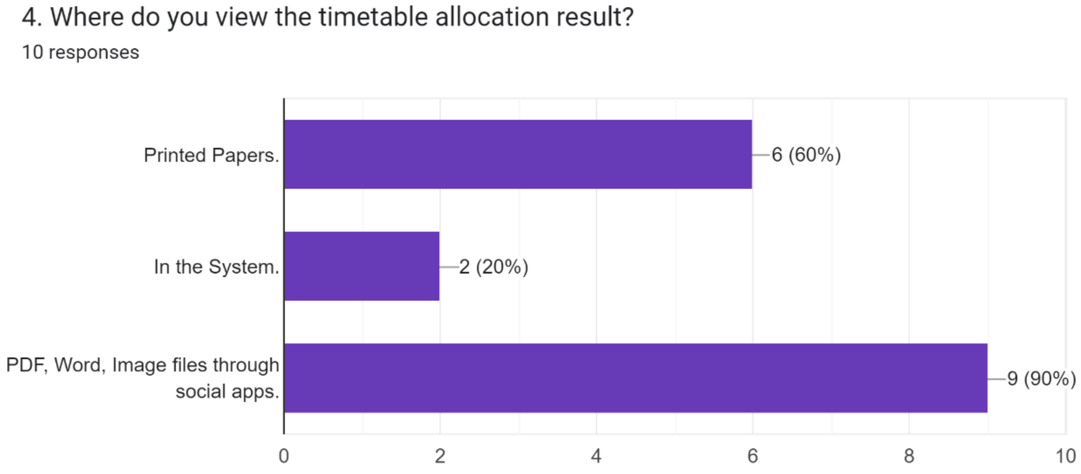
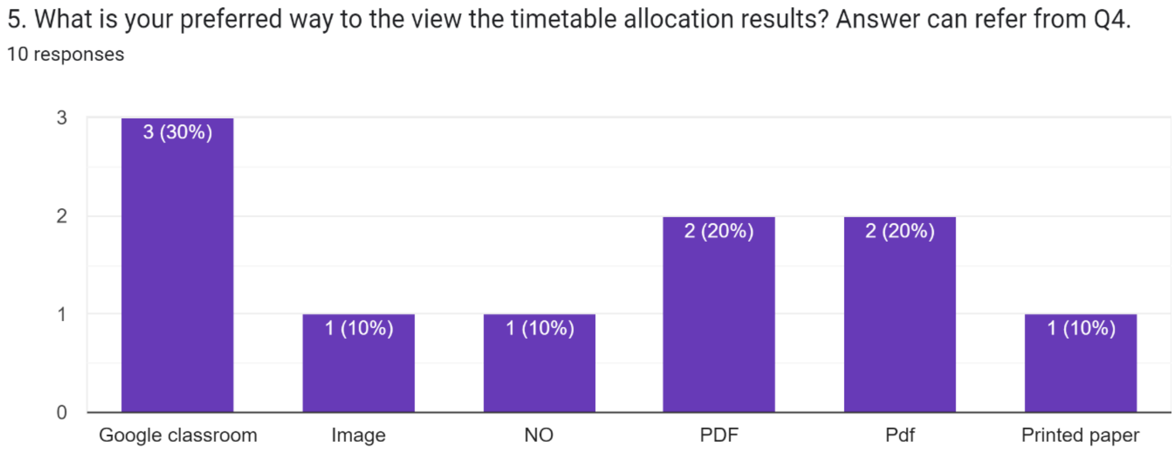


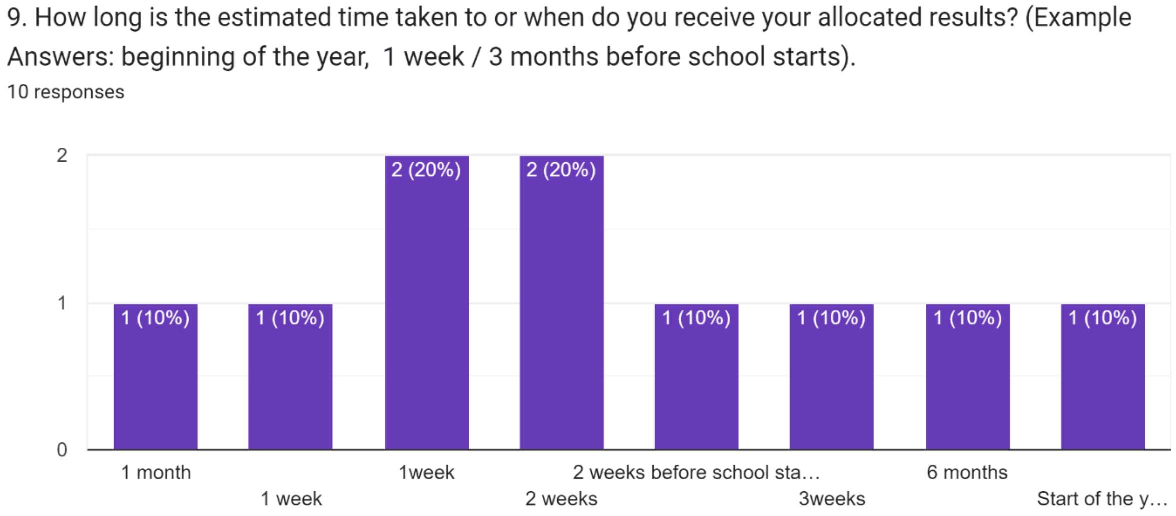
Figure 4.1.1.4: Survey Question 4.



5. What is your preferred way to the view the timetable allocation results? Answer can refer from Q4.

10 responses

Figure 4.1.1.5: Survey Question 5.



9. How long is the estimated time taken to or when do you receive your allocated results? (Example Answers: beginning of the year, 1 week / 3 months before school starts).

10 responses

Figure 4.1.1.6: Survey Question 9.

Similarly, question in Figure 4.1.1.4 also allow the teacher to give more than one answer. It is obvious that files are being sent through social apps by the system administrator to distribute the timetable allocation results as 9 out of the total 10 teachers voted for the "PDF, Word, Image files through social apps" option. Therefore, STSAMS should filter out the distribution process of the system administrators as results in Figure 4.1.1.6 revealed that it takes about 1 to 3 weeks for the timetable results to be passed to the teachers. As a resolution, the system should directly allow the results to be displayed to the teachers as soon as the system administrators have generated the timetables, so that the time taken for the timetables distribution will be as close as instant. To give more authority to the system administrators, they can decide if the generated timetables are displayed to the teacher as there might be mistakes occurring after generating the timetables. Then, both system administrators and teachers are able to view the timetables in the allotment module. This is because the teachers still prefer to view their timetables online as indicated in Figure 4.1.1.5. Long story short, the followings are the functional requirements included in STSAMS: -

Table 4.1.1.2: Problems converted as functional requirements of STSAMS.

Problems	**Functional Requirement ID**
Set timetables to be shown	FR51
Less preferable subjects to teach.	FR19

4.1.2 Virtual Interview

To get insight into the workflow of teacher-subject management in SMK Taman Ehsan, a system administrator has been chosen as a candidate to hold the virtual interview to know more about the system administrator's side of the perspective and workflow. From the questions asked, we can know that the school is using WinJaw5 to allocate the subjects. Then, surface-level information about the system has been collected from the interviewee. On top of that, an in-depth understanding and analysis of the system has been done by following the system manual prepared by WinJaws5 (n.d.) and testing the system itself in Section 2.3.2.

Table 4.1.2.1: Interview questions with answers from interviewee.

No.	Question Asked	Answers
1	Does the school use a software to manage allocation between teachers and subjects? If yes, what is the software name?	Yes, WinJaws5, but we are going to migrate to using AscTimetables.
2.	Does the system allocate subjects to teachers automatically (when generating timetable)?	Yes, but the adjustment after the generation is done manually.
3.	Do you face problems when you are generating timetables for teachers? Why? If not, what is the benefit the software provides?	Yes. WinJaw5 are not able to settle a lot of automatic problems like clashing problems. I must have to adjust the timetables manually after each generation.
4.	Does the software include AI to do data analysis? e.g., Teachers Subjects, and Allocations.	No.
5.	What are the criterions that is being considered by the software while allocating the subjects to teachers?	Teacher's Day Availability, Teacher's Hour Availability, Class's Hour Availability, Class's Empty Hours, Teacher Teaching Gaps between Classes, Maximum Teaching Hour per Week
6.	Does the replacement class allocation affect the original time schedule?	No.

4.2 Functional Requirements Specification

The functional requirements of STSAMS are organized in accordance with the requirements traceability matrix to ease tracing back of requirements from different requirement models such as the use case id, storyboard labelling, and the work breakdown structure of the project. The following is the functional requirements of STSAMS in an organized fashion:

Table 4.2.1: Requirement Traceability Matrix for STSAMS.

Unique Req. ID	**Requirement Description**		**Use Case ID**	**Status**	**Remarks**
Login Page					
FR1	The system should allow the system administrators and teachers to log into the system.		UC01	Done	-
FR2	The system should allow the system administrators and teachers to fill in their email and password to log in to the system.		UC01	Done	-
FR3	The system should validate the login credentials entered by the system administrators and teachers.		UC01	Done	-
FR4	The system should provide an option for the teachers to reset their passwords.		UC01	Done	-

FR5	The system should have an announcements section in the login page to display school announcements, and news.		-	Done	-
Forget Password Page					
FR6	The system should allow the teachers to reset their passwords.		UC02	Done	-
FR7	The system should allow the teachers to reset their password by validating through email addresses.		UC02	Done	-
FR8	The system should check if the verification code entered by the teachers is valid.		UC02	Done	
FR9	The system should allow the system teachers to send password reset requests to their email addresses bound to the system account to attain the verification code.		UC02	Done	-
FR10	The system should provide a new password and confirm new password input for the teachers to enter.		UC02	Done	-
FR11	The system should check the matching of the entered new password and confirm the new password.		-	Done	-

Create Teacher Page					
FR12	The system should allow the system administrators to create new teacher accounts for new teachers.		UC05	Done	-
FR13	The system should be able to set randomly generated default passwords for each newly created teacher account.		UC05	Done	-
First Login Page					
FR14	The system should prompt the teachers to complete their account profile for the first-time login.		UC01	Done	-
FR15	The system should allow the teachers to skip the profile completion process so that it can be set later.		-	Done	-
FR16	The system must prompt the teachers to change the default password to their new password.		UC01	Done	-

Teacher Profile Page					
FR17	The system should allow the teachers to update their profile credentials such as name, dob, and address		UC03	Done	-

FR18	The system should display the profile credentials of teachers.		UC03	Done	-
FR19	The system should allow the teachers to set their priority and sequence of subjects they preferred to teach.		UC03	Done	-
FR20	The system should display the sequence of subjects set by the teachers.		UC03	Done	-
Teacher Timetable Page					
FR21	The system should allow teachers to view the timetable or teacher-subject allocation results.		UC04	Done	-

View Teachers Page					
FR22	The system should allow the system administrators to view the details of teachers in a list.		UC06	Done	-
FR23	The system should display the list of teachers using the paging technique.		UC06	Done	-
FR24	The system should allow the system administrators to specify the number of teachers a page should display.		UC06	Done	-

FR25	The system should allow the system administrators to search teacher information by teacher’s name keyword.		UC06	Done	-
FR26	The system should allow the system administrators to sort the list of teachers by data fields alphabetically and numerically.		UC06	Done	-
FR27	The system should allow the system administrators to filter the list of teachers by data fields such as number range, and data types.		UC06	Done	-
View Teacher Details Page					
FR28	The system should display all the data fields related to a selected teacher.		UC07	Done	-
FR29	The system should allow the system administrators to enter the update state to make and save changes to the teacher information.		UC07	Done	-
FR30	The system should allow the system administrators to delete the teacher’s information.		UC07	Done	-
FR31	The system should allow the system administrators to delete the teacher’s account.		UC07	Done	-

View Subjects Page					
FR32	The system should allow the system administrators to view the details of subjects in a list.		UC08	Done	-
FR33	The system should display the list of subjects using the paging method.		UC08	Done	-
FR34	The system should allow the system administrator to specify the number of subjects a page should display.		UC08	Done	-
FR35	The system should allow the system administrator to search for subjects by subject name keyword.		UC08	Done	-
FR36	The system should allow the system administrators to sort the list of subjects by data fields alphabetically and numerically.		UC08	Done	-
FR37	The system should allow the system administrators to filter the list of teachers of subject types.		UC08	Done	-
FR38	The system should allow the system administrators to delete subjects.		UC08	Done	-

FR39	The system should display box to prompt for the confirmation of system administrators to delete subjects.		UC08	Done	-
FR40	The system should allow the system administrators to make updates the subjects data fields.		UC08	Done	-
View Class Page					
FR41	The system should display the list of created classes.		UC09	Done	-
FR42	The system should allow the system administrators to add new classes.		UC09	Done	-
FR43	The system should allow the system administrators to make changes to a selected class.		UC09	Done	-
FR44	The system should allow the system administrators to delete classes.		UC09	Done	-

Administrator Home Page					
FR45	The system should display dashboard showing the total of classes, subjects, and teachers.		-	Done	-

Subject Allotment Page					
FR46	The system should allow the system administrators to create multiple sets of allotments of teachers and subjects.		UC10	Done	-
FR47	The system should allow the system administrators to navigate to every existing allotment sets.		UC10	Done	-
FR48	The system should allow the system administrators to generate teacher-subject allotment results.		UC10	Done	The algorithm will be genetic algorithm which is a meta-heuristic type.
FR49	The system should allow the system administrators to set hard constraints as well as the soft constraints for the allotment algorithm to consider.		UC10	Done	-
FR50	The system should allow the system administrator to view the allocation results of multiple teachers.		UC10	Done	-
FR51	The system should allow the system administrator to toggle if the teacher could view the current set of allocation results.		UC10	Done	-
Announcement Management Page					

FR52	The system should allow the system administrator to add new announcements to the login page.		UC11	Done	-
FR53	The system should allow the system administrators to delete existing announcements.		UC11	Done	-
FR54	The system should allow the system administrators to update the contents in an announcement.		UC11	Done	-
FR55	The system should display all the created existing announcements to the system administrators.		UC11	Done	-
FR56	The system should give the system administrators to select which announcements to be selected to display on the login page.		UC11	Done	-

4.3 Non-Functional Requirements Specification

Table 4.3.1: Non-Functional Requirements of STSAMS.

Types	Non-Functional Requirements
Scalability	NFR01 – The STSAMS should be able to work with more than 15 subjects, classes, and teachers.
Security	NFR02 – The STSAMS will first check the verification code from users before changing new passwords. NFR03 – The STSAMS will prevent users from accessing unauthorized pages.
Usability	NFR04 – The STSAMS will display confirmation dialog before making any changes to their accounts or system data.
Portability	NFR05 – The STSAMS should allow its users to access the system using laptops and desktops through the web.
Compatibility	NFR06 – The STSAMS will work on multiple browsers such as Edge, Google Chrome, and OperaGX.

CHAPTER 5

SYSTEM DESIGN

In this chapter, we will be discussing the requirement modelling, as well as the system flow of the entire STSAMS.

5.1 Requirement Modelling

5.1.1 Use Case Diagram

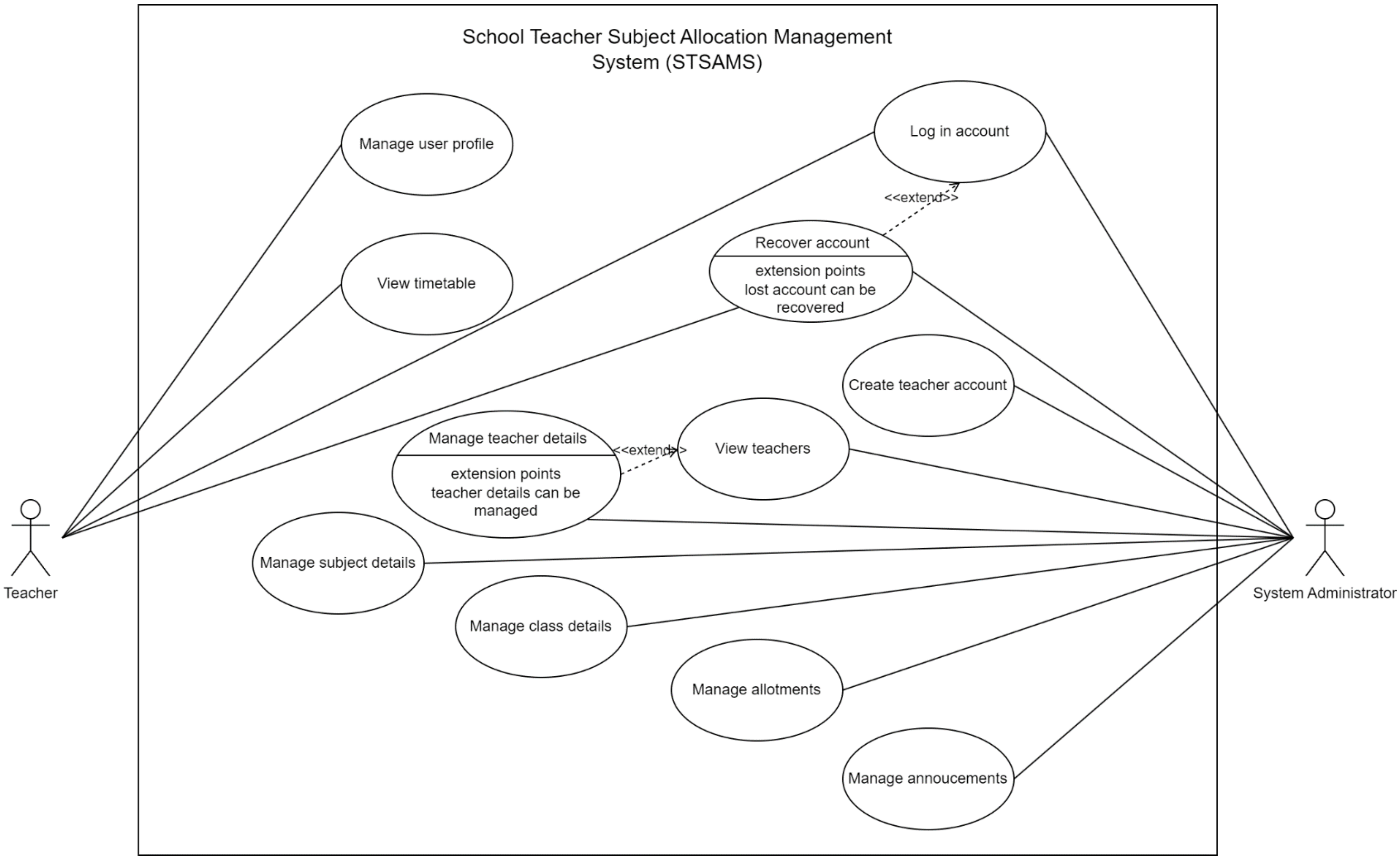


Figure 5.1.1.1: Use Case Diagram for STSAMS.

5.1.2 Use Case Descriptions

Use Case Name**: Log in account.**	ID**: UC01**	Importance Level**: High**
Primary Actor**: Teachers, System Administrators**	Use Case Type**: Detail, Essential**	
Stakeholders and Interests**:** **Teachers - logs in to STSAMS using a teacher account.** **System Administrator - logs in to STSAMS using system administrator account which is the super account with highest permission.**		
Brief Description**: This use case describes how account is logged in by different users.**		
Trigger**: Users (Teachers & System Administrators) want to log into STSAMS.**		
Relationships: Association**: Teachers, System Administrator** Include**: N/A** Extend**: UC02 – Recover account** Generalization**: N/A**		
Normal Flow of Events: **1. Users wants to log into STSAMS.** **2. Perform use case UC02-Recover account if teachers forget their password.** **3. Continue to Sub-flow 3.1 or 3.2.** **4. The system asks for the account ID and account password.** **5. Continue to Sub-flow 5.1 or 5.2.** **6. The user logged into STSAMS.**		
Sub-flows: **3.1 If the user is a teacher first logging in,** **3.1.1 The teacher logs into STSAMS using their email and the default password provided by system administrator.** **3.1.2 The teacher will be prompted to complete their profile and change default account password.**		

3.1.3 The teacher completes their user profiles and change new account passwords.

3.1.4 The teacher confirms on their changes made to their profile. Continue to Main Flow 2.

3.2 If the users are not first logging in, Continue to Main Flow 2.

5.1 If the login credentials are valid, the users will be brought to their respective home screens. Continue Main Flow 6.

5.2 If the login credentials are invalid, wrong password message will be displayed. Continue Main Flow 4.

Use Case Name**: Recover account.**	ID**: UC02**	Importance Level**: High**
Primary Actor**: Teachers**	Use Case Type**: Detail, Essential**	
Stakeholders and Interests**:** **Teachers - recovers their lost STSAMS teacher account.**		
Brief Description**: This use case describes how account is recovered by teachers.**		
Trigger**: Teachers want to recover their STSAMS accounts.**		
Relationships: Association**: Teachers** Include**: N/A** Extend**: N/A** Generalization**: N/A**		
Normal Flow of Events: **1. Users wants to recover their STSAMS account.** **2. Users choose the forget your password option to recover their account.** **3. Users are prompted by STSAMS to enter their email.** **4. STSAMS will send a verification code to the users through email.** **5. STSAMS prompts the users for verification code.** **6. The users enter the verification code.** **7. Continue to Sub-flow 8.1 or 8.2.** **8. STSAMS will set a new password for the user account.**		
Sub-flows: **8.1 If the verification code entered by users are valid,** **8.1.1 STSAMS will prompt the users for the new password.** **8.1.2 Users enter and confirm their new password for their account.** **8.2 If the verification code entered by users are invalid,**		

8.2.1 STSAMS will display invalid code message to the users. Continue to Main Flow 6.

<table>
<tr><td colspan="2">Use Case Name<b>: Manage user profile.</b></td><td>ID<b>: UC03</b></td><td>Importance Level<b>: High</b></td></tr>
<tr><td>Primary Actor<b>: Teachers</b></td><td colspan="3">Use Case Type<b>: Detail, Essential</b></td></tr>
<tr><td colspan="4">Stakeholders and Interests<b>:</b>
<b>Teachers – make changes to their information in their user profiles.</b></td></tr>
<tr><td colspan="4">Brief Description<b>: This use case describes how the teachers view and make changes to their information in their user profiles.</b></td></tr>
<tr><td colspan="4">Trigger<b>: Teachers want to make view and make modification on their user profiles.</b></td></tr>
<tr><td colspan="4">Relationships:
Association<b>: Teachers</b>
Include<b>: N/A</b>
Extend<b>: N/A</b>
Generalization<b>: N/A</b></td></tr>
<tr><td colspan="4">Normal Flow of Events:
<b>1. Teachers want to manage their user profiles.</b>
<b>2. STSAMS displays the user profiles to the teachers.</b>
<b>3. Continue to Sub-flow 3.1 or 3.2.</b>
4. <b>Teachers made changes to their user profiles.</b></td></tr>
<tr><td colspan="4">Sub-flows:
<b>3.1 If teachers want to edit the information in user profile,</b>
<b>3.1.1 Teachers choose to edit their user profile.</b>
<b>3.1.2 Teachers make changes to the information in their user profile.</b>
<b>3.1.3 Teachers choose to save to confirm their modification to the user profile. Continue to Main Flow 4.</b>
<b>3.2 If teachers want to set their preferences of subject to teach,</b>
<b>3.2.1 Teachers choose to set their preferences.</b>
<b>3.2.2 STSAMS displays the subjects priorities previously set by teachers.</b>
<b>3.2.3 Teachers set the priorities of subjects they are able and want to teach. Continue to Main Flow 4.</b></td></tr>
</table>

<table>
<tr><td colspan="2">Use Case Name<b>: View timetable.</b></td><td>ID<b>: UC04</b></td><td>Importance Level<b>: High</b></td></tr>
<tr><td>Primary Actor<b>: Teachers</b></td><td colspan="3">Use Case Type<b>: Detail, Essential</b></td></tr>
<tr><td colspan="4">Stakeholders and Interests<b>:</b>
<b>Teachers – view timetables generated by system administrator.</b></td></tr>
<tr><td colspan="4">Brief Description<b>: This use case describes how the teachers view timetables generated by the system administrators.</b></td></tr>
<tr><td colspan="4">Trigger<b>: Teachers want to view the generated timetables.</b></td></tr>
<tr><td colspan="4">Relationships:
Association<b>: Teachers</b>
Include<b>: N/A</b>
Extend<b>: N/A</b>
Generalization<b>: N/A</b></td></tr>
<tr><td colspan="4">Normal Flow of Events:
<b>1. Teachers want to view the generated timetables.</b>
<b>2. Teachers choose to view the generated timetables.</b>
<b>3. Teachers select a certain timetable to view from the multiple timetables.</b>
4. <b>Continue to Sub-flow 4.1 or 4.2.</b></td></tr>
<tr><td colspan="4">Sub-flows:
<b>4.1 If teacher want to print the timetable,</b>
<b>4.1.1 Teachers choose to print the timetable using their printer.</b>
<b>4.2 If teacher want to export the timetable using file format,</b>
<b>4.2.1 Teachers choose either PDF or image file format to export the timetable.</b></td></tr>
</table>

<table>
<tr><td colspan="2">Use Case Name<b>: Create teacher account.</b></td><td>ID<b>: UC05</b></td><td>Importance Level<b>: High</b></td></tr>
<tr><td>Primary Actor<b>: System Administrators</b></td><td colspan="3">Use Case Type<b>: Detail, Essential</b></td></tr>
<tr><td colspan="4">Stakeholders and Interests<b>:</b>
<b>System Administrator – create teacher account for newly enrolled teacher in the school.</b></td></tr>
<tr><td colspan="4">Brief Description<b>: This use case describes how the system administrator create teacher account.</b></td></tr>
<tr><td colspan="4">Trigger<b>: System Administrator want to create new teacher account for new teachers.</b></td></tr>
<tr><td colspan="4">Relationships:
Association<b>: System Administrator</b>
Include<b>: N/A</b>
Extend<b>: N/A</b>
Generalization<b>: N/A</b></td></tr>
<tr><td colspan="4">Normal Flow of Events:
<b>1. System administrators want to create new teacher account.</b>
<b>2. System administrators choose the option to create new teacher account.</b>
<b>3. STSAMS prompts system administrators to enter the new email.</b>
<b>4. System administrators enter the new email for the teacher.</b>
<b>5. System administrators choose to generate password randomly by STSAMS.</b>
<b>6. System administrators choose to copy the generated password for future use.</b>
7. <b>System administrators confirm and create the new account for teacher.</b></td></tr>
<tr><td colspan="4">Sub-flows:</td></tr>
</table>

Use Case Name**: View teachers.**

ID**: UC06**

Importance Level**: High**

Primary Actor**: System Administrators**

Use Case Type**: Detail, Essential**

Stakeholders and Interests**:**

System Administrator – view all of the teachers exist in the STSAMS.

Brief Description**: This use case describes how the system administrator view all teachers in the STSAMS.**

Trigger**: System Administrator want to view all teachers in STSAMS.**

Relationships:

Association**: System Administrator**

Include**: N/A**

Extend**: UC07 – Manage teacher details**

Generalization**: N/A**

Normal Flow of Events:

1. **System administrators want to view all teachers.**
2. **System administrators choose the option to view teachers.**
3. **STSAMS displays all of the existing teachers by pages.**
4. **If system administrators found their desired teacher to view details, perform use case UC07-Manage teacher details.**
5. **If not found, Continue to Sub-flow 5.1, 5.2, 5.3, 5.4 or 5.5.**

Sub-flows:

5.1 If system administrators want to filter some teacher out,

5.1.1 System administrators set the filtered to be done on the list of teachers.

5.1.2 STSAMS displays the teachers according to the filter set by system administrators.

5.2 If system administrators want to sort the teachers,

5.2.1 System administrators sort the list of teachers by the data column of table either alphabetically or numerically.

5.2.2 STSAMS displays the teachers according to the sorting done by system administrators.

5.3 If system administrators want to change items shown per page,

5.3.1 System administrators changes the number of teachers shown per page.

5.3.2 STSAMS shows the list of teachers by based on the items per page set by system administrators.

5.4 If system administrators want to navigate to another page of the shown list of teachers,

5.4.1 System administrators choose another page to display the teachers.

5.4.2 STSAMS displays only the specific page selected by system administrators.

5.5 If system administrators want to search for specific teachers.

5.5.1 System administrators enter teacher name keyword into the search box.

5.5.2 STSAMS displays only the teachers with names containing the keyword.

5.6 Continue to Main Flow 4 or 5.

Use Case Name**: Manage teacher details.** ID**: UC07** Importance Level**: High**

Primary Actor**: System Administrators** Use Case Type**: Detail, Essential**

Stakeholders and Interests**:**

System Administrator – manage teacher details by viewing, modifying, and deleting teacher user profile.

Brief Description**: This use case describes how the system administrator manage teacher details by viewing, modifying, and deleting teacher user profile.**

Trigger**: System Administrator want to manage teacher details.**

Relationships:

Association**: System Administrator**

Include**: N/A**

Extend**: N/A**

Generalization**: N/A**

Normal Flow of Events:

1. **System administrators want to manage teacher details.**
2. **STSAMS displays the complete information of the selected teacher.**
3. **Continue to sub-flow 3.1 or 3.2.**

Sub-flows:

3.1 If system administrators want to modify the content in teacher details,

3.1.1 System administrators choose to edit the profile of the selected teacher.

3.1.2 STSAMS will enter into update state for the system administrator to make changes to the teacher profile.

3.1.3 System administrators enter data for the teacher details.

3.1.4 System administrators confirm and choose to save changes.

3.1.5 STSAMS reflects the changes made.

3.2 If system administrators want to delete the teacher profile,

3.2.1 System administrators choose to delete the teacher profile.

3.2.2 STSAMS will prompt the confirmation of deletion to the system administrator.

3.2.3 System administrators confirm their decision, and the teacher account will be deleted from the STSAMS.

Use Case Name**: Manage subject details.**	ID**: UC08**	Importance Level**: High**
Primary Actor**: System Administrators**	Use Case Type**: Detail, Essential**	
Stakeholders and Interests**:** **System Administrator – manage subject details by viewing, modifying, and deleting subjects.**		
Brief Description**: This use case describes how the system administrator manage subject details by viewing, modifying, and deleting subjects.**		
Trigger**: System Administrator want to manage subject details.**		
Relationships: Association**: System Administrator** Include**: N/A** Extend**: N/A** Generalization**: N/A**		
Normal Flow of Events: **1. System administrators want to view all subjects.** **2. System administrators choose the option to view subjects.** **3. STSAMS displays all of the existing subjects by pages.** **4. If system administrators found their desired teacher to view details, continue to main flow 6.** **5. If not found, Continue to Sub-flow 5.1, 5.2, 5.3, 5.4 or 5.5.** 6. **Continue to Sub-flow 6.1, 6.2 or 6.3.**		
Sub-flows: **5.1 If system administrators want to filter some subjects out,** **5.1.1 System administrators set the filtered to be done on the list of subjects.** **5.1.2 STSAMS displays the subjects according to the filter set by system administrators.** **5.2 If system administrators want to sort the subjects,**		

5.2.1 System administrators sort the list of subjects by the data column of table either alphabetically or numerically.

5.2.2 STSAMS displays the subjects according to the sorting done by system administrators.

5.3 If system administrators want to change items shown per page,

5.3.1 System administrators changes the number of subjects shown per page.

5.3.2 STSAMS shows the list of subjects by based on the items per page set by system administrators.

5.4 If system administrators want to navigate to another page of the shown list of subjects,

5.4.1 System administrators choose another page to display the subjects.

5.4.2 STSAMS displays only the specific page selected by system administrators.

5.5 If system administrators want to search for specific subjects,

5.5.1 System administrators enter subject name keyword into the search box.

5.5.2 STSAMS displays only the subjects with names containing the keyword.

5.6 Continue to Main Flow 4 or 5.

6.1 If system administrators want to modify the content in subject details,

6.1.1 System administrators choose to edit the desired subject.

6.1.2 STSAMS will enter into update state for the system administrator to make changes to the subject details.

6.1.3 System administrators enter data for the subject details.

6.1.4 System administrators confirm and choose to save changes.

6.1.5 STSAMS reflects the changes made.

6.2 If system administrators want to delete the existing subject,

6.2.1 System administrators choose to delete desired subject.

6.2.2 STSAMS will prompt the confirmation of deletion to the system administrator.

6.2.3 System administrators confirm their decision, and the subject will be deleted from the STSAMS.

6.3 If system administrators want to create new subject,

6.3.1 System administrators choose to create new subject,

6.3.2 STSAMS will prompt the details regarding the new subject.

6.3.3 System administrators confirm on the creation of new subject.

6.3.4 New subject is added to the STSAMS.

Use Case Name**: Manage class details.**

ID**: UC09**

Importance Level**: High**

Primary Actor**: System Administrators**

Use Case Type**: Detail, Essential**

Stakeholders and Interests**:**

System Administrator – manage class details by viewing, modifying, and deleting classes.

Brief Description**: This use case describes how the system administrator manage class details by viewing, modifying, and deleting classes.**

Trigger**: System Administrator want to manage class details.**

Relationships:

Association**: System Administrator**

Include**: N/A**

Extend**: N/A**

Generalization**: N/A**

Normal Flow of Events:

1. **System administrators want to view all classes.**
2. **System administrators choose the option to view classes.**
3. **STSAMS displays all of the existing classes.**
4. **Continue to Sub-flow 4.1, 4.2 or 4.3.**

Sub-flows:

4.1 If system administrators want to modify the content in class details,

4.1.1 System administrators choose to edit the desired class.

4.1.2 STSAMS will enter into update state for the system administrator to make changes to the class details.

4.1.3 System administrators enter data for the class details.

4.1.4 System administrators confirm and choose to save changes.

4.1.5 STSAMS reflects the changes made.

4.2 If system administrators want to delete the existing class,

4.2.1 System administrators choose to delete desired class.

4.2.2 STSAMS will prompt the confirmation of deletion to the system administrator.

4.2.3 System administrators confirm their decision, and the class will be deleted from the STSAMS.

4.3 If system administrators want to create new class,

4.3.1 System administrators choose to create new class,

4.3.2 STSAMS will prompt the details regarding the new class.

4.3.3 System administrators confirm on the creation of new class.

4.3.4 New class is added to the STSAMS.

Use Case Name**: Manage allotments.**	ID**: UC10**	Importance Level**: High**
Primary Actor**: System Administrators**	Use Case Type**: Detail, Essential**	
Stakeholders and Interests**:** **System Administrator – manage allotments by choosing currently selected timetable sets, viewing teacher timetables, setting constraints, generating timetables, and toggling teacher permission to view them.**		
Brief Description**: This use case describes how the system administrator manage allotments by choosing currently selected timetable sets, viewing teacher timetables, setting constraints, generating timetables, and toggling teacher permission to view them.**		
Trigger**: System Administrator want to manage timetable allotments.**		
Relationships: Association**: System Administrator** Include**: N/A** Extend**: N/A** Generalization**: N/A**		
Normal Flow of Events: **1. System administrators want to manage timetable allotments.** **2. System administrators choose the option to manage allotments.** 3. **Continue to Sub-flow 3.1, 3.2 or 3.3.**		
Sub-flows: **3.1 If system administrators want to generate new timetable results,** **3.1.1 The system administrators choose the option to create new allotment set.** **3.1.2 System administrators fill up the necessary constraints for allocation algorithm to consider.** **3.1.3 System administrators choose the option to generate timetables.** **3.2 If system administrators want to allow teachers to view the timetables,** **3.2.1 System administrators select the desired allotment set.**		

3.2.2 System administrators toggle the option to allow teachers to view their allocation results.

3.3 If system administrators want to view the timetable results,

3.3.1 System administrators select the desired allotment set.

3.3.2 System administrators choose the option to view allocated results.

Use Case Name**: Manage announcements.**	ID**: UC11**	Importance Level**: High**
Primary Actor**: System Administrators**	Use Case Type**: Detail, Essential**	
Stakeholders and Interests**:** **System Administrator – manage announcements by selecting currently active announcements to be displayed in login page, adding, deleting, and updating announcements.**		
Brief Description**: This use case describes how the system administrator manage announcements by selecting currently active announcements to be displayed in login page, adding, deleting, and updating announcements.**		
Trigger**: System Administrator want to manage announcements.**		
Relationships: Association**: System Administrator** Include**: N/A** Extend**: N/A** Generalization**: N/A**		
Normal Flow of Events: **1. System administrators want to manage announcements.** **2. System administrators choose the option to manage announcements.** **3. STSAMS displays all of the existing announcements.** 4. **Continue to Sub-flow 4.1, 4.2 or 4.3.**		
Sub-flows: **4.1 If system administrators want to modify the content in announcement,** **4.1.1 System administrators choose to edit the announcement.** **4.1.2 STSAMS will enter into update state for the system administrator to make changes to the announcement.**		

4.1.3 System administrators enter data for the announcement.

4.1.4 System administrators confirm and choose to save changes.

4.1.5 STSAMS reflects the changes made.

4.2 If system administrators want to delete the existing announcement,

4.2.1 System administrators choose to delete desired announcement.

4.2.2 STSAMS will prompt the confirmation of deletion to the system administrator.

4.2.3 System administrators confirm their decision, and the announcement will be deleted from the STSAMS.

4.3 If system administrators want to create new announcement,

4.3.1 System administrators choose to create new announcement,

4.3.2 STSAMS will prompt the details regarding the new announcement.

4.3.3 System administrators confirm on the creation of new announcement.

4.3.4 New announcement is added to the STSAMS.

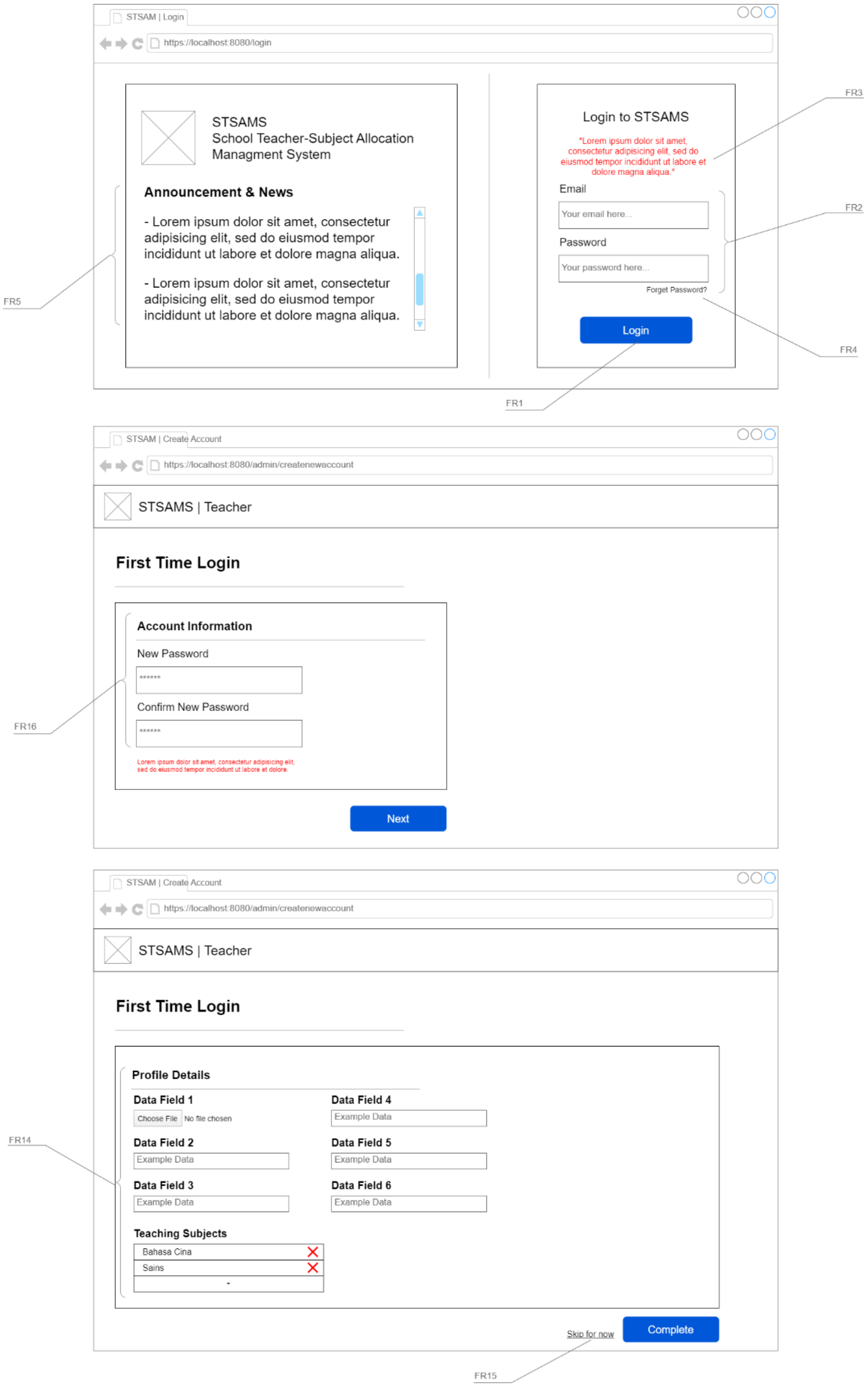
4.4 If system administrators want an announcement to be displayed in login page,

4.4.1 System administrators select announcements to be active.

4.4.2 STSAMS will display all announcements that are currently active in the login page.

5.2 User Interface Modelling

5.2.1 Storyboards



STSAM | Login

https://localhost:8080/login

STSAMS
School Teacher-Subject Allocation Managment System

Announcement & News

- Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

- Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Login to STSAMS

"Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua."

Email: Your email here...

Password: Your password here...

Forget Password?

STSAM | Create Account

https://localhost:8080/admin/createnewaccount

STSAMS | Teacher

First Time Login

Account Information

New Password: ******

Confirm New Password: ******

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore.

STSAM | Create Account

https://localhost:8080/admin/createnewaccount

STSAMS | Teacher

First Time Login

Profile Details

Data Field 1: Choose File No file chosen

Data Field 2: Example Data

Data Field 3: Example Data

Data Field 4: Example Data

Data Field 5: Example Data

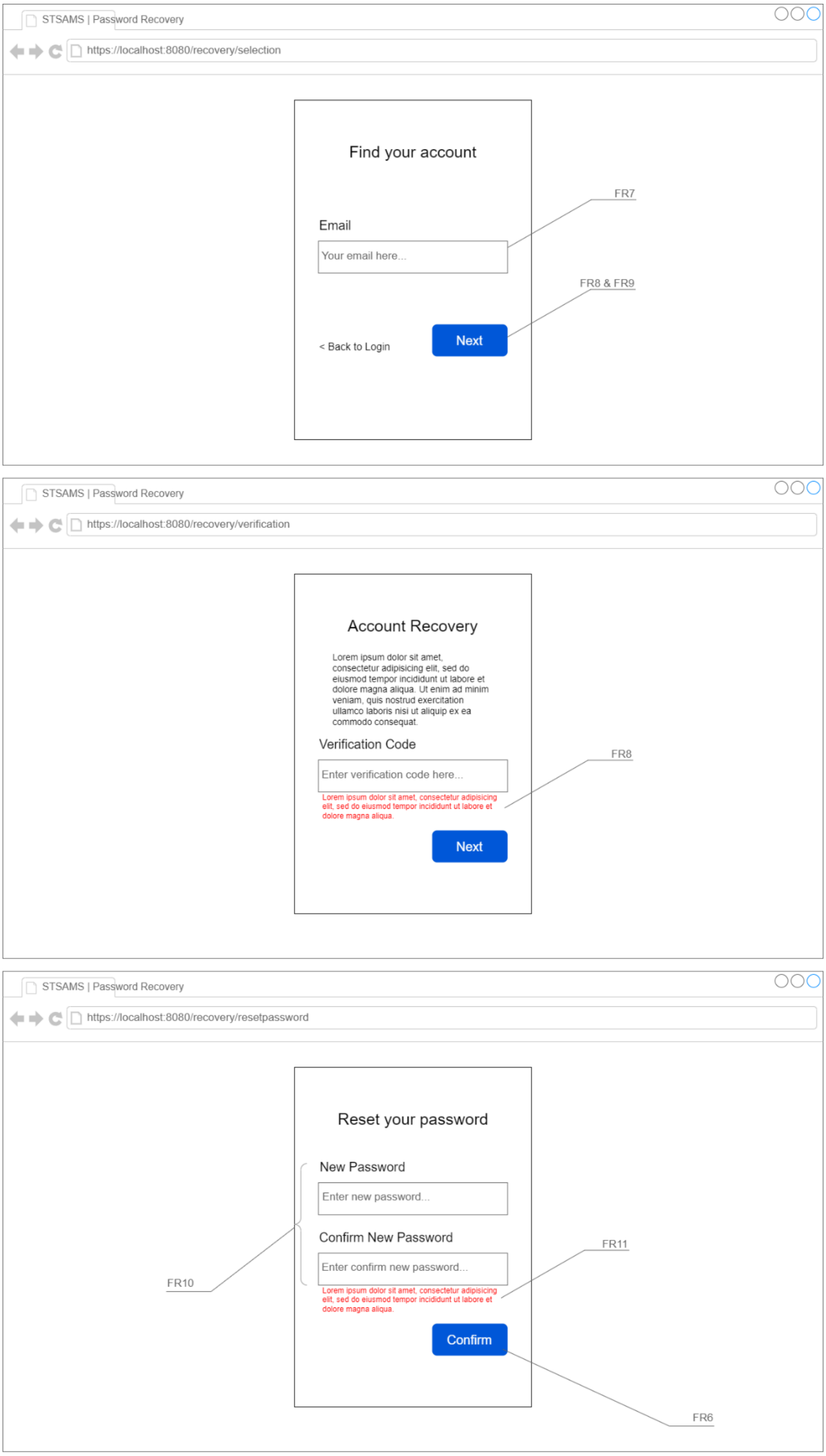
Data Field 6: Example Data

Teaching Subjects

- Bahasa Cina
- Sains

Skip for now

Figure 4.4.3.1: Storyboard for Login Page.

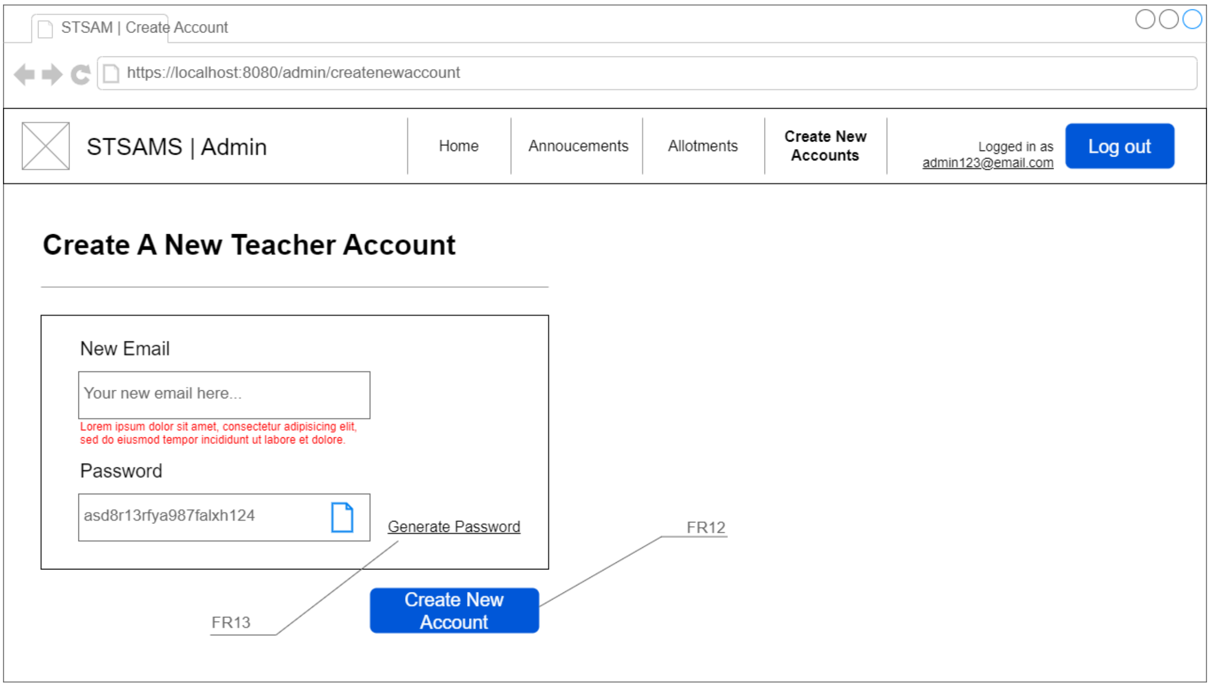

STSAMS | Password Recovery
https://localhost:8080/recovery/selection
Find your account
Email
Your email here...
< Back to Login
Next
FR7
FR8 & FR9
STSAMS | Password Recovery
https://localhost:8080/recovery/verification
Account Recovery
Verification Code
Enter verification code here...
Next
FR8
STSAMS | Password Recovery
https://localhost:8080/recovery/resetpassword
Reset your password
New Password
Enter new password...
Confirm New Password
Enter confirm new password...
Confirm
FR10
FR11
FR6

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Figure 4.4.3.2: Storyboard for Forget Password Page.



STSAM | Create Account

https://localhost:8080/admin/createnewaccount

Create A New Teacher Account

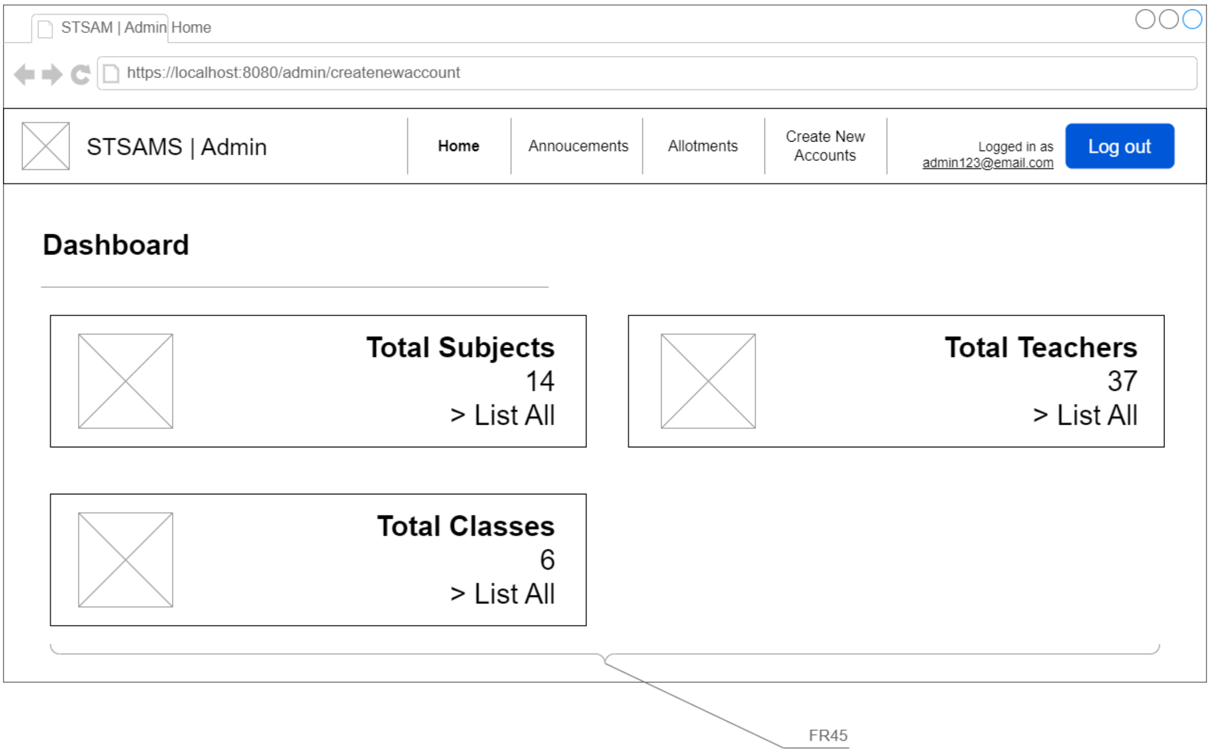
New Email: Your new email here...

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore.

Password: asd8r13rfya987falxh124

Logged in as admin123@email.com

Figure 4.4.3.3: Storyboard for Create Teacher Page.



STSAM | Admin Home

https://localhost:8080/admin/createnewaccount

Dashboard

Total Subjects 14

Total Teachers 37

Total Classes 6

Logged in as admin123@email.com

Figure 4.4.3.4: Storyboard for Admin Home Page.

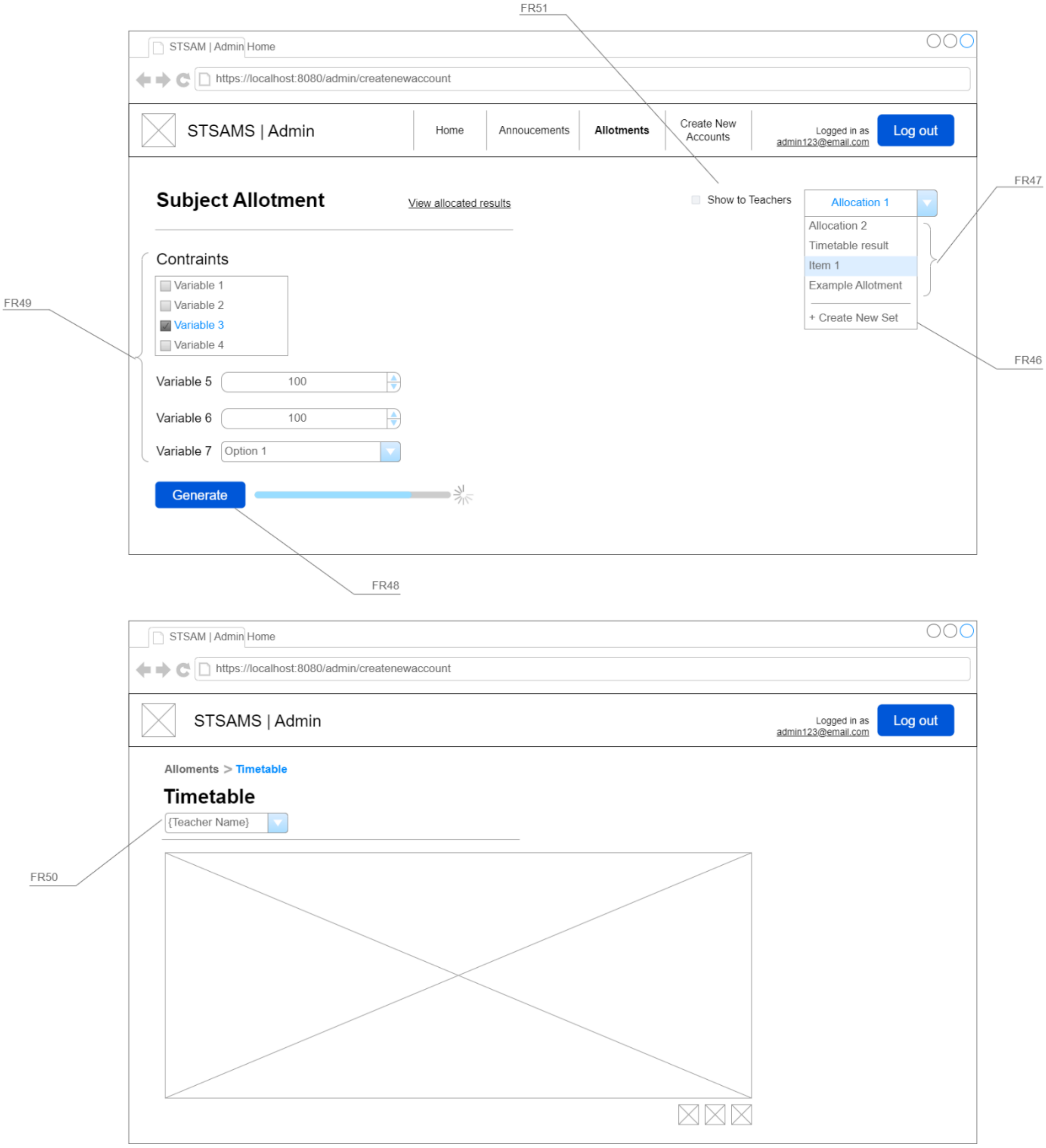
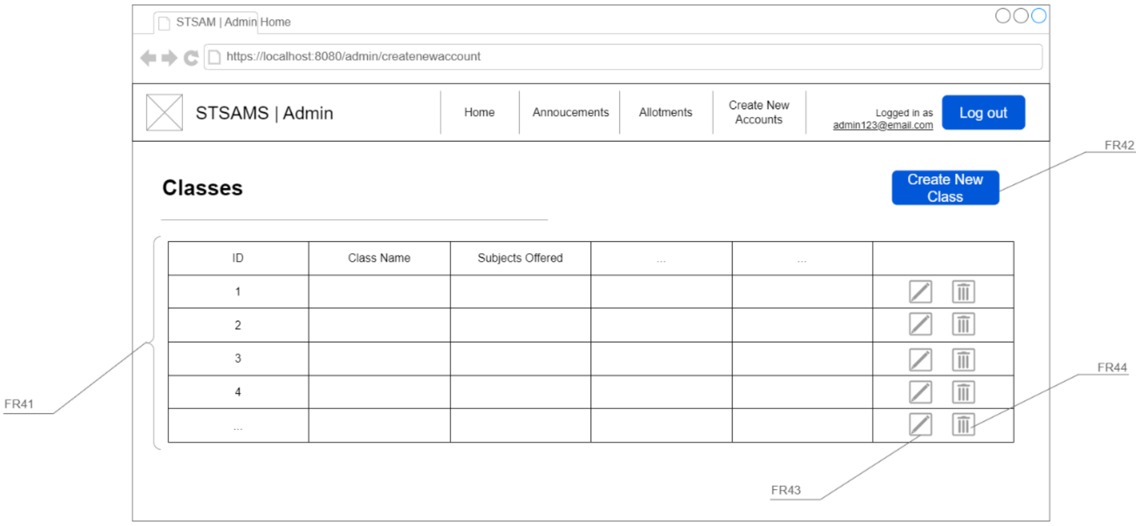
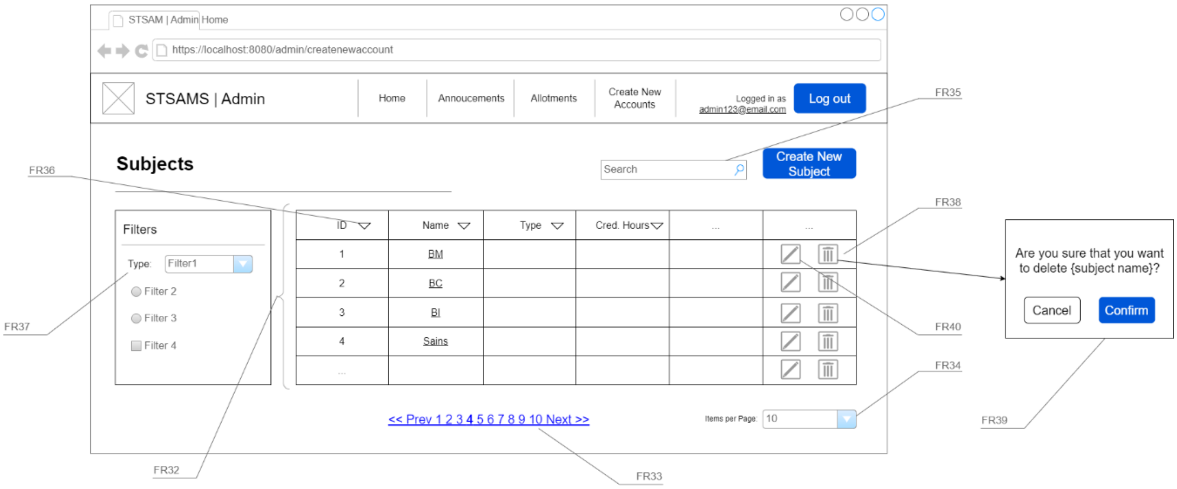


Figure 4.4.3.5: Storyboard for Allotment Page.



ID	Class Name	Subjects Offered	...	...	
1					
2					
3					
4					
...					

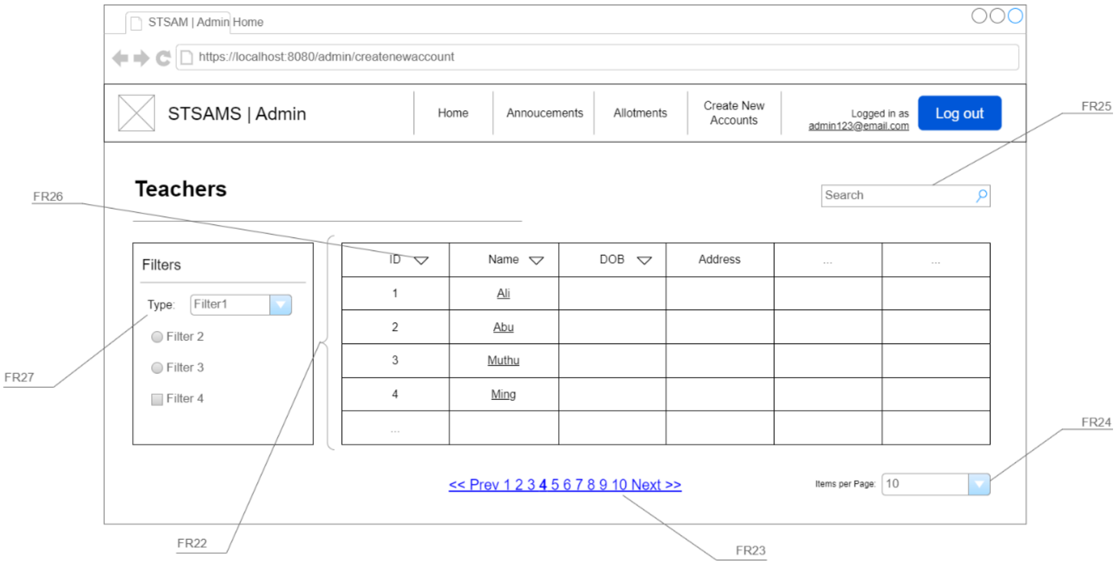
Figure 4.4.3.6: Storyboard for View Class Page.



ID	Name	Type	Cred. Hours	...	...
1	BM				
2	BC				
3	BI				
4	Sains				
...					

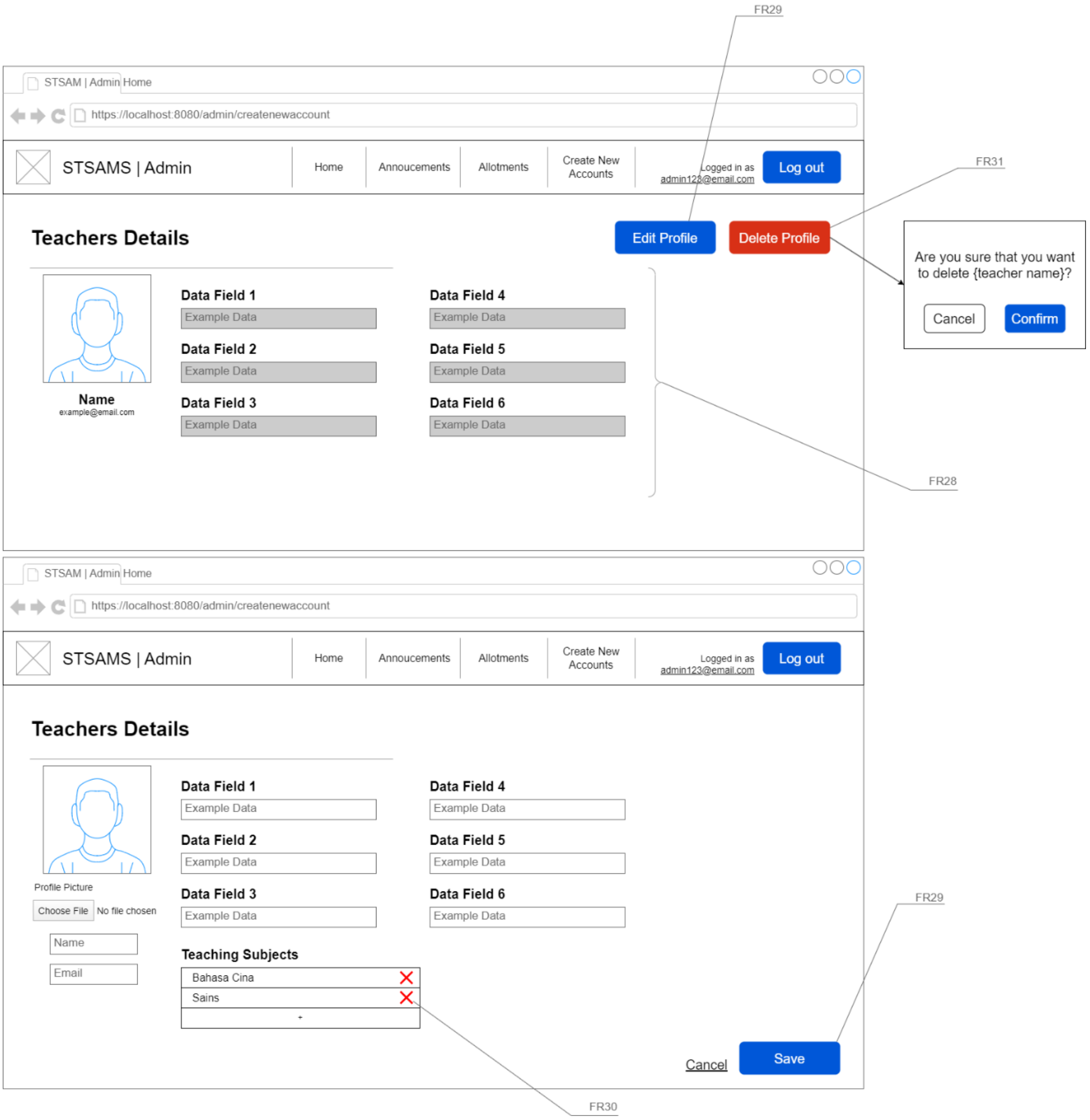
Are you sure that you want to delete {subject name}?

Figure 4.4.3.7: Storyboard for View Subjects Page.



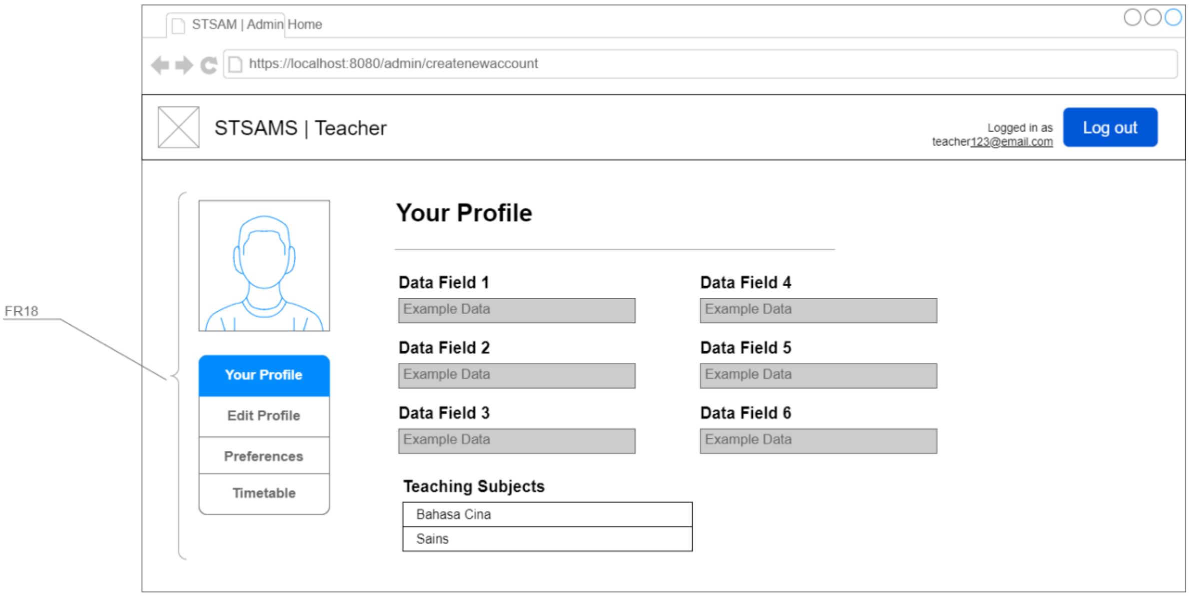
ID	Name	DOB	Address	...	...
1	Ali				
2	Abu				
3	Muthu				
4	Ming				
...					

Figure 4.4.3.8: Storyboard for View Teacher Page.



Are you sure that you want to delete {teacher name}?

Figure 4.4.3.9: Storyboard for View Teacher Details Page



Your Profile

Data Field 1: Example Data

Data Field 2: Example Data

Data Field 3: Example Data

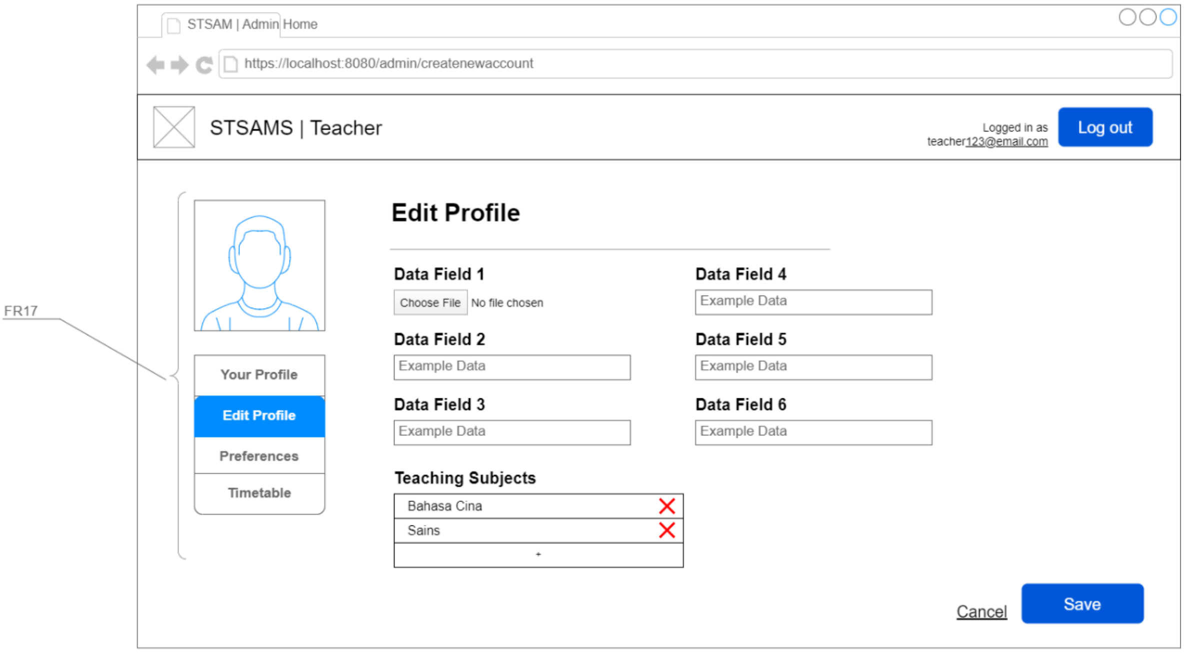
Data Field 4: Example Data

Data Field 5: Example Data

Data Field 6: Example Data

Teaching Subjects

- Bahasa Cina
- Sains



Edit Profile

Data Field 1: No file chosen

Data Field 2: Example Data

Data Field 3: Example Data

Data Field 4: Example Data

Data Field 5: Example Data

Data Field 6: Example Data

Teaching Subjects

- Bahasa Cina ✗
- Sains ✗

Figure 4.4.3.10: Part 1 Storyboard for Teacher Profile Page.

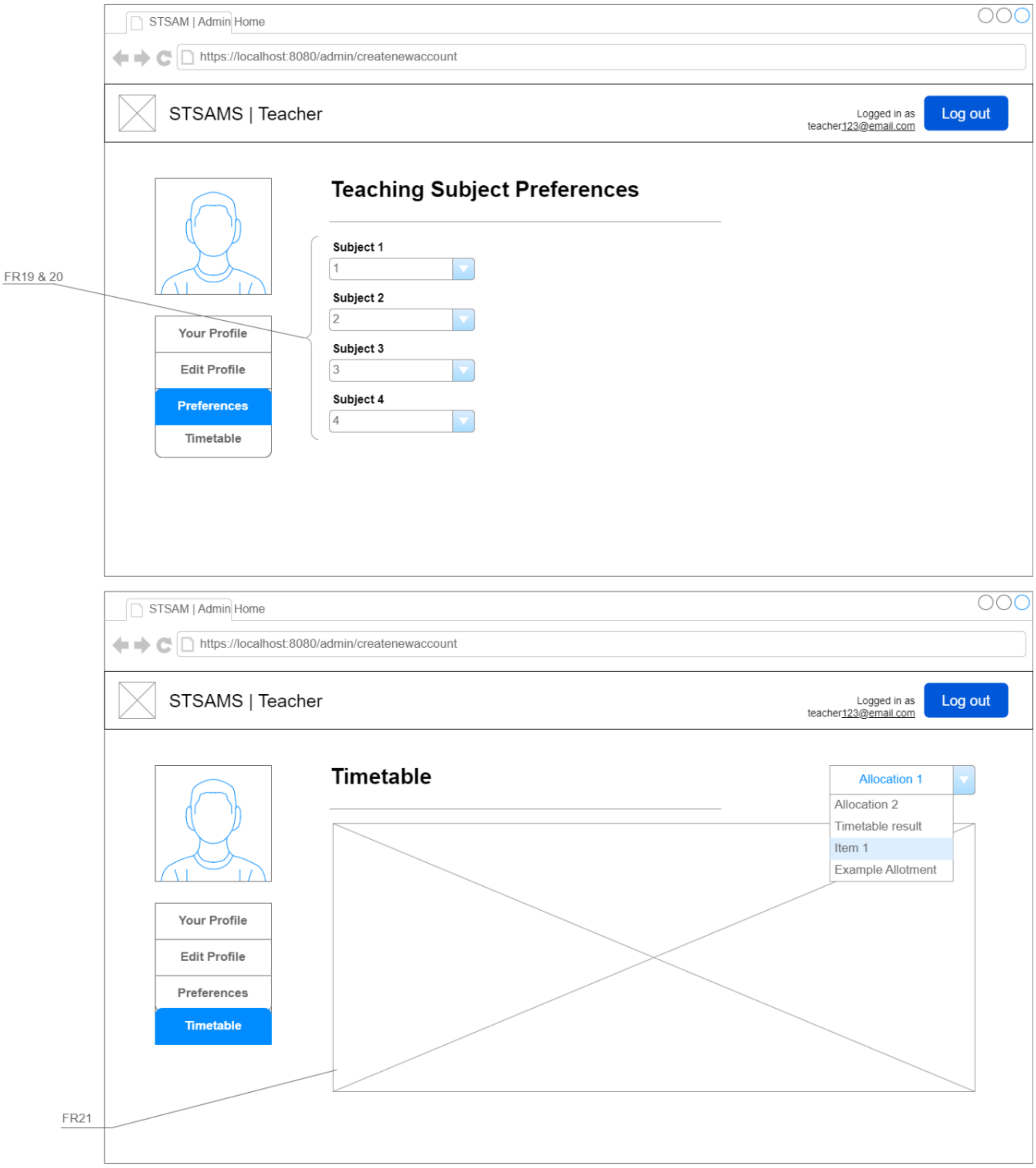
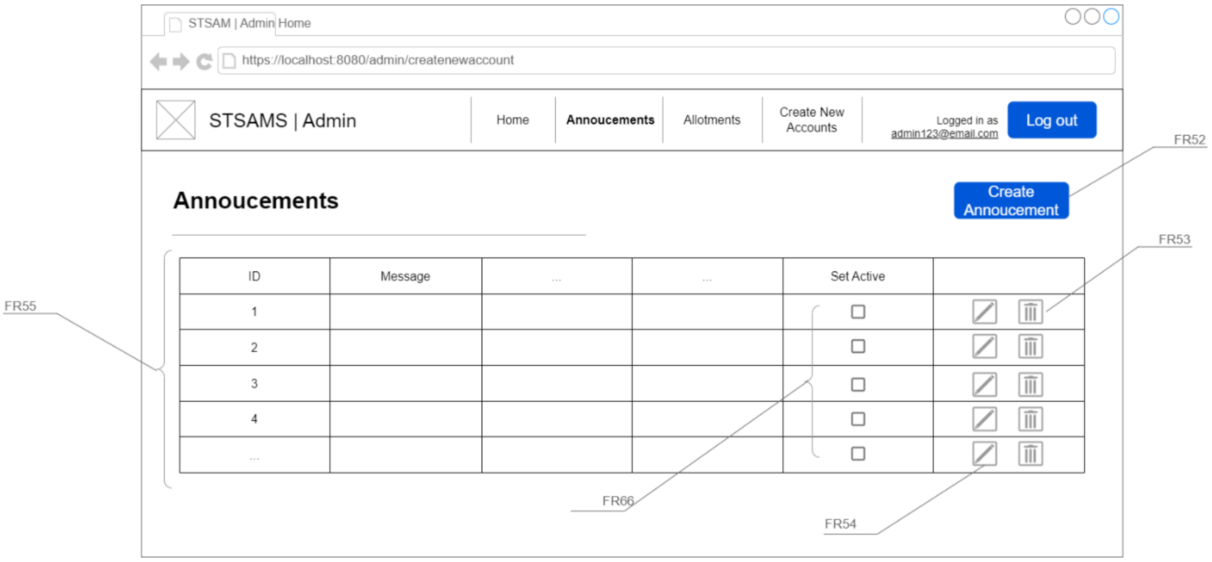


Figure 4.4.3.10: Part 2 Storyboard for Teacher Profile Page.



ID	Message	...	...	Set Active	
1				□	✎ 🗑
2				□	✎ 🗑
3				□	✎ 🗑
4				□	✎ 🗑
...				□	✎ 🗑

Figure 4.4.3.11: Storyboard for Announcement Management Page.

5.2.2 System UI Flow

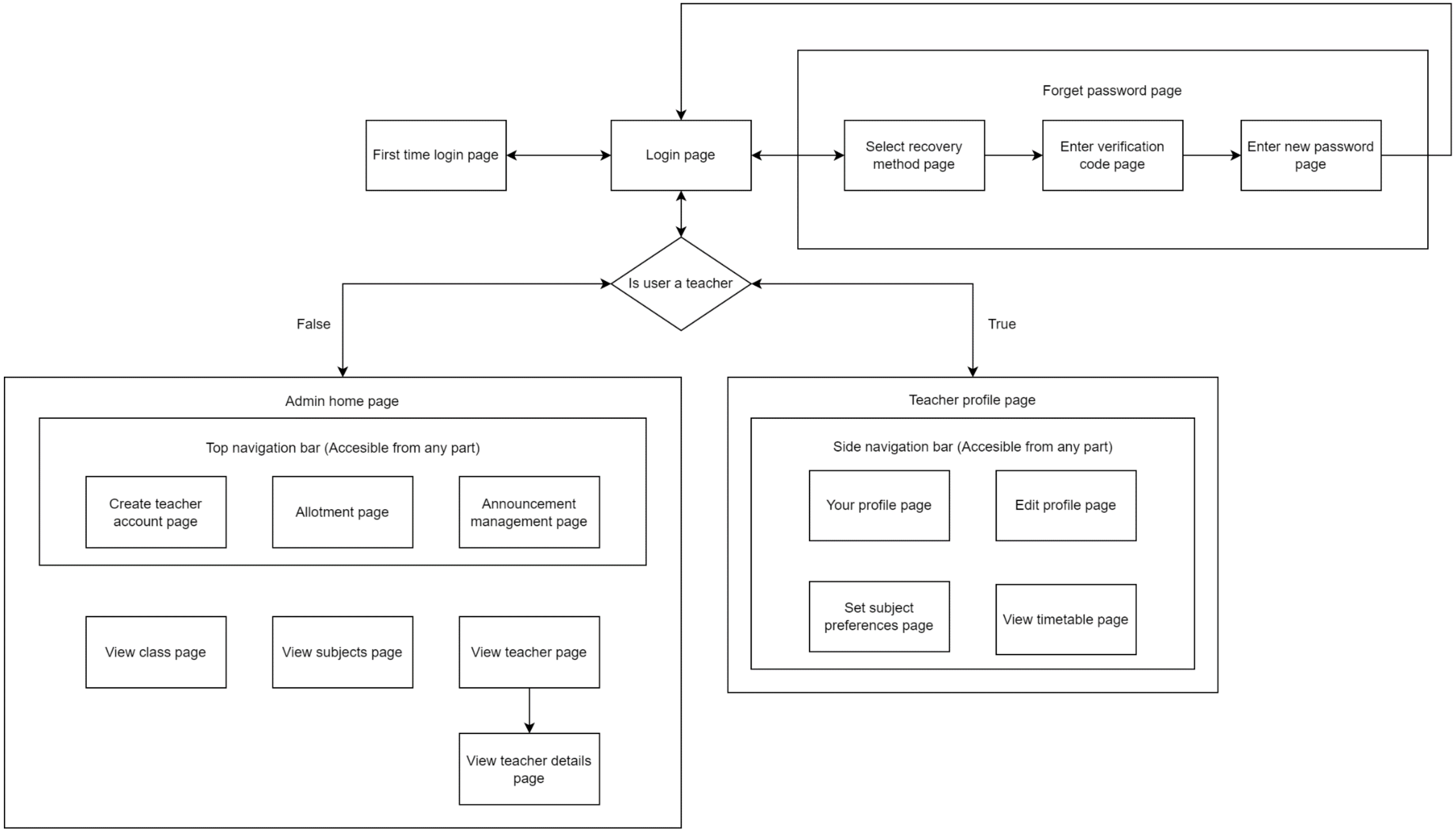

Forget password page
First time login page
Login page
Select recovery method page
Enter verification code page
Enter new password page
Is user a teacher
False
True
Admin home page
Top navigation bar (Accesible from any part)
Create teacher account page
Allotment page
Announcement management page
View class page
View subjects page
View teacher page
View teacher details page
Teacher profile page
Side navigation bar (Accesible from any part)
Your profile page
Edit profile page
Set subject preferences page
View timetable page

Figure 4.5.1: Proposed System Flow for STSAMS.

5.3 Data Persistence Design

In STSAMS, all data regarding to account credentials, timetabling constraint settings, generated timetables, and other data like subjects, classes, teachers, and venues will be saved in a Relational Database Management System (RDBMS) which is SQLite3. SQLite3 is chosen due to its lightweighted nature, and it is widely used when the backend is developed in the Python language. The frontend will be able to get access to the data in the database by communicating with an API server that is powered by Flask. The API server will handle all the requests sent from the frontend, do some operations on the database, ultimately return only the related data back to the frontend.

5.3.1 Entity Relationship Diagram (ERD)

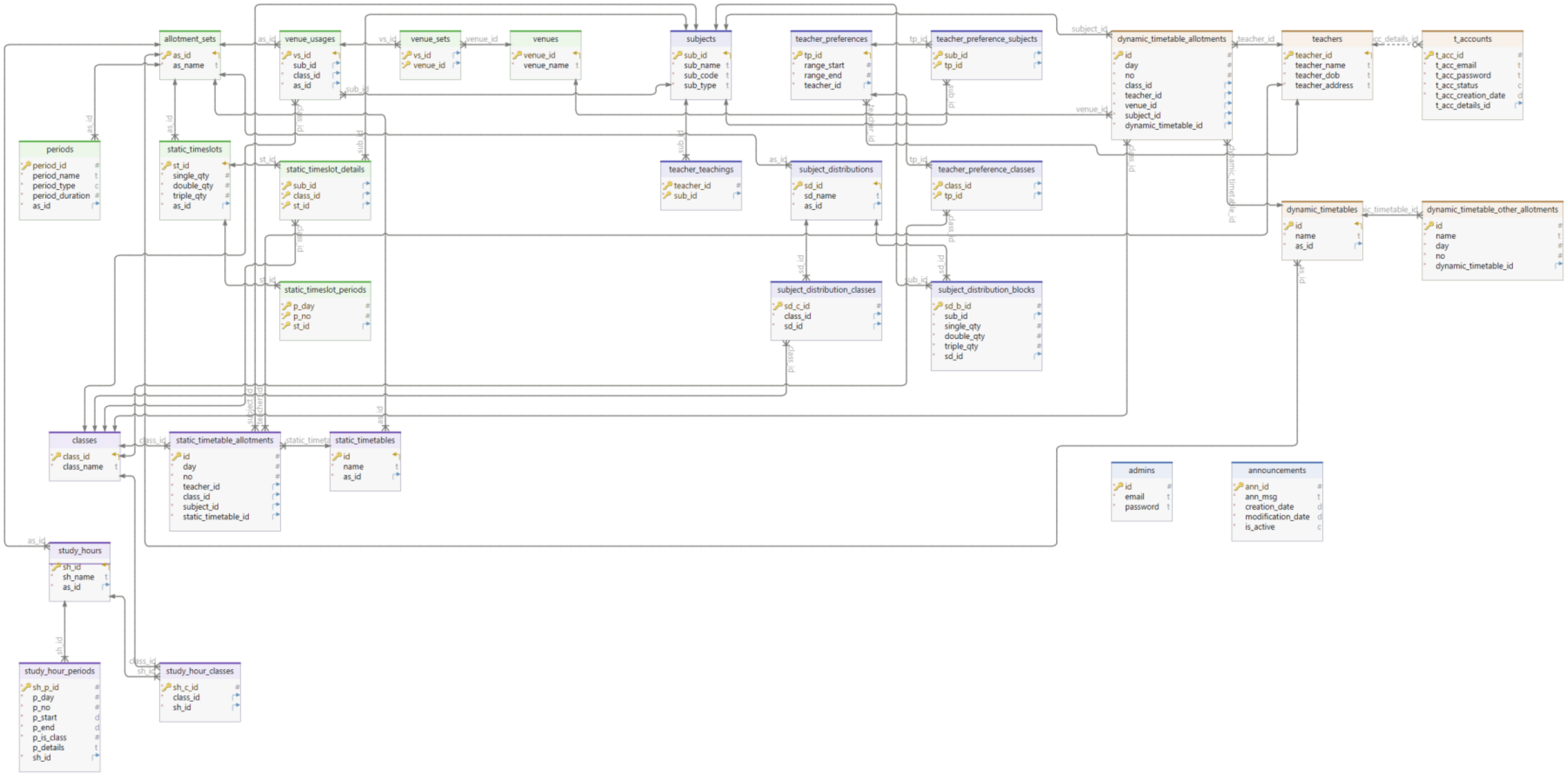


Figure 5.3.1.1: ERD for the database in STSAMS.

5.3.2 Table Descriptions in ERD

The description will be provided for tables in the database to explain what each of the columns' name represents and what are the columns' data type.

Announcements Table

Table 5.3.2.1: Description of columns in Announcements table.

Columns	Data Types	Descriptions
ann_id	INTEGER	Unique ID that represents an announcement detail.
ann_msg	TEXT	The message content of the announcement.
creation_date	DATETIME	The creation date of the announcement.
modification_date	DATETIME	The date and time the announcement detail is modified.
is_active	CHAR	Decides if the announcement detail will be shown in certain pages.

Subjects Table

Table 5.3.2.2: Description of columns in Subjects table.

Columns	Data Types	Descriptions
sub_id	INTEGER	Unique ID that represents a subject detail.
sub_name	TEXT	The name of the subject.
sub_code	TEXT	The abbreviation of the subject name.
sub_type	TEXT	The weight of the subject.

Classes Table

Table 5.3.2.3: Description of columns in Classes table.

Columns	Data Types	Descriptions
class_id	INTEGER	Unique ID that represents a class detail.
class_name	TEXT	The name of the class.

Venues Table

Table 5.3.2.4: Description of columns in Venues table.

Columns	Data Types	Descriptions
venue_id	INTEGER	Unique ID that represents a venue detail.
venue_name	TEXT	The name of the venue.

Teachers Table

Table 5.3.2.5: Description of columns in Teachers table.

Columns	Data Types	Descriptions
teacher_id	INTEGER	Unique ID that represents a teacher detail.
teacher_name	TEXT	The name of the teacher.
teacher_dob	TEXT	The date of birth of the teacher.
teacher_address	TEXT	The address of the teacher.

T_accounts Table

Table 5.3.2.6: Description of columns in T_accounts table.

Columns	Data Types	Descriptions
t_acc_id	INTEGER	Unique ID that represents a teacher account.
t_acc_email	TEXT	The email of the teacher account.
t_acc_password	TEXT	The password of the teacher account.
t_acc_status	CHAR	The status of the teacher account.
t_acc_creation_date	TIMESTAMP	The creation date of the teacher account.
t_acc_details_id	INTEGER	The unique ID of the teacher.

Allotment_sets Table

Table 5.3.2.7: Description of columns in Allotment_sets table.

Columns	**Data Types**	**Descriptions**
as_id	INTEGER	Unique ID that represents an allotment set detail.
as_name	TEXT	The name of the allotment set.

Teacher_teachings Table

Table 5.3.2.8: Description of columns in Teacher_teachings table.

Columns	**Data Types**	**Descriptions**
teacher_id	INTEGER	The unique ID that represents a teacher.
sub_id	INTEGER	The unique ID that represents a subject.

Study_hours Table

Table 5.3.2.9: Description of columns in Study_hours table.

Columns	**Data Types**	**Descriptions**
sh_id	INTEGER	Unique ID that represents a study hour detail.
sh_name	TEXT	The name of the study hour.
as_id	INTEGER	The unique ID that represents an allotment set.

Study_hour_classes Table

Table 5.3.2.10: Description of columns in Study_hour_classes table.

Columns	**Data Types**	**Descriptions**
sh_c_id	INTEGER	Unique ID that represents a study hour classes detail.
class_id	INTEGER	The unique ID that represents a class.
sh_id	INTEGER	The unique ID that represents a study hour.

Study_hour_periods Table

Table 5.3.2.11: Description of columns in Study_hour_periods table.

Columns	Data Types	Descriptions
sh_p_id	INTEGER	Unique ID that represents a study hour periods detail.
p_day	INTEGER	The day of a period.
p_no	INTEGER	The position number of a period in a specific day.
p_start	TIME	The start time of a period.
p_end	TIME	The end time of a period.
p_is_class	TINYINT	The Boolean value decide if a period is class type.
p_details	TEXT	The description of a period.
sh_id	INTEGER	The unique ID that represents a study hour.

Subject_distribution Table

Table 5.3.2.12: Description of columns in Subject_distribution table.

Columns	Data Types	Descriptions
sd_id	INTEGER	Unique ID that represents a subject distribution detail.
sd_name	TEXT	The name of the subject distribution.
as_id	INTEGER	The unique ID that represents an allotment set.

Subject_distribution_classes Table

Table 5.3.2.13: Description of columns in Subject_distribution_classes table.

Columns	Data Types	Descriptions
sd_c_id	INTEGER	Unique ID that represents a subject distribution classes detail.
class_id	INTEGER	The unique ID that represents a class.
sd_id	INTEGER	The unique ID that represents a subject distribution.

Subject_distribution_blocks Table

Table 5.3.2.14: Description of columns in Subject_distribution_blocks table.

Columns	Data Types	Descriptions
sd_b_id	INTEGER	Unique ID that represents a subject distribution blocks detail.
sub_id	INTEGER	The unique ID that represents a subject.
single_qty	INTEGER	The quantity of a subject with single block.
double_qty	INTEGER	The quantity of a subject with two consecutive blocks.
triple_qty	INTEGER	The quantity of a subject with three consecutive blocks.
sd_id	INTEGER	The unique ID that represents a subject distribution.

Venue_usages Table

Table 5.3.2.15: Description of columns in Venue_usages table.

Columns	Data Types	Descriptions
vs_id	INTEGER	Unique ID that represents a venue usages detail.
sub_id	INTEGER	The unique ID that represents a subject.
class_id	INTEGER	The unique ID that represents a class.
as_id	INTEGER	The unique ID that represents an allotment set.

Venue_sets Table

Table 5.3.2.16: Description of columns in Venue_sets table.

Columns	Data Types	Descriptions
vs_id	INTEGER	The unique ID that represents a venue usage.
venue_id	INTEGER	The unique ID that represents a venue.

Teacher_preferences Table

Table 5.3.2.17: Description of columns in Teacher_preferences table.

Columns	**Data Types**	**Descriptions**
tp_id	INTEGER	The unique ID that represents a teacher preference detail.
range_start	INTEGER	The start range of the preferred period position.
range_end	INTEGER	The end range of the preferred period position.
teacher_id	INTEGER	The unique ID that represents a teacher.

Teacher_preference_classes Table

Table 5.3.2.18: Description of columns in Teacher_preference_classes table.

Columns	**Data Types**	**Descriptions**
class_id	INTEGER	The unique ID that represents a class.
tp_id	INTEGER	The unique ID that represents a teacher preference.

Teacher_preference_subjects Table

Table 5.3.2.19: Description of columns in Teacher_preference_subjects table.

Columns	**Data Types**	**Descriptions**
sub_id	INTEGER	The unique ID that represents a subject.
tp_id	INTEGER	The unique ID that represents a teacher preference.

Static_timeslots Table

Table 5.3.2.20: Description of columns in Static_timeslots table.

Columns	**Data Types**	**Descriptions**
st_id	INTEGER	The unique ID that represents a static timeslot.
single_qty	INTEGER	The quantity of a static timeslot with single block.
double_qty	INTEGER	The quantity of a static timeslot with two consecutive blocks.
triple_qty	INTEGER	The quantity of a static timeslot with three consecutive blocks.
as_id	INTEGER	The unique ID that represents an allotment set.

Static_timeslot_details Table

Table 5.3.2.21: Description of columns in Static_timeslot_details table.

Columns	**Data Types**	**Descriptions**
sub_id	INTEGER	The unique ID that represents a subject.
class_id	INTEGER	The unique ID that represents a class.
st_id	INTEGER	The unique ID that represents a static timeslot.

Static_timeslot_periods Table

Table 5.3.2.22: Description of columns in Static_timeslot_periods table.

Columns	**Data Types**	**Descriptions**
p_day	INTEGER	The day of a period.
p_no	INTEGER	The position number of a period in a specific day.
st_id	INTEGER	The unique ID that represents a static timeslot.

Static_timetables

Table 5.3.2.23: Description of columns in Static_timetables table.

Columns	**Data Types**	**Descriptions**
id	INTEGER	The unique ID that represents a static timetable.
name	INTEGER	The name of a static timetable.
as_id	INTEGER	The unique ID that represents an allotment set.

Static_timetable_allotments Table

Table 5.3.2.24: Description of columns in Static_timetable_allotments table.

Columns	**Data Types**	**Descriptions**
id	INTEGER	The unique ID that represents a static timetable allotment.
day	INTEGER	The day of a period.
no	INTEGER	The position number of a period in a specific day.
teacher_id	INTEGER	The unique ID that represents a teacher.
class_id	INTEGER	The unique ID that represents a class.
subject_id	INTEGER	The unique ID that represents a subject.
static_timetable_id	INTEGER	The unique ID that represents a static timetable.

Dynamic_timetables

Table 5.3.2.25: Description of columns in Dynamic_timetables table.

Columns	**Data Types**	**Descriptions**
id	INTEGER	The unique ID that represents a dynamic timetable.
name	INTEGER	The name of a dynamic timetable.
as_id	INTEGER	The unique ID that represents an allotment set.

Dynamic_timetable_allotments Table

Table 5.3.2.26: Description of columns in Dynamic_timetable_allotments table.

Columns	**Data Types**	**Descriptions**
id	INTEGER	The unique ID that represents a dynamic timetable allotment.
day	INTEGER	The day of a period.
no	INTEGER	The position number of a period in a specific day.
teacher_id	INTEGER	The unique ID that represents a teacher.
class_id	INTEGER	The unique ID that represents a class.
venue_id	INTEGER	The unique ID that represents a venue.
subject_id	INTEGER	The unique ID that represents a subject.
dyanmic_timetable_id	INTEGER	The unique ID that represents a dynamic timetable.

Dynamic_timetable_allotments Table

Table 5.3.2.27: Description of columns in Dynamic_timetable_allotments table.

Columns	**Data Types**	**Descriptions**
id	INTEGER	The unique ID that represents a dynamic timetable other allotment.
name	TEXT	The name of the allotted not class-type period.
day	INTEGER	The day of a period.
no	INTEGER	The position number of a period in a specific day.
dyanmic_timetable_id	INTEGER	The unique ID that represents a dynamic timetable.

5.4 Algorithm Design

As there are two types of allocation in STSAMS, which is static allocation and dynamic allocation, they may have minor different definitions of object classes, fitness evaluation formula. Besides that, both algorithms are using different constraints during execution. The objective of this separation is to reduce the complexity of the algorithm as modularizing functionalities in the system could increase maintainability and learnability. However, the execution steps of these two algorithms are mostly similar in terms of execution steps and result structuring.

5.4.1 Class Definition

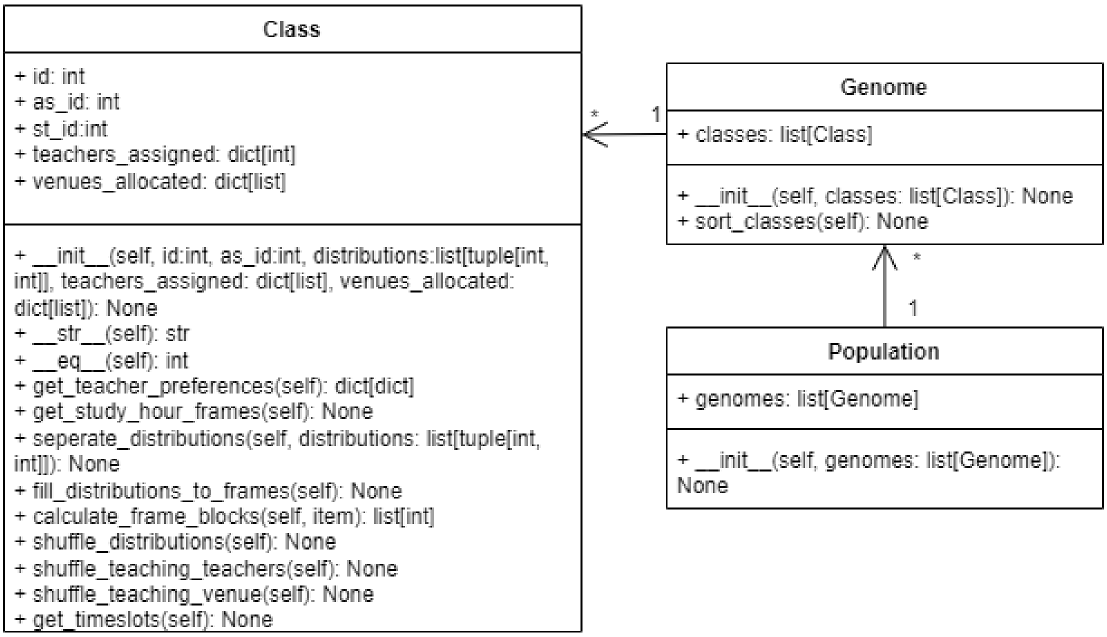

Class
+ id: int
+ as_id: int
+ st_id:int
+ teachers_assigned: dict[int]
+ venues_allocated: dict[list]
+ __init__(self, id:int, as_id:int, distributions:list[tuple[int, int]], teachers_assigned: dict[list], venues_allocated: dict[list]): None
+ __str__(self): str
+ __eq__(self): int
+ get_teacher_preferences(self): dict[dict]
+ get_study_hour_frames(self): None
+ seperate_distributions(self, distributions: list[tuple[int, int]]): None
+ fill_distributions_to_frames(self): None
+ calculate_frame_blocks(self, item): list[int]
+ shuffle_distributions(self): None
+ shuffle_teaching_teachers(self): None
+ shuffle_teaching_venue(self): None
+ get_timeslots(self): None
*
1
Genome
+ classes: list[Class]
+ __init__(self, classes: list[Class]): None
+ sort_classes(self): None
*
1
Population
+ genomes: list[Genome]
+ __init__(self, genomes: list[Genome]): None

Figure 5.4.1.1: Class Diagram of STSAMS Algorithm for Dynamic Allocation.

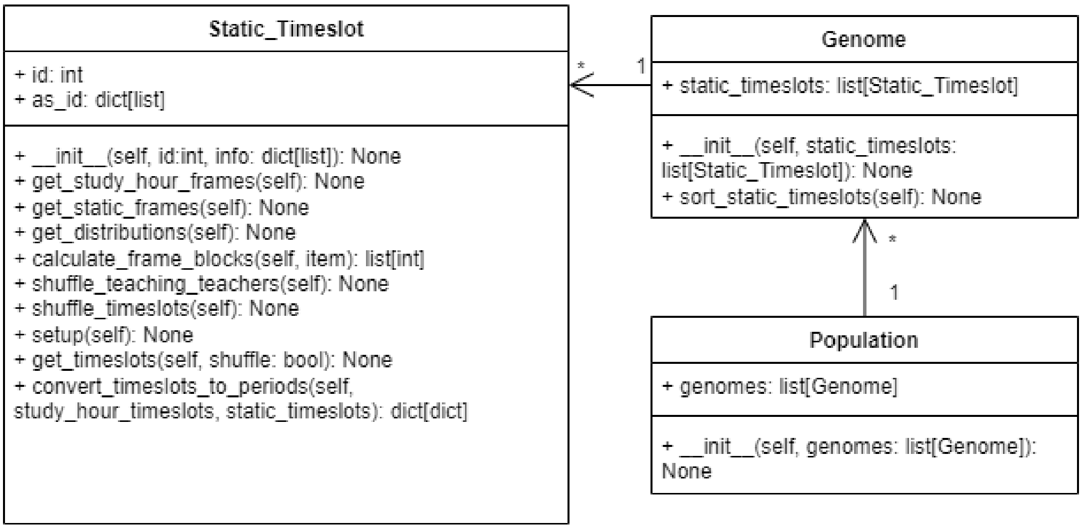

Static_Timeslot
+ id: int
+ as_id: dict[list]
+ __init__(self, id:int, info: dict[list]): None
+ get_study_hour_frames(self): None
+ get_static_frames(self): None
+ get_distributions(self): None
+ calculate_frame_blocks(self, item): list[int]
+ shuffle_teaching_teachers(self): None
+ shuffle_timeslots(self): None
+ setup(self): None
+ get_timeslots(self, shuffle: bool): None
+ convert_timeslots_to_periods(self, study_hour_timeslots, static_timeslots): dict[dict]
*
1
Genome
+ static_timeslots: list[Static_Timeslot]
+ __init__(self, static_timeslots: list[Static_Timeslot]): None
+ sort_static_timeslots(self): None
*
1
Population
+ genomes: list[Genome]
+ __init__(self, genomes: list[Genome]): None

Figure 5.4.1.1: Class Diagram of STSAMS Algorithm for Static Allocation.

5.4.2 Execution Flow

The type of meta-heuristic algorithm used in STSAMS is genetic algorithm. The principle behind genetic algorithm is that the step of execution consists of initialization, fitness calculation, selection, crossover, and mutation. To visualize the execution process, Figure 5.4.2.1 below shows how the designed genetic algorithm's flow of allocations.

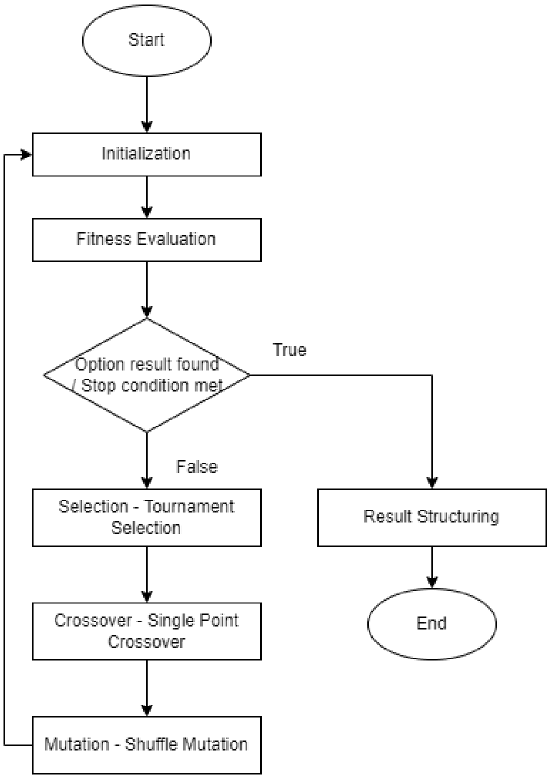

Start
Initialization
Fitness Evaluation
Option result found / Stop condition met
True
False
Selection - Tournament Selection
Result Structuring
Crossover - Single Point Crossover
End
Mutation - Shuffle Mutation

Figure 5.4.2.1 Execution flow of the STSAMS Genetic Algorithm.

The genetic algorithm will first be starting with initializing random population of genomes containing the timetables for the very first time. After that, fitness evaluation will be conducted on the genomes in the generated timetable populations. The evaluated timetable population may achieve optimal solution in the first or Nth time of fitness evaluation. So, if the best result or one of the stop conditions is met, the best genome in the population will be selected to perform result structuring. This structuring process will make sure that the returned value feasible to be supported in the frontend and saved in the database. In contrast, the population timetable genomes will be sent to the Tournament Selection phase to select the genomes with the best fitness value in a population. Then, the selected genomes are the parent genomes that will be crossed using the single point crossover method to create offsprings. Like the nature of genetic crossover, there is a set chance that the offspring may experience mutations. Hence, the algorithm will mutate the offsprings by shuffling the genes in affected genomes. The entire process will be looped until an optimal timetable solution is produced or stop condition is met.

5.4.3 Execution Steps

This section will provide the detailed design and explanation of each step in the algorithm execution flow.

5.4.3.1 Initialization

The first phase of the generation process is the initialization phase. In this phase, population of genomes containing timetable allocation will be generated randomly based on the given population size. This is because increasing the population size could increase the search range of the algorithm for the optimal solution. Nonetheless, it will cause the algorithm to slow down due to low computational power resulting from too large of a search space.

5.4.3.1.1 Static Allotment Algorithm (SAA)

To generate the genomes for each population, the SAA will require the allotment set identifier to continue. This is because STSAMS has the feature to group the allotments into different sets to isolate the context of use including constraint settings and class sessions like morning or afternoon classes. So, SAA will only consider the constrain settings for specific allotment set. Then, SAA will extract the constrain settings for static timeslots from the database to initialize the genome. In static timeslot constraint, it contains data of the classes and subjects. Using the subject information, SAA can extract a list of teachers that could teach the specific subjects. After that, SSA will find distributions details of the static timeslot from database to sort out the static timeslots with different block size. For example, if a static timeslot has 1 single block and 2 triple blocks, the result would be “1, 3, 3”.

During the constraint setting for static timeslot, admins are also required to set the frame details for each static timeslot. This frame details are referring to the days and period positions that the static timeslots can be allocated. If there are no spaces left in the static timeslot frame, the blocks will be allocated randomly in positions stated in the study hour constraint. These data are then merged into a single class named “Static_Timeslot”. If there is more than one static timeslot in the allotment set, the process will be repeated to initialize all static timeslots. Lastly, SSA will do this process again by few numbers of time according to the set population size. The structure of the genome will look like the figure shown below:

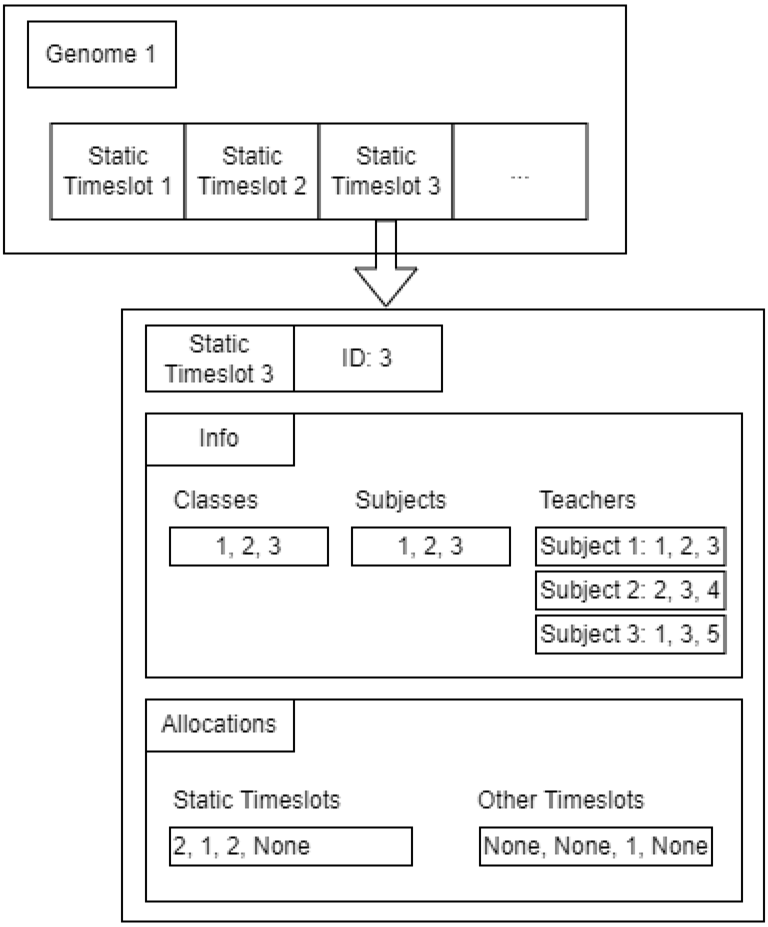


Figure 5.4.3.1.1.1: Genetic Encoding Design of SAA.

As stated before, each static timeslot instances will consist of the information regarding to the involved classes, subjects, and teaching teachers. All teaching teachers are mapped to the subject identifier stated in the in the list of involved subjects. For example, teachers with the ID of 1, 2, and 3 can teach the subject with an ID of 1. In the allocation section of the genetic encoding, it has two different allocation types. The distribution blocks that could fit into the specified frame for the static timeslot will be stored into the static timeslots list, while other timeslots list will contain the leftovers. Then, the other timeslots list will have the frame size based on the study hour constraint set. The example on how the distribution blocks is allocated into both lists is stated in the explanation below: -

Static Timeslot Frame Size

Table 5.4.3.1.1.1: Example Frame Size.

3	1	2	-	-

Distribution Blocks

Table 5.4.3.1.1.2: Example Distribution Blocks.

2	1	2	1	-

SSA will first evaluate the static timeslot frame size. Then, the distribution blocks will be filled into the frame. According to the frame, the first frame could accommodate total distribution blocks of 3. So, the first and second distribution blocks could fit in it because they totaled 3, leaving two more blocks to be allocated. The frame size of 1 is now unable to hold any blocks as the first distribution blocks after allocating 2 and 1 is 2. So, SSA will fill a null value into the frame. SSA will continue until the distribution blocks list is exhausted. The result of the allocation will be shown in the following tables: -

Static Timeslot List

Table 5.4.3.1.1.3: Example Static Timeslot List.

2	1	None	2	-

Other Timeslot List

Table 5.4.3.1.1.4: Example Timeslot List.

1	None	None	None	None

Ultimately, the content in the lists will be shuffled to add randomness to the allocation. As aforementioned, there are two types of algorithms with differ constraints checking. So, the genetic encodings for both algorithms will be appearing to be mostly similar in its structure.

5.4.3.1.2 Dynamic Allotment Algorithm (DAA)

The algorithm will now require two initial important information to continue its execution which are the allotment set ID and static timetable ID. The allotment set ID will be the group of different contexts of use for different allocations, while static timetable ID is the identifier to reference to the saved static timetable that is generated by SAA. The reserved periods by the static timetable will act as a constraint to prevent slot overlapping in DAA. Like SAA, DAA will also extract the data of the related constraint settings stated in Table 0. The table below will show the description of each constraints involved: -

Table 5.4.3.1.2.1: Description of the constraints involved in DAA.

No.	Constraint	Description
1	Teacher Preferences	This constraint settings contains data regarding to the preferred range of period position by the teachers
2	Study Hours	This constraint settings contains the frame for timetable allocation which is the periods that are available to accommodate subject blocks.
3	Subject Distributions	The constraint settings contain the information on the quantity of single, double, and triple blocks of subjects.
4	Venue Usages	The constraint settings decide the venues allocations for different subjects.

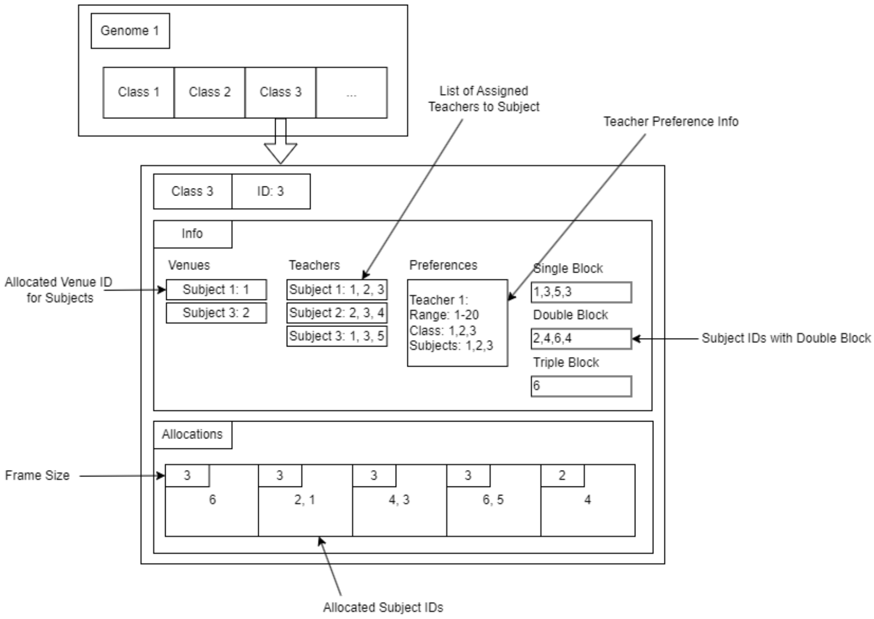


Figure 5.4.3.1.2.1: Genetic Encoding Design of DAA.

In DDA, the classes involved are the genes of a genome instead of an information in a gene. So, the class ID will represent ID of each gene. The information of a gene consists of venues allocation, teacher assignments, teacher preferences, and distribution block quantities. Then, the distribution blocks are allocated in slots with different frame size in the allocation section. The value from the distribution block list will utilize the Queue data structure to pop out the subject IDs. For instances, if Subject 6 takes 3 spaces, it will completely occupy the frame. Hence, Subject 2, and Subject 1 will need to fill in the second box. To apply randomness to the allocations, the order of the subject IDs in the block distribution lists will be shuffled, then the assignment of frame blocks in allocation section is performed again.

In the scenario shown in Figure 5.4.3.1.2.1, subject with the ID of 6 will have 1 double and 1 triple distribution blocks. The teacher with the ID of 1 does not prefer to teach Subject 6 since the list of preferred subjects only include Subject 1, Subject 2, and Subject 3. In this case, Teacher 1 will be satisfied with teaching in the class because Class 3 is in the preferred class list. In the teacher assignment, it shows that Teacher 1 will teacher only Subject 1 or Subject 3 depending on the allotment. If Teacher 1 was selected to teach Subject 1, the teacher will conduct the class in Venue 1.

5.4.3.2 Fitness Evaluation

In SSA, the genomes will only be evaluated on clashing between teachers and classes. There is no zero chance that the subjects will experience clashing because same subjects can be taught in a school at the same time. However, teachers cannot be in the different classes and classes cannot have more than one lesson happening concurrently unless the subjects and classes are in the same static timeslot. All the static timeslots in a genome will be collected and transposed to find clashing. The transposing logic is shown the following steps: -

Table 5.4.3.2.1: Allocation for Static Timeslot 1.

Class: 1, 2 / Teacher: 4, 2					
Periods	1	2	3	4	5
Allocated?	0	1	1	0	0

Table 5.4.3.2.2: Allocation for Static Timeslot 2.

Class: 2, 3 / Teacher: 1, 2					
Periods	1	2	3	4	5
Allocated?	1	1	0	1	0

First, the allocations for static timeslot 1 and 2 will be transposed, resulting in answer shown in Table 0. According to the “Allocated?” column is referred to as whether the period is allocated with the static timeslot.

Table 5.4.3.2.3: Transposed periods of Static Timeslot 1 and 2.

Periods	1	2	3	4	5
Allocated Classes and Teachers	Class: 2, 3 Teacher: 1, 2	Class: 1, 2, 2, 3 Teacher: 1, 2, 2, 4	Class: 1, 2 Teacher: 2, 4	Class: 2, 3 Teacher: 1, 2	Class: - Teacher: -

By looking at the class and teacher lists, clashes are decided by the repetitive values. To optimize the clash checking, the original size of class list containing 1, 2, and 3 will be compared to the size of the transposed lists such as class lists with elements of 1, 2, 2, and 3 in the second period. It can be deducted that there is clashing if the comparing sizes is different. Then, the following formula will be used to calculate the fitness value of the genomes.

$$Fitness\ Value = \frac{1}{(1 + \#Classes\ Clashes + \#Teacher\ Clashes)}$$

If an optimal solution is found, the fitness value will be 1.

In DAA, the genomes will be evaluated on clashing between venues and teachers. In this case, classes are not included in evaluate directly because clashes between classes will be depended on the venue allocated and assigned teachers. Like SAA, DAA will also compile all the timeslot allocation in the classes to be transposed.

Table 5.4.3.2.4: Allocation for Class 1.

<table>
<tr><td colspan="6">Subject 1: (Teachers: 1, 2 / Venues: -)
Subject 2: (Teachers: 2, 3 / Venues: 1, 2)
Subject 3: (Teachers: 4, 5 / Venues: -)
Subject 4: (Teachers: 4, 6 / Venues: -)</td></tr>
<tr><td>Periods</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr>
<tr><td>Subject</td><td>1</td><td>3</td><td>2</td><td>2</td><td>4</td></tr>
</table>

Table 5.4.3.2.5: Allocation for Class 2.

<table>
<tr><td colspan="6">Subject 1: (Teachers: 1, 2 / Venues: -)
Subject 2: (Teachers: 3, 2 / Venues: 1, 2)
Subject 3: (Teachers: 5, 4 / Venues: -)</td></tr>
<tr><td>Periods</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr>
<tr><td>Subject</td><td>1</td><td>2</td><td>2</td><td>1</td><td>3</td></tr>
</table>

First, the allocations for static timeslot 1 and 2 will be transposed, resulting in answer shown in Table 0. The subject column is referring to the ID of a subject. The IDs can be used to obtain information about the assigning teachers and allocating venues. Following the design the initialization phase, the first element of both teacher list and venue list will be selected as the chosen value.

Table 5.4.3.2.6: Transposed periods of Static Timeslot 1 and 2.

Periods	1	2	3	4	5
Selected Teachers	1, 1	4, 2	2, 3	2, 3	4, 5
Selected Venues	-, -	-, 1	1, 1	-, -	-, -

By looking at the venue and teacher lists, clashes are decided by the repetitive values. To optimize the clash checking, the size of selected teacher list containing 1, and 1 will be compared to the size of the lists after removing the repetitive values. It can be deducted that there is clashing if the comparing sizes is different. From Table 0, it can be seen that Period 1 is having clashes for Teacher 1 and Period 3 is having clashes for Venue 1. Then, the following formula will be used to calculate the fitness value of the genomes.

$$Fitness\ Value = \frac{1}{(1 + \#Venue\ Clashes + \#Teacher\ Clashes)}$$

5.4.3.3 Selection

The type of selection used in both algorithms is Roulette Wheel Selection. This methodology will select the genomes based on their probability. The probability of selection is provided as fitness value and SAA and DAA. So, the fitness values evaluated in the fitting phase can be used to represent the probability for each genome to be selected in the Roulette Wheel. This method of selection will ensure that the most fit timetable solution can be chosen to perform the cross-over to produce better offspring. By luck, the timetable solutions or genomes that are worst in fitness can still be chosen as parent genomes. This will enable all genomes to have the chance to be participating in the selection phase. This randomness will ensure better results can be produce as it is not confirmed that two best parent genomes will produce offsprings that are better than the parents.

Table 5.4.3.3.1: The example genomes with mock fitness value and its percentage of occupation in the Roulette Wheel.

Genomes	Fitness Value	Percentage in Roulette Wheel
A	0.3	30%
B	0.2	20%
C	0.35	35%
D	0.15	15%

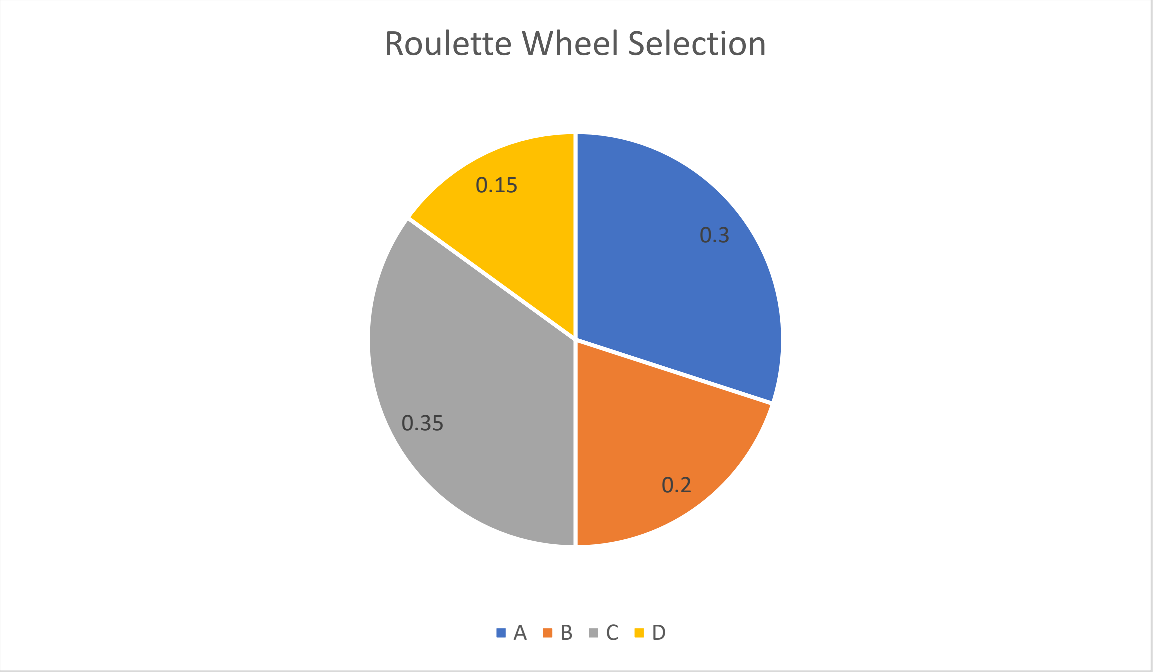


Figure 5.4.3.2.1: Pie Chart on the Probability of Genome to be Chosen.

From Figure 5.4.3.2.1, it can be imagined that the wheel is spun. Genome C will have the highest chance of landing on the wheel's arrow. So, it is very likely that Genome C will be chosen to be the parent genomes in the crossover phase. At the same time, the other genomes may still win the selection and become the parent genome for the crossover phase. The likeliness of the genomes getting chosen are C > A > B > D.

5.4.3.4 Crossover

Single point crossover method will used in the crossover phase. The crossover process will cut genomes into two parts, switch the parts, and lastly join them back together as shown in Figure. In SAA and DAA, the cutting point of the genomes are randomized, so that there is higher probability to create offsprings that are better in fitness.

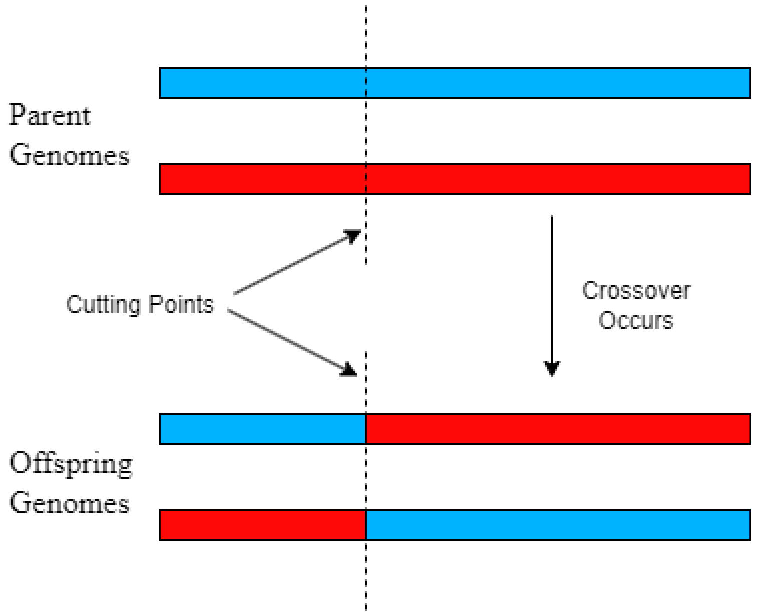


Figure 5.4.3.4.1: Visualization of Single Point Crossover

To demonstrate the crossover process, example is shown in Table 0. Parent genome A and B will have a cutting point. After separating the genomes into two pieces, the back piece of Genome A will exchange with Genome B's, creating a completely new offsprings. The offsprings will have the genes of both parents making them identical to their parents but still having the genes structured differently. In SAA and DAA the genomes will be first sorted according to the ID of classes or static timeslots. Different colors of text is used to provide clearly separations between genes of Parent A and Parent B.

Table 5.4.3.4.1: Example Result of Single Point Crossover.

Parent A	1	3	4	2	1	5
Parent B	2	7	2	8	9	3
Offspring A	1	3	2	8	9	3
Offspring B	2	7	4	2	1	5

5.4.3.5 Mutation

In the mutation phase, the shuffle mutation strategy will be applied. This type of mutation will shuffle the sequence of the genes are ordered in, creating a genome completely different from its original form. The mutation will only happen after each crossover phase. Just like in real life, genetic errors may happen during the crossover of chromosomes. The chances of a genome mutating are 50% and 30% for SAA and DAA respectively.

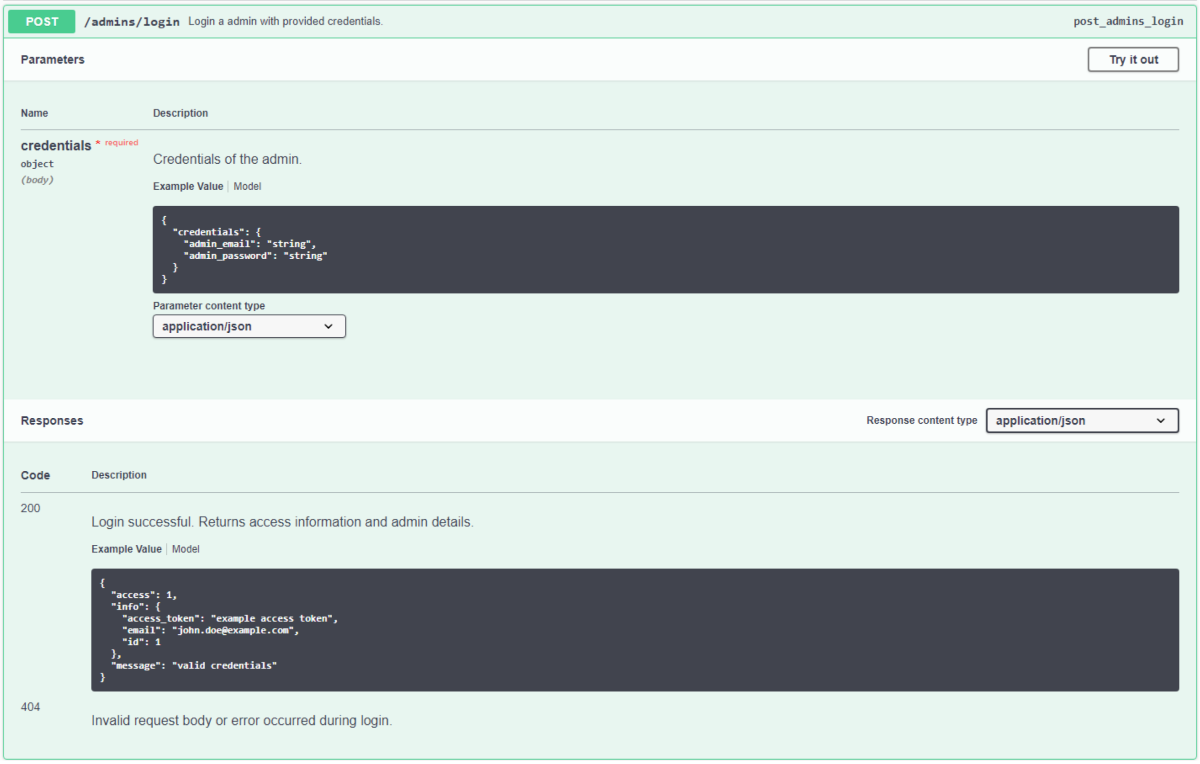
Table 5.4.3.5.1: Example Result of Shuffle Mutation.

Original Genome	5	2	1	2	3
Mutated Genome	1	5	2	3	2

5.5 API Routes Design

In this project, Swagger API Documentation tool is utilized to create the design for every route in STSAMS. Every route will contain information about the parameters used in the requests and the possible response structure will be frontend receive including the response codes and example values. The routes will be grouped into different section to ease the reading and reduce the complexity of the documentation.

5.5.1 Admins Tag



POST `/admins/login` Login a admin with provided credentials. post_admins_login

Parameters

Name	Description
credentials * required object *(body)*	Credentials of the admin.

```
{
  "credentials": {
    "admin_email": "string",
    "admin_password": "string"
  }
}
```

Parameter content type: application/json

Responses

Response content type: application/json

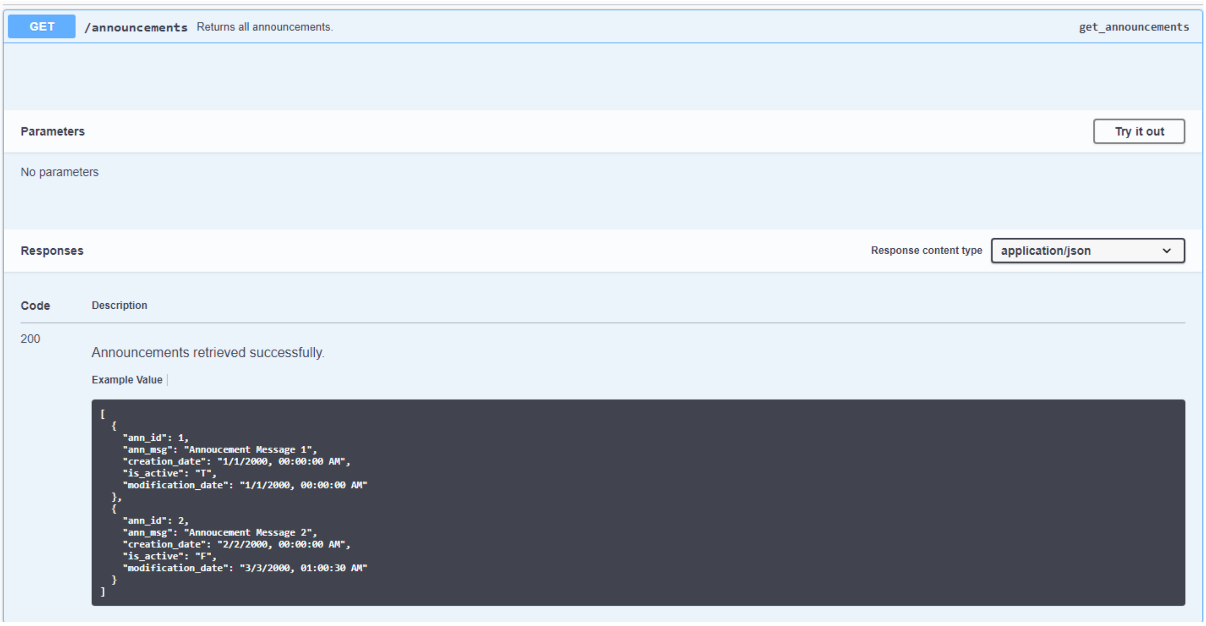
Code	Description
200	Login successful. Returns access information and admin details.

```
{
  "access": 1,
  "info": {
    "access_token": "example access token",
    "email": "john.doe@example.com",
    "id": 1
  },
  "message": "valid credentials"
}
```

Code	Description
404	Invalid request body or error occurred during login.

Figure 5.5.1.1: The design of admin login route.

5.5.2 Announcements Tag



/announcements Returns all announcements. get_announcements

Parameters

No parameters

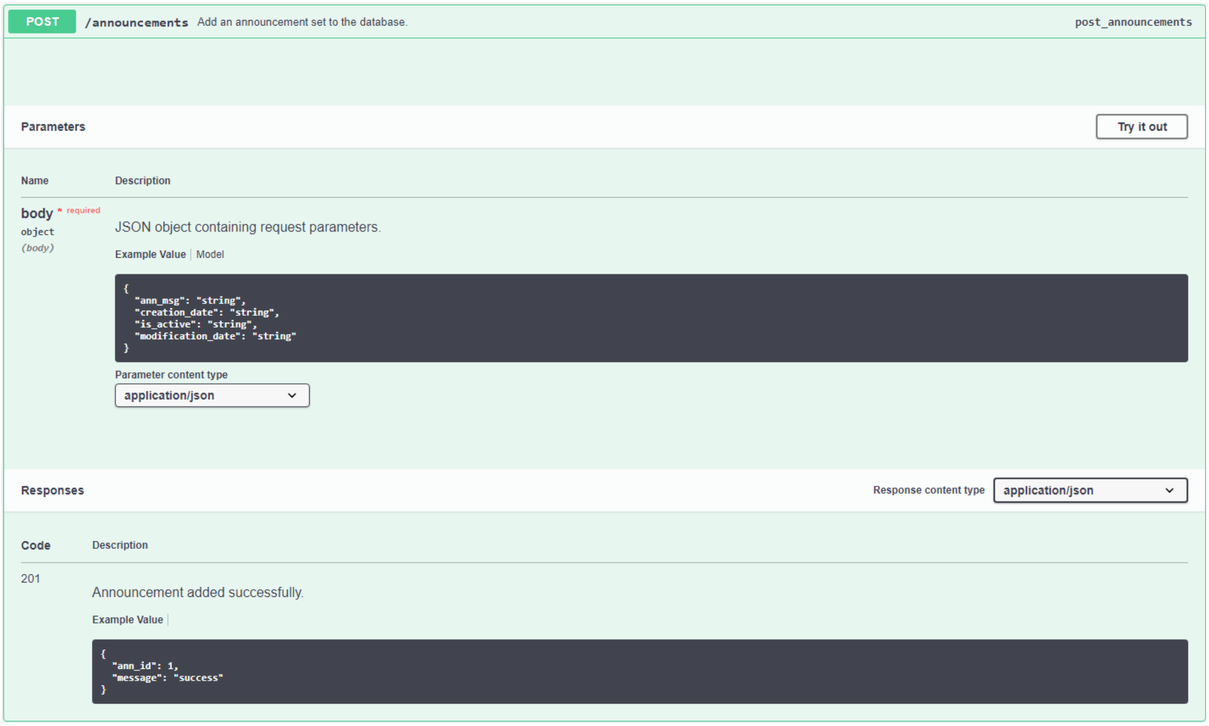
Responses — Response content type: application/json

Code | Description
200 | Announcements retrieved successfully.

Example Value

```
[
  {
    "ann_id": 1,
    "ann_msg": "Annoucement Message 1",
    "creation_date": "1/1/2000, 00:00:00 AM",
    "is_active": "T",
    "modification_date": "1/1/2000, 00:00:00 AM"
  },
  {
    "ann_id": 2,
    "ann_msg": "Annoucement Message 2",
    "creation_date": "2/2/2000, 00:00:00 AM",
    "is_active": "F",
    "modification_date": "3/3/2000, 01:00:30 AM"
  }
]
```

Figure 5.5.2.1: The design of getting all announcements route.



/announcements Add an announcement set to the database. post_announcements

Parameters

Name | Description
body * required
object
(body) | JSON object containing request parameters.

Example Value

```
{
  "ann_msg": "string",
  "creation_date": "string",
  "is_active": "string",
  "modification_date": "string"
}
```

Parameter content type: application/json

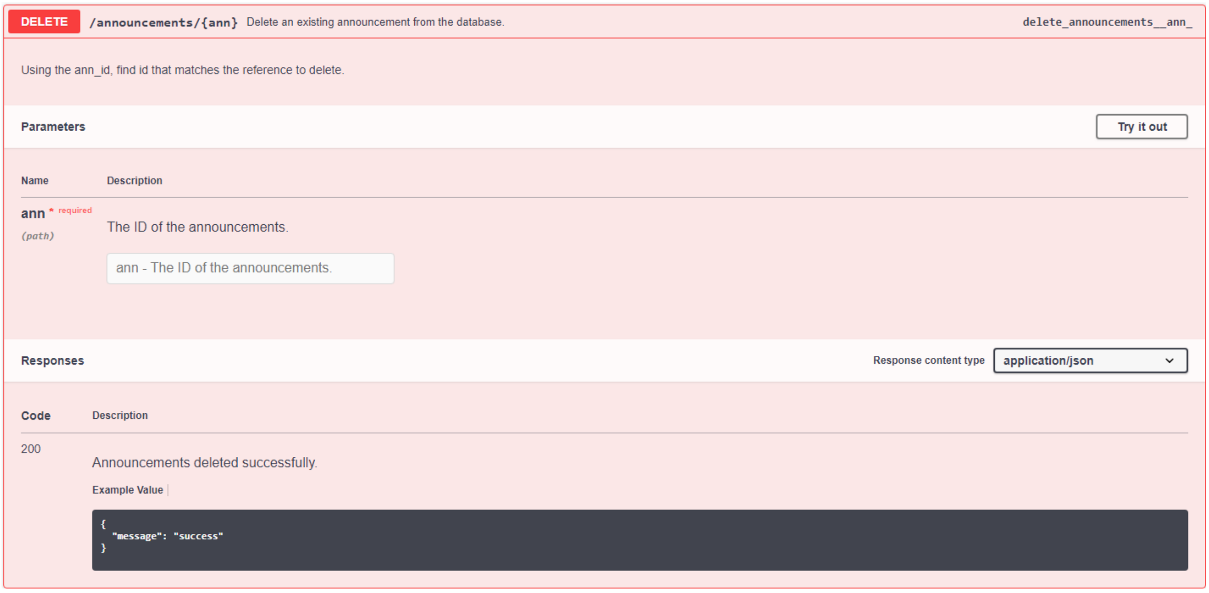
Responses — Response content type: application/json

Code | Description
201 | Announcement added successfully.

Example Value

```
{
  "ann_id": 1,
  "message": "success"
}
```

Figure 5.5.2.2: The design of announcement creation route.



/announcements/{ann} Delete an existing announcement from the database.

delete_announcements__ann_

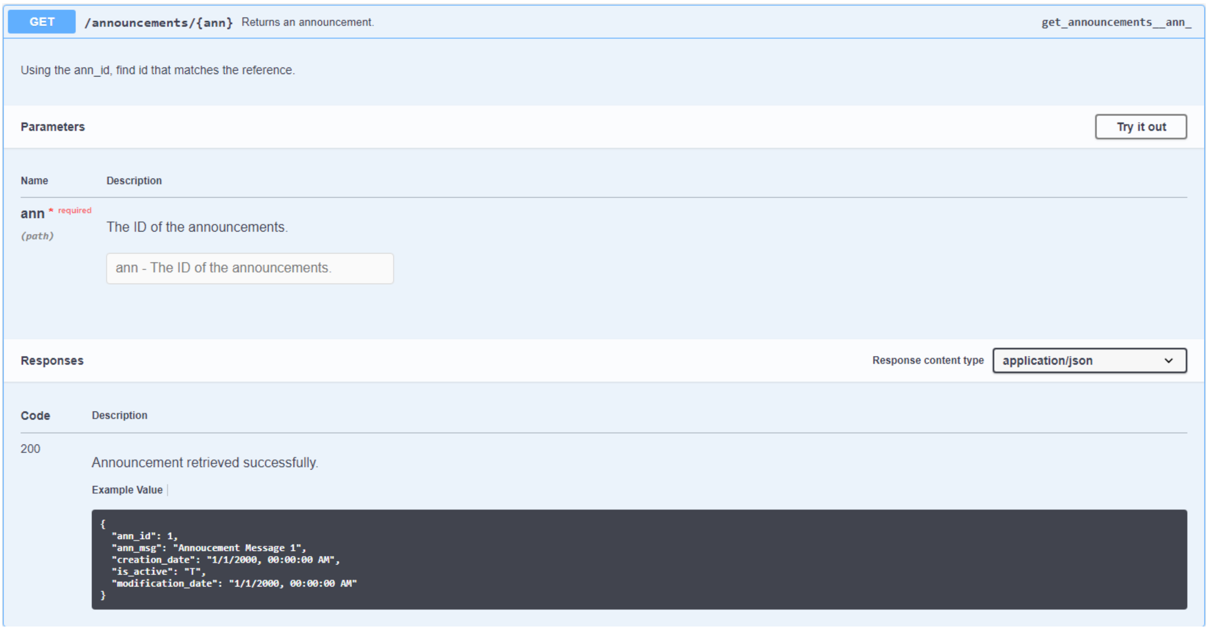
Using the ann_id, find id that matches the reference to delete.

Name	Description
ann * required (path)	The ID of the announcements. ann - The ID of the announcements.

Code	Description
200	Announcements deleted successfully. Example Value

```
{
  "message": "success"
}
```

Figure 5.5.2.3: The design of announcement deletion route.



/announcements/{ann} Returns an announcement.

get_announcements__ann_

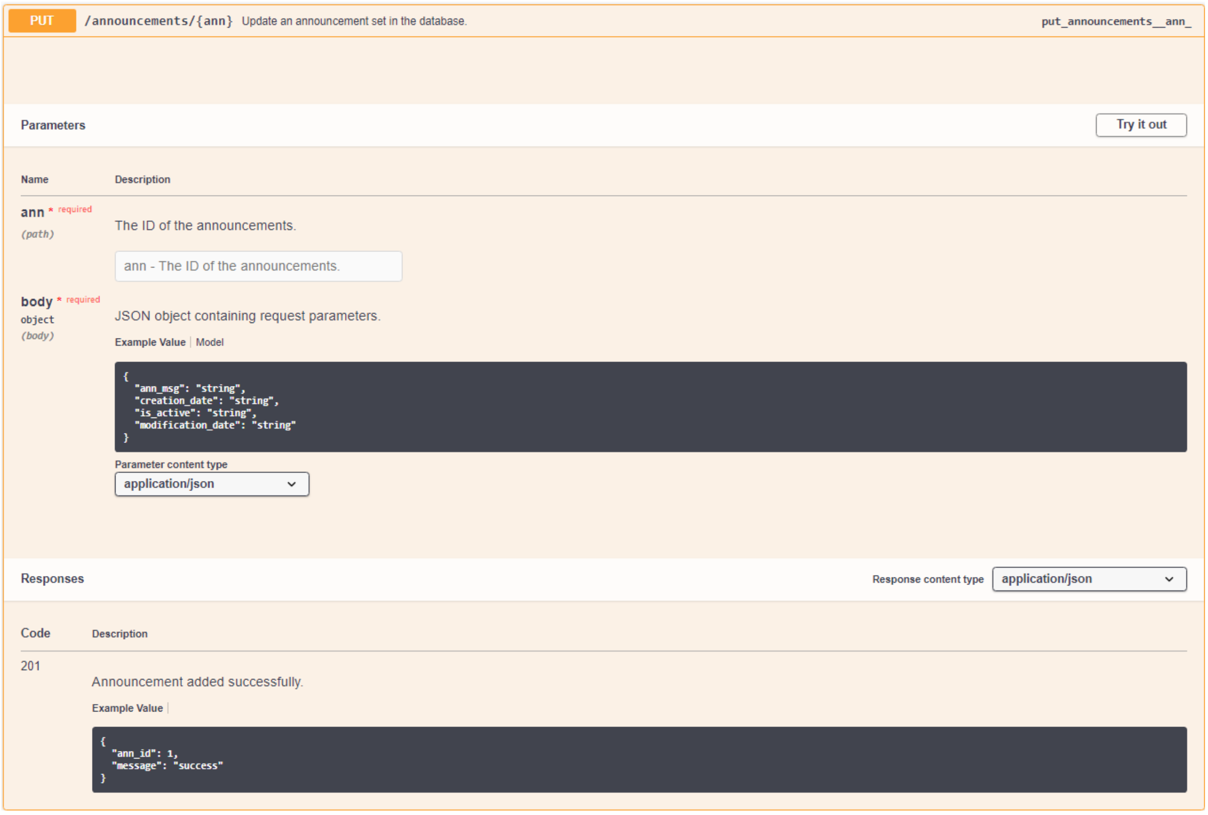
Using the ann_id, find id that matches the reference.

Name	Description
ann * required (path)	The ID of the announcements. ann - The ID of the announcements.

Code	Description
200	Announcement retrieved successfully. Example Value

```
{
  "ann_id": 1,
  "ann_msg": "Annoucement Message 1",
  "creation_date": "1/1/2000, 00:00:00 AM",
  "is_active": "T",
  "modification_date": "1/1/2000, 00:00:00 AM"
}
```

Figure 5.5.2.4: The design of getting specific announcement route.



`/announcements/{ann}` Update an announcement set in the database.

Name	Description
ann * required *(path)*	The ID of the announcements. ann - The ID of the announcements.
body * required object *(body)*	JSON object containing request parameters.

```
{
  "ann_msg": "string",
  "creation_date": "string",
  "is_active": "string",
  "modification_date": "string"
}
```

Parameter content type: application/json

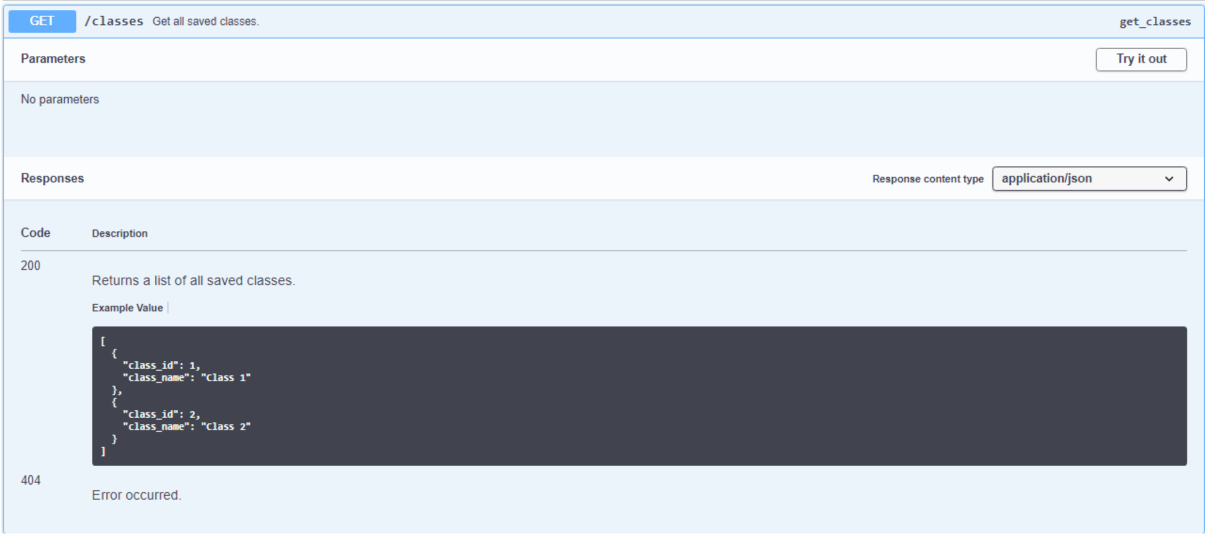
Response content type: application/json

Code	Description
201	Announcement added successfully.

```
{
  "ann_id": 1,
  "message": "success"
}
```

Figure 5.5.2.5: The route design for updating specific announcement.

5.5.3 Classes Tag



GET /classes Get all saved classes. get_classes

No parameters

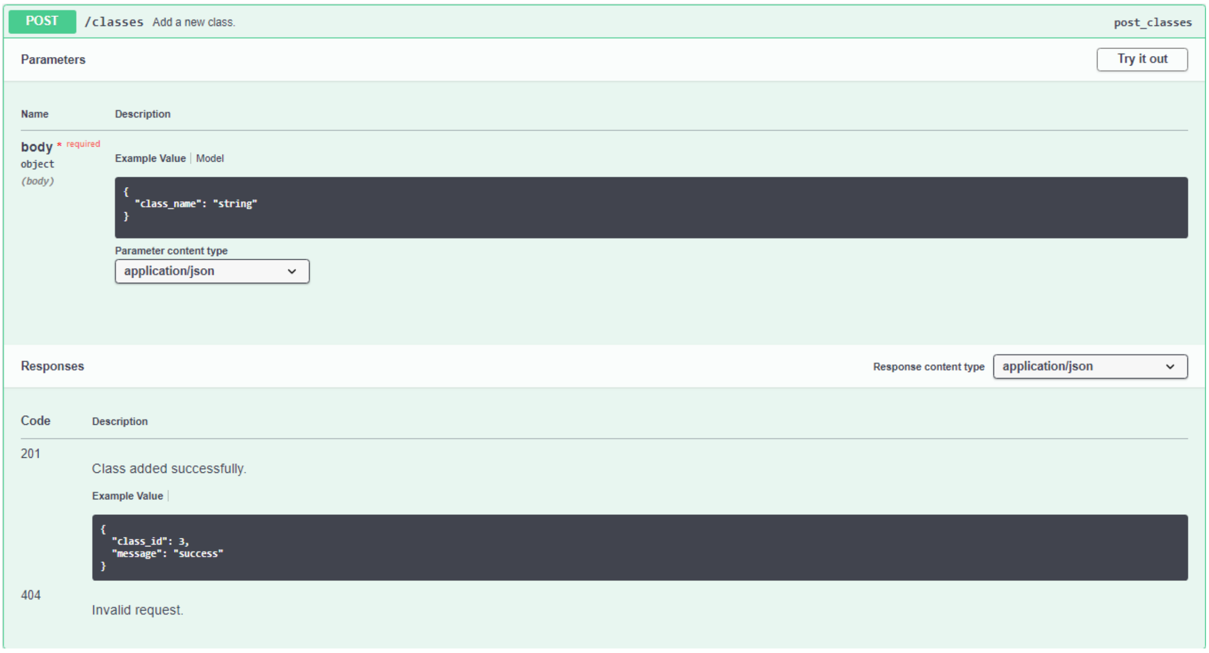
Response content type: application/json

Code	Description
200	Returns a list of all saved classes.
404	Error occurred.

Example Value

```
[
  {
    "class_id": 1,
    "class_name": "Class 1"
  },
  {
    "class_id": 2,
    "class_name": "Class 2"
  }
]
```

Figure 5.5.3.1: The route design for getting all classes.



POST /classes Add a new class. post_classes

Name	Description
body * required object (body)	Example Value | Model

```
{
  "class_name": "string"
}
```

Parameter content type: application/json

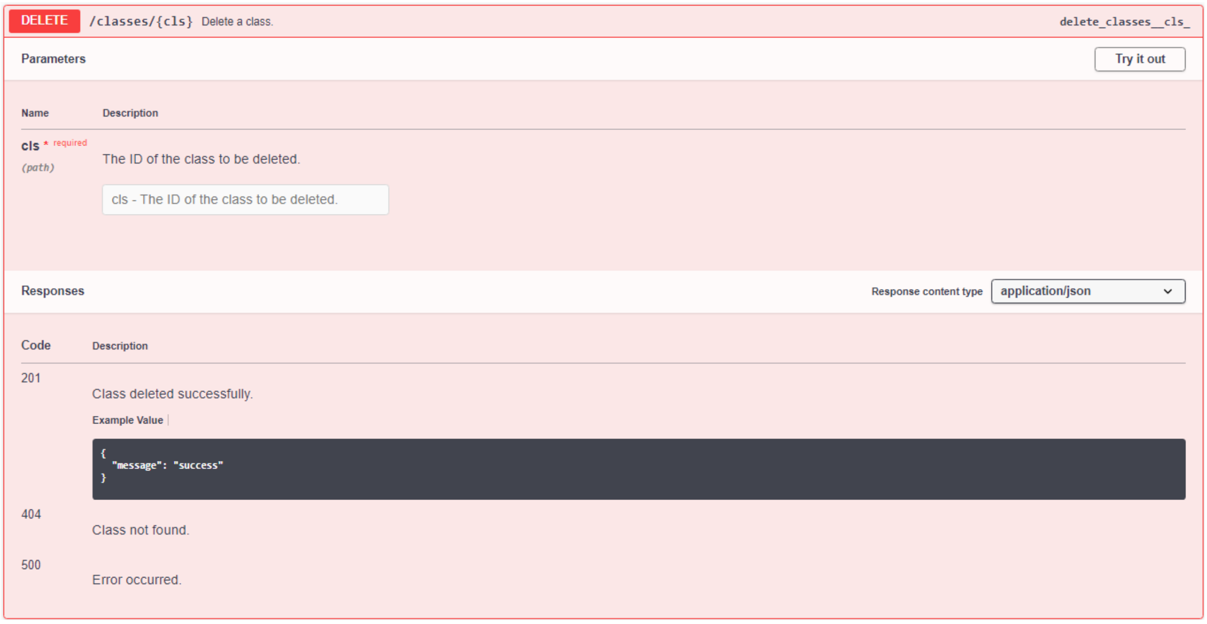
Response content type: application/json

Code	Description
201	Class added successfully.
404	Invalid request.

Example Value

```
{
  "class_id": 3,
  "message": "success"
}
```

Figure 5.5.3.2: The route design for creating new classes.



/classes/{cls} Delete a class. delete_classes__cls_

Parameters

Name	Description
cls * required (path)	The ID of the class to be deleted. cls - The ID of the class to be deleted.

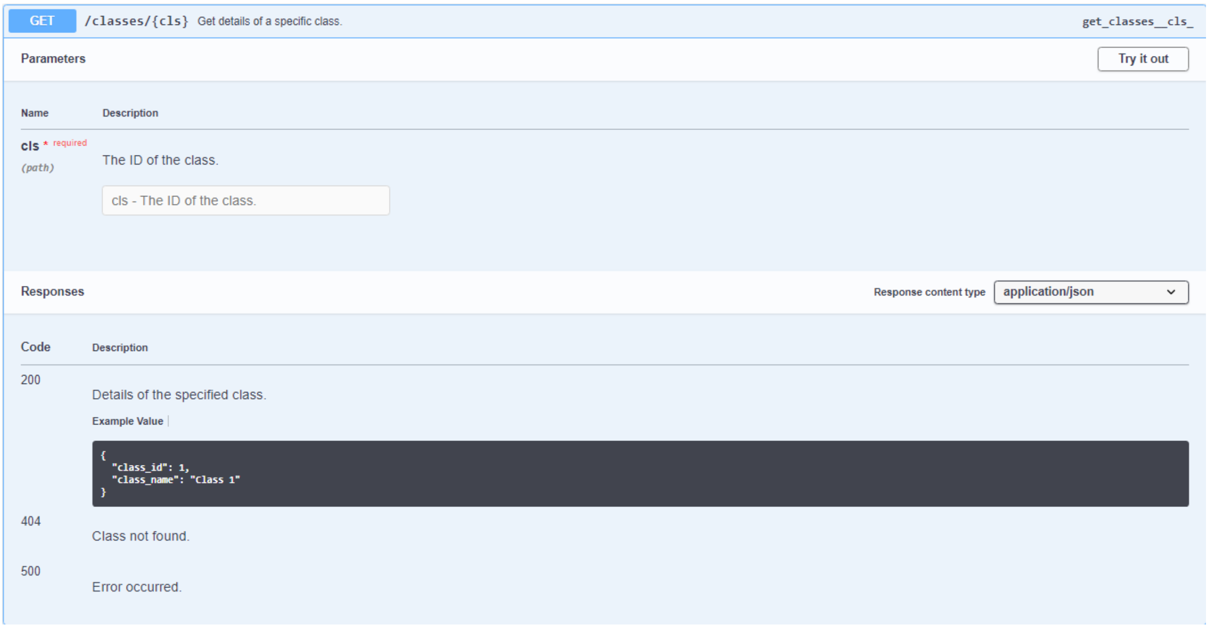
Responses

Response content type: application/json

Code	Description
201	Class deleted successfully. Example Value
404	Class not found.
500	Error occurred.

```
{
  "message": "success"
}
```

Figure 5.5.3.3: The route design for deleting existing classes.



/classes/{cls} Get details of a specific class. get_classes__cls_

Parameters

Name	Description
cls * required (path)	The ID of the class. cls - The ID of the class.

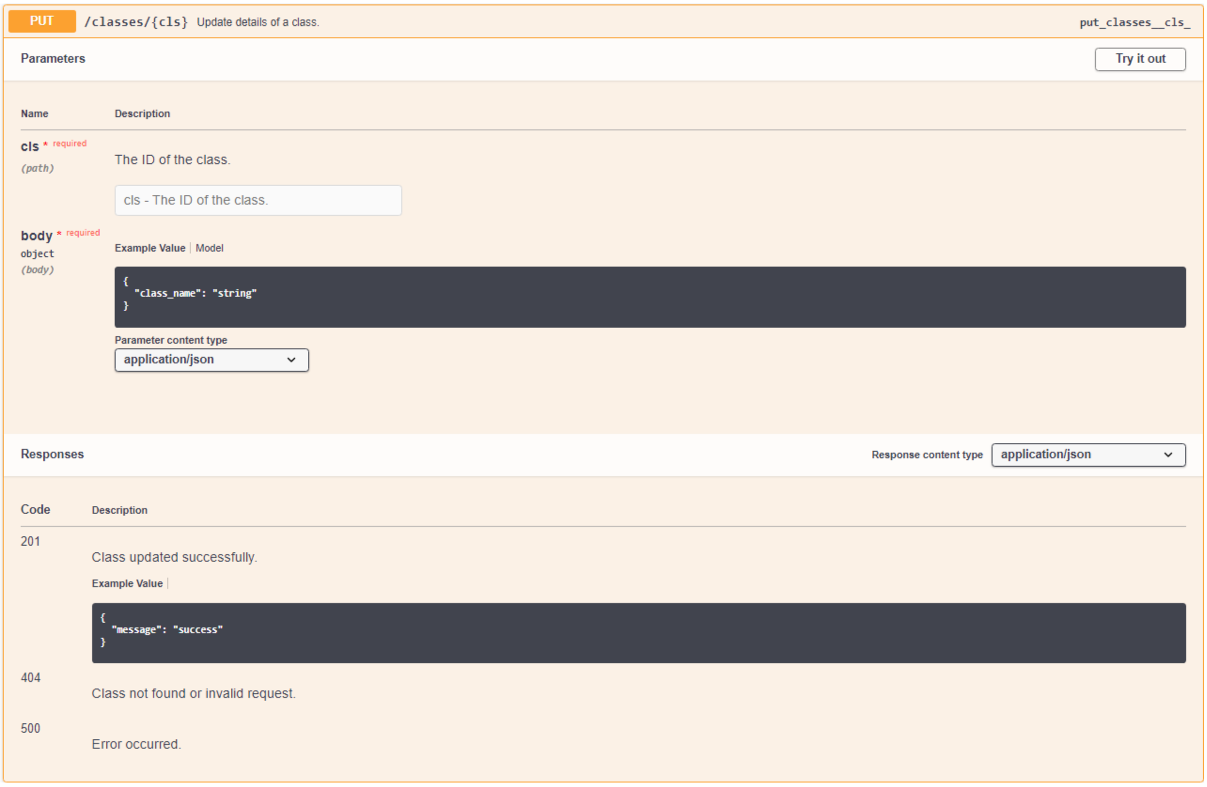
Responses

Response content type: application/json

Code	Description
200	Details of the specified class. Example Value
404	Class not found.
500	Error occurred.

```
{
  "class_id": 1,
  "class_name": "Class 1"
}
```

Figure 5.5.3.4: The route design for getting a specific class.



PUT /classes/{cls} Update details of a class. put_classes__cls_

Parameters

Try it out

Name	Description
cls * required (path)	The ID of the class. cls - The ID of the class.
body * required object (body)	Example Value | Model

```
{
  "class_name": "string"
}
```

Parameter content type: application/json

Responses

Response content type: application/json

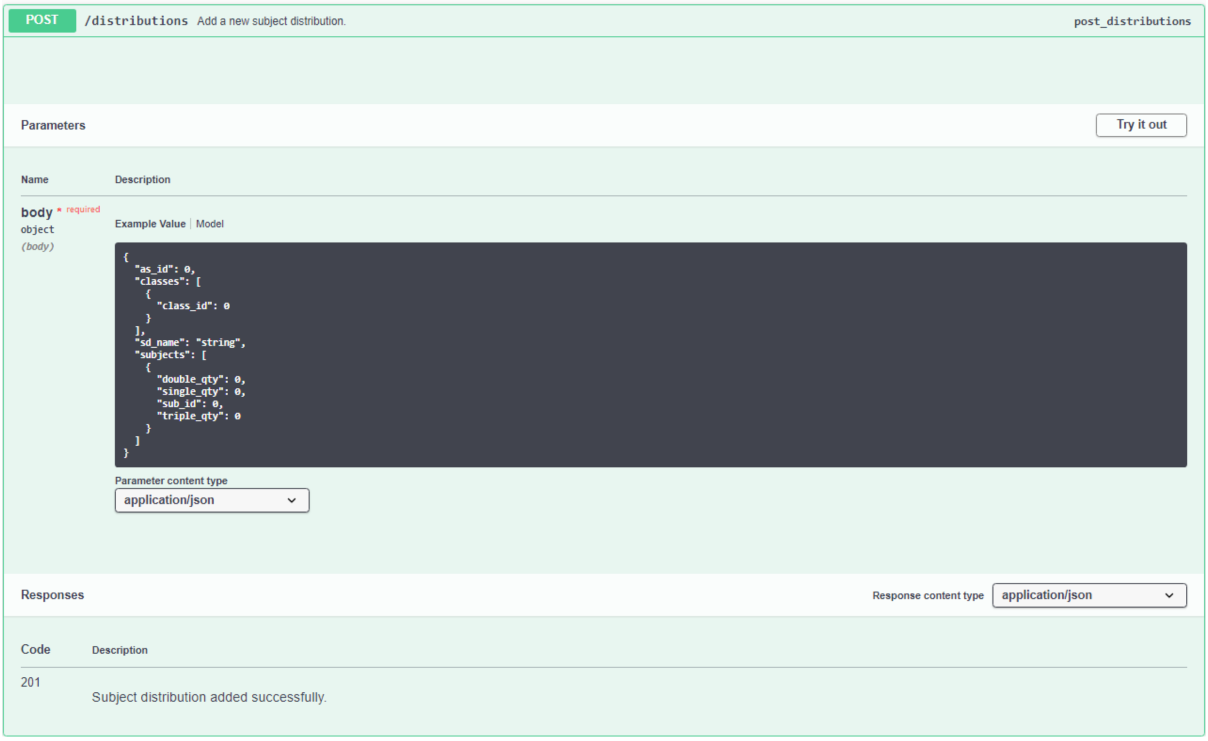
Code	Description
201	Class updated successfully. Example Value

```
{
  "message": "success"
}
```

Code	Description
404	Class not found or invalid request.
500	Error occurred.

Figure 5.5.3.5: The route design for updating an existing class.

5.5.4 Subject Distributions Tag



POST /distributions Add a new subject distribution. post_distributions

Parameters

Name	Description
body * required object *(body)*	Example Value | Model

```
{
  "as_id": 0,
  "classes": [
    {
      "class_id": 0
    }
  ],
  "sd_name": "string",
  "subjects": [
    {
      "double_qty": 0,
      "single_qty": 0,
      "sub_id": 0,
      "triple_qty": 0
    }
  ]
}
```

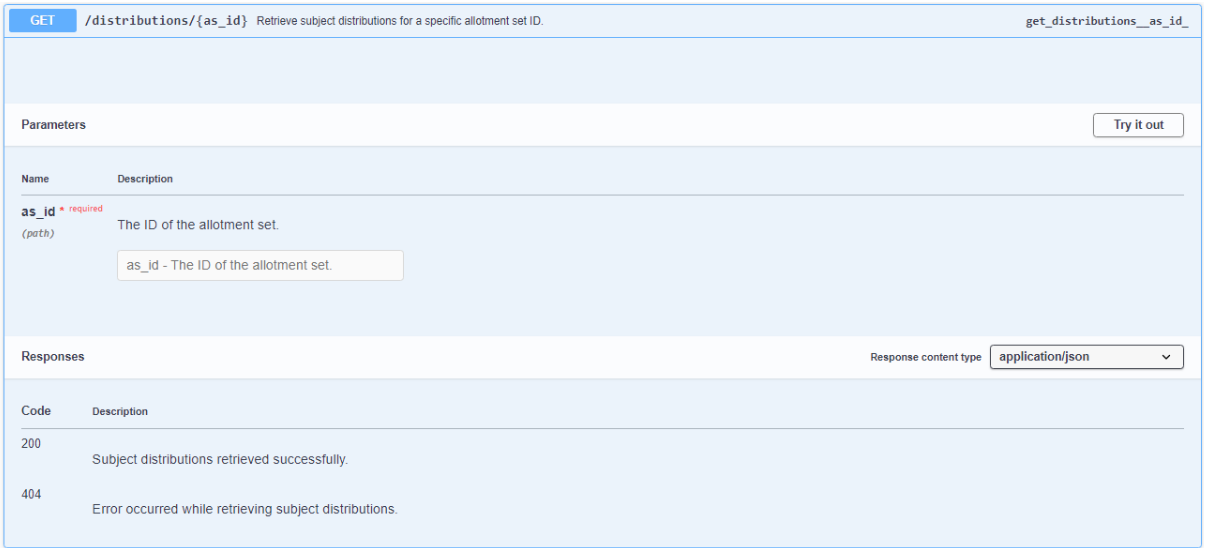
Parameter content type: application/json

Responses

Response content type: application/json

Code	Description
201	Subject distribution added successfully.

Figure 5.5.4.1: The route design for creating new subject distributions.



GET /distributions/{as_id} Retrieve subject distributions for a specific allotment set ID. get_distributions__as_id_

Parameters

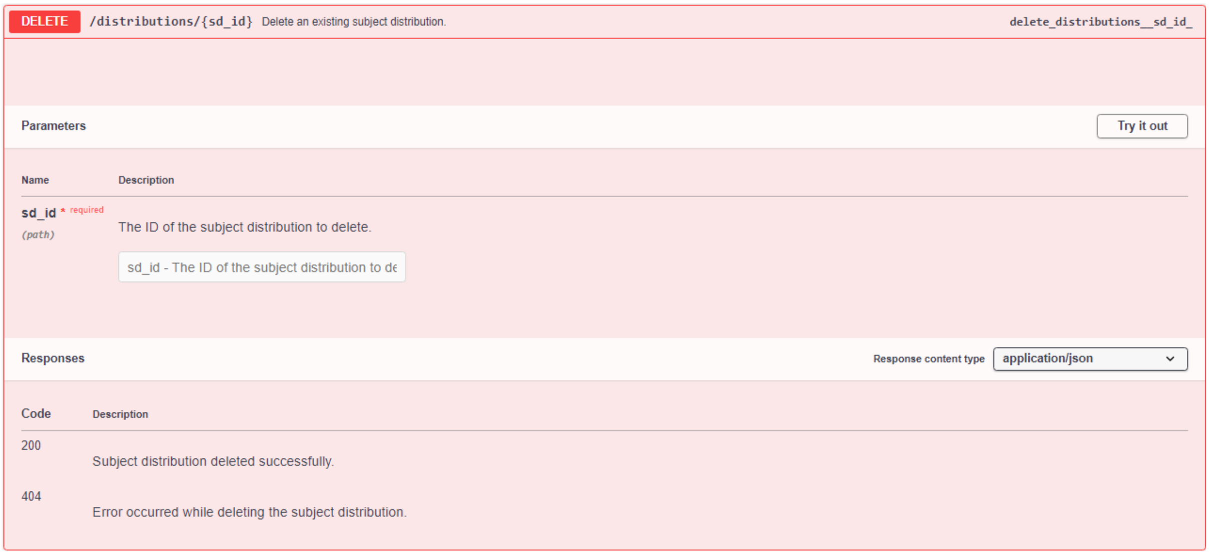
Name	Description
as_id * required *(path)*	The ID of the allotment set. as_id - The ID of the allotment set.

Responses

Response content type: application/json

Code	Description
200	Subject distributions retrieved successfully.
404	Error occurred while retrieving subject distributions.

Figure 5.5.4.2: The route design for getting specific subject distribution.



/distributions/{sd_id} Delete an existing subject distribution. delete_distributions__sd_id_

Parameters

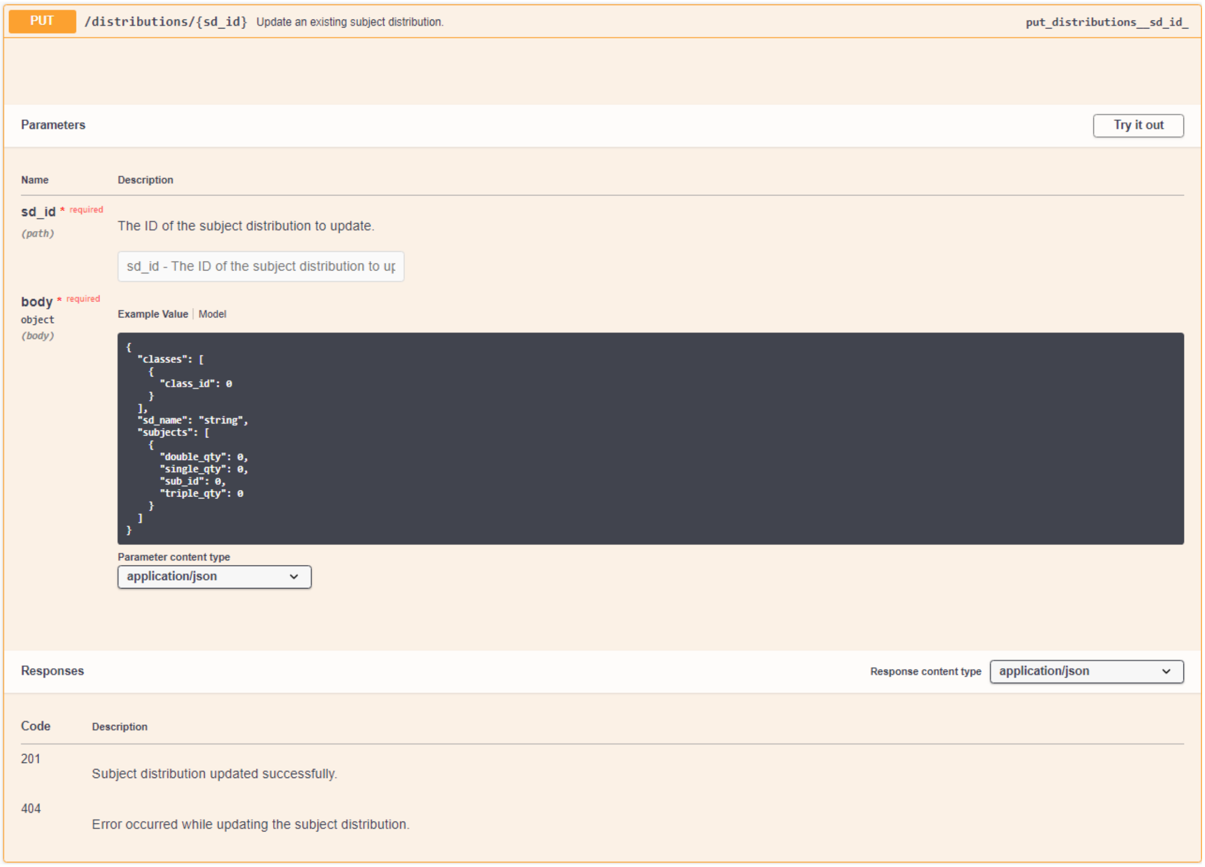
Name	Description
sd_id * required (path)	The ID of the subject distribution to delete. sd_id - The ID of the subject distribution to de

Responses

Response content type application/json

Code	Description
200	Subject distribution deleted successfully.
404	Error occurred while deleting the subject distribution.

Figure 5.5.4.3: The route design for deleting a subject distribution.



/distributions/{sd_id} Update an existing subject distribution. put_distributions__sd_id_

Parameters

Name	Description
sd_id * required (path)	The ID of the subject distribution to update. sd_id - The ID of the subject distribution to up
body * required object (body)	Example Value | Model

```
{
  "classes": [
    {
      "class_id": 0
    }
  ],
  "sd_name": "string",
  "subjects": [
    {
      "double_qty": 0,
      "single_qty": 0,
      "sub_id": 0,
      "triple_qty": 0
    }
  ]
}
```

Parameter content type application/json

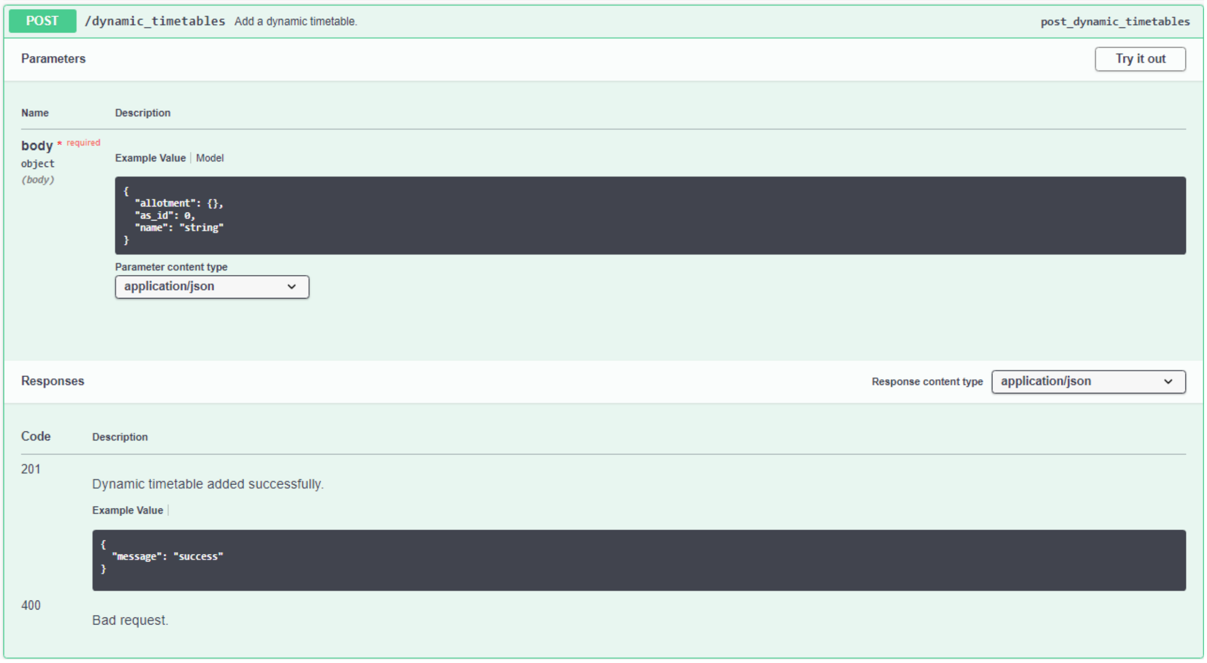
Responses

Response content type application/json

Code	Description
201	Subject distribution updated successfully.
404	Error occurred while updating the subject distribution.

Figure 5.5.4.4: The route design for updating a subject distribution.

5.5.5 Dynamic Timetables Tag



/dynamic_timetables Add a dynamic timetable. post_dynamic_timetables

Parameters

Name Description

body * required
object
(body)

Example Value | Model

```
{
  "allotment": {},
  "as_id": 0,
  "name": "string"
}
```

Parameter content type
application/json

Responses Response content type application/json

Code Description

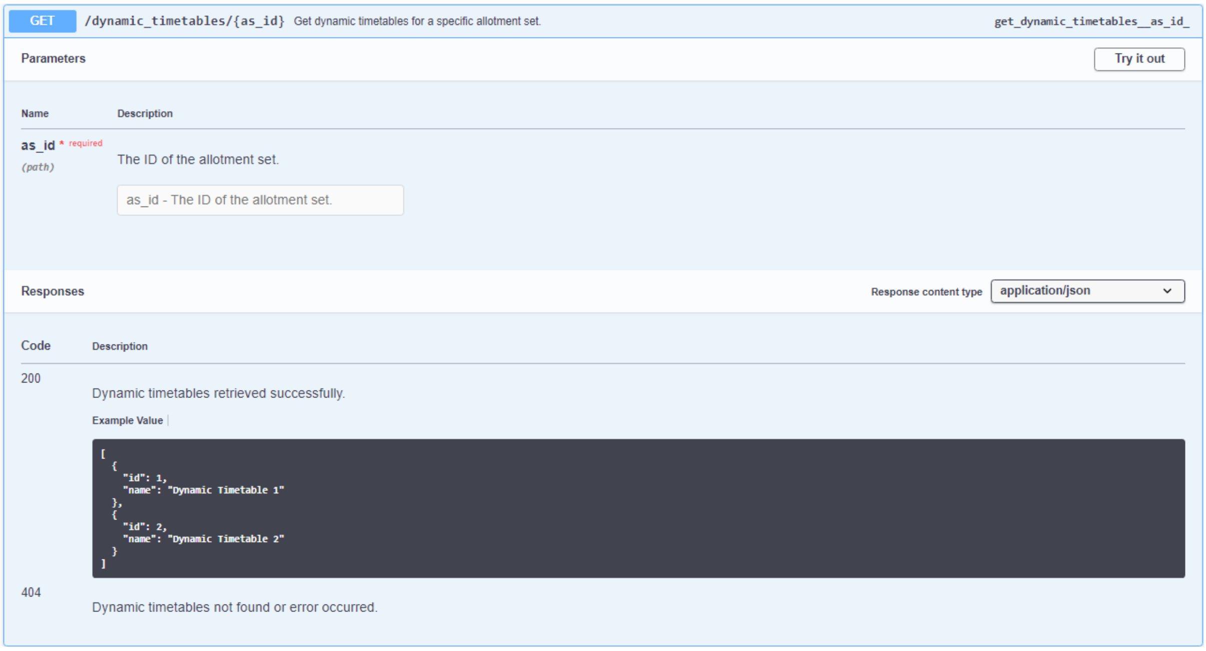
201 Dynamic timetable added successfully.

Example Value

```
{
  "message": "success"
}
```

400 Bad request.

Figure 5.5.5.1: The route design for adding a dynamic timetable.



/dynamic_timetables/{as_id} Get dynamic timetables for a specific allotment set. get_dynamic_timetables__as_id_

Parameters

Name Description

as_id * required
(path)

The ID of the allotment set.

as_id - The ID of the allotment set.

Responses Response content type application/json

Code Description

200 Dynamic timetables retrieved successfully.

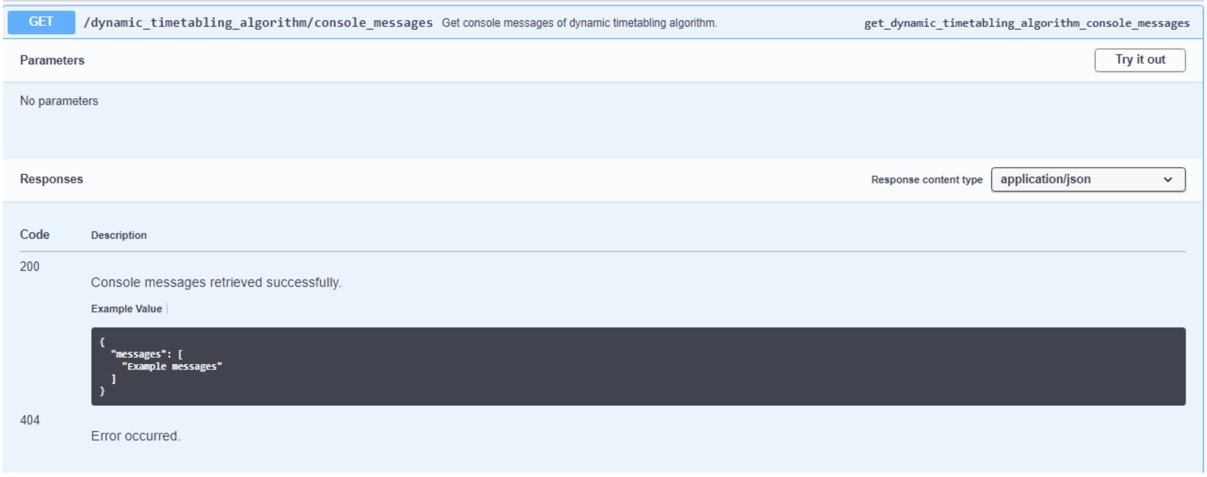
Example Value

```
[
  {
    "id": 1,
    "name": "Dynamic Timetable 1"
  },
  {
    "id": 2,
    "name": "Dynamic Timetable 2"
  }
]
```

404 Dynamic timetables not found or error occurred.

Figure 5.5.5.2: The route design for getting a dynamic timetable.

5.5.6 Dynamic Timetabling Algorithm



GET /dynamic_timetabling_algorithm/console_messages Get console messages of dynamic timetabling algorithm. get_dynamic_timetabling_algorithm_console_messages

Parameters

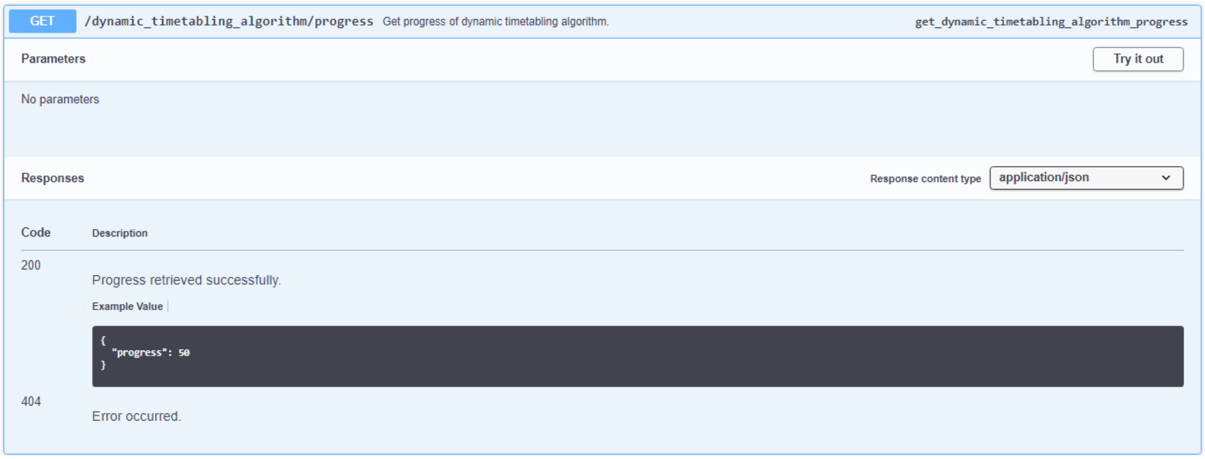
No parameters

Responses

Code	Description
200	Console messages retrieved successfully. Example Value
404	Error occurred.

```
{
  "messages": [
    "Example messages"
  ]
}
```

Figure 5.5.6.1: The route design for getting console messages during dynamic timetable generation.



GET /dynamic_timetabling_algorithm/progress Get progress of dynamic timetabling algorithm. get_dynamic_timetabling_algorithm_progress

Parameters

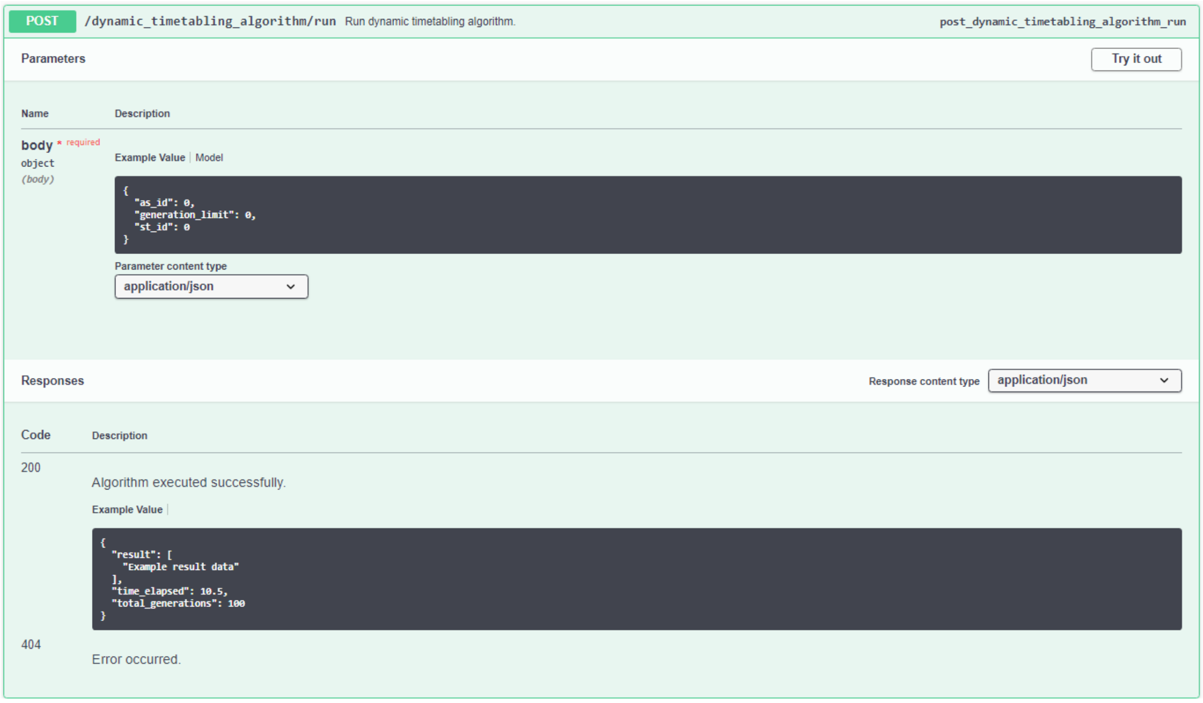
No parameters

Responses

Code	Description
200	Progress retrieved successfully. Example Value
404	Error occurred.

```
{
  "progress": 50
}
```

Figure 5.5.6.2: The route design for getting progress of the dynamic timetable generation.



POST /dynamic_timetabling_algorithm/run Run dynamic timetabling algorithm. post_dynamic_timetabling_algorithm_run

Parameters

Name: body * required — object (body)

Example Value

```
{
  "as_id": 0,
  "generation_limit": 0,
  "st_id": 0
}
```

Parameter content type: application/json

Responses

Response content type: application/json

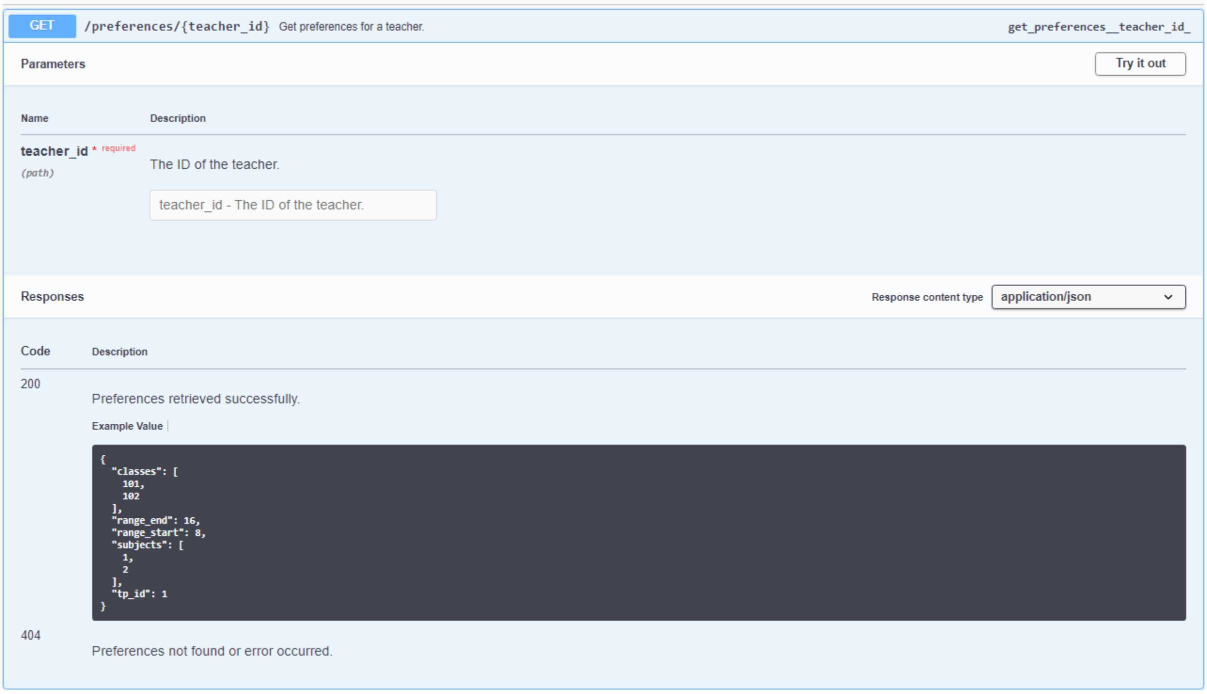
Code	Description
200	Algorithm executed successfully.
404	Error occurred.

Example Value (200)

```
{
  "result": [
    "Example result data"
  ],
  "time_elapsed": 10.5,
  "total_generations": 100
}
```

Figure 5.5.6.3: The route design for running the dynamic timetabling algorithm.

5.5.7 Preferences Tag



/preferences/{teacher_id} Get preferences for a teacher. get_preferences__teacher_id_

Parameters

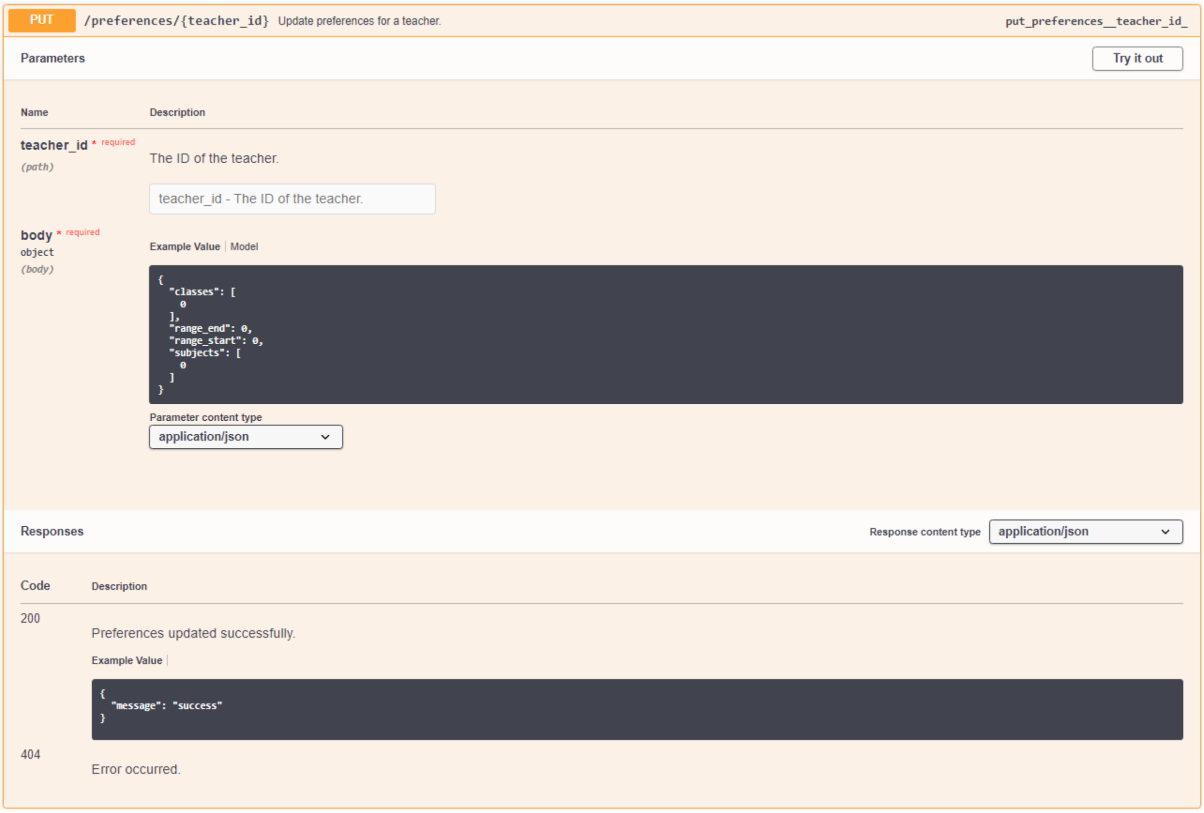
Name	Description
teacher_id * required (path)	The ID of the teacher. teacher_id - The ID of the teacher.

Responses

Code	Description
200	Preferences retrieved successfully. Example Value
404	Preferences not found or error occurred.

```
{
  "classes": [
    101,
    102
  ],
  "range_end": 16,
  "range_start": 8,
  "subjects": [
    1,
    2
  ],
  "tp_id": 1
}
```

Figure 5.5.7.1: The route design for getting a teacher's preferences.



/preferences/{teacher_id} Update preferences for a teacher. put_preferences__teacher_id_

Parameters

Name	Description
teacher_id * required (path)	The ID of the teacher. teacher_id - The ID of the teacher.
body * required object (body)	Example Value | Model

```
{
  "classes": [
    0
  ],
  "range_end": 0,
  "range_start": 0,
  "subjects": [
    0
  ]
}
```

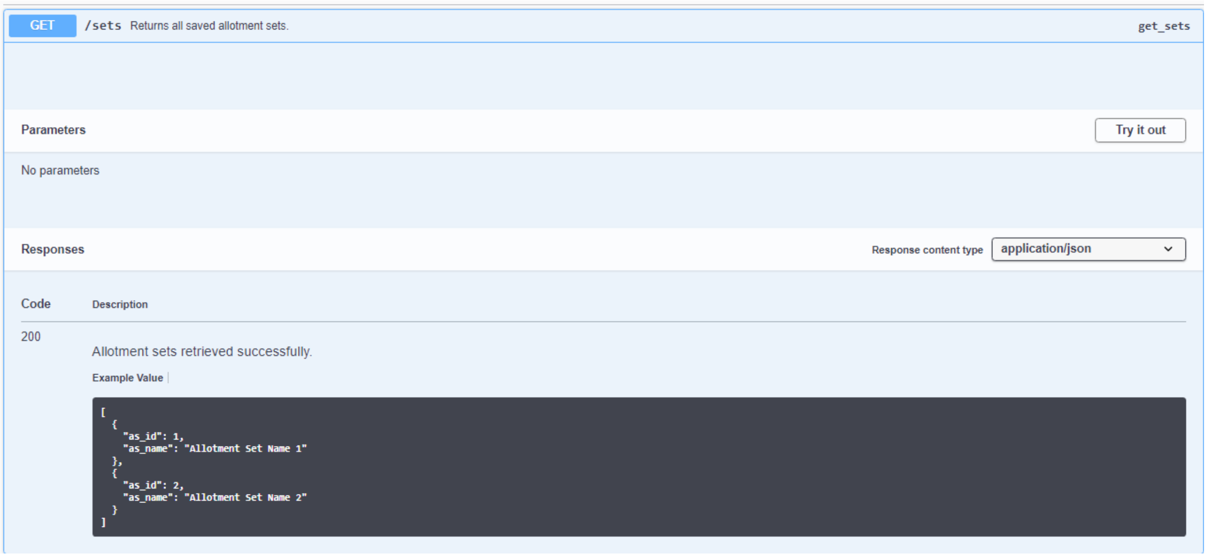
Responses

Code	Description
200	Preferences updated successfully. Example Value
404	Error occurred.

```
{
  "message": "success"
}
```

Figure 5.5.7.2: The route design for updating a teacher's preferences.

5.5.8 Allotment Sets Tag



GET /sets Returns all saved allotment sets. get_sets

Parameters

No parameters

Responses

Response content type: application/json

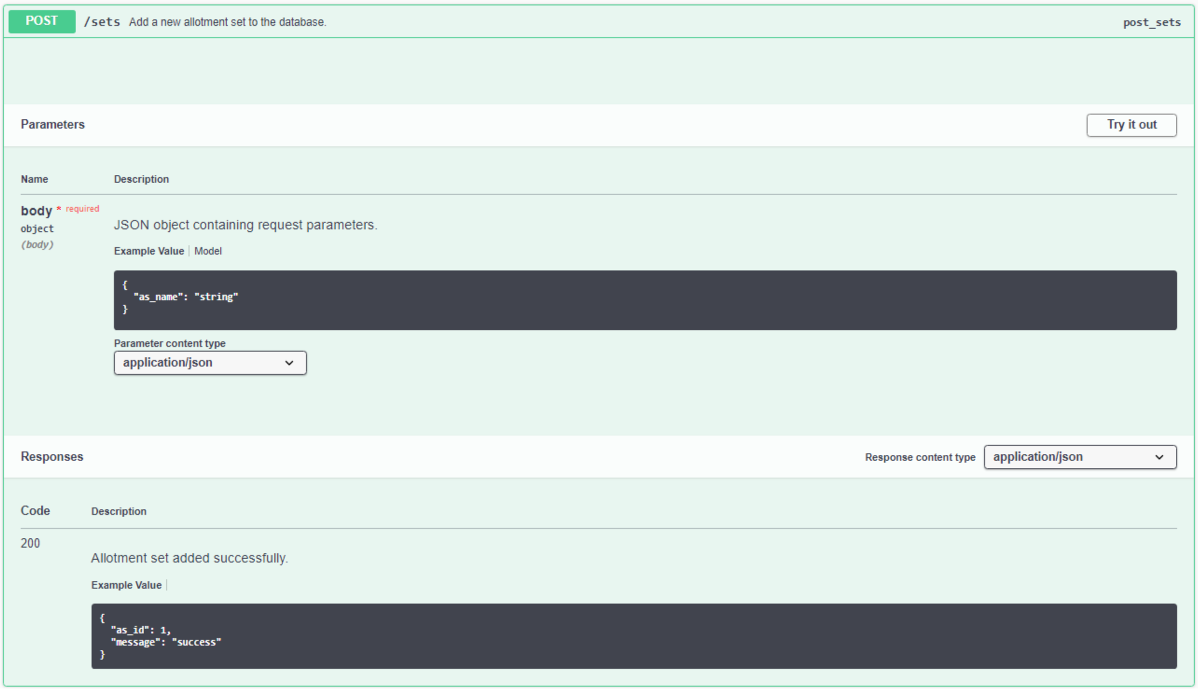
Code | Description

200 Allotment sets retrieved successfully.

Example Value

```
[
  {
    "as_id": 1,
    "as_name": "Allotment Set Name 1"
  },
  {
    "as_id": 2,
    "as_name": "Allotment Set Name 2"
  }
]
```

Figure 5.5.8.1: The route design for getting all allotment sets.



POST /sets Add a new allotment set to the database. post_sets

Parameters

Name | Description

body * required object (body) JSON object containing request parameters.

```
{
  "as_name": "string"
}
```

Parameter content type: application/json

Responses

Response content type: application/json

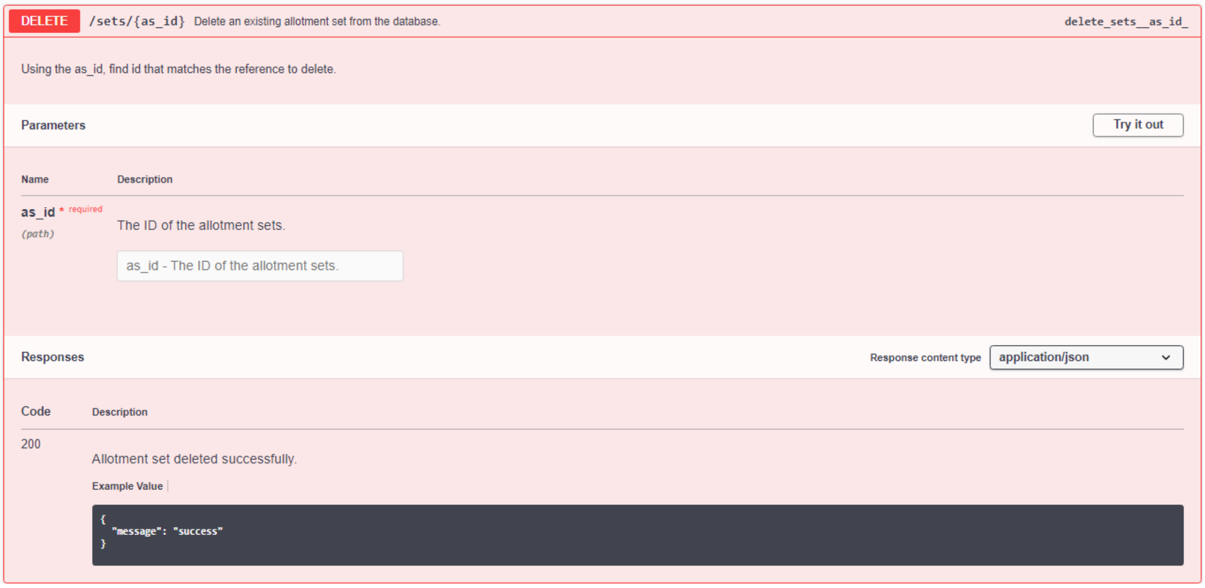
Code | Description

200 Allotment set added successfully.

Example Value

```
{
  "as_id": 1,
  "message": "success"
}
```

Figure 5.5.8.2: The route design for creating new allotment sets.



DELETE /sets/{as_id} Delete an existing allotment set from the database. delete_sets__as_id_

Using the as_id, find id that matches the reference to delete.

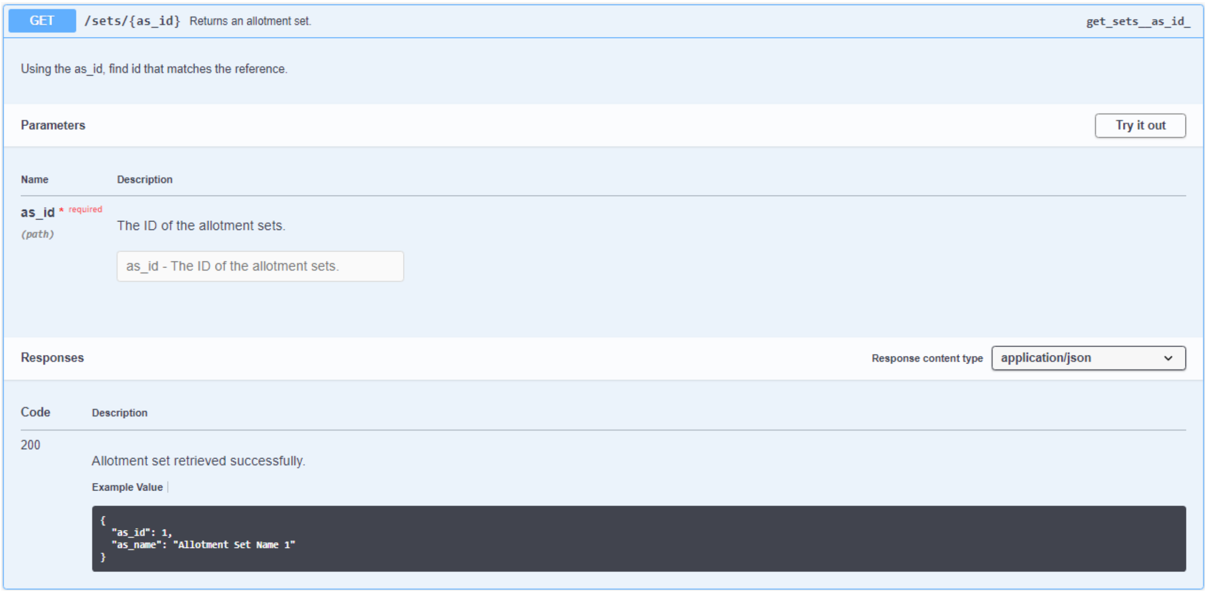
as_id * required (path) The ID of the allotment sets.

as_id - The ID of the allotment sets.

200 Allotment set deleted successfully.

```
{
  "message": "success"
}
```

Figure 5.5.8.3: The route design for deleting an allotment set.



GET /sets/{as_id} Returns an allotment set. get_sets__as_id_

Using the as_id, find id that matches the reference.

as_id * required (path) The ID of the allotment sets.

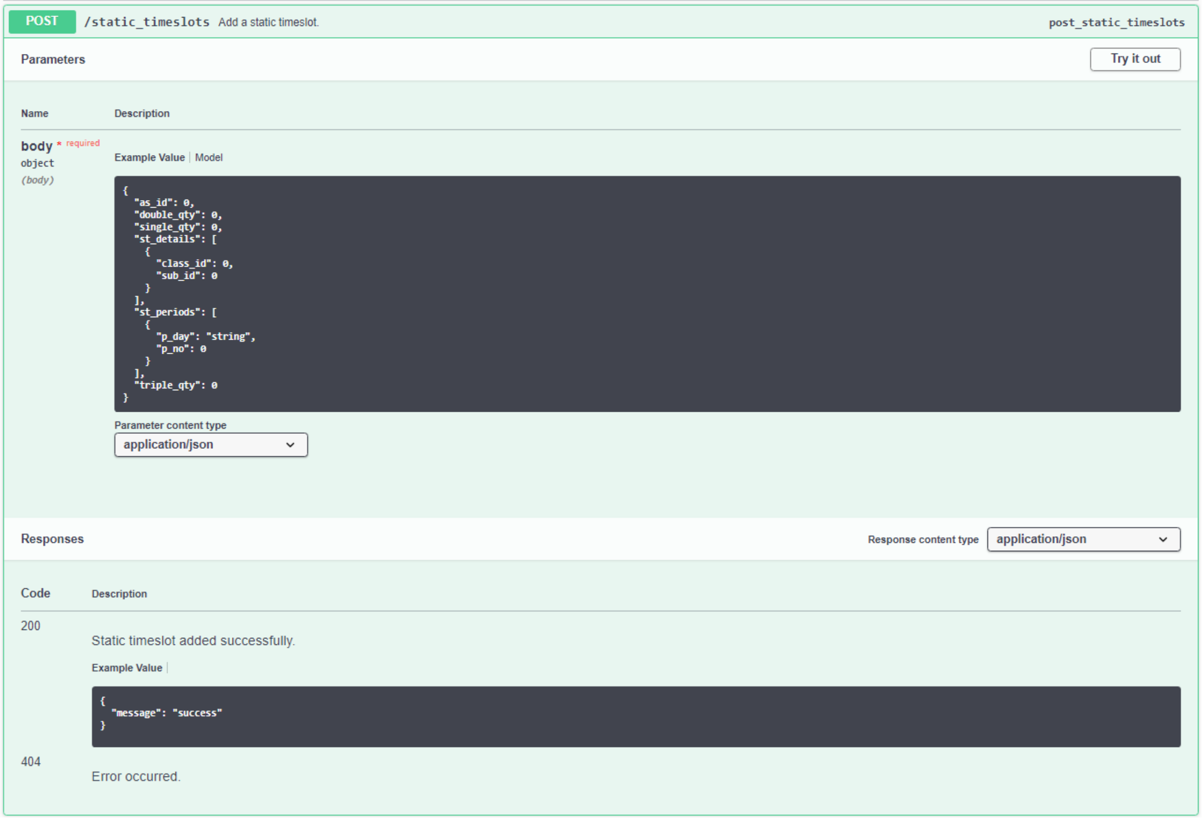
as_id - The ID of the allotment sets.

200 Allotment set retrieved successfully.

```
{
  "as_id": 1,
  "as_name": "Allotment Set Name 1"
}
```

Figure 5.5.8.4: The route design for getting an allotment set.

5.5.9 Static Timeslots Tag



POST /static_timeslots Add a static timeslot. post_static_timeslots

body * required
object
(body)

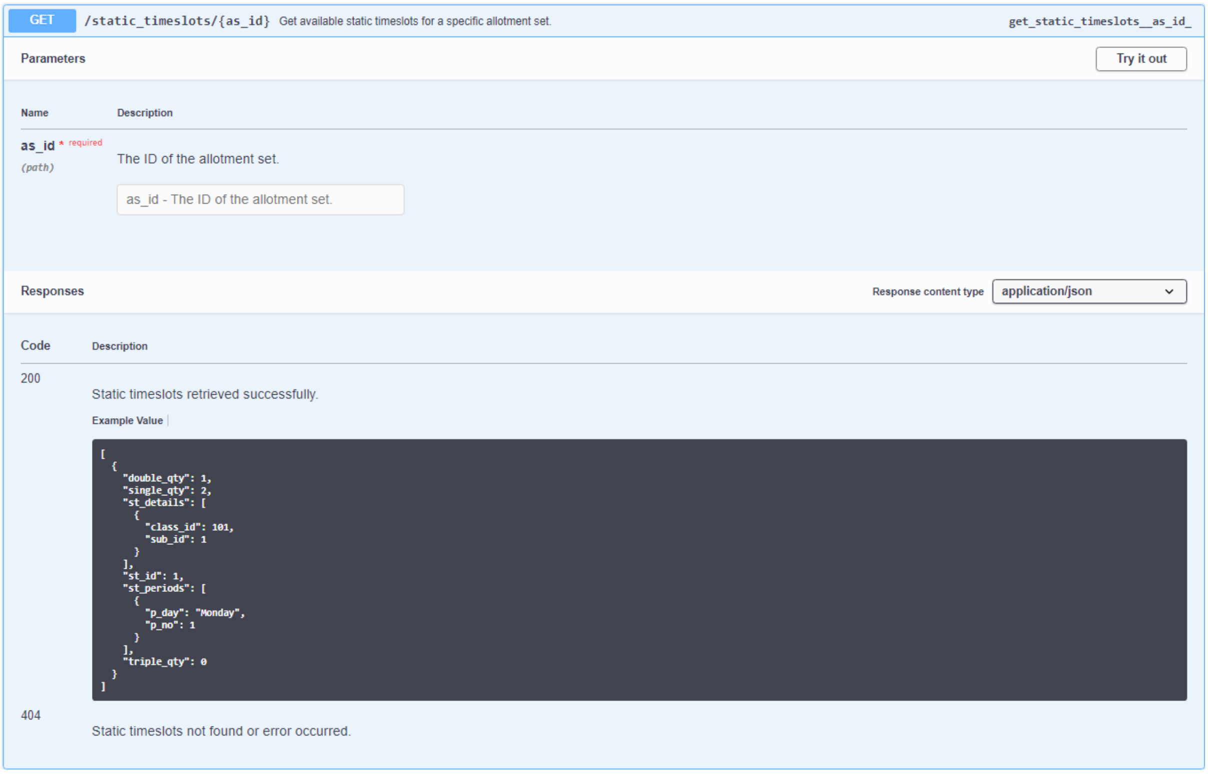
```
{
  "as_id": 0,
  "double_qty": 0,
  "single_qty": 0,
  "st_details": [
    {
      "class_id": 0,
      "sub_id": 0
    }
  ],
  "st_periods": [
    {
      "p_day": "string",
      "p_no": 0
    }
  ],
  "triple_qty": 0
}
```

200 Static timeslot added successfully.

```
{
  "message": "success"
}
```

404 Error occurred.

Figure 5.5.9.1: The route design for creating a new static timeslot.



GET /static_timeslots/{as_id} Get available static timeslots for a specific allotment set. get_static_timeslots__as_id_

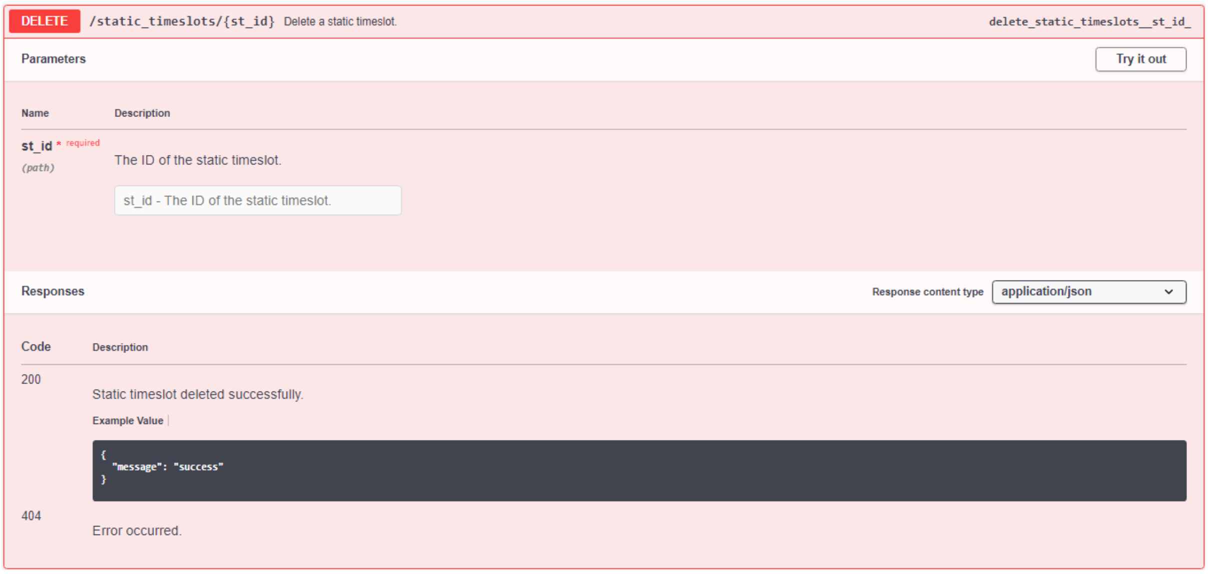
as_id * required
(path)
The ID of the allotment set.
as_id - The ID of the allotment set.

200 Static timeslots retrieved successfully.

```
[
  {
    "double_qty": 1,
    "single_qty": 2,
    "st_details": [
      {
        "class_id": 101,
        "sub_id": 1
      }
    ],
    "st_id": 1,
    "st_periods": [
      {
        "p_day": "Monday",
        "p_no": 1
      }
    ],
    "triple_qty": 0
  }
]
```

404 Static timeslots not found or error occurred.

Figure 5.5.9.2: The route design for getting a static timeslot.



DELETE /static_timeslots/{st_id} Delete a static timeslot. delete_static_timeslots__st_id_

Parameters

Name	Description
st_id * required (path)	The ID of the static timeslot. st_id - The ID of the static timeslot.

Responses

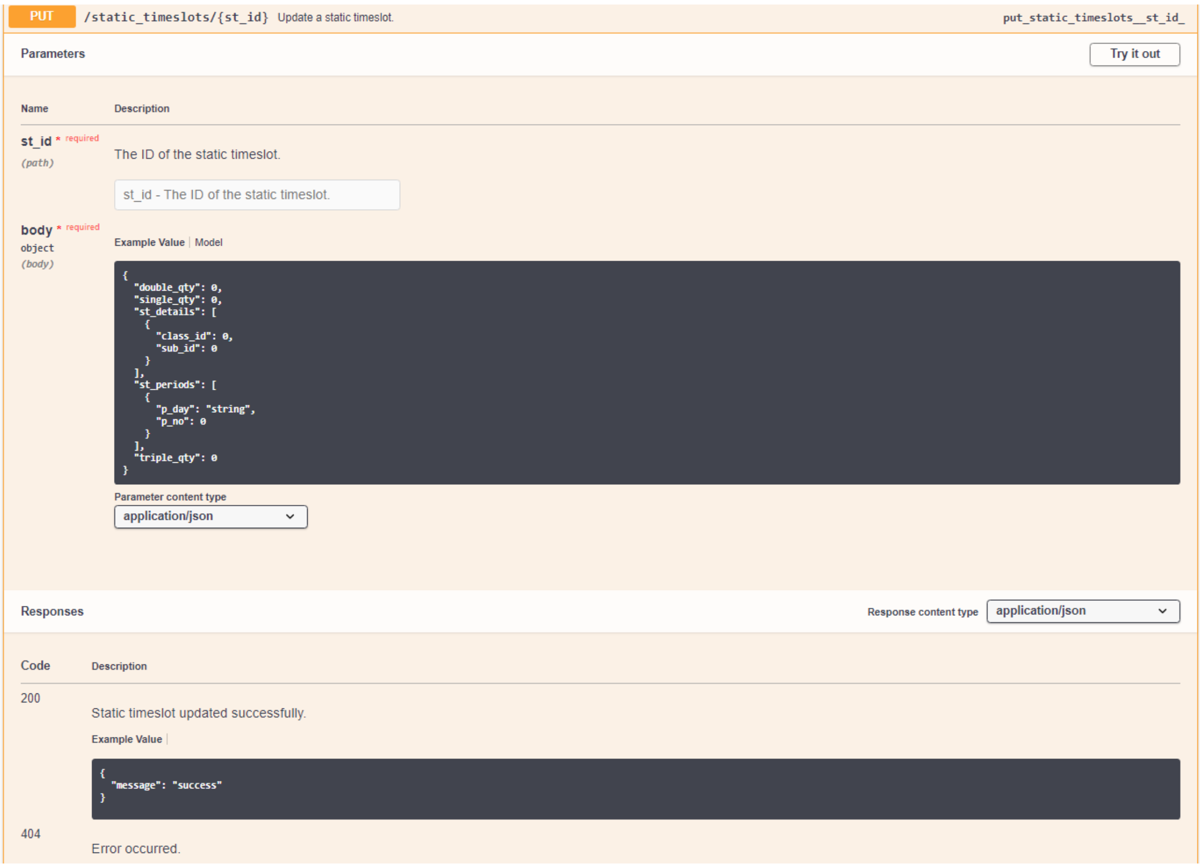
Response content type: application/json

Code	Description
200	Static timeslot deleted successfully.
404	Error occurred.

Example Value

```
{
  "message": "success"
}
```

Figure 5.5.9.3: The route design for deleting a static timeslot.



PUT /static_timeslots/{st_id} Update a static timeslot. put_static_timeslots__st_id_

Parameters

Name	Description
st_id * required (path)	The ID of the static timeslot. st_id - The ID of the static timeslot.
body * required object (body)	Example Value | Model

```
{
  "double_qty": 0,
  "single_qty": 0,
  "st_details": [
    {
      "class_id": 0,
      "sub_id": 0
    }
  ],
  "st_periods": [
    {
      "p_day": "string",
      "p_no": 0
    }
  ],
  "triple_qty": 0
}
```

Parameter content type: application/json

Responses

Response content type: application/json

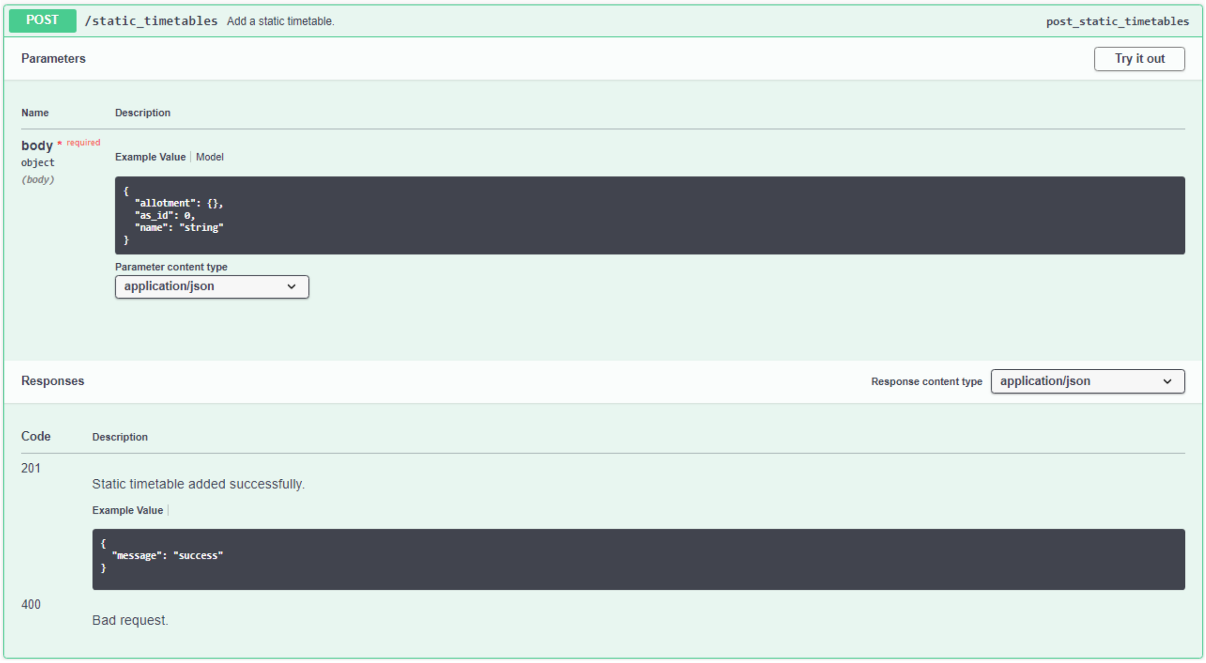
Code	Description
200	Static timeslot updated successfully.
404	Error occurred.

Example Value

```
{
  "message": "success"
}
```

Figure 5.5.9.4: The route design for updating a static timeslot.

5.5.10 Static Timetables Tag



POST /static_timetables Add a static timetable. post_static_timetables

Parameters

Name	Description
body * required object (body)	Example Value | Model

```
{
  "allotment": {},
  "as_id": 0,
  "name": "string"
}
```

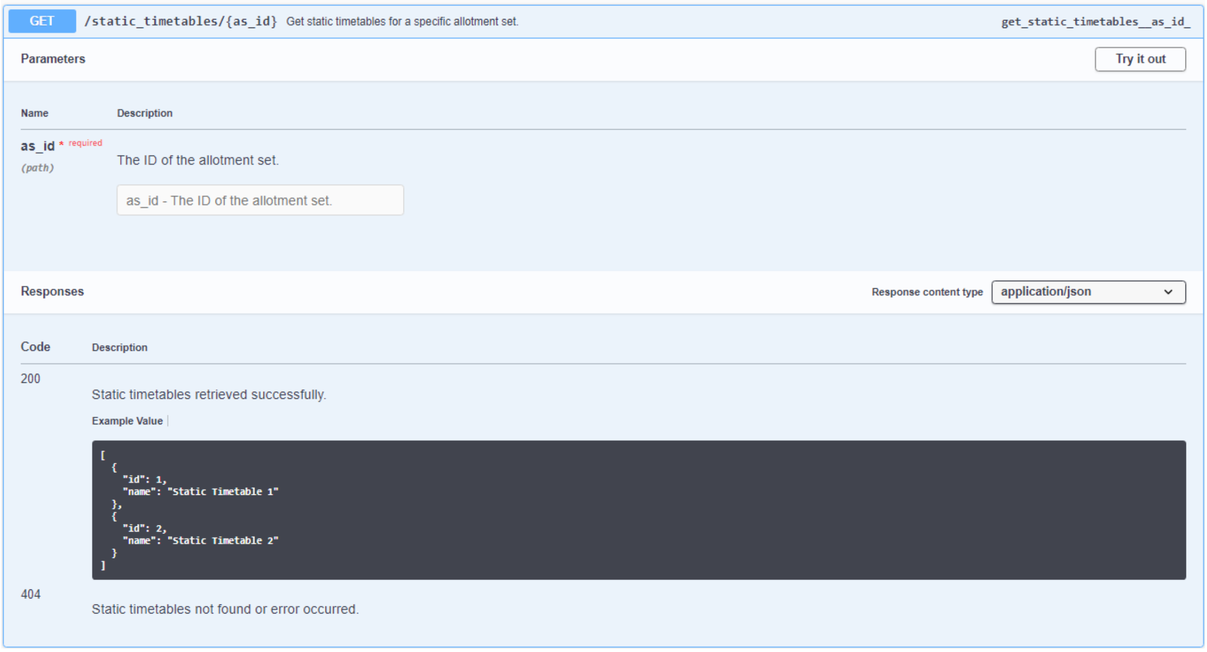
Responses

Code	Description
201	Static timetable added successfully.
400	Bad request.

Example Value (201)

```
{
  "message": "success"
}
```

Figure 5.5.10.1: The route design for creating a new static timetable.



GET /static_timetables/{as_id} Get static timetables for a specific allotment set. get_static_timetables__as_id_

Parameters

Name	Description
as_id * required (path)	The ID of the allotment set. as_id - The ID of the allotment set.

Responses

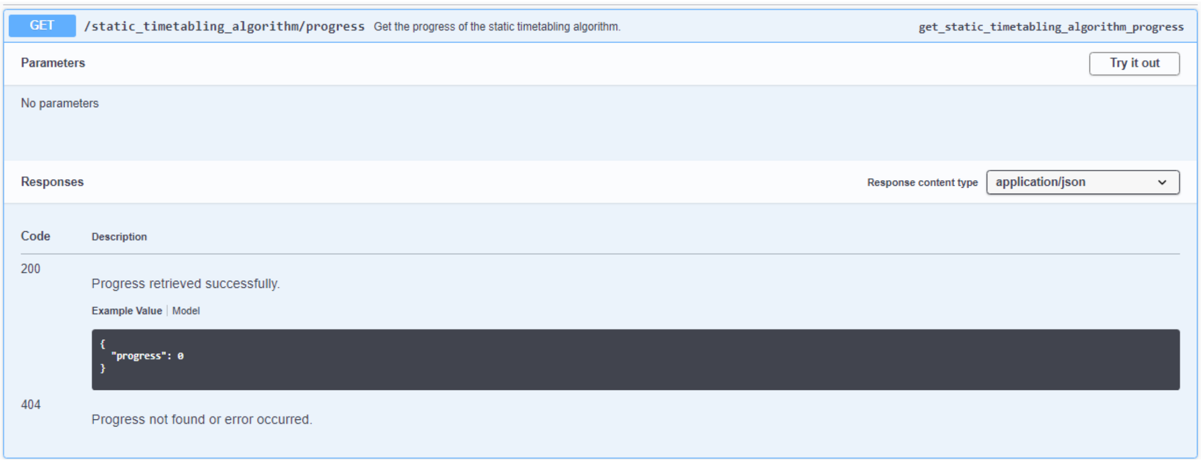
Code	Description
200	Static timetables retrieved successfully.
404	Static timetables not found or error occurred.

Example Value (200)

```
[
  {
    "id": 1,
    "name": "Static Timetable 1"
  },
  {
    "id": 2,
    "name": "Static Timetable 2"
  }
]
```

Figure 5.5.10.2: The route design for getting a static timetable.

5.5.11 Static Timetabling Algorithm



GET /static_timetabling_algorithm/progress Get the progress of the static timetabling algorithm. get_static_timetabling_algorithm_progress

Parameters

No parameters

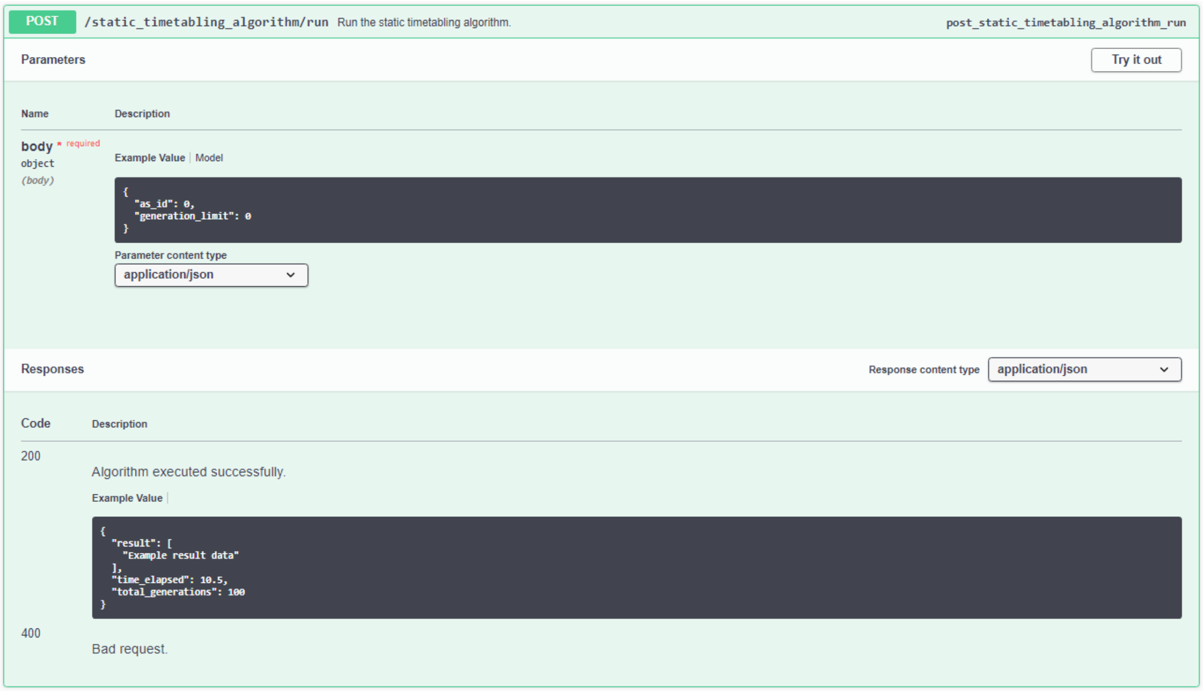
Responses

Response content type: application/json

Code	Description
200	Progress retrieved successfully. Example Value | Model
404	Progress not found or error occurred.

```
{
  "progress": 0
}
```

Figure 5.5.11.1: The route design for getting process of the static timetabling algorithm.



POST /static_timetabling_algorithm/run Run the static timetabling algorithm. post_static_timetabling_algorithm_run

Parameters

Name	Description
body * required object *(body)*	Example Value | Model

```
{
  "as_id": 0,
  "generation_limit": 0
}
```

Parameter content type: application/json

Responses

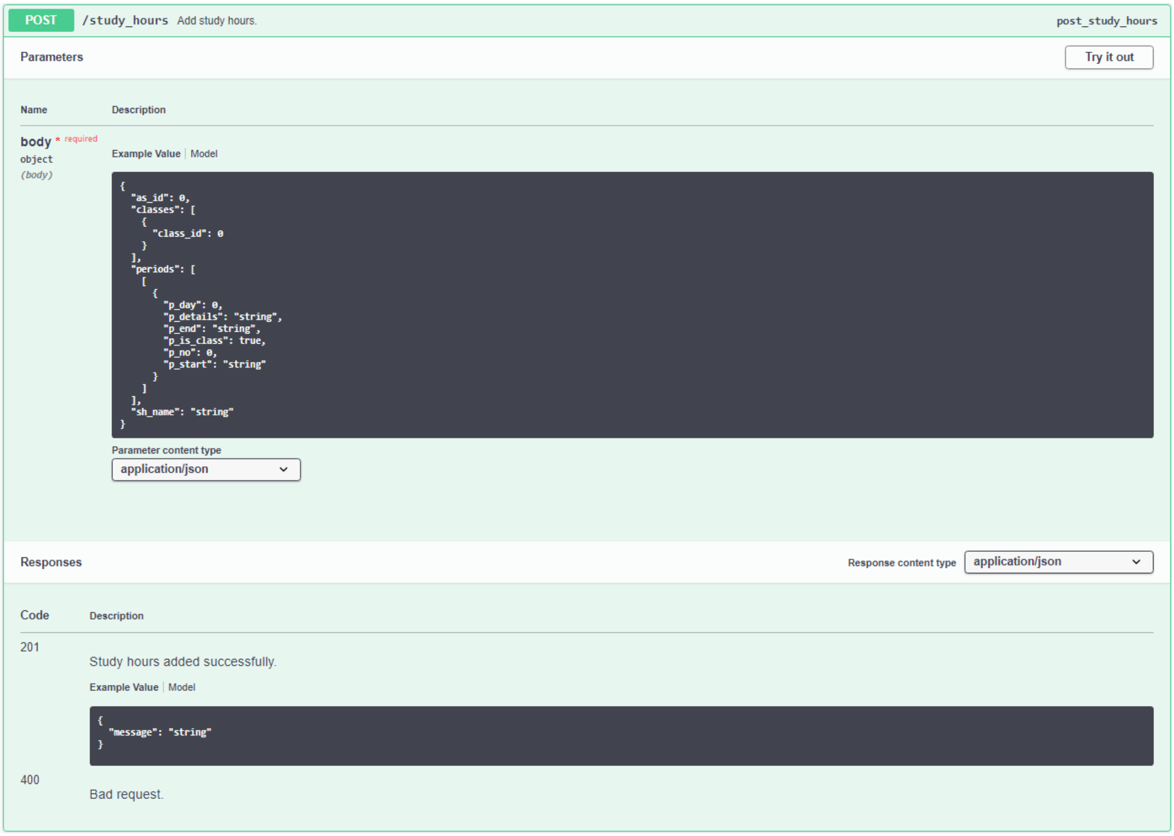
Response content type: application/json

Code	Description
200	Algorithm executed successfully. Example Value
400	Bad request.

```
{
  "result": [
    "Example result data"
  ],
  "time_elapsed": 10.5,
  "total_generations": 100
}
```

Figure 5.5.11.2: The route design for running the static timetabling algorithm.

5.5.12 Study Hours



/study_hours Add study hours. post_study_hours

Parameters

Name	Description
body * required object (body)	Example Value | Model

```
{
  "as_id": 0,
  "classes": [
    {
      "class_id": 0
    }
  ],
  "periods": [
    [
      {
        "p_day": 0,
        "p_details": "string",
        "p_end": "string",
        "p_is_class": true,
        "p_no": 0,
        "p_start": "string"
      }
    ]
  ],
  "sh_name": "string"
}
```

Parameter content type: application/json

Responses

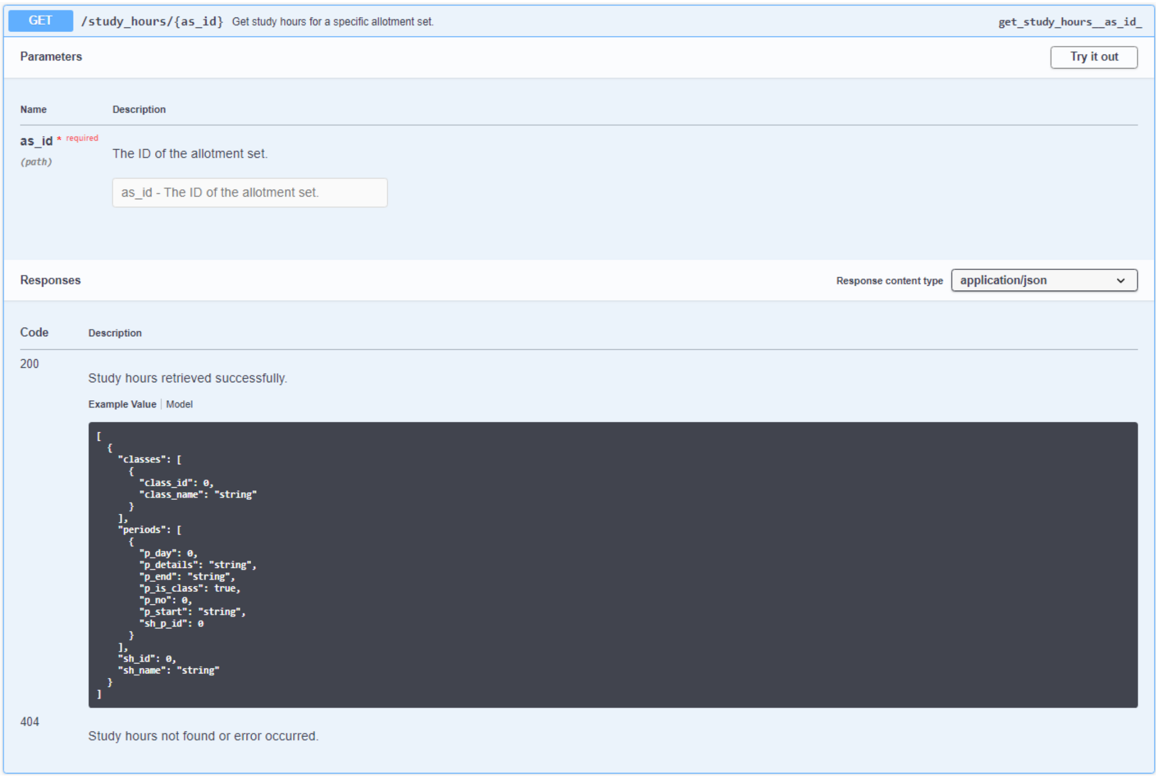
Response content type: application/json

Code	Description
201	Study hours added successfully. Example Value | Model

```
{
  "message": "string"
}
```

Code	Description
400	Bad request.

Figure 5.5.12.1: The route design for creating a new study hour.



/study_hours/{as_id} Get study hours for a specific allotment set. get_study_hours__as_id_

Parameters

Name	Description
as_id * required (path)	The ID of the allotment set. as_id - The ID of the allotment set.

Responses

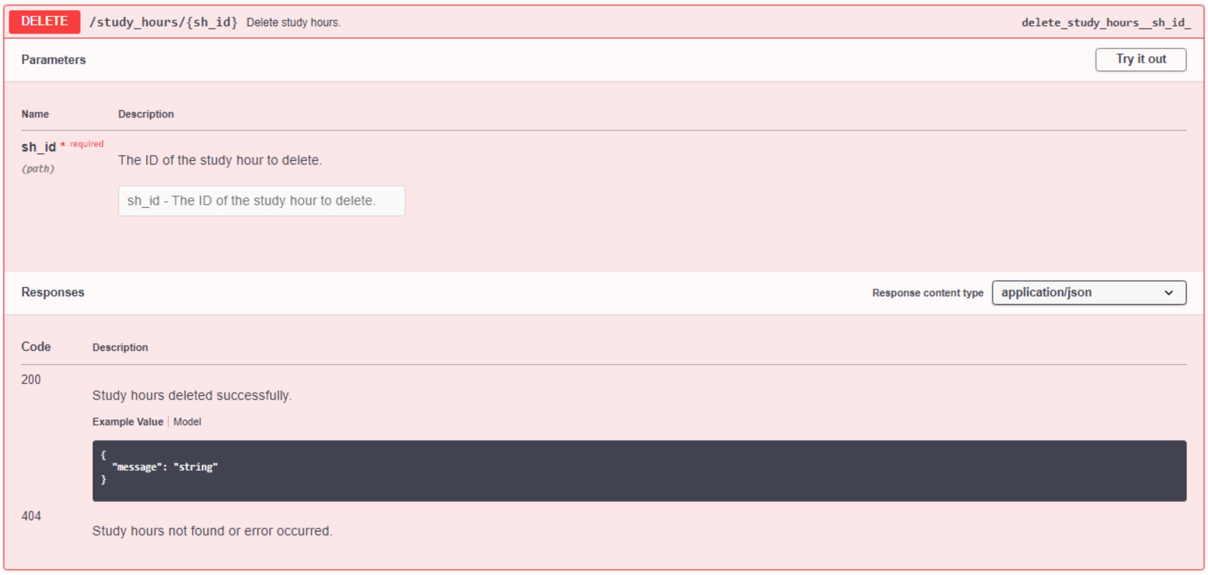
Response content type: application/json

Code	Description
200	Study hours retrieved successfully. Example Value | Model

```
[
  {
    "classes": [
      {
        "class_id": 0,
        "class_name": "string"
      }
    ],
    "periods": [
      {
        "p_day": 0,
        "p_details": "string",
        "p_end": "string",
        "p_is_class": true,
        "p_no": 0,
        "p_start": "string",
        "sh_p_id": 0
      }
    ],
    "sh_id": 0,
    "sh_name": "string"
  }
]
```

Code	Description
404	Study hours not found or error occurred.

Figure 5.5.12.2: The route design for getting a study hour.



DELETE /study_hours/{sh_id} Delete study hours. delete_study_hours__sh_id_

Parameters

Name	Description
sh_id * required (path)	The ID of the study hour to delete. sh_id - The ID of the study hour to delete.

Responses

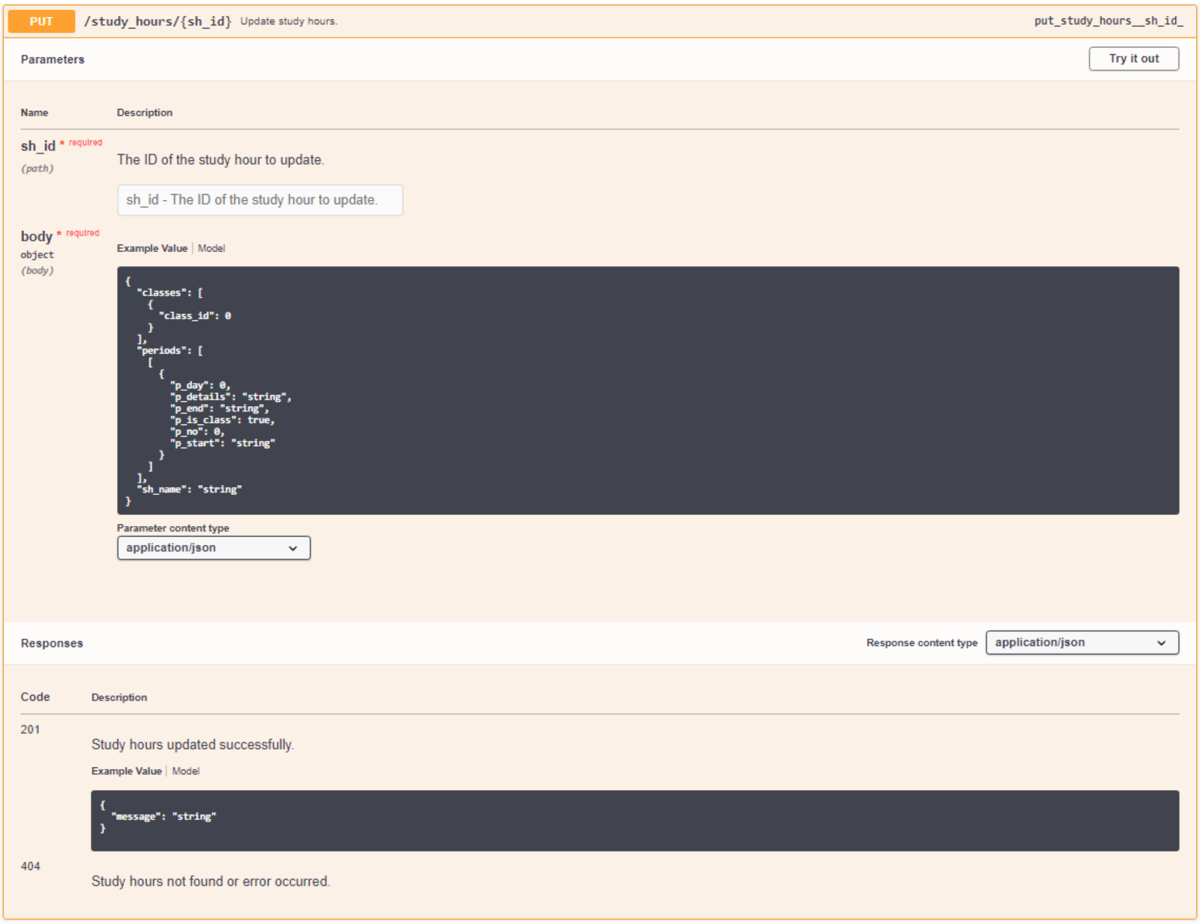
Response content type: application/json

Code	Description
200	Study hours deleted successfully.
404	Study hours not found or error occurred.

Example Value | Model

```
{
  "message": "string"
}
```

Figure 5.5.12.3: The route design for deleting a study hour.



PUT /study_hours/{sh_id} Update study hours. put_study_hours__sh_id_

Parameters

Name	Description
sh_id * required (path)	The ID of the study hour to update. sh_id - The ID of the study hour to update.
body * required object (body)	Example Value | Model

```
{
  "classes": [
    {
      "class_id": 0
    }
  ],
  "periods": [
    [
      {
        "p_day": 0,
        "p_details": "string",
        "p_end": "string",
        "p_is_class": true,
        "p_no": 0,
        "p_start": "string"
      }
    ]
  ],
  "sh_name": "string"
}
```

Parameter content type: application/json

Responses

Response content type: application/json

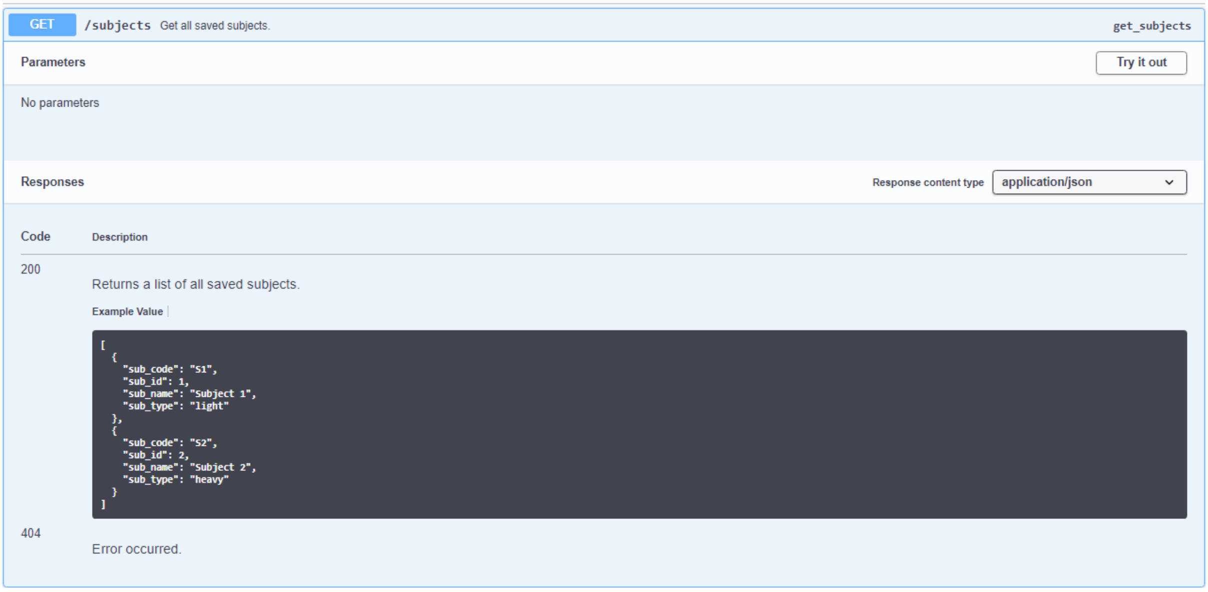
Code	Description
201	Study hours updated successfully.
404	Study hours not found or error occurred.

Example Value | Model

```
{
  "message": "string"
}
```

Figure 5.5.12.4: The route design for updating a study hour.

5.5.13 Subjects Tag



/subjects Get all saved subjects.

Parameters

No parameters

Responses

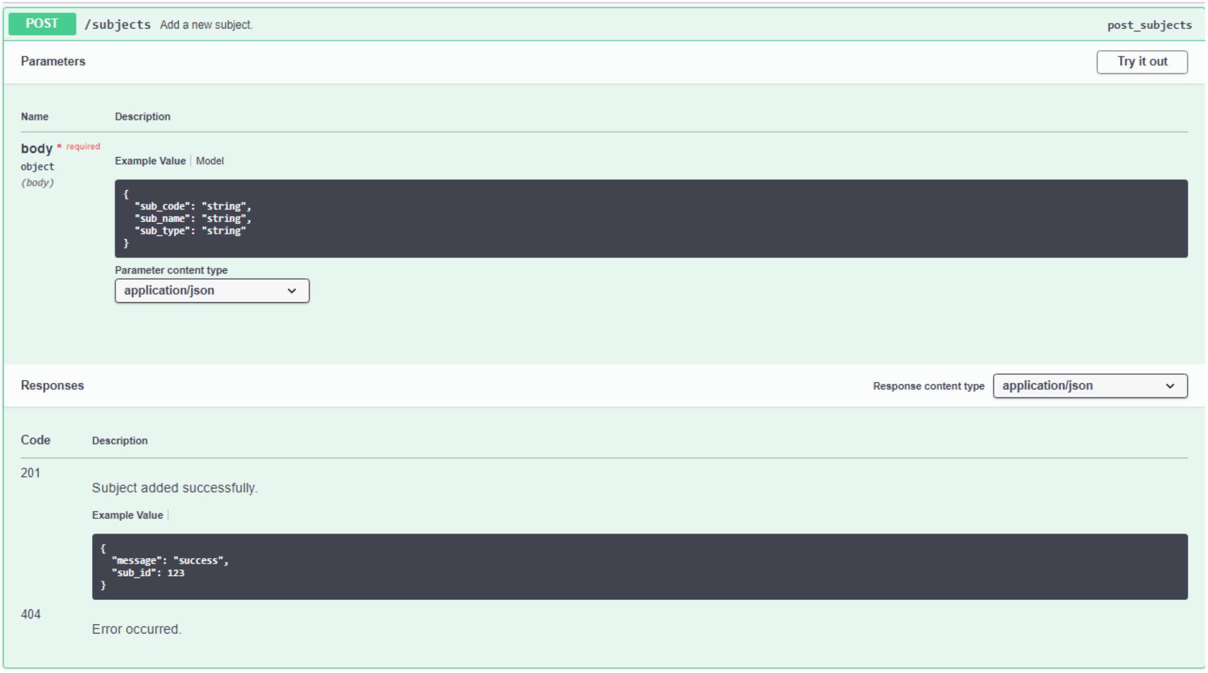
Response content type: application/json

Code	Description
200	Returns a list of all saved subjects.
404	Error occurred.

Example Value

```
[
  {
    "sub_code": "S1",
    "sub_id": 1,
    "sub_name": "Subject 1",
    "sub_type": "light"
  },
  {
    "sub_code": "S2",
    "sub_id": 2,
    "sub_name": "Subject 2",
    "sub_type": "heavy"
  }
]
```

Figure 5.5.13.1: The route design for getting all subjects.



/subjects Add a new subject.

Parameters

Name	Description
body * required object (body)	Example Value | Model

```
{
  "sub_code": "string",
  "sub_name": "string",
  "sub_type": "string"
}
```

Parameter content type: application/json

Responses

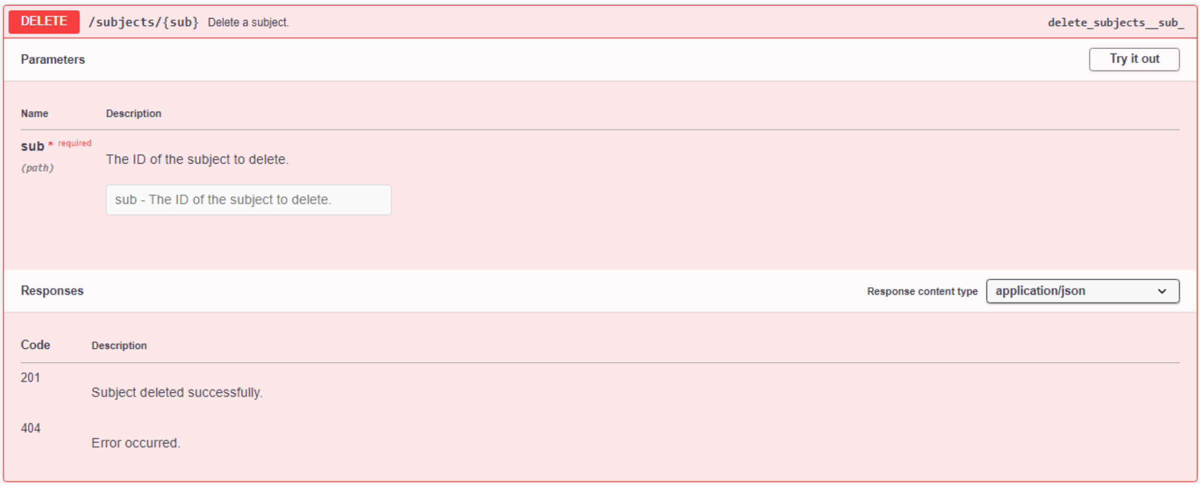
Response content type: application/json

Code	Description
201	Subject added successfully.
404	Error occurred.

Example Value

```
{
  "message": "success",
  "sub_id": 123
}
```

Figure 5.5.13.2: The route design for creating a new subject.



DELETE /subjects/{sub} Delete a subject. delete_subjects__sub_

Parameters

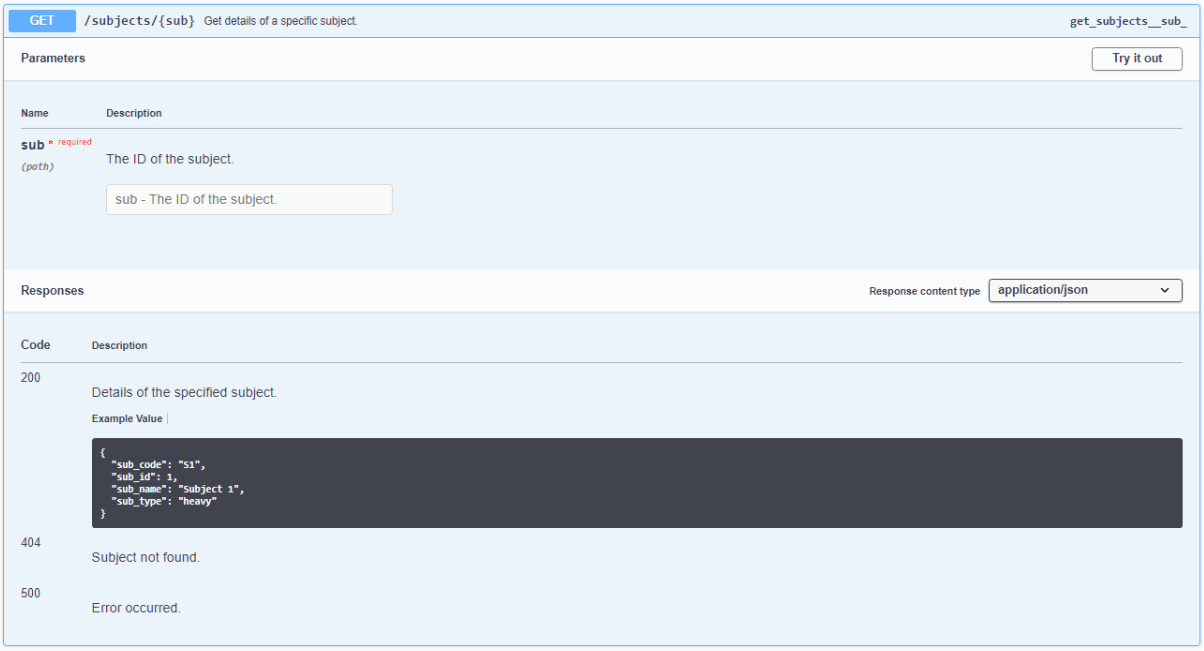
Name	Description
sub * required (path)	The ID of the subject to delete. sub - The ID of the subject to delete.

Responses

Response content type: application/json

Code	Description
201	Subject deleted successfully.
404	Error occurred.

Figure 5.5.13.3: The route design for deleting a subject.



GET /subjects/{sub} Get details of a specific subject. get_subjects__sub_

Parameters

Name	Description
sub * required (path)	The ID of the subject. sub - The ID of the subject.

Responses

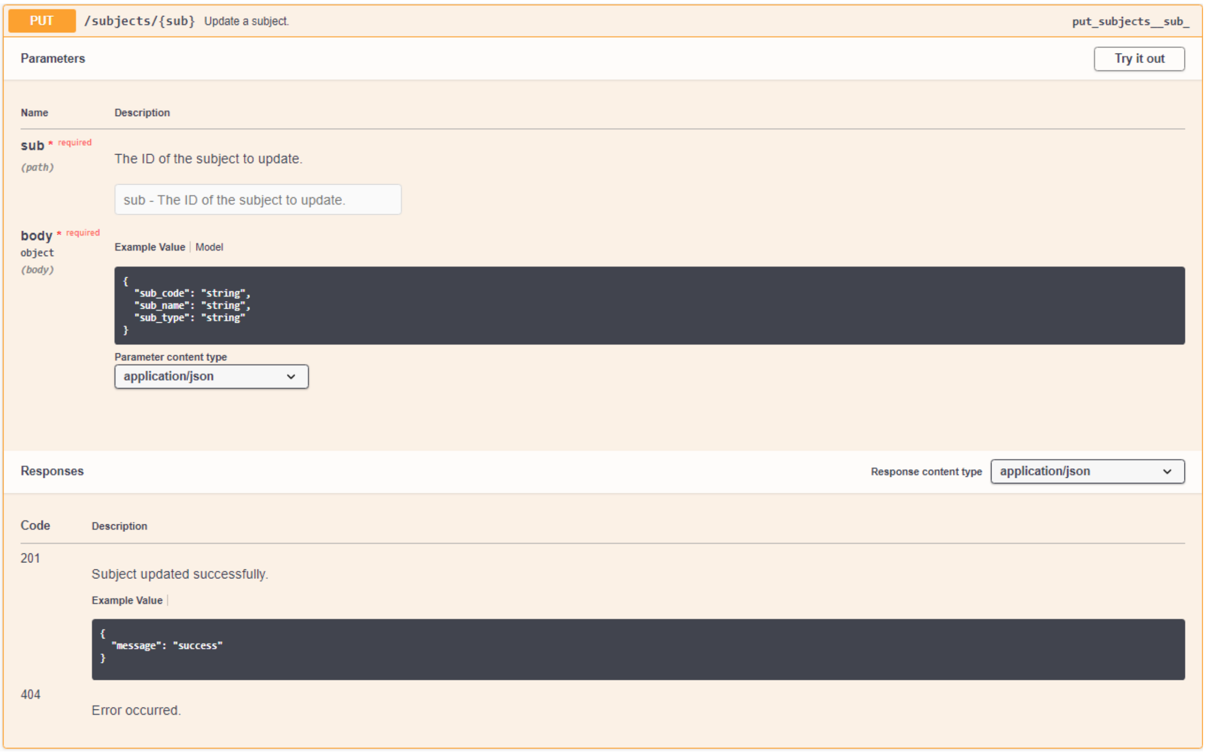
Response content type: application/json

Code	Description
200	Details of the specified subject.
404	Subject not found.
500	Error occurred.

Example Value (200):

```
{
  "sub_code": "S1",
  "sub_id": 1,
  "sub_name": "Subject 1",
  "sub_type": "heavy"
}
```

Figure 5.5.13.4: The route design for getting a subject detail.



PUT /subjects/{sub} Update a subject.

sub * required (path) The ID of the subject to update.

sub - The ID of the subject to update.

body * required object (body)

```
{
  "sub_code": "string",
  "sub_name": "string",
  "sub_type": "string"
}
```

Parameter content type: application/json

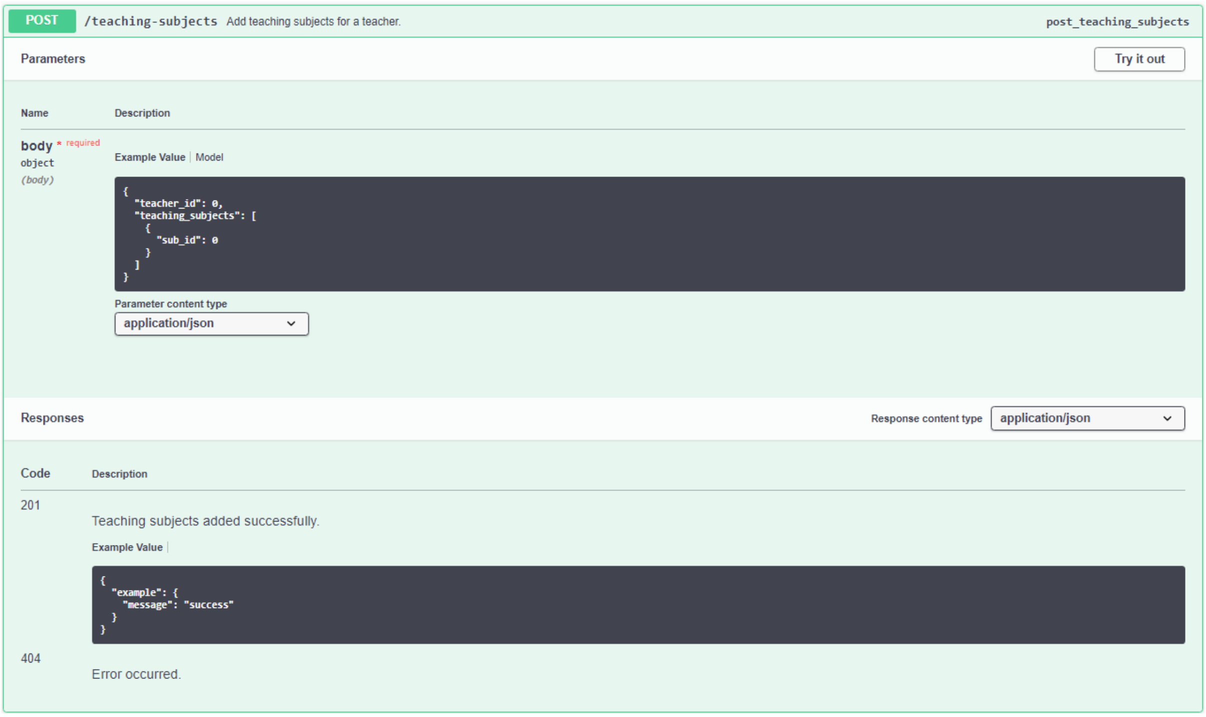
Response content type: application/json

201 Subject updated successfully.

```
{
  "message": "success"
}
```

404 Error occurred.

Figure 5.5.13.5: The route design for updating a subject.



POST /teaching-subjects Add teaching subjects for a teacher.

body * required object (body)

```
{
  "teacher_id": 0,
  "teaching_subjects": [
    {
      "sub_id": 0
    }
  ]
}
```

Parameter content type: application/json

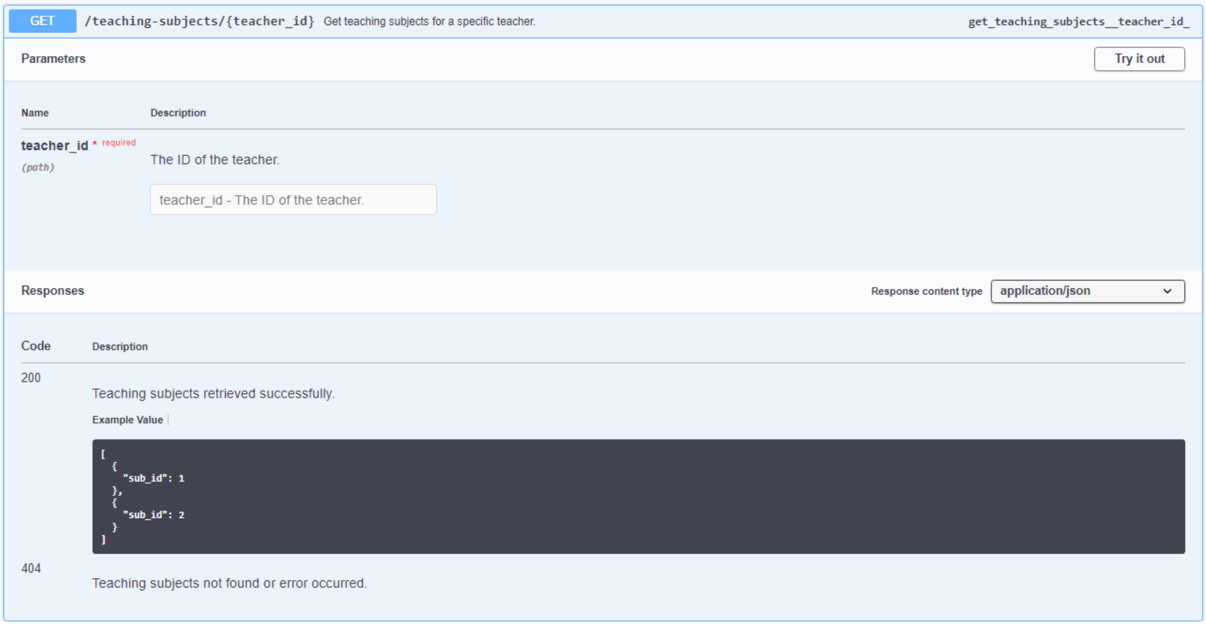
Response content type: application/json

201 Teaching subjects added successfully.

```
{
  "example": {
    "message": "success"
  }
}
```

404 Error occurred.

Figure 5.5.13.6: The route design for adding teaching subjects to a teacher.



GET /teaching-subjects/{teacher_id} Get teaching subjects for a specific teacher. get_teaching_subjects__teacher_id_

teacher_id * required (path) The ID of the teacher.

teacher_id - The ID of the teacher.

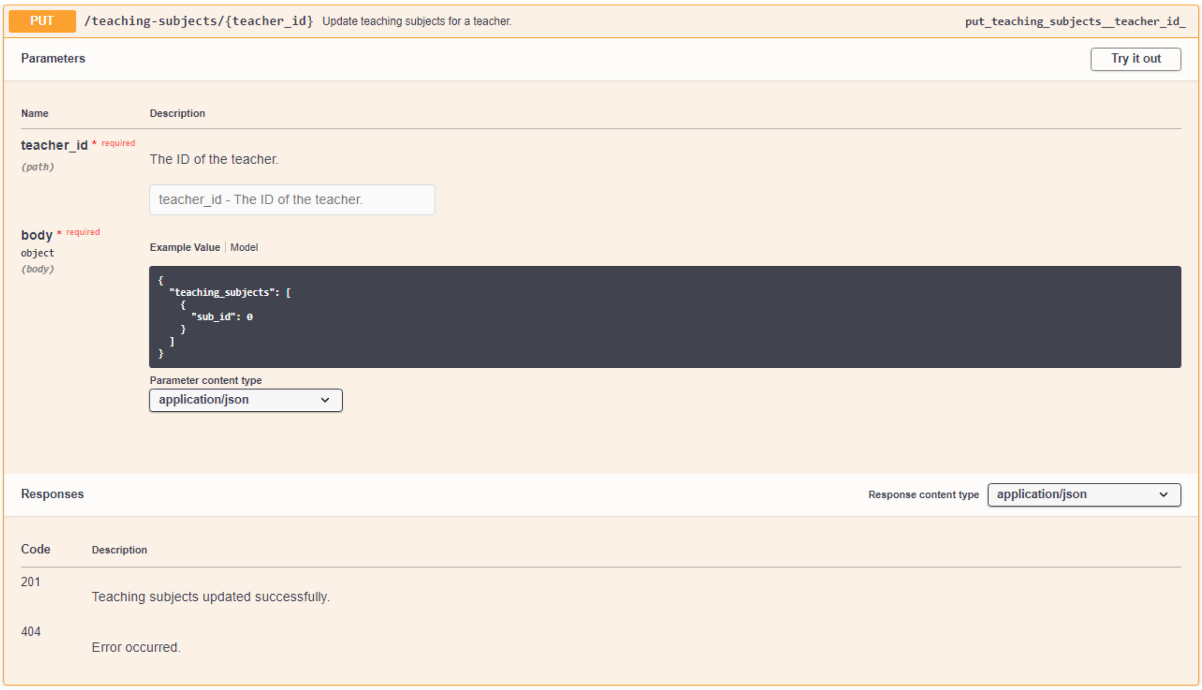
200 Teaching subjects retrieved successfully.

Example Value

```
[
  {
    "sub_id": 1
  },
  {
    "sub_id": 2
  }
]
```

404 Teaching subjects not found or error occurred.

Figure 5.5.13.7: The route design for getting teaching subjects of a teacher.



PUT /teaching-subjects/{teacher_id} Update teaching subjects for a teacher. put_teaching_subjects__teacher_id_

teacher_id * required (path) The ID of the teacher.

teacher_id - The ID of the teacher.

body * required object (body)

Example Value | Model

```
{
  "teaching_subjects": [
    {
      "sub_id": 0
    }
  ]
}
```

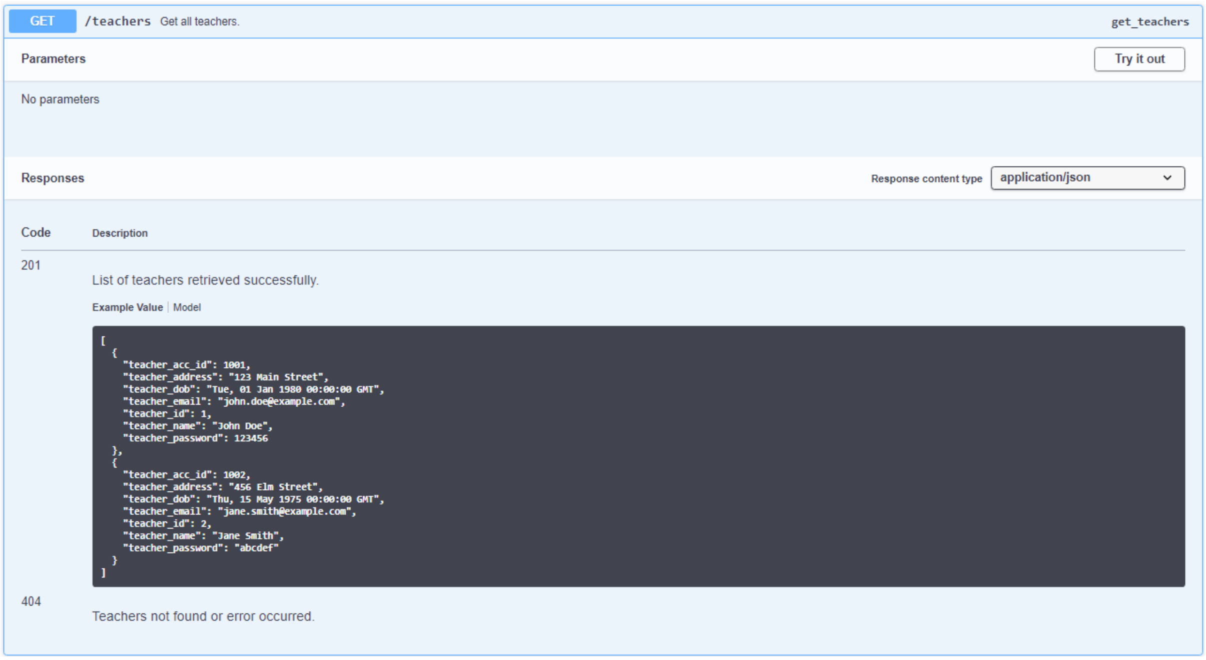
Parameter content type: application/json

201 Teaching subjects updated successfully.

404 Error occurred.

Figure 5.5.13.8: The route design for updating teaching subjects of a teacher.

5.5.14 Teachers Tag



GET /teachers Get all teachers.

No parameters

Response content type: application/json

201

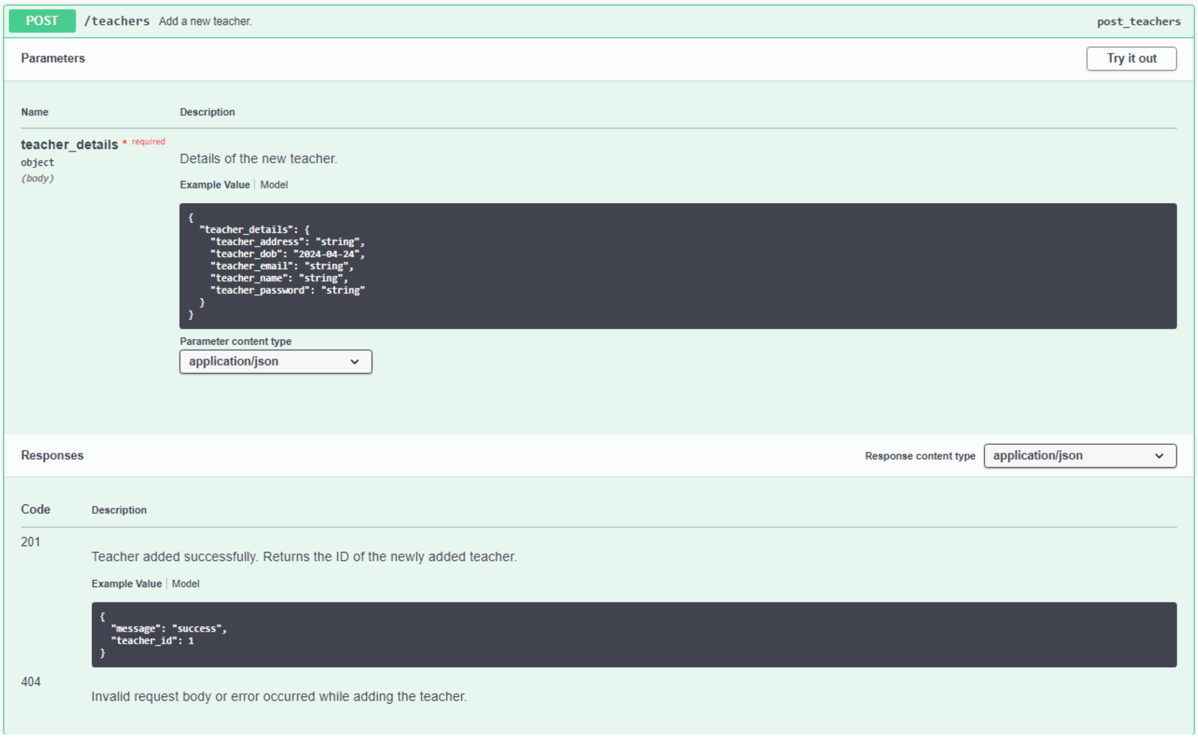
List of teachers retrieved successfully.

```
[
  {
    "teacher_acc_id": 1001,
    "teacher_address": "123 Main Street",
    "teacher_dob": "Tue, 01 Jan 1980 00:00:00 GMT",
    "teacher_email": "john.doe@example.com",
    "teacher_id": 1,
    "teacher_name": "John Doe",
    "teacher_password": 123456
  },
  {
    "teacher_acc_id": 1002,
    "teacher_address": "456 Elm Street",
    "teacher_dob": "Thu, 15 May 1975 00:00:00 GMT",
    "teacher_email": "jane.smith@example.com",
    "teacher_id": 2,
    "teacher_name": "Jane Smith",
    "teacher_password": "abcdef"
  }
]
```

404

Teachers not found or error occurred.

Figure 5.5.14.1: The route design for getting all teachers.



POST /teachers Add a new teacher.

teacher_details * required
object
(body)

Details of the new teacher.

```
{
  "teacher_details": {
    "teacher_address": "string",
    "teacher_dob": "2024-04-24",
    "teacher_email": "string",
    "teacher_name": "string",
    "teacher_password": "string"
  }
}
```

Parameter content type: application/json

Response content type: application/json

201

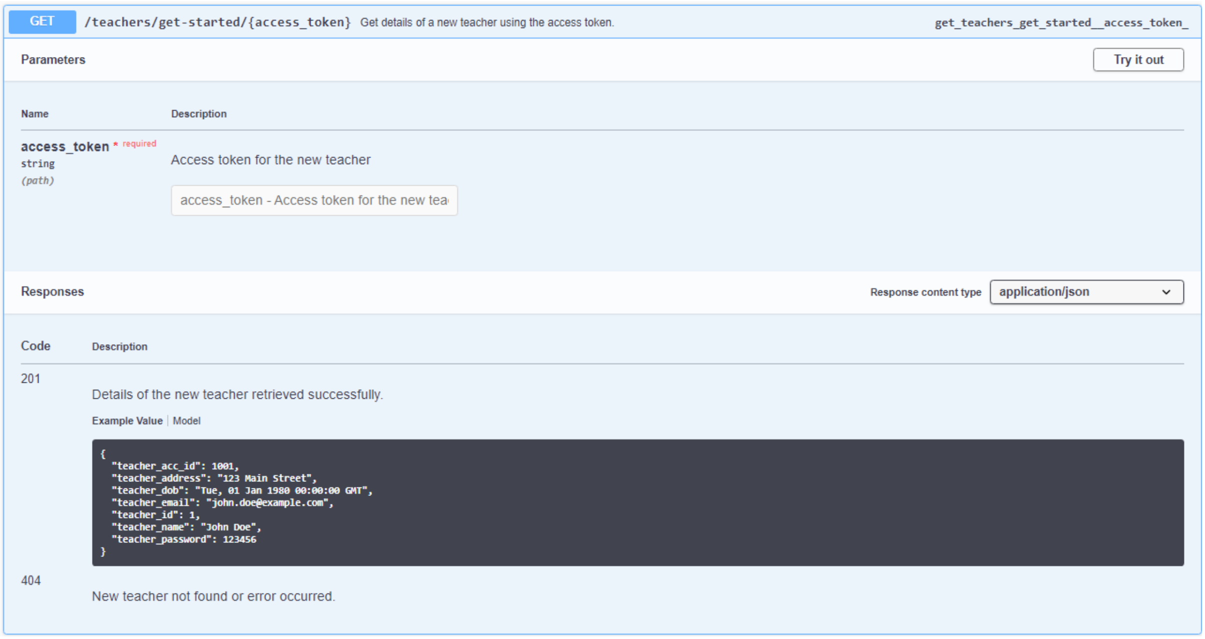
Teacher added successfully. Returns the ID of the newly added teacher.

```
{
  "message": "success",
  "teacher_id": 1
}
```

404

Invalid request body or error occurred while adding the teacher.

Figure 5.5.14.2: The route design for creating a new teacher.



GET `/teachers/get-started/{access_token}` Get details of a new teacher using the access token. `get_teachers_get_started__access_token_`

Parameters

Name	Description
access_token * required string *(path)*	Access token for the new teacher access_token - Access token for the new tea

Responses

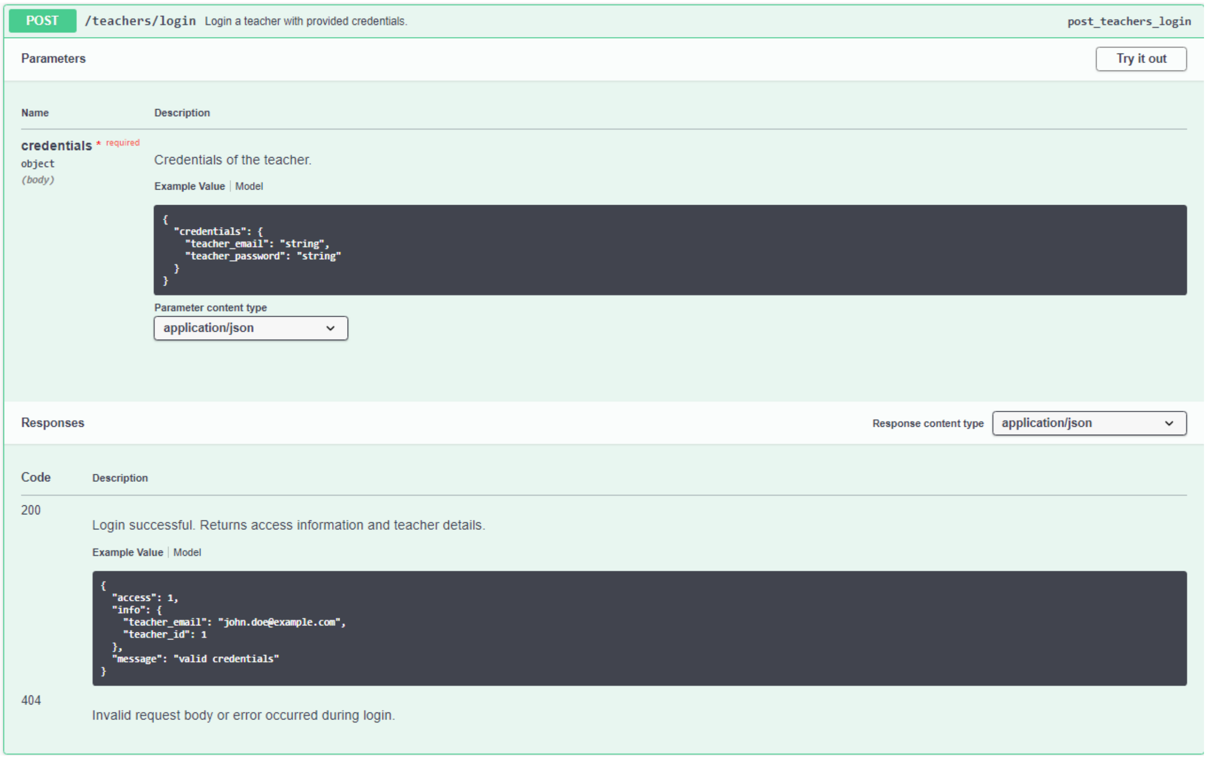
Response content type: application/json

Code	Description
201	Details of the new teacher retrieved successfully.
404	New teacher not found or error occurred.

Example Value | Model

```
{
  "teacher_acc_id": 1001,
  "teacher_address": "123 Main Street",
  "teacher_dob": "Tue, 01 Jan 1980 00:00:00 GMT",
  "teacher_email": "john.doe@example.com",
  "teacher_id": 1,
  "teacher_name": "John Doe",
  "teacher_password": 123456
}
```

Figure 5.5.14.3: The route design for getting teacher details of first-time login teachers.



POST `/teachers/login` Login a teacher with provided credentials. `post_teachers_login`

Parameters

Name	Description
credentials * required object *(body)*	Credentials of the teacher.

Example Value | Model

```
{
  "credentials": {
    "teacher_email": "string",
    "teacher_password": "string"
  }
}
```

Parameter content type: application/json

Responses

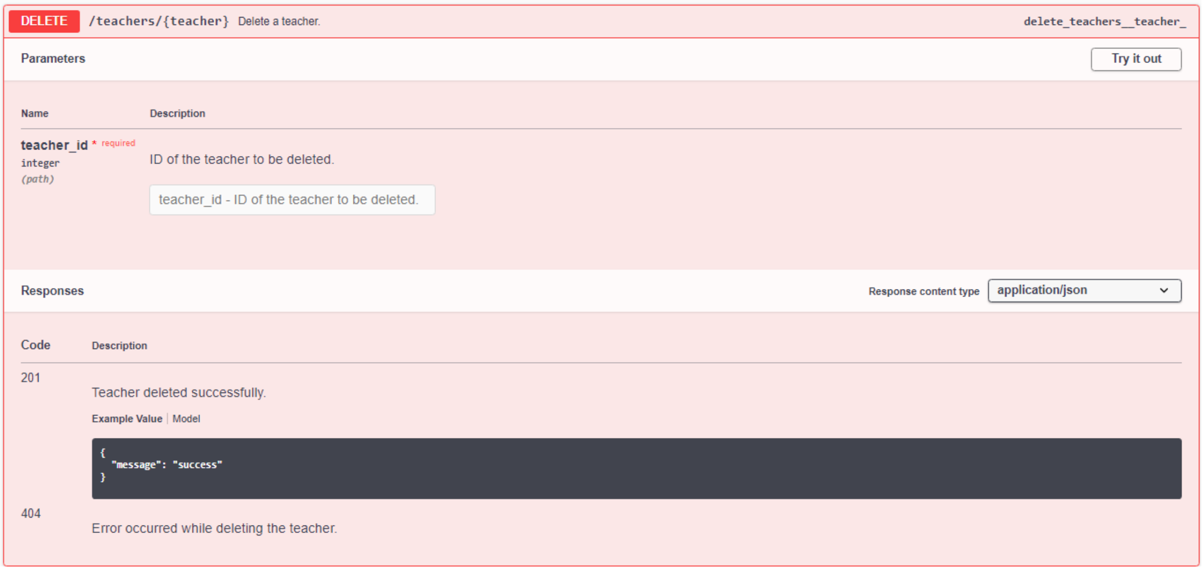
Response content type: application/json

Code	Description
200	Login successful. Returns access information and teacher details.
404	Invalid request body or error occurred during login.

Example Value | Model

```
{
  "access": 1,
  "info": {
    "teacher_email": "john.doe@example.com",
    "teacher_id": 1
  },
  "message": "valid credentials"
}
```

Figure 5.5.14.4: The route design for logging in teachers into STSAMS.



DELETE `/teachers/{teacher}` Delete a teacher. `delete_teachers__teacher_`

Parameters

Name	Description
teacher_id * required integer (path)	ID of the teacher to be deleted. teacher_id - ID of the teacher to be deleted.

Responses

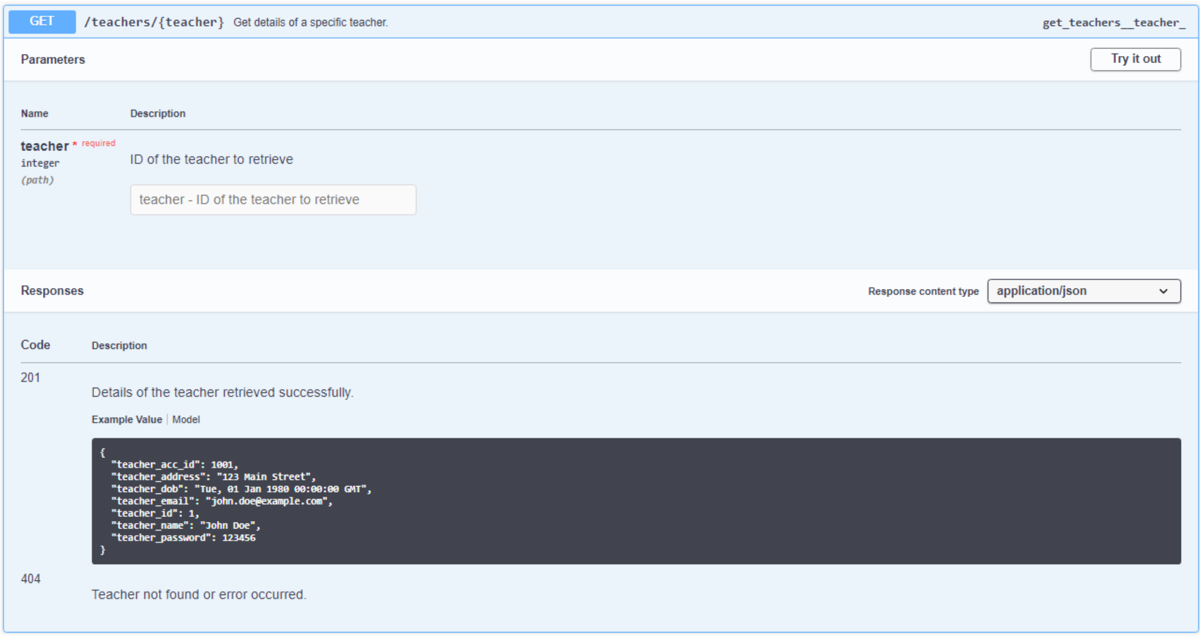
Response content type: application/json

Code	Description
201	Teacher deleted successfully.
404	Error occurred while deleting the teacher.

Example Value | Model

```
{
  "message": "success"
}
```

Figure 5.5.14.5: The route design for deleting a teacher.



GET `/teachers/{teacher}` Get details of a specific teacher. `get_teachers__teacher_`

Parameters

Name	Description
teacher * required integer (path)	ID of the teacher to retrieve teacher - ID of the teacher to retrieve

Responses

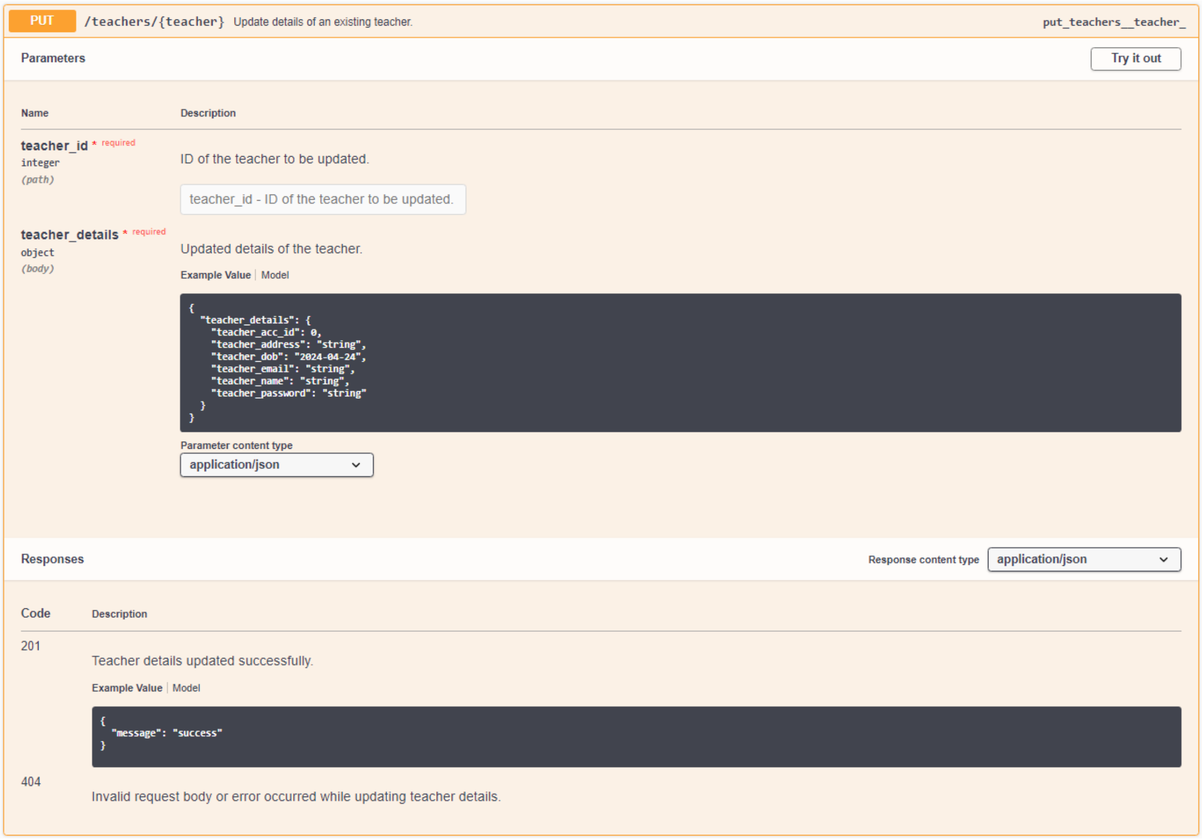
Response content type: application/json

Code	Description
201	Details of the teacher retrieved successfully.
404	Teacher not found or error occurred.

Example Value | Model

```
{
  "teacher_acc_id": 1001,
  "teacher_address": "123 Main Street",
  "teacher_dob": "Tue, 01 Jan 1980 00:00:00 GMT",
  "teacher_email": "john.doe@example.com",
  "teacher_id": 1,
  "teacher_name": "John Doe",
  "teacher_password": 123456
}
```

Figure 5.5.14.6: The route design for getting details of a teacher.



`/teachers/{teacher}` Update details of an existing teacher.

Parameters

Name	Description
teacher_id * required integer (path)	ID of the teacher to be updated. teacher_id - ID of the teacher to be updated.
teacher_details * required object (body)	Updated details of the teacher.

Example Value | Model

```
{
  "teacher_details": {
    "teacher_acc_id": 0,
    "teacher_address": "string",
    "teacher_dob": "2024-04-24",
    "teacher_email": "string",
    "teacher_name": "string",
    "teacher_password": "string"
  }
}
```

Parameter content type: application/json

Responses

Response content type: application/json

Code	Description
201	Teacher details updated successfully.

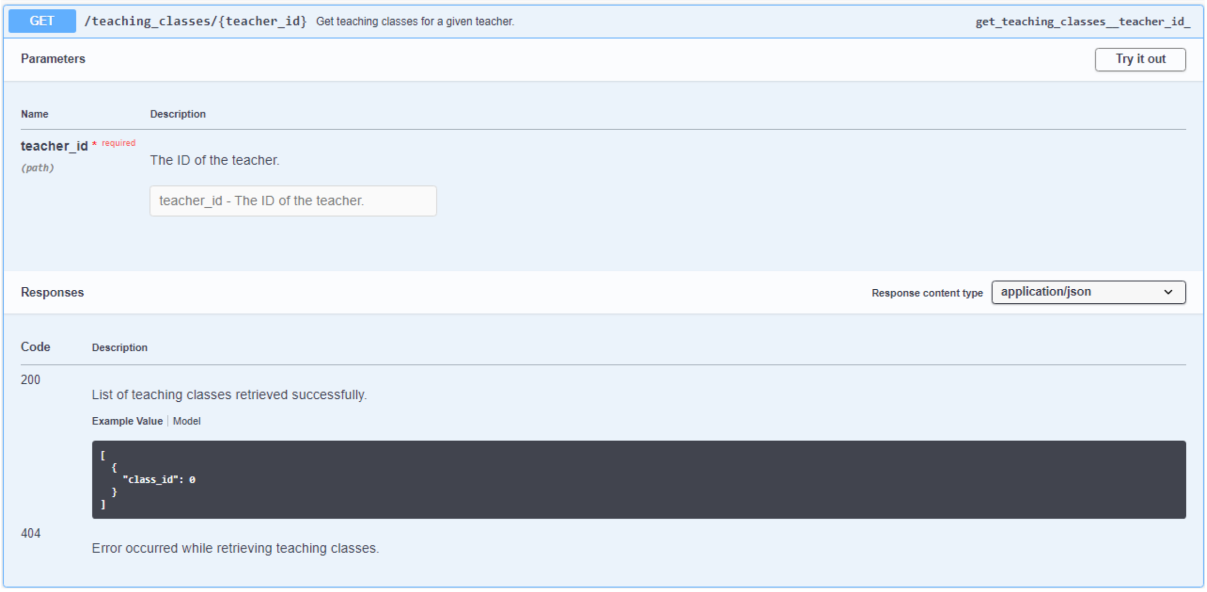
Example Value | Model

```
{
  "message": "success"
}
```

Code	Description
404	Invalid request body or error occurred while updating teacher details.

Figure 5.5.14.7: The route design for updating the details of a teacher.

5.5.15 Teaching Classes Tag



`/teaching_classes/{teacher_id}` Get teaching classes for a given teacher. `get_teaching_classes__teacher_id_`

Parameters

Name	Description
teacher_id * required (path)	The ID of the teacher. teacher_id - The ID of the teacher.

Responses

Response content type: application/json

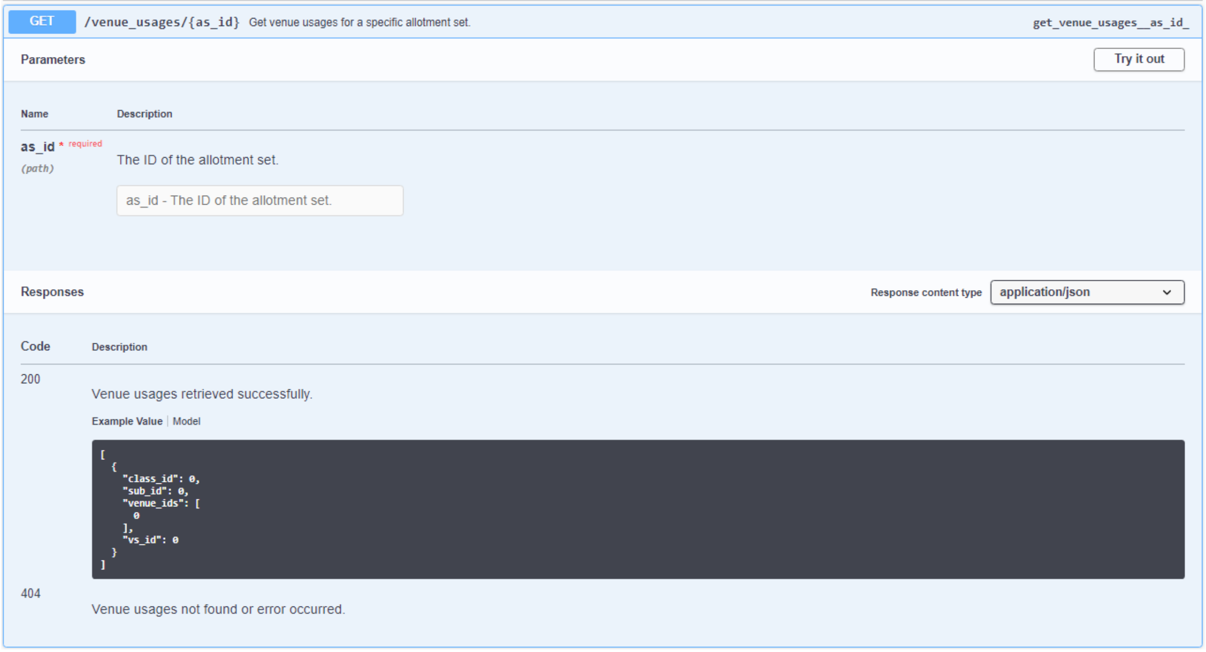
Code	Description
200	List of teaching classes retrieved successfully.
404	Error occurred while retrieving teaching classes.

Example Value | Model

```
[
  {
    "class_id": 0
  }
]
```

Figure 5.5.15.1: The route design for getting teaching classes of a teacher.

5.5.16 Venue Usages Tag



GET /venue_usages/{as_id} Get venue usages for a specific allotment set. get_venue_usages__as_id_

Parameters

Name	Description
as_id * required *(path)*	The ID of the allotment set. as_id - The ID of the allotment set.

Responses

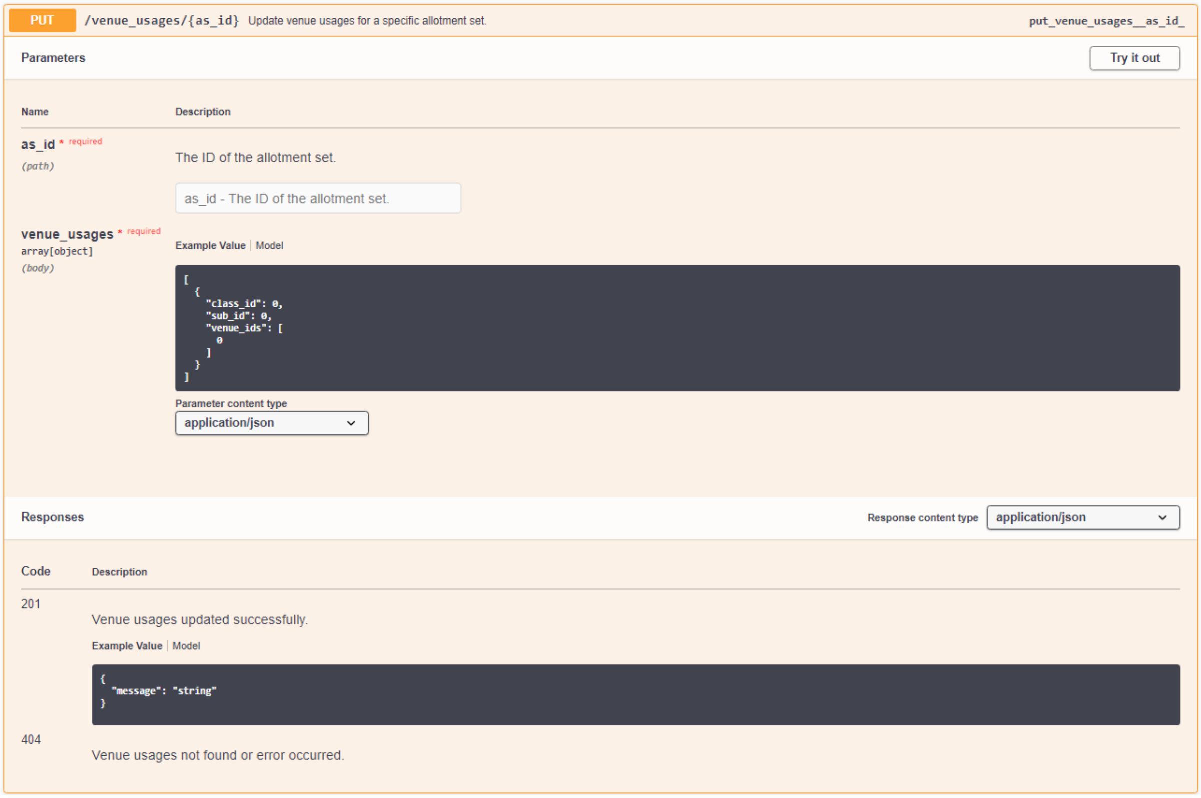
Response content type: application/json

Code	Description
200	Venue usages retrieved successfully.
404	Venue usages not found or error occurred.

Example Value | Model

```
[
  {
    "class_id": 0,
    "sub_id": 0,
    "venue_ids": [
      0
    ],
    "vs_id": 0
  }
]
```

Figure 5.5.16.1: The route design for getting venue usages.



PUT /venue_usages/{as_id} Update venue usages for a specific allotment set. put_venue_usages__as_id_

Parameters

Name	Description
as_id * required *(path)*	The ID of the allotment set. as_id - The ID of the allotment set.
venue_usages * required array[object] *(body)*	Example Value | Model

```
[
  {
    "class_id": 0,
    "sub_id": 0,
    "venue_ids": [
      0
    ]
  }
]
```

Parameter content type: application/json

Responses

Response content type: application/json

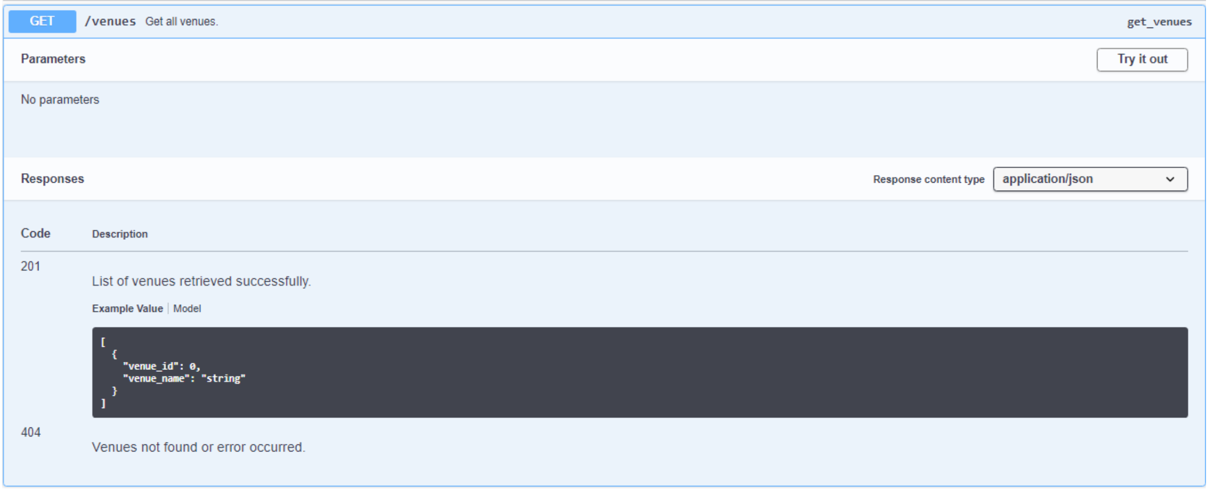
Code	Description
201	Venue usages updated successfully.
404	Venue usages not found or error occurred.

Example Value | Model

```
{
  "message": "string"
}
```

Figure 5.5.16.2: The route design for updating venue usages.

5.5.17 Venues Tag



/venues Get all venues. get_venues

Parameters

No parameters

Responses

Response content type: application/json

Code | Description

201

List of venues retrieved successfully.

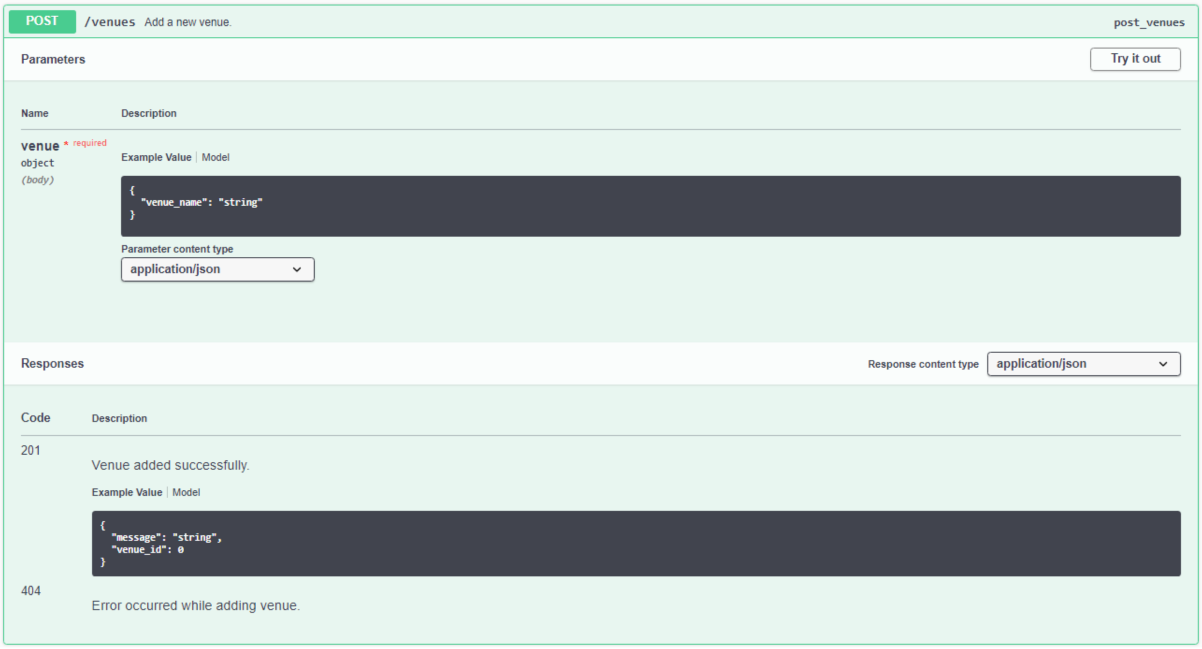
Example Value | Model

```
[
  {
    "venue_id": 0,
    "venue_name": "string"
  }
]
```

404

Venues not found or error occurred.

Figure 5.5.17.1: The route design for getting all venues.



/venues Add a new venue. post_venues

Parameters

Name | Description

venue * required
object
(body)

Example Value | Model

```
{
  "venue_name": "string"
}
```

Parameter content type: application/json

Responses

Response content type: application/json

Code | Description

201

Venue added successfully.

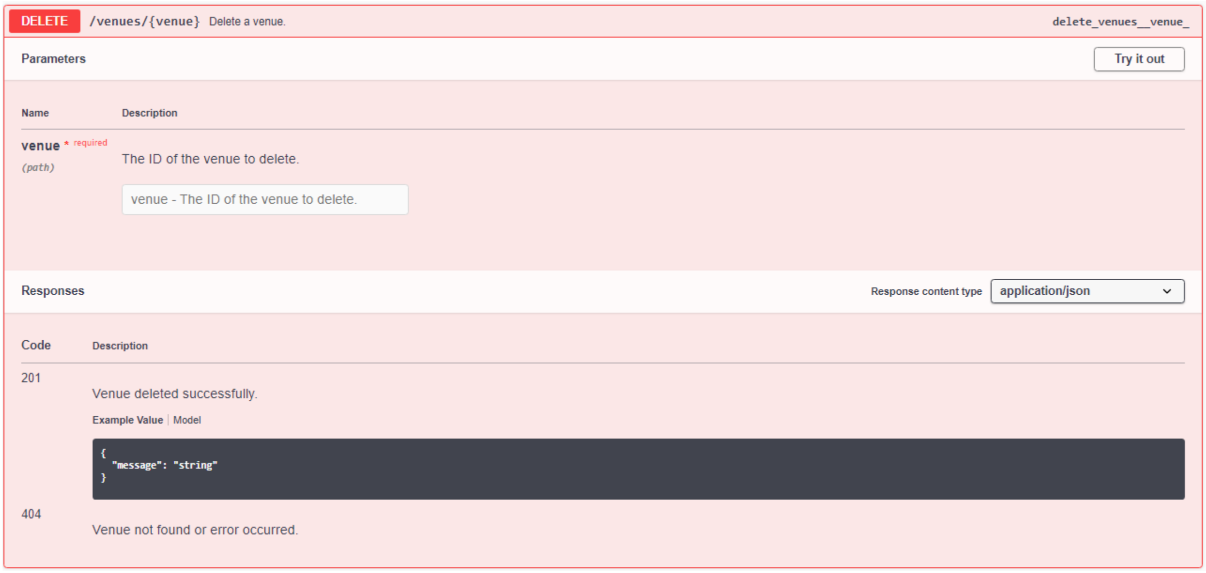
Example Value | Model

```
{
  "message": "string",
  "venue_id": 0
}
```

404

Error occurred while adding venue.

Figure 5.5.17.2: The route design for creating a venue.



DELETE /venues/{venue} Delete a venue. delete_venues__venue_

Parameters — Try it out

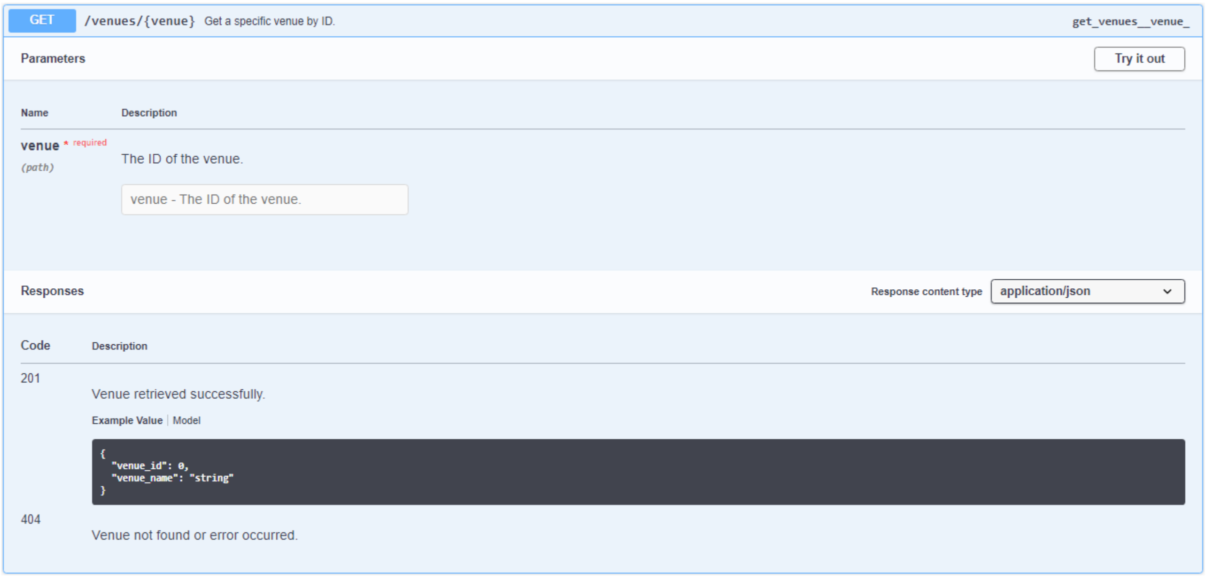
Name	Description
venue * required (path)	The ID of the venue to delete. venue - The ID of the venue to delete.

Responses — Response content type: application/json

Code	Description
201	Venue deleted successfully. Example Value | Model
404	Venue not found or error occurred.

```
{
  "message": "string"
}
```

Figure 5.5.17.3 The route design for deleting a venue.



GET /venues/{venue} Get a specific venue by ID. get_venues__venue_

Parameters — Try it out

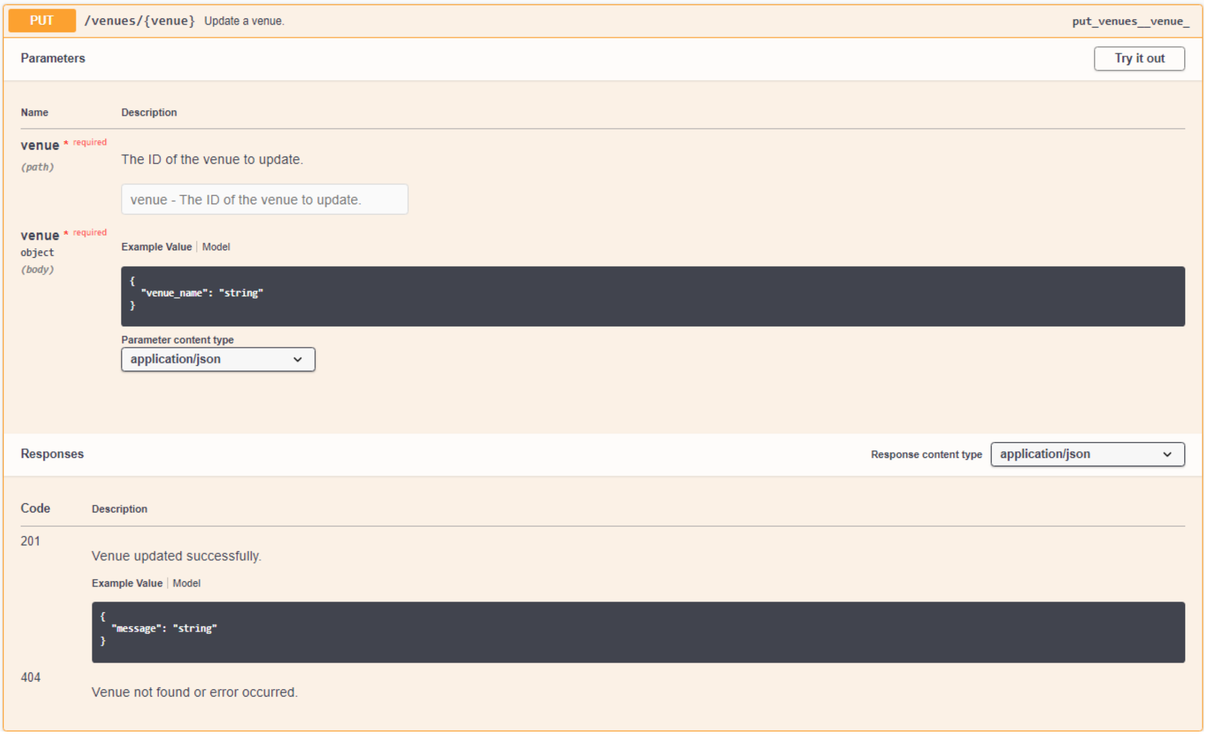
Name	Description
venue * required (path)	The ID of the venue. venue - The ID of the venue.

Responses — Response content type: application/json

Code	Description
201	Venue retrieved successfully. Example Value | Model
404	Venue not found or error occurred.

```
{
  "venue_id": 0,
  "venue_name": "string"
}
```

Figure 5.5.17.4 The route design for getting a venue.



/venues/{venue} Update a venue.

Parameters

Name	Description
venue * required (path)	The ID of the venue to update. venue - The ID of the venue to update.
venue * required object (body)	Example Value | Model

```
{
  "venue_name": "string"
}
```

Parameter content type: application/json

Responses

Response content type: application/json

Code	Description
201	Venue updated successfully. Example Value | Model

```
{
  "message": "string"
}
```

Code	Description
404	Venue not found or error occurred.

Figure 5.5.17.5 The route design for updating a venue.

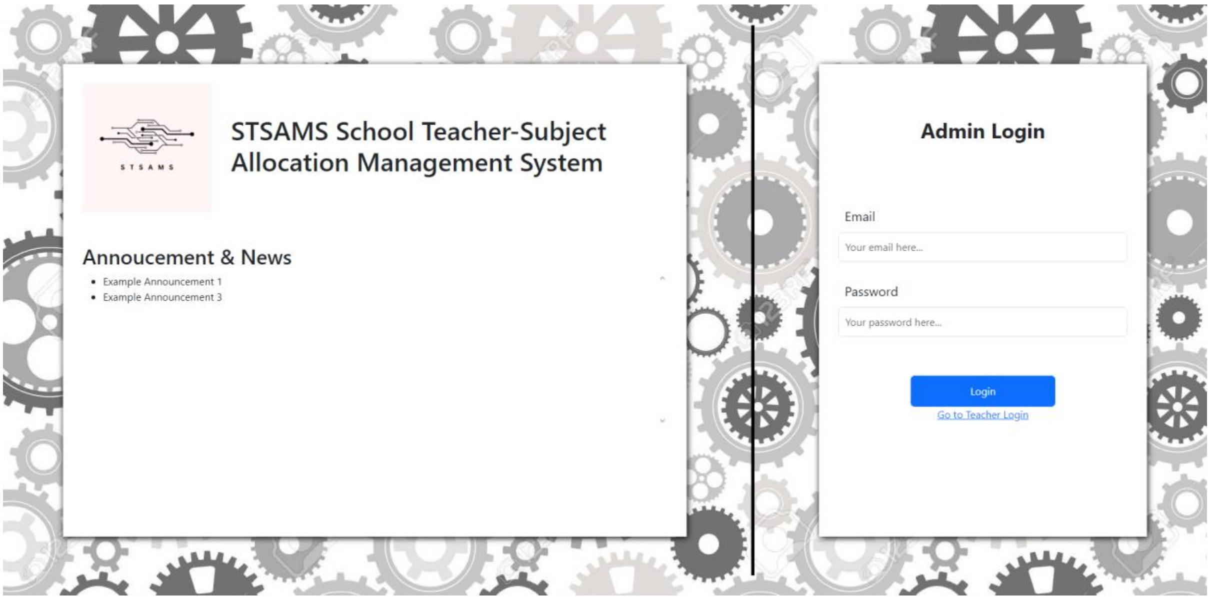
CHAPTER 6

SYSTEM IMPLEMENTATION

In this chapter, the implementation and functionality of the system will be explained. Steps and guides on how to use the system will be showed to increase user's understanding STSAMS.

6.1 Authentication & Authorization

As there are two different users using STSAMS, two user roles are defined to implement a clear separation between system admins and teachers. There are a few functionalities in the system to prevent users from accessing the unauthorized URLs to increase the security of the system. In STSAMS, there are two login pages either for system administrators or teacher. In UI design, they share the same structure to ensure consistency. Visualization for both login pages in shown in Figure 6.1.1 and Figure 6.1.2.



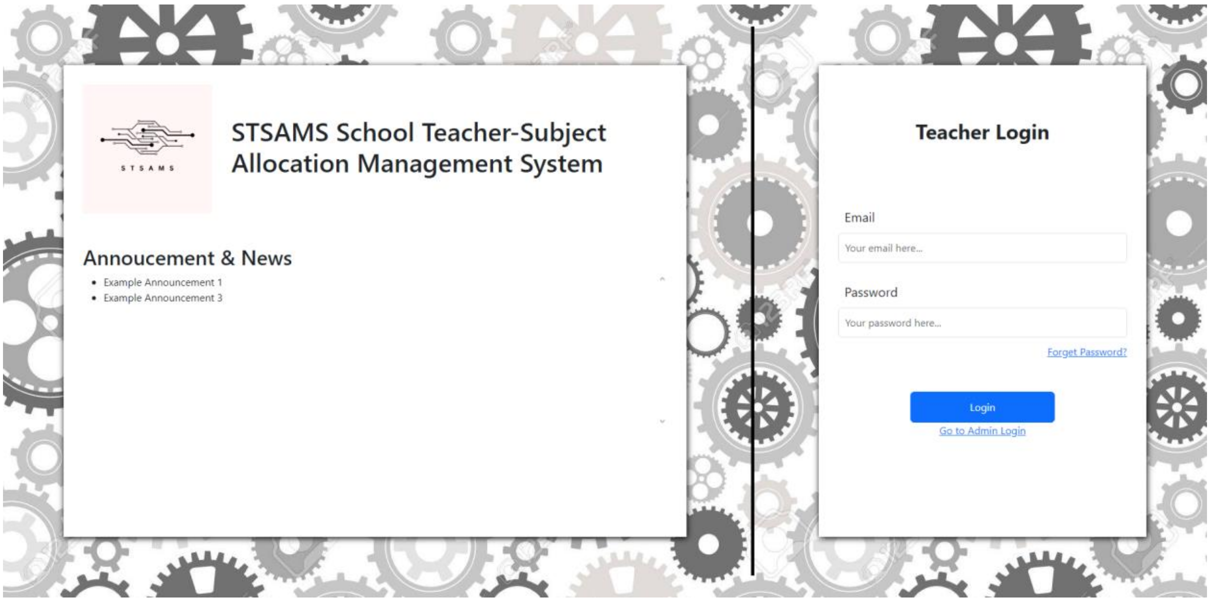
STSAMS School Teacher-Subject Allocation Management System

Annoucement & News

- Example Announcement 1
- Example Announcement 3

Admin Login

Figure 6.1.1: Login Page for System Administrators.



STSAMS School Teacher-Subject Allocation Management System

Annoucement & News

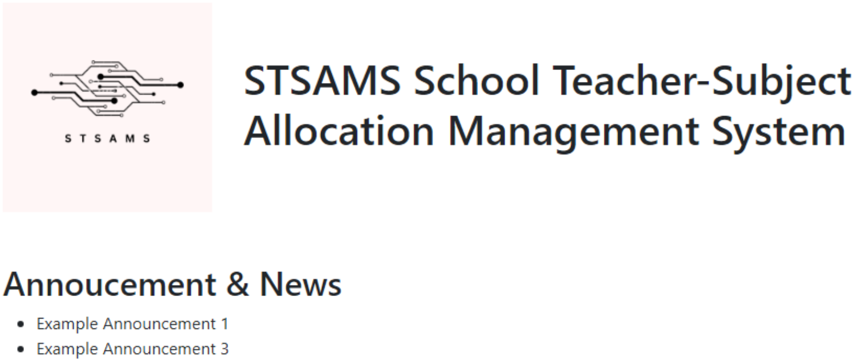
- Example Announcement 1
- Example Announcement 3

Figure 6.1.2: Login Page for Teachers.

To ease navigating between these two pages, a link is added below the Login button like Figure 6.1.3. So, the system administrators and teachers will not have to change the URL to get access to their desired login page.

Figure 6.1.3: Navigation link to Admin Login.

In both login pages, there is also an announcements section to the show the announcement set by the system administrators such as Figure 6.1.4. The data management for announcements will be shown in the later section.



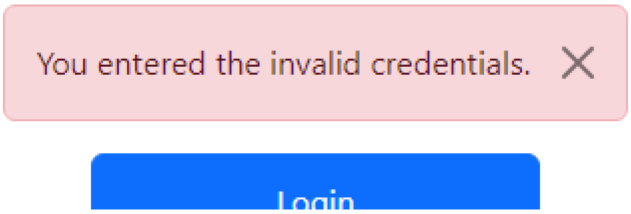
STSAMS School Teacher-Subject Allocation Management System

Annoucement & News

- Example Announcement 1
- Example Announcement 3

Figure 6.1.4: Announcement Section in the Login Page.

Then, imagine a scenario of one of users wanting to log into the system. If the user entered the wrong credentials or accessed to the wrong login page, the system will show an alert, displaying an error message of "You entered the invalid credentials." as shown in the Figure 6.1.5.



You entered the invalid credentials.

Figure 6.1.5: Alert Message for Invalid Logins.

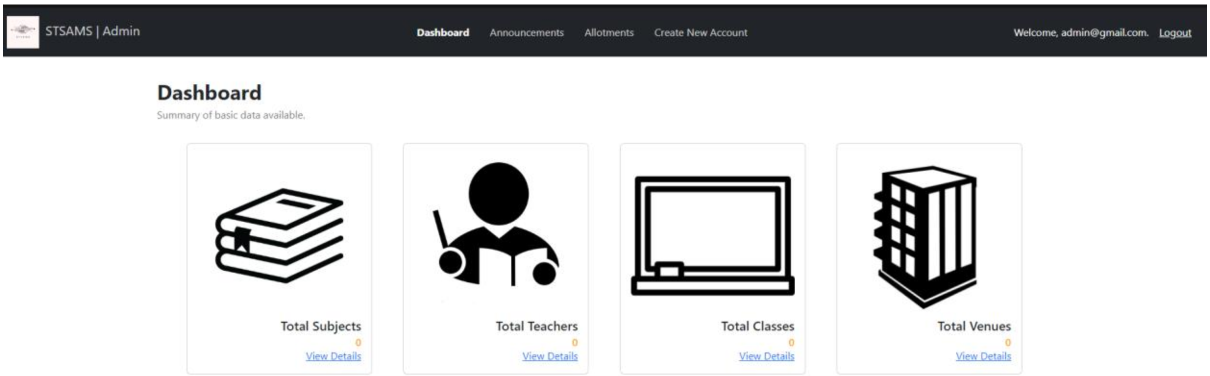
Besides that, the system will check for authentications before letting any users to access any pages. As roles are defined for each type of user, the system will know if the user to authenticated as a teacher or an administrator. Using this logic, an authenticated teacher will not be allowed to access the system administrator's page even though they are authenticated. This will also apply similarly to the system administrators. If both users try to do so, the system will simply send them back to the login page if they are not authenticated or their home pages if they are already authenticated. The example of navigation is shown in Table 6.1.1.

Table 6.1.1: Example of navigation and redirection based on authentication.

User Type	Trying to Access	Authenticated?	Redirection
Admin	/teacher/home	No	/login/teacher
Admin	/admin/home	No	/login/admin
Admin	/admin/login	Yes	/admin/home
Teacher	/admin/home	Yes	/teacher/home
Teacher	/teacher/home	Yes	-
Teacher	/teacher/home	No	/login/teacher

6.2 Data Management

This data management will only be available to the administrator's module. This is because only system administrators are authorized to modify the data in the system. Once a system administrator is logged in to the system, he/she will be brought to the home page which can be also called as the Dashboard. The dashboard allows the system administrator to take get the overview of the data in the system. For example, the data is categorized into 4 sections which are the subject section, teacher section, classes section, and venues section followed by the total count of the data as shown in Figure 6.2.1.



Dashboard

Summary of basic data available.

Total Subjects 0

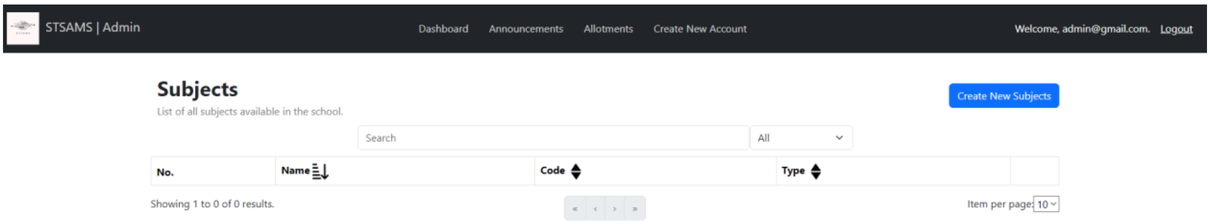
Total Teachers 0

Total Classes 0

Total Venues 0

Figure 6.2.1: The Dashboard in the Admin Module.

Then, the system administrators can click on view details to navigate to the page that shows the detailed list of specific type of data. To illustrate, the administrator will be navigated to the subject's management page as shown in Figure 6.2.2 after clicking on "View Details" link in the subject section.



Subjects

List of all subjects available in the school.

No.	Name	Code	Type	

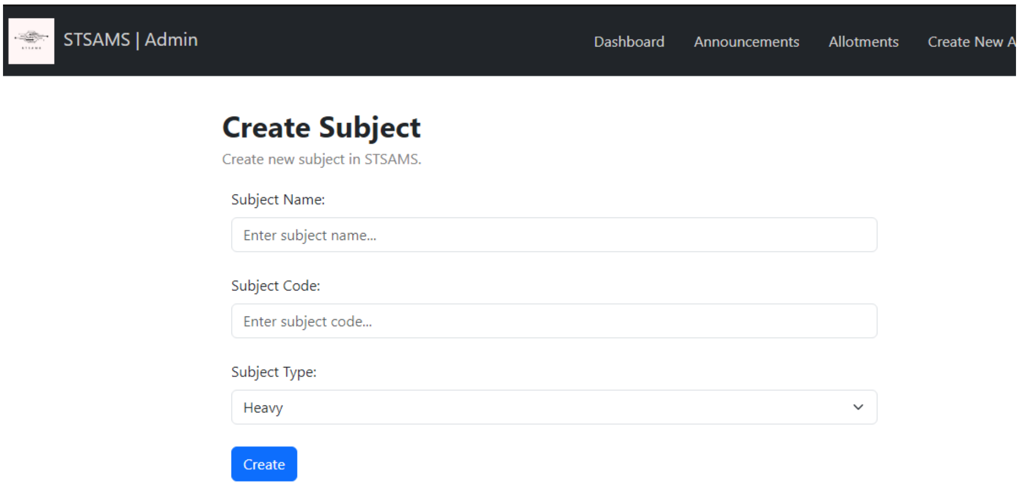
Showing 1 to 0 of 0 results.

Item per page: 10

Figure 6.2.2 Subject Management Page in the Admin Module.

The functionalities in Figure 6.2.2 will be explained from the left to the right then progressing downwards. On the top left corner, it shows the current page title the system administrator is currently accessing, then a brief description below explaining the page context.

Then, clicking on the button will bring the admin to the subject creation page as shown in Figure 6.2.3 to enter the details for a new subject.



Create Subject

Create new subject in STSAMS.

Subject Name:

Enter subject name...

Subject Code:

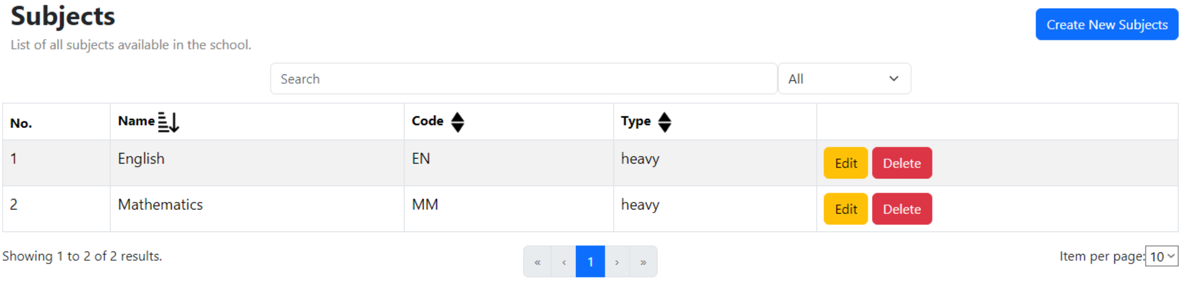
Enter subject code...

Subject Type:

Heavy

Figure 6.2.3: Subject Creation Page.

After filling in the details for a new subject then confirming the subject creation, the administrator will be brought back to the Subjects page. Then, the administrator will see the newly created subject in the table content.



Subjects

List of all subjects available in the school.

Search

All

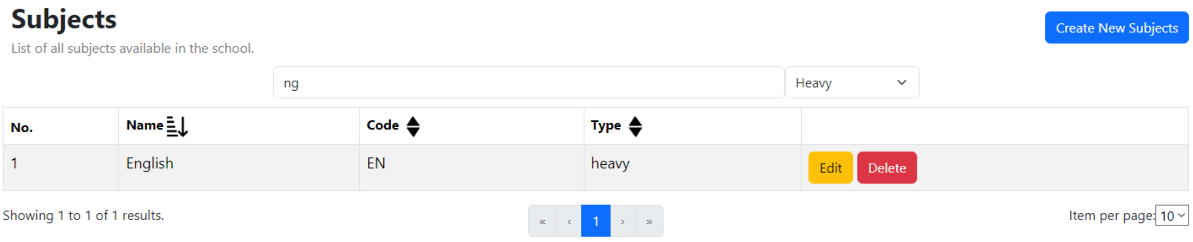
No.	Name	Code	Type	
1	English	EN	heavy	Edit Delete
2	Mathematics	MM	heavy	Edit Delete

Showing 1 to 2 of 2 results.

Item per page: 10

Figure 6.2.4: New Subject Added to the Table.

Then, there is the search bar and type selection input box. Both functionalities will filter out the subjects in table based on specific conditions. For example, the search bar will filter out subjects based on its name and the selection input box will filter out subjects by its type. After filling in the filter section, the English subject will only be displayed as shown in Figure 6.2.5.



Subjects

List of all subjects available in the school.

Create New Subjects

ng

Heavy

No.	Name	Code	Type	
1	English	EN	heavy	Edit Delete

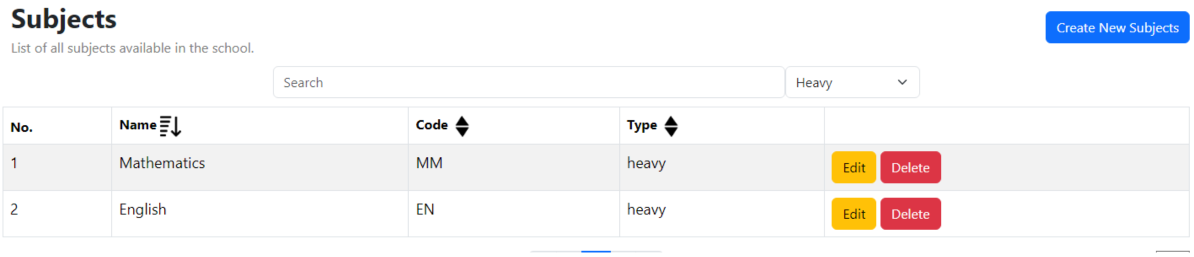
Showing 1 to 1 of 1 results.

« ‹ 1 › »

Item per page: 10

Figure 6.2.5: Filtered Subject Table.

Other than that, the admin can also sort the subjects by their names, codes, or types. In Figure 6.2.6, it is showing that the subjects are now sorted based on their names in descending order. The different types of sorting mode are presented in Table 6.2.1.



Subjects

List of all subjects available in the school.

Create New Subjects

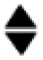
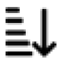
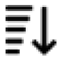
Search

Heavy

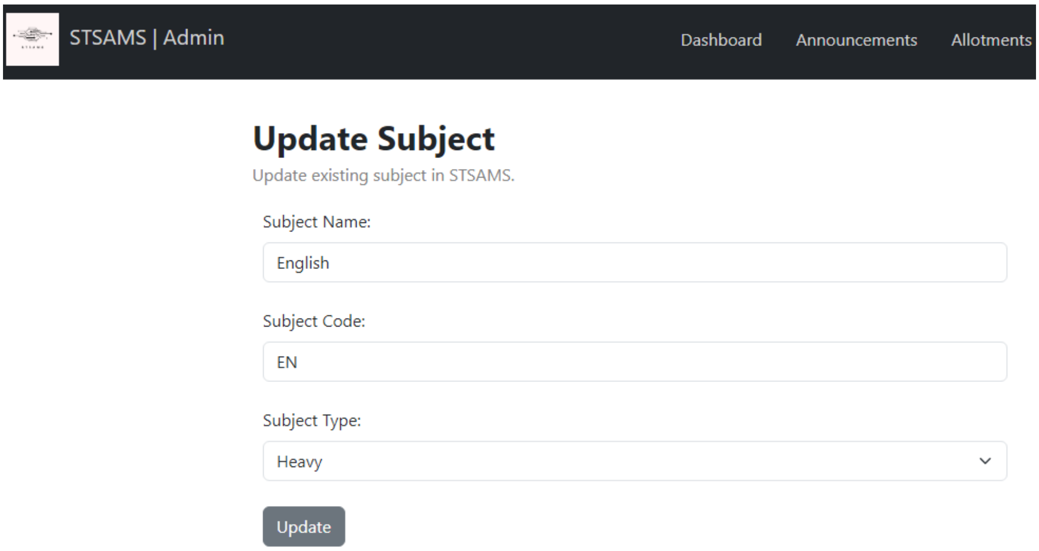
No.	Name	Code	Type	
1	Mathematics	MM	heavy	Edit Delete
2	English	EN	heavy	Edit Delete

Figure 6.2.6: Sorted Subject Table.

Table 6.2.1: The Available Sorting Modes and its Symbol Used.

No.	Sort Modes	Symbols Used
1.	Unsorted	 Figure 6.2.7: Symbol for Unsorted Mode.
2.	Sorted in Ascending Order	 Figure 6.2.8: Symbol for Ascending Order Mode.
3.	Sorted in Descending Order	 Figure 6.2.9: Symbol for Descending Order Mode.

If the admins want to edit the details for the subject, they can click on the Edit button. They will be brought to the edit subject page as shown in Figure 6.2.7 to modify the subject data like subject name, subject code, and subject type.



Update Subject

Update existing subject in STSAMS.

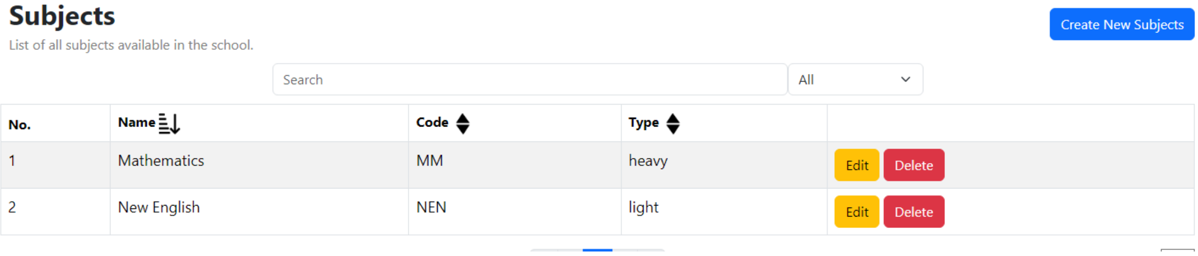
Subject Name: English

Subject Code: EN

Subject Type: Heavy

Figure 6.2.7: Subject Modification Page.

After changing and confirming the details of the selected subject, the admins will be navigated back to the Subject Page and the updated subject will be reflected in the subjects table instantly like in Figure 6.2.8.



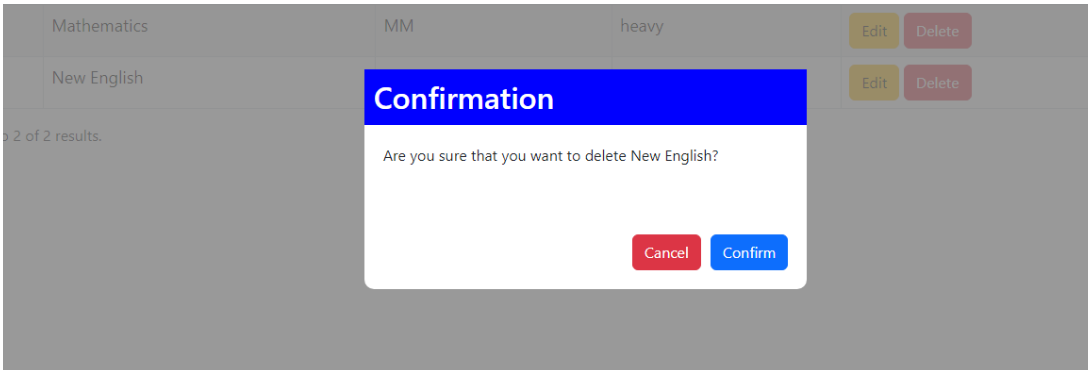
Subjects

List of all subjects available in the school.

No.	Name	Code	Type	
1	Mathematics	MM	heavy	Edit Delete
2	New English	NEN	light	Edit Delete

Figure 6.2.8: The Newly Update Subject Data.

To delete a subject, the system admins can click on the Delete button. To avoid accidental deletion, a confirmation dialog like Figure 6.2.9 will be displayed before deleting a subject.



Mathematics	MM	heavy	Edit Delete
New English			Edit Delete

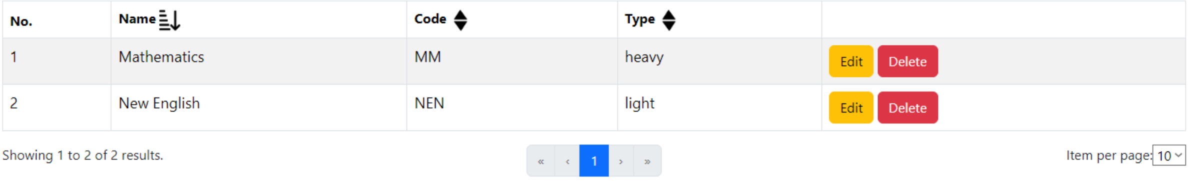
2 of 2 results.

Confirmation

Are you sure that you want to delete New English?

Figure 6.2.9: Confirmation Dialog before Deletions.

To prevent information overload and excessive scrolling, the pagination functionality is implemented and placed at the bottom of the table as shown in Figure 6.2.10. For example, the system admins can see which range of subjects that are currently being displayed, navigate through the pages, and set number of subjects will be displayed in one page.



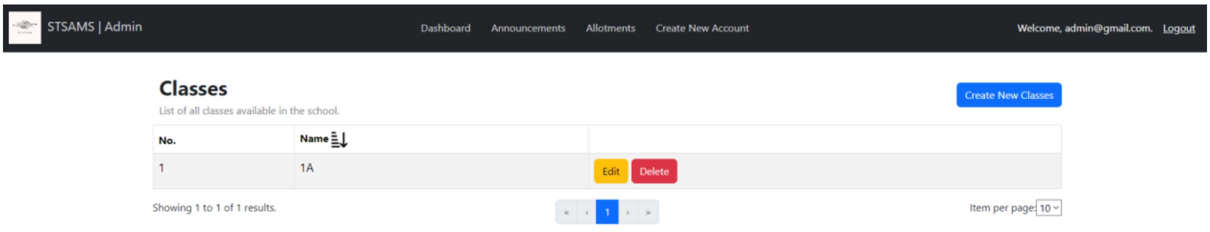
No.	Name ↓	Code ⬍	Type ⬍	
1	Mathematics	MM	heavy	Edit Delete
2	New English	NEN	light	Edit Delete

Showing 1 to 2 of 2 results.

Figure 6.2.10: Pagination in Subject Page.

This will conclude the guides on subject management. To ensure consistency, all the other data like classes, venues, teachers, and announcements will be mostly similar to subject page. However, there are still some explanations need to be made in the teachers and announcements page.

The figures below will show in the implementations in the classes page.



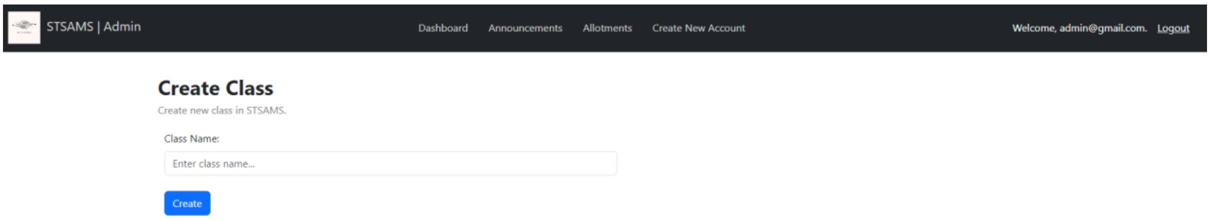
Classes

List of all classes available in the school.

No.	Name ↓	
1	1A	Edit Delete

Showing 1 to 1 of 1 results.

Figure 6.2.11: Main Page for Class Management.



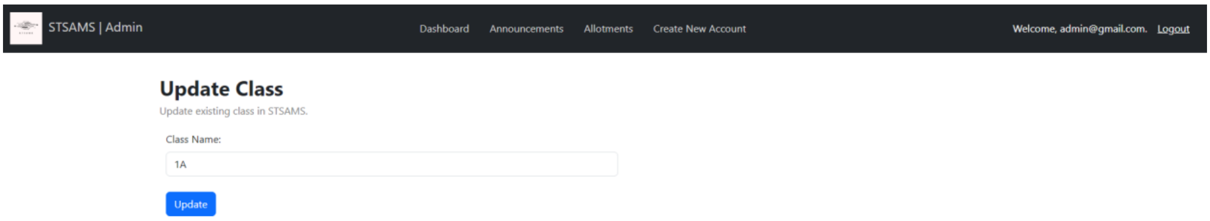
Create Class

Create new class in STSAMS.

Class Name:

Enter class name...

Figure 6.2.12: Class Creation Page.



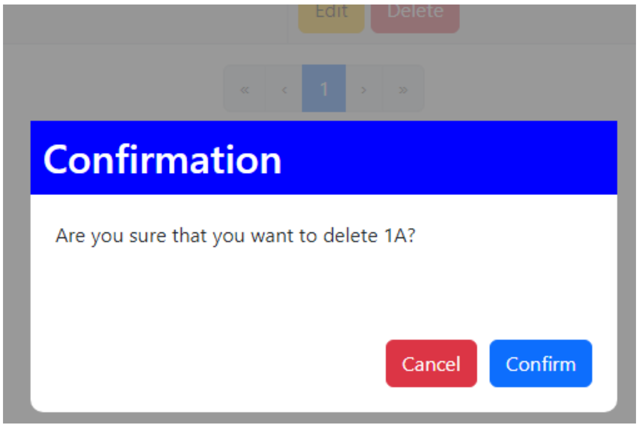
Update Class

Update existing class in STSAMS.

Class Name:

1A

Figure 6.2.13: Class Modification Page.

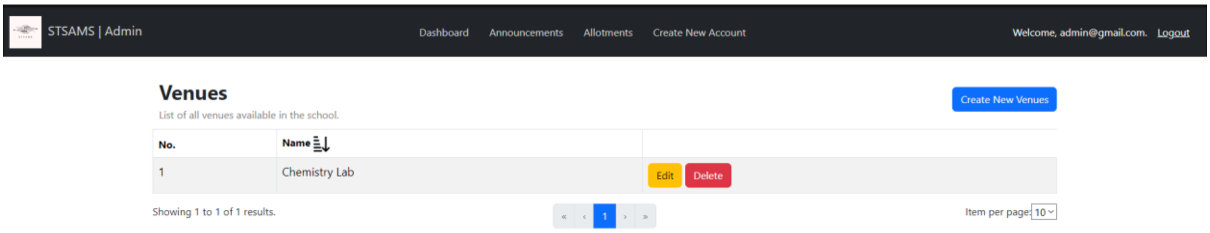


Confirmation

Are you sure that you want to delete 1A?

Figure 6.2.14: Confirmation Dialog before Class Deletions.

The figures below will show in the implementations in the venues page.



Venues

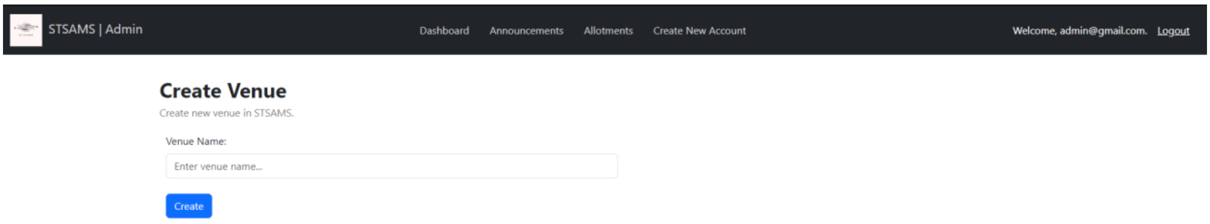
List of all venues available in the school.

No.	Name	
1	Chemistry Lab	Edit Delete

Showing 1 to 1 of 1 results.

Item per page: 10

Figure 6.2.15: Main Page for Venue Management.

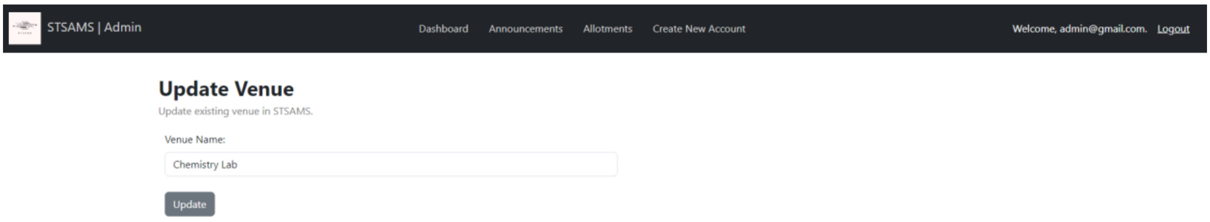


Create Venue

Create new venue in STSAMS.

Venue Name: Enter venue name...

Figure 6.2.16: Venue Creation Page.

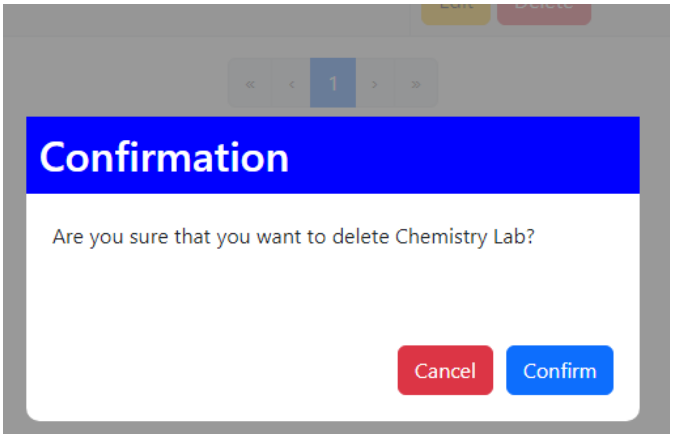


Update Venue

Update existing venue in STSAMS.

Venue Name: Chemistry Lab

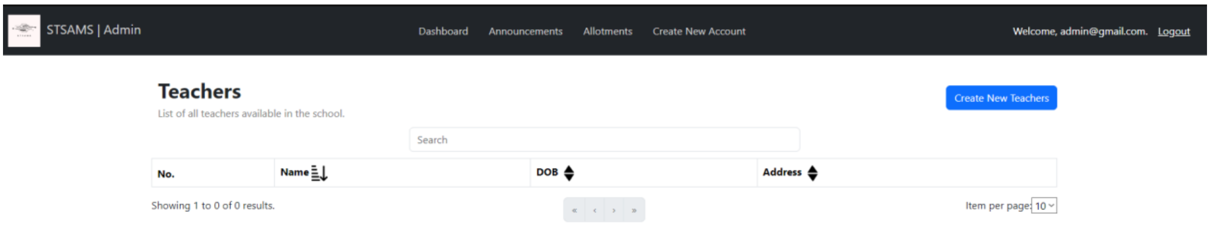
Figure 6.2.17: Venue Modification Page.



Are you sure that you want to delete Chemistry Lab?

Figure 6.2.14: Confirmation Dialog before Venue Deletions.

The figures below will show in the implementations in the Teacher page.



Teachers

List of all teachers available in the school.

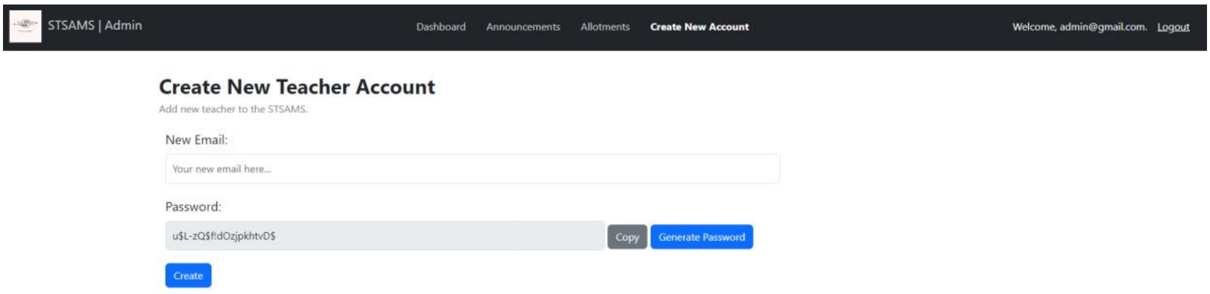
No.	Name	DOB	Address

Showing 1 to 0 of 0 results.

Item per page: 10

Figure 6.2.15: Main Page for Teacher Management.

Then, the system admins can create a new teacher by pressing the Create New Teachers button. It will navigate the admins to the teacher creation screen as displayed in Figure 6.2.16.



Create New Teacher Account

Add new teacher to the STSAMS.

New Email:

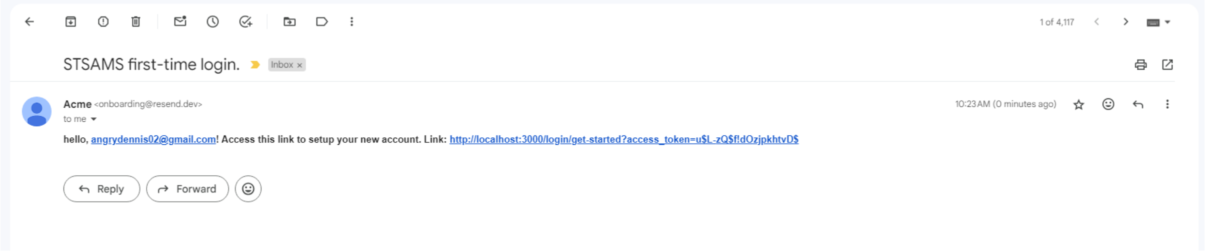
Your new email here...

Password:

u$L-zQ$f!dOzjpkhtvD$

Figure 6.2.16: Teacher Creation Page.

The system enables system admins to enter email for the new teacher as well as generating a default password for the teacher. If the admins are not happy with the generated password, they can create a new random password by pressing on the Generate Password button. To copy the password to the clipboard, they can press on the Copy button. After confirming the new teacher's details, the Create button can be pressed and an email like in Figure 6.2.17 will be sent to the teacher to set up his/her account.



STSAMS first-time login.

Acme <onboarding@resend.dev>

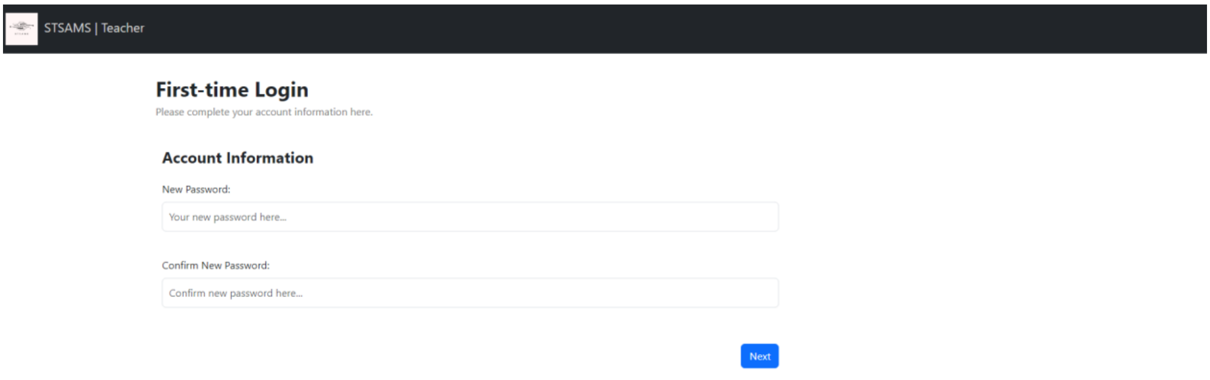
to me

10:23 AM (0 minutes ago)

hello, angrydennis02@gmail.com! Access this link to setup your new account. Link: http://localhost:3000/login/get-started?access_token=u$L-zQ$f!dOzjpkhtvD$

Figure 6.2.17: Email Received by the Teacher.

The first-time login page illustrated in Figure 6.2.18 will be opened after the teacher clicked on the access link.



First-time Login

Please complete your account information here.

Account Information

New Password:

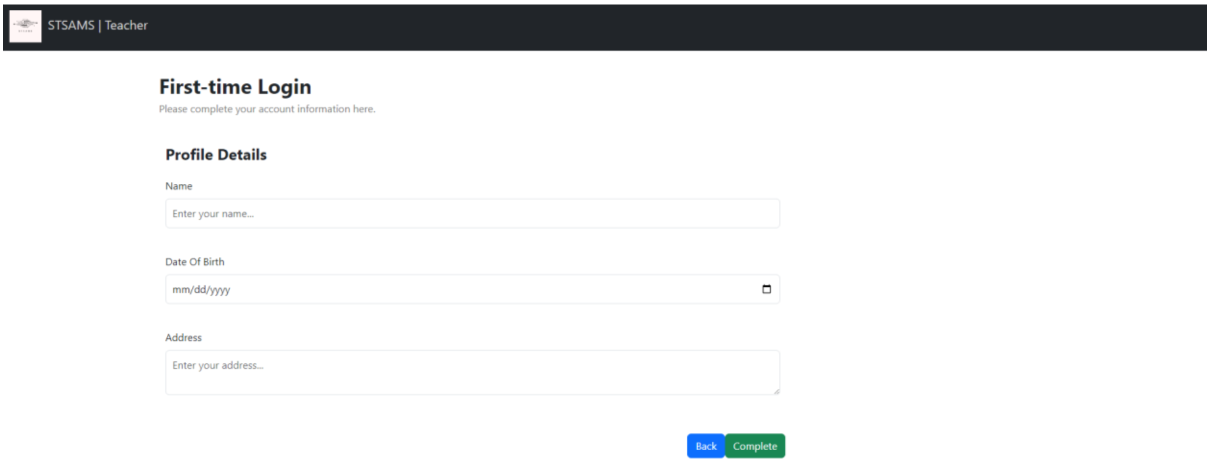
Your new password here...

Confirm New Password:

Confirm new password here...

Figure 6.2.18: First-time Login Page.

The system will require the teacher to update their default password to a new strong password that will have at least one special, numeric, uppercase, or lowercase character, longer than 12 characters. To do further validation, this system will make sure that the password and password confirmation are matched before allowing the teacher to go to the next page as shown in Figure 6.2.19.



First-time Login

Please complete your account information here.

Profile Details

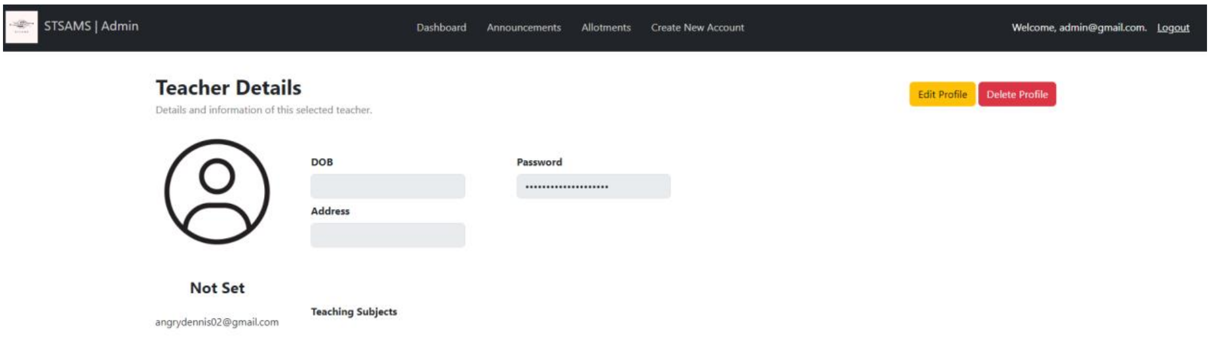
Name: Enter your name...

Date Of Birth: mm/dd/yyyy

Address: Enter your address...

Figure 6.2.19: Profile Details Form in First-time Login Page.

After filling in the details, clicking on the Complete button will bring the teacher to the login page. Moreover, the system admins can view and update the details for the teacher.



Teacher Details

Details and information of this selected teacher.

Not Set

angrydennis02@gmail.com

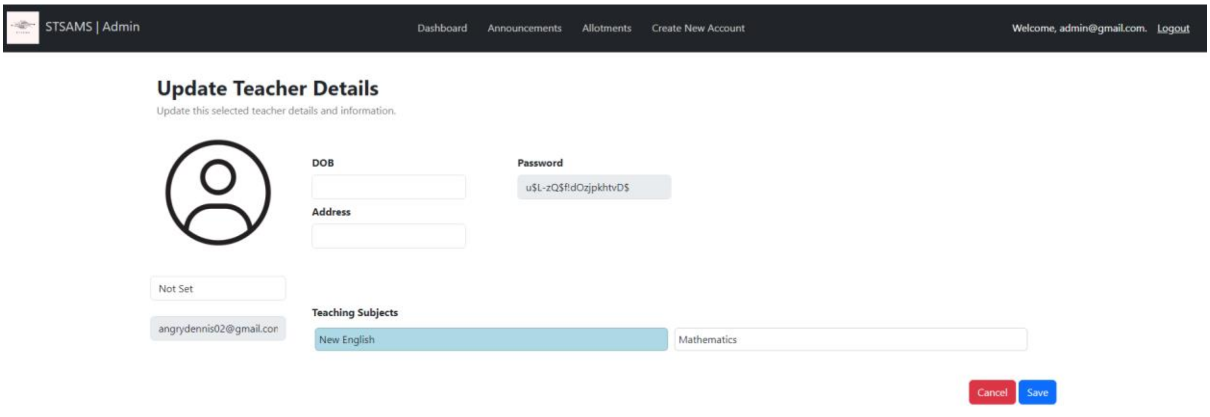
DOB

Address

Password: ••••••••••••••••••••

Teaching Subjects

Figure 6.2.20: View Details Page for Teacher.



Update Teacher Details

Update this selected teacher details and information.

Not Set

angrydennis02@gmail.cor

DOB

Address

Password: u$L-zQ$f!dOzjpkhtvD$

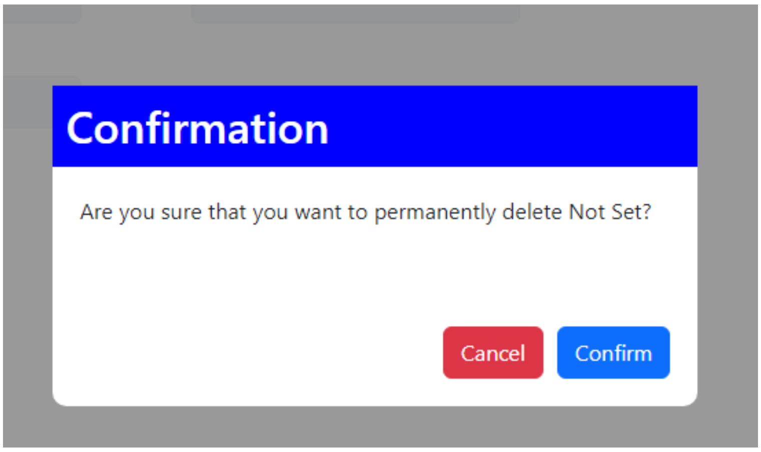
Teaching Subjects

New English

Mathematics

Figure 6.2.21: Update Teacher Page.

If one of the teachers in school is transferred away or retired, the system admins can choose to delete the profile from the system. To prevent accidental deletion, a confirmation dialog will also be shown like in Figure 6.2.22.



Confirmation

Are you sure that you want to permanently delete Not Set?

Figure 6.2.22: Confirmation Dialog before Teacher Profile Deletion.

As for the announcements management, the announcements' main page can be accessed through the top navigation bar as shown in Figure 6.2.23.

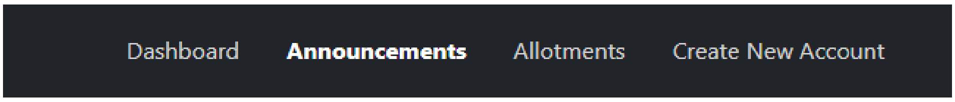
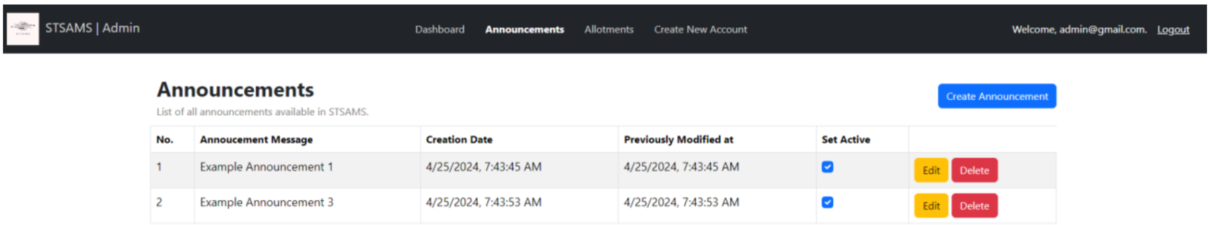


Figure 6.2.23: Top Navigation Bar of STSAMS.

The figures below will show in the implementations in the Announcements page.

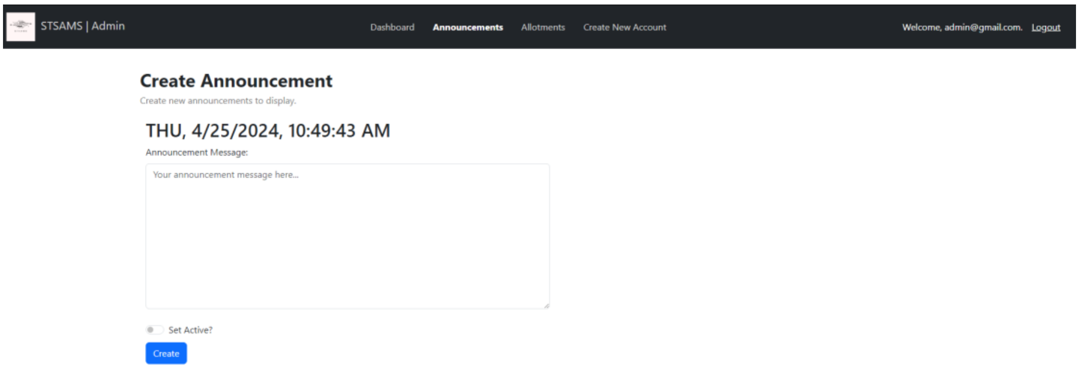


Announcements

List of all announcements available in STSAMS.

No.	Annoucement Message	Creation Date	Previously Modified at	Set Active	
1	Example Announcement 1	4/25/2024, 7:43:45 AM	4/25/2024, 7:43:45 AM	☑	Edit Delete
2	Example Announcement 3	4/25/2024, 7:43:53 AM	4/25/2024, 7:43:53 AM	☑	Edit Delete

Figure 6.2.24: Main Page for Announcements Management.



Create Announcement

Create new announcements to display.

THU, 4/25/2024, 10:49:43 AM

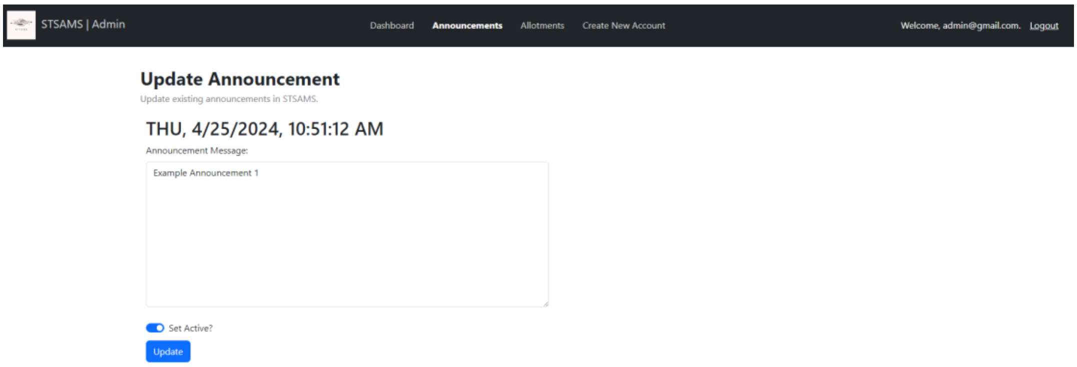
Announcement Message:

Your announcement message here...

○ Set Active?

Figure 6.2.25: Announcement Creation Page.

Once the “Set Active?” switch is enabled, this option will allow the announcement to be shown in the login page for both teachers and administrators.



Update Announcement

Update existing announcements in STSAMS.

THU, 4/25/2024, 10:51:12 AM

Announcement Message:

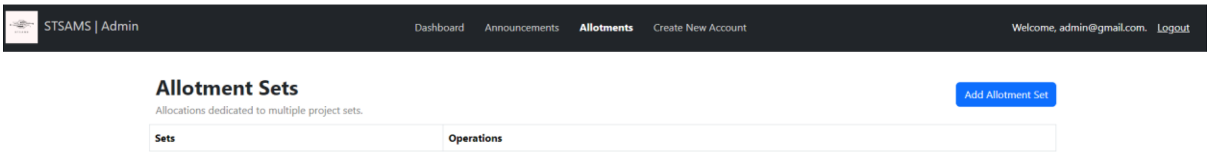
Example Announcement 1

● Set Active?

Figure 6.2.26: Announcement Modification Page.

6.3 Allotment Sets & Constraint Settings

This is most important section of the system as the system administrators will define relationship between the previously created data. To allow the system admins to modularize their work, allotment used in different context can be grouped into allotment sets. It can be said that it is very similar to a project file for a software. The page to manage the allotment sets is shown in Figure 6.3.1.



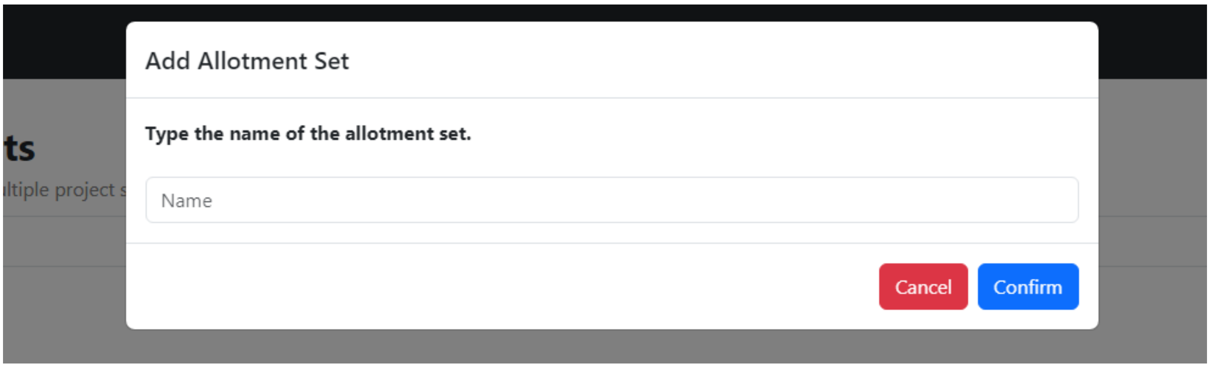
Allotment Sets

Allocations dedicated to multiple project sets.

Sets	Operations

Figure 6.3.1: Main Page for Allotment Sets Management.

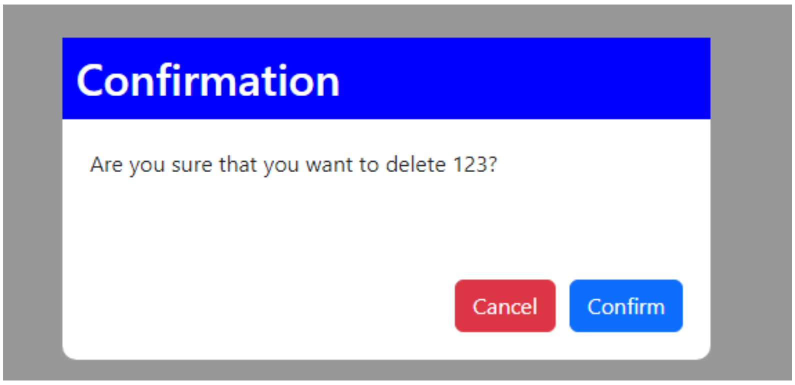
The system admins can add new allotment set by pressing on the Add Allotment Set button. Then, a modal illustrated in Figure 6.3.2 will be displayed, requiring the name of the allotment set. Then, confirmation dialog like in Figure 6.3.3 will also be displayed before confirming to delete an allotment set.



Add Allotment Set

Type the name of the allotment set.

Figure 6.3.2: Modal for Allotment Set Creation.

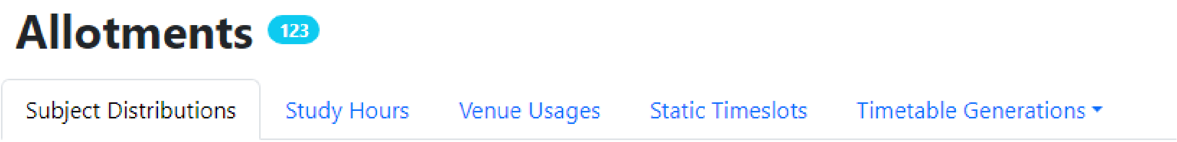


Confirmation

Are you sure that you want to delete 123?

Figure 6.3.3: Confirmation Dialog before Deleting Allotment Set.

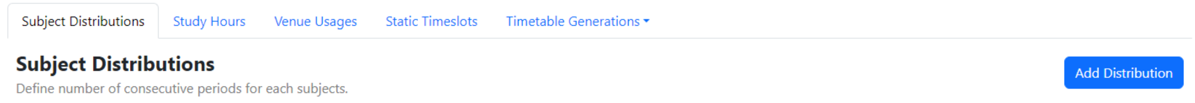
After clicking on the name of the newly created allotment set, it will bring the admins to the Allotments Page that will allow admins to define the constraint settings for Study Hours, Subject Distributions, Venue Usages and Static Timeslots as well as the timetable generation options for Static Generation and Dynamic Generation as shown in Figure 6.3.4.



Allotments 123

Figure 6.3.4: The Options Available in Allotments Page.

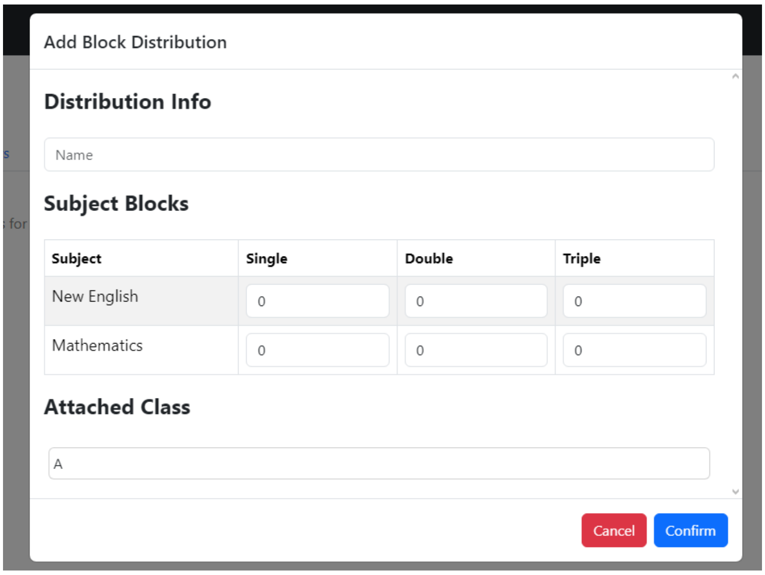
The constraint setting options are intentionally ordered in the way that admins must follow to ensure that the process of generating timetables is without any problem. So, the first constraint to be set will be the subject distributions. The system admins can add distribution by pressing on the Add Distribution button. After that, a modal will be popped up for the system admins to fill in the distribution details for the allotment.



Subject Distributions

Define number of consecutive periods for each subjects.

Figure 6.3.5: Content of Subject Distribution Tab.



Add Block Distribution

Distribution Info

Subject Blocks

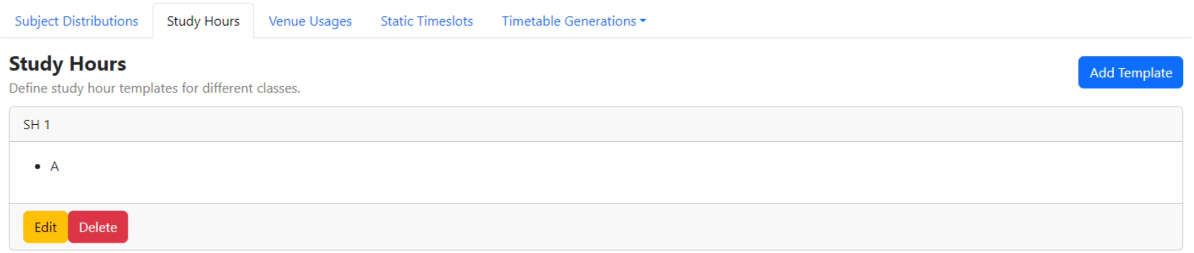
Subject	Single	Double	Triple
New English	0	0	0
Mathematics	0	0	0

Attached Class

A

Figure 6.3.6: Modal of Distribution Creation.

The functionalities are also like the Study Hours constraint and Static Timeslots constraint.



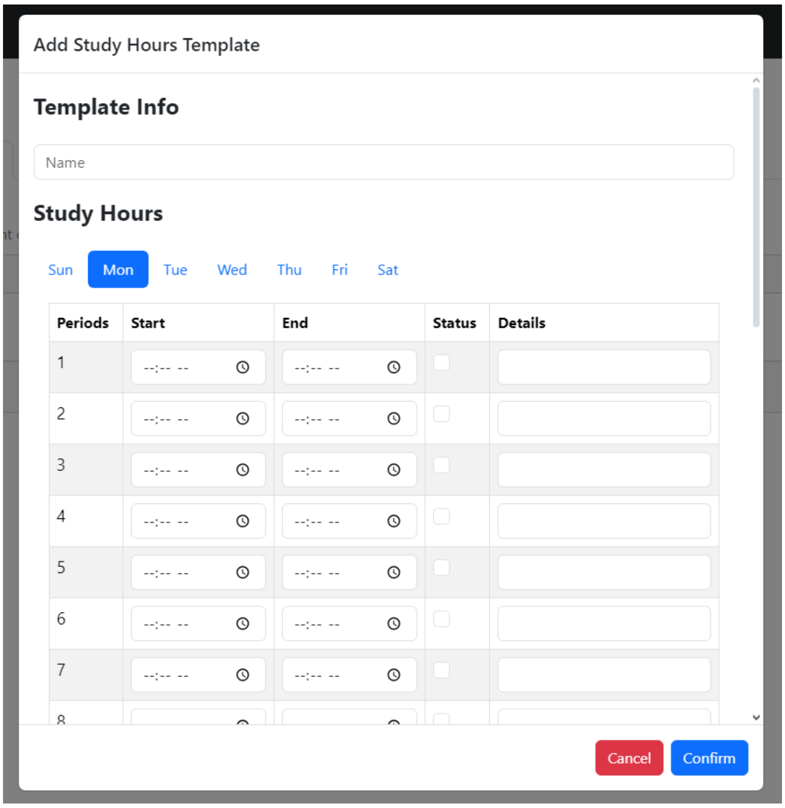
Study Hours

Define study hour templates for different classes.

SH 1

- A

Figure 6.3.7: Content of Study Hours Tab.



Add Study Hours Template

Template Info

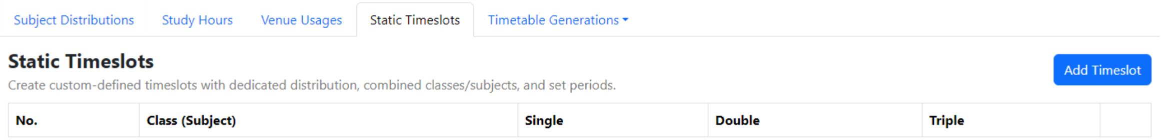
Name

Study Hours

Periods	Start	End	Status	Details
1	--:-- --	--:-- --	☐	
2	--:-- --	--:-- --	☐	
3	--:-- --	--:-- --	☐	
4	--:-- --	--:-- --	☐	
5	--:-- --	--:-- --	☐	
6	--:-- --	--:-- --	☐	
7	--:-- --	--:-- --	☐	
8			☐	

Figure 6.3.8: Modal of Study Hours Creation.

In Figure 6.3.8, it shows a modal that can allow system admins to set the study hour frame for specific classes that will be attached to this frame. The usage of the frame is to define the number of periods that the timetabling algorithm can work with. For example, if the 1st until the 10th period on Monday is defined as the study hour frame for Class 1A, then the subject blocks set in the subject distribution will only be allocated in the specified frame and not be overflowed.

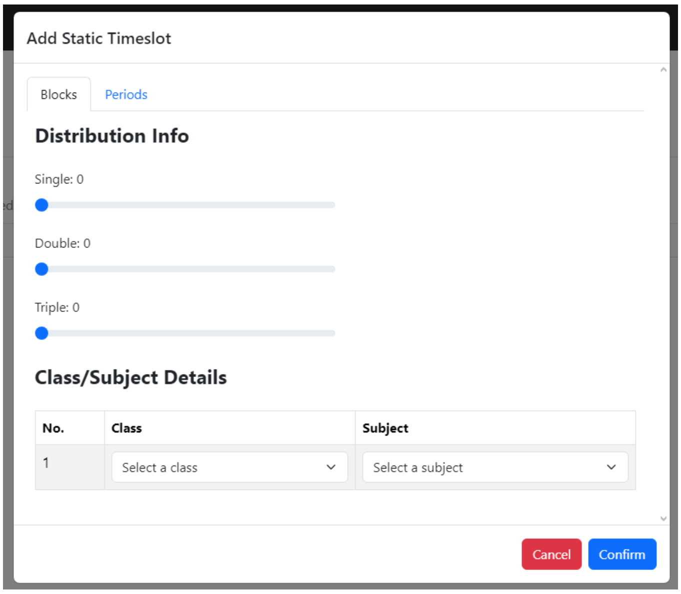


Static Timeslots

Create custom-defined timeslots with dedicated distribution, combined classes/subjects, and set periods.

No.	Class (Subject)	Single	Double	Triple	

Figure 6.3.9: Content of Static Timeslots Tab.



Add Static Timeslot

Distribution Info

Single: 0

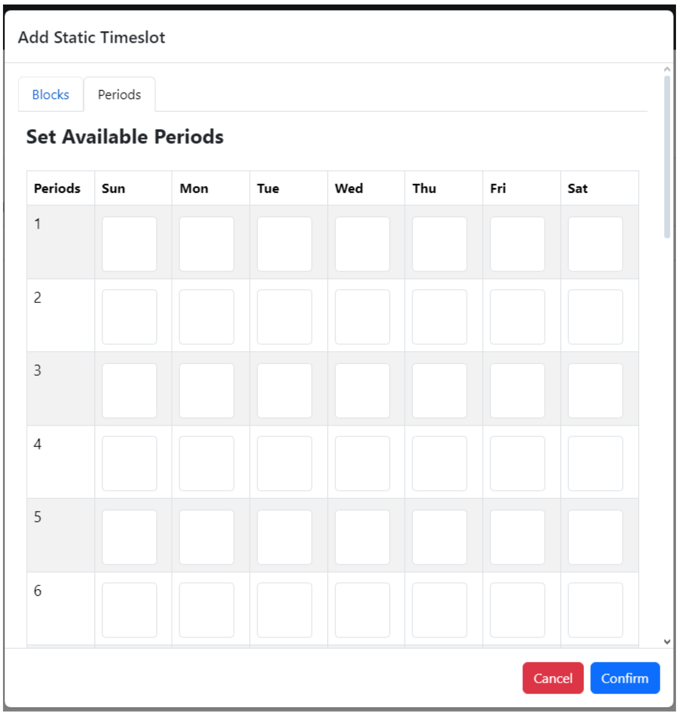
Double: 0

Triple: 0

Class/Subject Details

No.	Class	Subject
1	Select a class	Select a subject

Figure 6.3.10: Modal in Blocks Tab for Static Timeslots Creation.



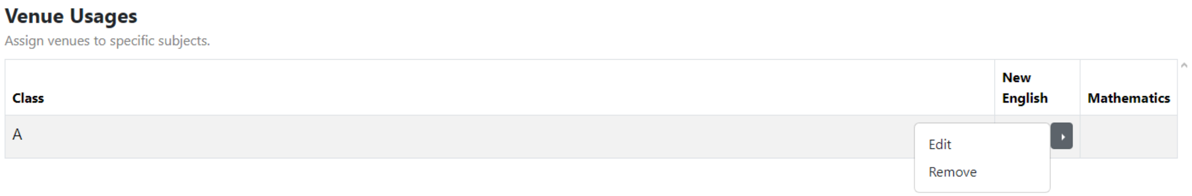
Add Static Timeslot

Set Available Periods

Periods	Sun	Mon	Tue	Wed	Thu	Fri	Sat
1							
2							
3							
4							
5							
6							

Figure 6.3.11: Modal in Periods Tab for Static Timeslots Creation.

Furthermore, it can be said that the Static Timeslots constraint is optional. This is because it will only define the periods containing subjects and classes that happen at the same time. For example, Class 1A and Class 1B may be having P.E concurrently with the same teacher. Not just that, Class 1A may have two different subjects taught at the same time with different teachers like Pendidikan Islam and Bahasa Cina. Then, this static timeslot will also have its own dedicated block distribution as stated in Figure 6.3.10. The system admins also can set the specific period that the static timeslot will be allocated in. This will consider scenario when P.E only will happen in the morning or in the evening.

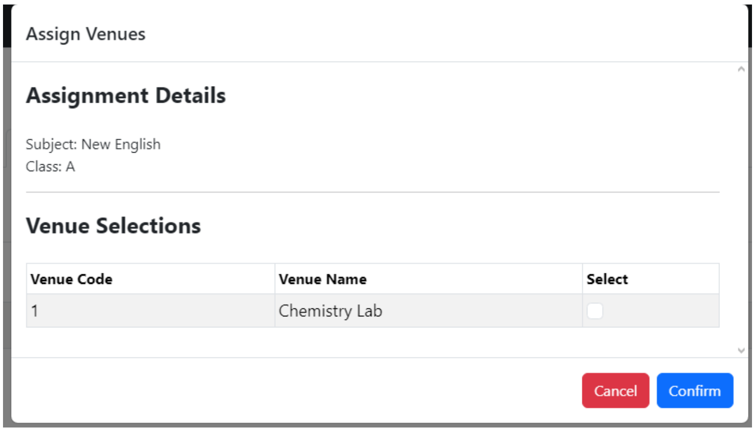


Venue Usages

Assign venues to specific subjects.

Class	New English	Mathematics
A		

Figure 6.3.12: Content of Venue Usages Tab.



Assign Venues

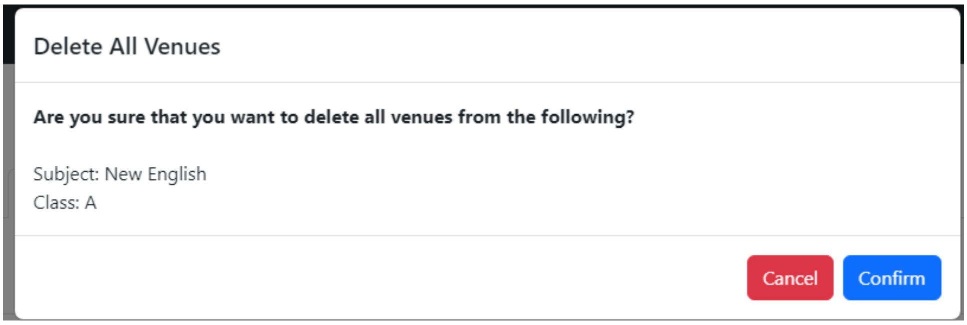
Assignment Details

Subject: New English
Class: A

Venue Selections

Venue Code	Venue Name	Select
1	Chemistry Lab	□

Figure 6.3.13: Modal to Set Venue Usages.



Delete All Venues

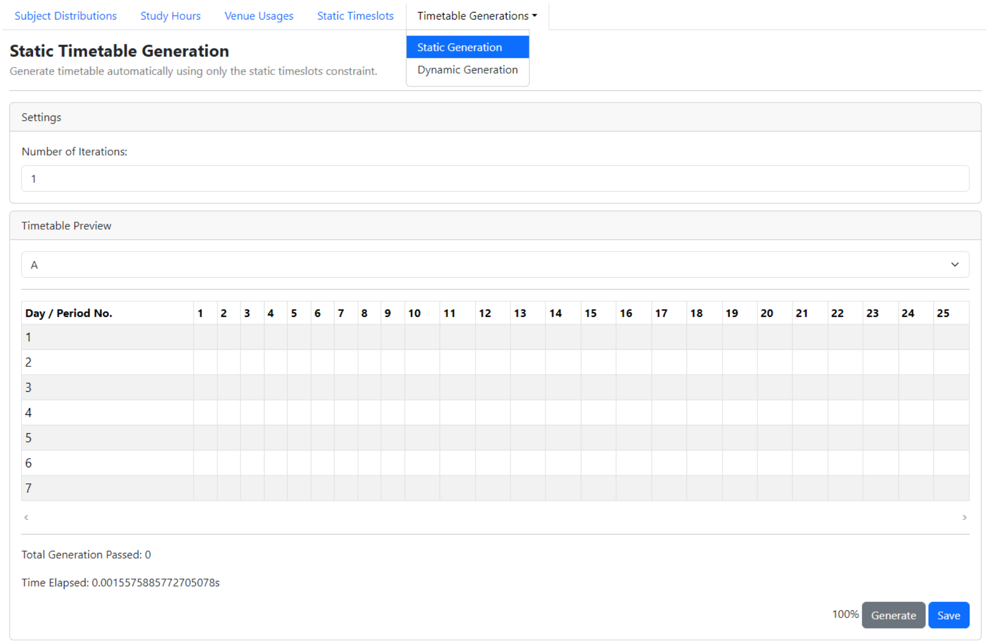
Are you sure that you want to delete all venues from the following?

Subject: New English
Class: A

Figure 6.3.14: Confirmation Modal before Deleting Venue Information.

On top that, Venue Usages constraint will decide whether a subject for class will have a venue. For instance, Chemistry subject will not be taught in class but in a chemistry lab. The unset venue usages for subjects will happen in class by default.

The figures below will show in the implementations in the Timetable Generations page.



Static Timetable Generation

Generate timetable automatically using only the static timeslots constraint.

Number of Iterations: 1

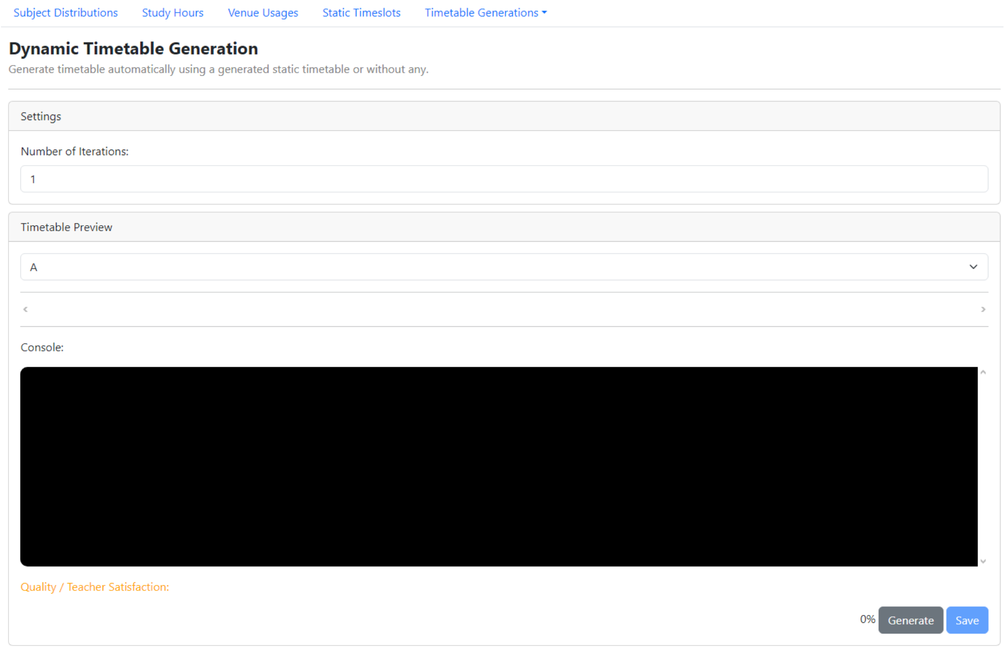
A

Day / Period No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
1																									
2																									
3																									
4																									
5																									
6																									
7																									

Total Generation Passed: 0

Time Elapsed: 0.0015575885772705078s

Figure 6.3.15: Static Generation Page.



Dynamic Timetable Generation

Generate timetable automatically using a generated static timetable or without any.

Number of Iterations: 1

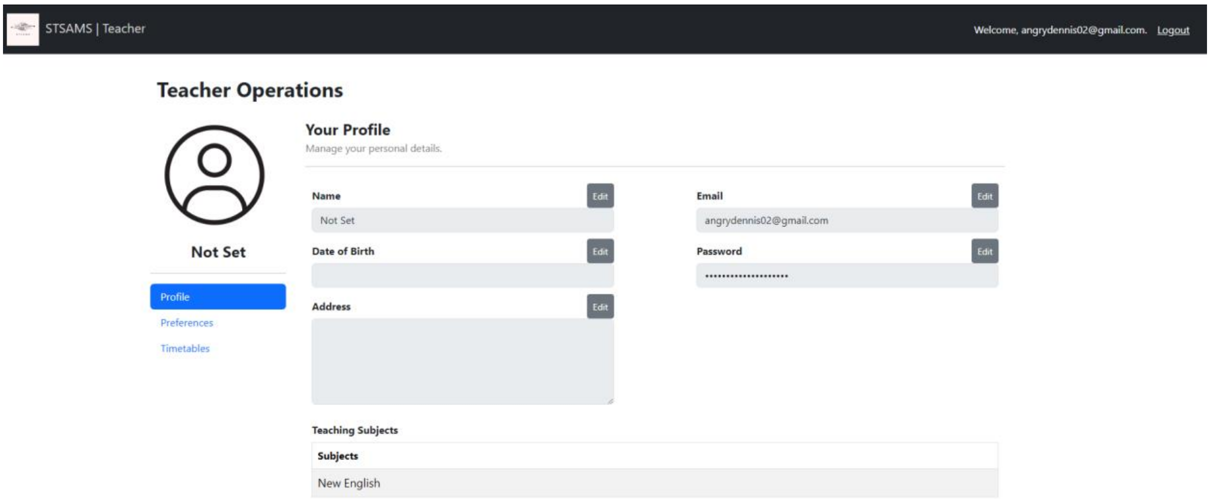
A

Quality / Teacher Satisfaction:

Figure 6.3.16: Dynamic Generation Page.

6.4 Teacher’s Module

In teacher’s module, teachers can view and modify their profile and set their preferences in timetable allocation.



Teacher Operations

Not Set

Your Profile

Manage your personal details.

Name: Not Set

Email: angrydennis02@gmail.com

Date of Birth:

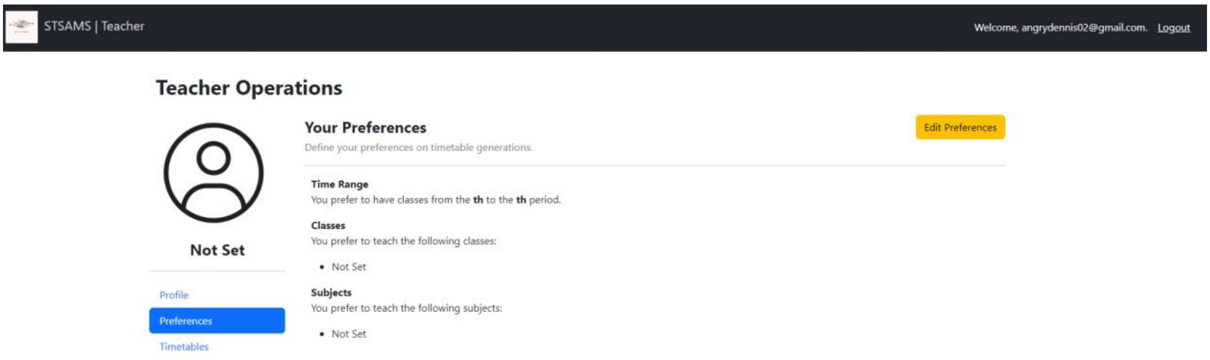
Password: ••••••••••••••••••••

Address:

Teaching Subjects

Subjects
New English

Figure 6.4.1: Teacher Profile Page.



Teacher Operations

Not Set

Your Preferences

Define your preferences on timetable generations.

Time Range

You prefer to have classes from the **th** to the **th** period.

Classes

You prefer to teach the following classes:

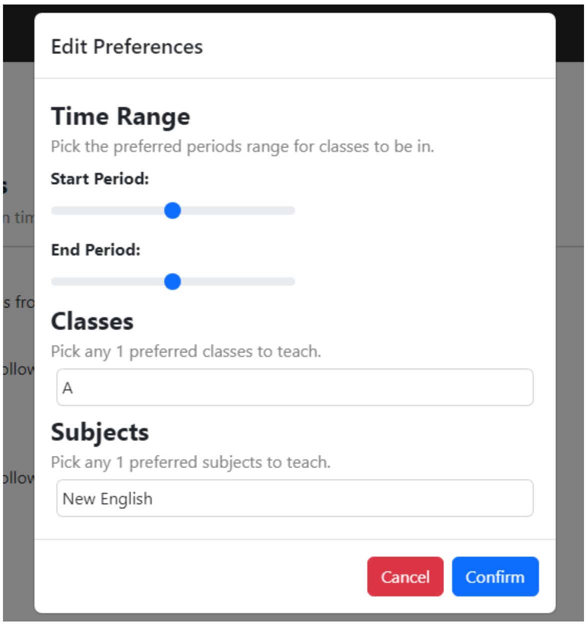
- Not Set

Subjects

You prefer to teach the following subjects:

- Not Set

Figure 6.4.2: Teacher Preferences Page.



Edit Preferences

Time Range

Pick the preferred periods range for classes to be in.

Start Period:

End Period:

Classes

Pick any 1 preferred classes to teach.

A

Subjects

Pick any 1 preferred subjects to teach.

New English

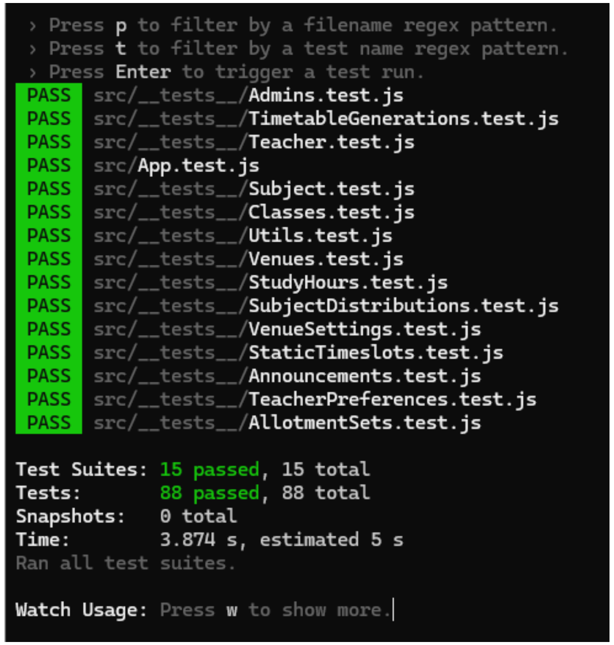
Figure 6.4.3: Modal to Edit Teacher Preferences.

CHAPTER 7

SYSTEM TESTING

In this chapter, we will be discussing the testing and verification done on STSAMS to achieve the testing objectives.

7.1 Test Cases and Results for Frontend Webpages



```
 › Press p to filter by a filename regex pattern.
 › Press t to filter by a test name regex pattern.
 › Press Enter to trigger a test run.
 PASS  src/__tests__/Admins.test.js
 PASS  src/__tests__/TimetableGenerations.test.js
 PASS  src/__tests__/Teacher.test.js
 PASS  src/App.test.js
 PASS  src/__tests__/Subject.test.js
 PASS  src/__tests__/Classes.test.js
 PASS  src/__tests__/Utils.test.js
 PASS  src/__tests__/Venues.test.js
 PASS  src/__tests__/StudyHours.test.js
 PASS  src/__tests__/SubjectDistributions.test.js
 PASS  src/__tests__/VenueSettings.test.js
 PASS  src/__tests__/StaticTimeslots.test.js
 PASS  src/__tests__/Announcements.test.js
 PASS  src/__tests__/TeacherPreferences.test.js
 PASS  src/__tests__/AllotmentSets.test.js

Test Suites: 15 passed, 15 total
Tests:       88 passed, 88 total
Snapshots:   0 total
Time:        3.874 s, estimated 5 s
Ran all test suites.

Watch Usage: Press w to show more.
```

Figure 7.1.1: Unit Test Overall Result using Jest.

Table 7.1.1: Test Cases and Results for Allotment Sets.

Project Name:	**STSAMS**	**Test Designed by:**	**Dennis Yap Jian Yuan**					
Module Name:	**Allotment Sets**	**Test Designed date:**	**15th April 2024**					
Release Version:	**1**	**Test Executed by:**	**Dennis Yap Jian Yuan**					
		Test Execution date:	**22nd April 2024**					
Pre-condition	Input value is valid.							
Dependencies:								
Test Priority	**Medium**							
Test Case#	**Test Title**	**Test Summary**	**Test Data**	**Expected Result**	**Post-condition**	**Actual Result**	**Status**	**Notes**
TC-01	Test postAllotmentSet method	Verify that the postAllotmentSet method sends request to the correct route & method and receives the response correctly.	{data: “example”}	Route: http://127.0.0.1:5000/sets Method: POST Response: {data: “example”}	Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/sets Method: POST Response: {data: “example”}	Pass	-
TC-02	Test getAllotmentSet method	Verify that the getAllotmentSet method sends	ID: 1	Route: http://127.0.0.1:5000/sets/1	Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/sets/1	Pass	-

		request to the correct route & method and receives the response correctly.		Method: GET Response: {data: “example”}		Method: GET Response: {data: “example”}		
TC-03	Test getAllotmentSets method	Verify that the getAllotmentSets method sends request to the correct route & method and receives the response correctly.	-	Route: http://127.0.0.1:5000/sets Method: GET Response: {data: “example”}	Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/sets Method: GET Response: {data: “example”}	Pass	-
TC-04	Test deleteAllotmentSets method	Verify that the deleteAllotmentSets method sends request to the correct route & method and receives the response correctly.	ID: 1	Route: http://127.0.0.1:5000/sets/1 Method: DELETE Response: {data: “example”}	Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/sets/1 Method: DELETE Response: {data: “example”}	Pass	-

Table 7.1.2: Test Cases and Results for Announcements.

Project Name:	**STSAMS**	**Test Designed by:**	**Dennis Yap Jian Yuan**					
Module Name:	**Announcements**	**Test Designed date:**	**15th April 2024**					
Release Version:	**1**	**Test Executed by:**	**Dennis Yap Jian Yuan**					
		Test Execution date:	**22nd April 2024**					
Pre-condition	Input value is valid.							
Dependencies:								
Test Priority	**Medium**							
Test Case#	**Test Title**	**Test Summary**	**Test Data**	**Expected Result**	**Post-condition**	**Actual Result**	**Status**	**Notes**
TC-05	Test postAnnouncement method	Verify that the postAnnouncement method sends request to the correct route & method and receives the response correctly.	{data: “example”}	Route: http://127.0.0.1:5000/announcements Method: POST Response: {data: “example”}	Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/announcements Method: POST Response: {data: “example”}	Pass	-
TC-06	Test getAnnouncements method	Verify that the getAnnouncement method sends request to the	-	Route: http://127.0.0.1:5000/announcements	Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/announcements	Pass	-

		correct route & method and receives the response correctly.		Method: GET Response: {data: “example”}		Method: GET Response: {data: “example”}		
TC-07	Test getAnnouncement method	Verify that the getAnnouncement method sends request to the correct route & method and receives the response correctly.	ID: 1	Route: http://127.0.0.1:5000/announcements/1 Method: GET Response: {data: “example”}	Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/announcements/1 Method: GET Response: {data: “example”}	Pass	-
TC-08	Test putAnnouncement method	Verify that the putAnnouncement method sends request to the correct route & method and receives the response correctly.	ID: 1 JSON: {data: “example”}	Route: http://127.0.0.1:5000/announcements/1 Method: PUT Response: {data: “example”}	Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/announcements/1 Method: PUT Response: {data: “example”}	Pass	-
TC-09	Test deleteAnnouncement method	Verify that the deleteAnnouncement method sends request to the correct route & method and receives the response correctly.	ID: 1	Route: http://127.0.0.1:5000/announcements/1 Method: DELETE Response: {data: “example”}	Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/announcements/1 Method: DELETE Response: {data: “example”}	Pass	-

Table 7.1.3: Test Cases and Results for Classes.

Project Name:	**STSAMS**	**Test Designed by:**	**Dennis Yap Jian Yuan**					
Module Name:	**Classes**	**Test Designed date:**	**15th April 2024**					
Release Version:	**1**	**Test Executed by:**	**Dennis Yap Jian Yuan**					
		Test Execution date:	**22nd April 2024**					
Pre-condition	Input value is valid.							
Dependencies:								
Test Priority	**Medium**							
Test Case#	**Test Title**	**Test Summary**	**Test Data**	**Expected Result**	**Post-condition**	**Actual Result**	**Status**	**Notes**

TC-10	Test postClass method	Verify that the postClass method sends request to the correct route & method and receives the response correctly.	{data: “example”}	Route: http://127.0.0.1:5000/classes
				Method: POST
				Body: {data: “example”}
				Response: {data: “example”}
					Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/classes
						Method: POST
						Body: {data: “example”}
						Response: {data: “example”}
							Pass	-

TC-11	Test getClasses method	Verify that the getClasses method sends request to the correct route & method and receives the response correctly.	-	Route: http://127.0.0.1:5000/classes Method: GET Response: {data: “example”}	Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/classes Method: GET Response: {data: “example”}	Pass	-
TC-12	Test getClass method	Verify that the getClass method sends request to the correct route & method and receives the response correctly.	ID: 1	Route: http://127.0.0.1:5000/classes/1 Method: GET Response: {data: “example”}	Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/classes/1 Method: GET Response: {data: “example”}	Pass	-

TC-13	Test putClass method	Verify that the putClass method sends request to the correct route & method and receives the response correctly.	ID: 1
			JSON: {data: “example”}
				Route: http://127.0.0.1:5000/classes/1
				Method: PUT
				Body: {data: “example”}
				Response: {data: “example”}
					Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/classes/1
						Method: PUT
						Body: {data: “example”}
						Response: {data: “example”}
							Pass	-

TC-14	Test deleteClass method	Verify that the deleteClass method sends request to the correct route & method and	ID: 1	Route: http://127.0.0.1:5000/classes/1 Method: DELETE	Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/classes/1 Method: DELETE	Pass	-

		receives the response correctly.		Response: {data: “example”}		Response: {data: “example”}		
TC-15	Test handleSorting method	Verify that the handleSorting method swtiches sorting modes correctly.	Name: “name” sortField: { name: “name”, Sorts.Ascending }	{ name: “name”, sort_type: “Sorts.Descending” }	Actual result is equals to Expected Result	{ name: “name”, sort_type: “Sorts.Descending” }	Pass	
TC-16	Test handleSorting method	Verify that the handleSorting method swtiches sorting modes correctly when column name changes.	Name: “other_name” sortField: { name: “name”, Sorts.Ascending }	{ name: “other_name”, sort_type: “Sorts.Ascending” }	Actual result is equals to Expected Result	{ name: “other_name”, sort_type: “Sorts.Ascending” }	Pass	
TC-17	Test handleFilterAndSort method	Verify that the handleFilterAndSort filter and sorts out the classes correctly.	Classes: [{name: “Class A”, id: 1}, {name: “Class B”, id: 2}] sortField: { name: “name”, Sorts.Descending }	[{name: “Class B”, id: 2}, {name: “Class A”, id: 1}]	Actual result is equals to Expected Result	[{name: “Class B”, id: 2}, {name: “Class A”, id: 1}]	Pass	

Table 7.1.4: Test Cases and Results for Static Timeslots.

Project Name:	**STSAMS**	**Test Designed by:**	**Dennis Yap Jian Yuan**
Module Name:	**Static Timeslots**	**Test Designed date:**	**15th April 2024**
Release Version:	**1**	**Test Executed by:**	**Dennis Yap Jian Yuan**
		Test Execution date:	**22nd April 2024**

Pre-condition	Input value is valid.
Dependencies:	
Test Priority	**Medium**

Test Case#	**Test Title**	**Test Summary**	**Test Data**	**Expected Result**	**Post-condition**	**Actual Result**	**Status**	**Notes**

TC-18	Test postStaticTimeslot method	Verify that the postStaticTimeslot method sends request to the correct route & method and receives the response correctly.	{data: “example”}	Route: http://127.0.0.1:5000/static_timeslots
				Method: POST
				Body: {data: “example”}
				Response: {data: “example”}
					Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/static_timeslots
						Method: POST
						Body: {data: “example”}
						Response: {data: “example”}
							Pass	-

TC-19	Test getStaticTimeslots method	Verify that the getClasses method sends request to the correct route & method and receives the response correctly.	-	Route: http://127.0.0.1:5000/static_timeslots Method: GET Response: {data: “example”}	Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/static_timeslots Method: GET Response: {data: “example”}	Pass	-
TC-20	Test getStaticTimeslot method	Verify that the getStaticTimeslot method sends request to the correct route & method and receives the response correctly.	ID: 1	Route: http://127.0.0.1:5000/static_timeslots/1 Method: GET Response: {data: “example”}	Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/static_timeslots/1 Method: GET Response: {data: “example”}	Pass	-

TC-21	Test putStaticTimeslot method	Verify that the putStaticTimeslot method sends request to the correct route & method and receives the response correctly.	ID: 1
			JSON: {data: “example”}
				Route: http://127.0.0.1:5000/static_timeslots/1
				Method: PUT
				Body: {data: “example”}
				Response: {data: “example”}
					Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/static_timeslots/1
						Method: PUT
						Body: {data: “example”}
						Response: {data: “example”}
							Pass	-

TC-22	Test deleteStaticTimeslot method	Verify that the deleteStaticTimeslot method sends request to the correct route & method and receives the response correctly.	ID: 1	Route: http://127.0.0.1:5000/static_timeslots/1 Method: DELETE Response: {data: "example"}	Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/static_timeslots/1 Method: DELETE Response: {data: "example"}	Pass	-

Table 7.1.5: Test Cases and Results for Study Hours.

Project Name:	**STSAMS**	**Test Designed by:**	**Dennis Yap Jian Yuan**					
Module Name:	**Study Hours**	**Test Designed date:**	**15th April 2024**					
Release Version:	**1**	**Test Executed by:**	**Dennis Yap Jian Yuan**					
		Test Execution date:	**22nd April 2024**					
Pre-condition	Input value is valid.							
Dependencies:								
Test Priority	**Medium**							
Test Case#	**Test Title**	**Test Summary**	**Test Data**	**Expected Result**	**Post-condition**	**Actual Result**	**Status**	**Notes**

TC-23	Test postStudyHours method	Verify that the postStudyHours method sends request to the correct route & method and receives the response correctly.	{data: “example”}	Route: http://127.0.0.1:5000/study_hours
				Method: POST
				Body: {data: “example”}
				Response: {data: “example”}
					Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/study_hours
						Method: POST
						Body: {data: “example”}
						Response: {data: “example”}
							Pass	-

TC-24	Test getStudyHours method	Verify that the getStudyHours method sends request to the correct route & method and receives the response correctly.	-	Route: http://127.0.0.1:5000/study_hours Method: GET Response: {data: “example”}	Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/study_hours Method: GET Response: {data: “example”}	Pass	-

TC-25	Test putStudyHour method	Verify that the putStudyHour method sends request to the correct route & method and receives the response correctly.	ID: 1
			JSON: {data: “example”}
				Route: http://127.0.0.1:5000/study_hours/1
				Method: PUT
				Body: {data: “example”}
				Response: {data: “example”}
					Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/study_hours/1
						Method: PUT
						Body: {data: “example”}
						Response: {data: “example”}
							Pass	-

TC-26	Test deleteStudyHour method	Verify that the deleteStudyHour method sends request to the correct route & method and receives the response correctly.	ID: 1	Route: http://127.0.0.1:5000/study_hours/1 Method: DELETE Response: {data: “example”}	Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/study_hours/1 Method: DELETE Response: {data: “example”}	Pass	-

Table 7.1.6: Test Cases and Results for Subjects.

Project Name:	**STSAMS**	**Test Designed by:**	**Dennis Yap Jian Yuan**					
Module Name:	**Subjects**	**Test Designed date:**	**15th April 2024**					
Release Version:	**1**	**Test Executed by:**	**Dennis Yap Jian Yuan**					
		Test Execution date:	**22nd April 2024**					
Pre-condition	Input value is valid.							
Dependencies:								
Test Priority	**Medium**							
Test Case#	**Test Title**	**Test Summary**	**Test Data**	**Expected Result**	**Post-condition**	**Actual Result**	**Status**	**Notes**

TC-27	Test postSubject method	Verify that the postSubject method sends request to the correct route & method and receives the response correctly.	{data: “example”}	Route: http://127.0.0.1:5000/subjects
				Method: POST
				Body: {data: “example”}
				Response: {data: “example”}
					Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/subjects
						Method: POST
						Body: {data: “example”}
						Response: {data: “example”}
							Pass	-

TC-28	Test getSubjects method	Verify that the getSubjects method sends request to the correct route & method and receives the response correctly.	-	Route: http://127.0.0.1:5000/subjects Method: GET Response: {data: “example”}	Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/subjects Method: GET Response: {data: “example”}	Pass	-
TC-29	Test getSubject method	Verify that the getSubject method sends request to the correct route & method and receives the response correctly.	ID: 1	Route: http://127.0.0.1:5000/subjects/1 Method: GET Response: {data: “example”}	Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/subjects/1 Method: GET Response: {data: “example”}	Pass	-

TC-30	Test putSubject method	Verify that the putSubject method sends request to the correct route & method and receives the response correctly.	ID: 1
			JSON: {data: “example”}
				Route: http://127.0.0.1:5000/subjects/1
				Method: PUT
				Body: {data: “example”}
				Response: {data: “example”}
					Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/subjects/1
						Method: PUT
						Body: {data: “example”}
						Response: {data: “example”}
							Pass	-

TC-31	Test deleteSubject method	Verify that the deleteSubject method sends request to the correct route & method and	ID: 1	Route: http://127.0.0.1:5000/subjects/1 Method: DELETE	Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/subjects/1 Method: DELETE	Pass	-

		receives the response correctly.		Response: {data: “example”}		Response: {data: “example”}		

TC-32	Test handleSorting method	Verify that the handleSorting method swtiches sorting modes correctly.	1)
			Name: “sub_name”
			sortField: { name: “sub_name”, Sorts.Ascending }
			2)
			Name: “sub_code”
			sortField: { name: “sub_code”, Sorts.Descending }
			3)
			Name: “sub_type”
			sortField: { name: “sub_type”, Sorts.Unsorted }
				1)
				{ name: “sub_name”, sort_type: “Sorts.Descending” }
				2)
				{ name: “sub_code”, sort_type: “Sorts.Unsorted” }
				3)
				{ name: “sub_type”, sort_type: “Sorts.Ascending” }
					Actual result is equals to Expected Result	1)
						{ name: “sub_name”, sort_type: “Sorts.Descending” }
						2)
						{ name: “sub_code”, sort_type: “Sorts.Unsorted” }
						3)
						{ name: “sub_type”, sort_type: “Sorts.Ascending” }
							Pass	

TC-33	Test handleSorting method	Verify that the handleSorting method swtiches sorting modes correctly when column name changes.	1) Name: “sub_name” sortField: { name: “sub_code”, Sorts.Descending } 2) Name: “sub_code”	1) { name: “sub_name”, sort_type: “Sorts.Ascending” } 2)	Actual result is equals to Expected Result	1) { name: “sub_name”, sort_type: “Sorts.Ascending” } 2)	Pass	

			sortField: { name: “sub_name”, Sorts.Unsorted }	{ name: “sub_code”, sort_type: “Sorts.Ascending” }		{ name: “sub_code”, sort_type: “Sorts.Ascending” }		
TC-34	Test handleFilterAndSort method	Verify that the handleFilterAndSort filter and sorts out the classes correctly.	1) [[{sub_id: 1, sub_name: "Math", sub_code: "MM", sub_type: "heavy"}], "Mathh", "heavy", "sub_name"] 2) [[{sub_id: 2, sub_name: "Geografi", sub_code: "GEO", sub_type: "light"}], "gr", "light", "sub_name"] 3) [[{sub_id: 1, sub_name: "Math", sub_code: "MM", sub_type: "heavy"}], "", "light", "sub_name"]	1)[] 2) [{sub_id: 2, sub_name: "Geografi", sub_code: "GEO", sub_type: "light"}] 3)[]	Actual result is equals to Expected Result	1)[] 2) [{sub_id: 2, sub_name: "Geografi", sub_code: "GEO", sub_type: "light"}] 3)[]	Pass	

			4) [[{sub_id: 2, sub_name: "Geografi", sub_code: "GEO", sub_type: "light"}], "", "light", "sub_name"] 5) [[{sub_id: 1, sub_name: "Math", sub_code: "MM", sub_type: "heavy"}, {sub_id: 2, sub_name: "Geografi", sub_code: "GEO", sub_type: "heavy"}], "", "heavy", {name: "sub_name", sort_type: Sorts.Ascending}] 6) [[{sub_id: 3, sub_name: "Bahasa Melayu", sub_code: "BM", sub_type: "light"}, {sub_id: 4,	4) [{sub_id: 2, sub_name: "Geografi", sub_code: "GEO", sub_type: "light"}] 5) [{sub_id: 2, sub_name: "Geografi", sub_code: "GEO", sub_type: "heavy"}, {sub_id: 1, sub_name: "Math", sub_code: "MM", sub_type: "heavy"}] 6) [{sub_id: 3, sub_name: "Bahasa Melayu", sub_code: "BM", sub_type: "light"}, {sub_id: 4, sub_name: "Geografi", sub_code:		4) [{sub_id: 2, sub_name: "Geografi", sub_code: "GEO", sub_type: "light"}] 5) [{sub_id: 2, sub_name: "Geografi", sub_code: "GEO", sub_type: "heavy"}, {sub_id: 1, sub_name: "Math", sub_code: "MM", sub_type: "heavy"}] 6) [{sub_id: 3, sub_name: "Bahasa Melayu", sub_code: "BM", sub_type: "light"}, {sub_id: 4, sub_name: "Geografi", sub_code:		

			sub_name: "Geografi", sub_code: "GEO", sub_type: "light"}], "", "light", {name: "sub_code", sort_type: Sorts.Ascending}]	"GEO", sub_type: "light"}]		"GEO", sub_type: "light"}]		

Table 7.1.7: Test Cases and Results for Subject Distributions.

Project Name:	**STSAMS**	**Test Designed by:**	**Dennis Yap Jian Yuan**
Module Name:	**Subject Distributions**	**Test Designed date:**	**15th April 2024**
Release Version:	**1**	**Test Executed by:**	**Dennis Yap Jian Yuan**
		Test Execution date:	**22nd April 2024**

Pre-condition	Input value is valid.
Dependencies:	
Test Priority	**Medium**

Test Case#	**Test Title**	**Test Summary**	**Test Data**	**Expected Result**	**Post-condition**	**Actual Result**	**Status**	**Notes**

TC-35	Test postDistribution method	Verify that the postDistribution method sends request to the correct route & method and receives the response correctly.	{data: “example”}	Route: http://127.0.0.1:5000/distributions
				Method: POST
				Body: {data: “example”}
				Response: {data: “example”}
					Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/distributions
						Method: POST
						Body: {data: “example”}
						Response: {data: “example”}
							Pass	-

TC-36	Test getDistributions method	Verify that the getDistributions method sends request to the correct route & method and receives the response correctly.	-	Route: http://127.0.0.1:5000/distributions Method: GET Response: {data: “example”}	Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/distributions Method: GET Response: {data: “example”}	Pass	-	

TC-37	Test putDistribution method	Verify that the putDistribution method sends request to the correct route & method and receives the response correctly.	ID: 1
			JSON: {data: “example”}
				Route: http://127.0.0.1:5000/distributions/1
				Method: PUT
				Body: {data: “example”}
				Response: {data: “example”}
					Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/distributions/1
						Method: PUT
						Body: {data: “example”}
						Response: {data: “example”}
							Pass	-	

TC-38	Test deleteDistribution method	Verify that the deleteDistribution method sends request to the correct route & method and receives the response correctly.	ID: 1	Route: http://127.0.0.1:5000/distributions/1 Method: DELETE Response: {data: “example”}	Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/distributions/1 Method: DELETE Response: {data: “example”}	Pass	-	
TC-39	Test validateInputData method	Verify that the validateInputData method check if name, subjects,	1) Name: “Test Name”	1) {name: false, blocks: true, classes: true}	Actual result is equals to Expected Result	1) {name: false, blocks: true, classes: true}	Pass	-	

		and classes are empty correctly.	Subjects: [] Classes: [] 2) Name: “”, Subjects: [] Classes: []	2) {name: true, blocks: true, classes: true}		2) {name: true, blocks: true, classes: true}		

Table 7.1.8: Test Cases and Results for Teachers.

Project Name:	**STSAMS**	**Test Designed by:**	**Dennis Yap Jian Yuan**					
Module Name:	**Teachers**	**Test Designed date:**	**15th April 2024**					
Release Version:	**1**	**Test Executed by:**	**Dennis Yap Jian Yuan**					
		Test Execution date:	**22nd April 2024**					
Pre-condition	Input value is valid.							
Dependencies:								
Test Priority	**Medium**							
Test Case#	**Test Title**	**Test Summary**	**Test Data**	**Expected Result**	**Post-condition**	**Actual Result**	**Status**	**Notes**

TC-40	Test postTeacher method	Verify that the postTeacher method sends request to the correct route & method and receives the response correctly.	{data: “example”}	Route: http://127.0.0.1:5000/teachers
				Method: POST
				Body: {data: “example”}
				Response: {data: “example”}
					Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/teachers
						Method: POST
						Body: {data: “example”}
						Response: {data: “example”}
							Pass	-

TC-41	Test postEmail method	Verify that the postEmail method sends request to the correct route & method and receives the response correctly.	{data: “example”}	Route: http://127.0.0.1:4000/email/send
				Method: POST
				Body: {data: “example”}
				Response: {data: “example”}
					Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/email/send
						Method: POST
						Body: {data: “example”}
						Response: {data: “example”}
							Pass	-

TC-42	Test getTeacher method	Verify that the getTeacher method sends request to the correct route & method and receives the response correctly.	Access Token: access_token	Route: http://127.0.0.1:5000/teachers/get-started/access_token Method: GET Response: {data: “example”}	Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/teachers/get-started/access_token Method: GET Response: {data: “example”}	Pass	-

TC-43	Test putTeacher method	Verify that the putTeacher method sends request to the correct route & method and receives the response correctly.	ID: 1
			JSON: {data: “example”}
				Route: http://127.0.0.1:5000/teachers/1
				Method: PUT
				Body: {data: “example”}
				Response: {data: “example”}
					Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/teachers/1
						Method: PUT
						Body: {data: “example”}
						Response: {data: “example”}
							Pass	-

TC-44	Test getTeachers method	Verify that the getTeachers method sends request to the correct route & method and receives the response correctly.	-	Route: http://127.0.0.1:5000/teachers Method: GET Response: {data: “example”}	Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/teachers Method: GET Response: {data: “example”}	Pass	-
TC-45	Test getTeacherDetails method	Verify that the getTeacherDetails method sends request to the correct route & method and receives the response correctly.	ID: 1	Route: http://127.0.0.1:5000/teachers/1 Method: GET Response: {data: “example”}	Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/teachers/1 Method: GET Response: {data: “example”}	Pass	-
TC-46	Test getTeachingSubjects method	Verify that the getTeachingSubjects method sends request to the correct route & method and receives the response correctly.	ID: 1	Route: http://127.0.0.1:5000/teaching-subjects/1 Method: GET Response: {data: “example”}	Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/teaching-subjects/1 Method: GET Response: {data: “example”}	Pass	-
TC-47	Test putTeachingSubjects method	Verify that the putTeachingSubjects method sends request to the correct route & method and	ID: 1 JSON: {data: “example”}	Route: http://127.0.0.1:5000/teaching-subjects/1 Method: PUT	Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/teaching-subjects/1 Method: PUT	Pass	-

		receives the response correctly.		Body: {data: “example”}
				Response: {data: “example”}
						Body: {data: “example”}
						Response: {data: “example”}		

TC-48	Test deleteTeacher method	Verify that the deleteTeacher method sends request to the correct route & method and receives the response correctly.	ID: 1	Route: http://127.0.0.1:5000/teachers/1 Method: DELETE Response: {data: “example”}	Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/teachers/1 Method: DELETE Response: {data: “example”}	Pass	-

TC-49	Test teacherLogin method	Verify that the teacherLogin method sends request to the correct route & method and receives the response correctly.	{username: “example_username”, password: “example_password}	Route: http://127.0.0.1:5000/teachers/login
				Method: POST
				Body: {username: “example_username”, password: “example_password}
				Response: {data: “example”}
					Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/teachers/login
						Method: POST
						Body: {username: “example_username”, password: “example_password}
						Response: {data: “example”}
							Pass	-

Table 7.1.9: Test Cases and Results for Teacher Preferences.

Project Name:	**STSAMS**	**Test Designed by:**	**Dennis Yap Jian Yuan**					
Module Name:	**Teacher Preferences**	**Test Designed date:**	**15th April 2024**					
Release Version:	**1**	**Test Executed by:**	**Dennis Yap Jian Yuan**					
		Test Execution date:	**22nd April 2024**					
Pre-condition	Input value is valid.							
Dependencies:								
Test Priority	**Medium**							
Test Case#	**Test Title**	**Test Summary**	**Test Data**	**Expected Result**	**Post-condition**	**Actual Result**	**Status**	**Notes**
TC-50	Test getPreferences method	Verify that the getPreferences method sends request to the correct route & method and receives the response correctly.	ID: 123	Route: http://127.0.0.1:5000/preferences/123 Method: GET Response: {data: “example”}	Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/preferences/123 Method: GET Response: {data: “example”}	Pass	-
TC-51	Test putPreferences method	Verify that the putPreferences method sends	ID: 123	Route: http://127.0.0.1:5000/preferences/123	Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/preferences/123	Pass	-

		request to the correct route & method and receives the response correctly.	JSON: {data: “example”}	Method: PUT
				Body: {data: “example”}
				Response: {data: “example”}
						Method: PUT
						Body: {data: “example”}
						Response: {data: “example”}		

TC-52	Test getTeachingClasses method	Verify that the getTeachingClasses method sends request to the correct route & method and receives the response correctly.	ID: 123	Route: http://127.0.0.1:5000/teaching_classes/123 Method: GET Response: {data: “example”}	Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/teaching_classes/123 Method: GET Response: {data: “example”}	Pass	-
TC-53	Test getOrdinal method	Verify that the getOrdinal method will return the ordinal of an inputted integer correctly.	1) Number: 1 2) Number: 2 3) Number: 3 4) Number: 4 5) Number: 11 6)	1) Ordinal: “st” 2) Ordinal: “nd” 3) Ordinal: “rd” 4) Ordinal: “th” 5) Ordinal: “th” 6)	Actual result is equals to Expected Result	1) Ordinal: “st” 2) Ordinal: “nd” 3) Ordinal: “rd” 4) Ordinal: “th” 5) Ordinal: “th” 6)	Pass	-

			Number: 12 7) Number: 13	Ordinal: “th” 7) Ordinal: “th”		Ordinal: “th” 7) Ordinal: “th”		

Table 7.1.10: Test Cases and Results for Timetable Generation.

Project Name:	**STSAMS**	**Test Designed by:**	**Dennis Yap Jian Yuan**					
Module Name:	**Timetable Generation**	**Test Designed date:**	**15th April 2024**					
Release Version:	**1**	**Test Executed by:**	**Dennis Yap Jian Yuan**					
		Test Execution date:	**22nd April 2024**					
Pre-condition	Input value is valid.							
Dependencies:								
Test Priority	**Medium**							
Test Case#	**Test Title**	**Test Summary**	**Test Data**	**Expected Result**	**Post-condition**	**Actual Result**	**Status**	**Notes**

TC-54	Test generateStaticTimetable method	Verify that the generateStaticTimetable method sends request to the correct route & method and receives the response correctly.	{data: “example”}	Route: http://127.0.0.1:5000/static_timetabling_algorithm/run
				Method: POST
				Body: {data: “example”}
				Response: {data: “example”}
					Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/static_timetabling_algorithm/run
						Method: POST
						Body: {data: “example”}
						Response: {data: “example”}
							Pass	-

TC-55	Test getStaticProgress method	Verify that the getStaticProgress method sends request to the correct route & method and receives the response correctly.	-	Route: http://127.0.0.1:5000/static_timetabling_algorithm/progress Method: GET Response: {data: “example”}	Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/static_timetabling_algorithm/progress Method: GET Response: {data: “example”}	Pass	-

TC-56	Test postStaticTimetable method	Verify that the postStaticTimetable method sends request to the correct route & method and receives the response correctly.	{data: “example”}	Route: http://127.0.0.1:5000/static_timetables
				Method: POST
				Body: {data: “example”}
				Response: {data: “example”}
					Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/static_timetables
						Method: POST
						Body: {data: “example”}
						Response: {data: “example”}
							Pass	-

TC-57	Test generateDynamicTimetable method	Verify that the generateDynamicTimetable method sends request to the correct route & method and receives the response correctly.	{data: “example”}	Route: http://127.0.0.1:5000/dynamic_timetabling_algorithm/run
				Method: POST
				Body: {data: “example”}
				Response: {data: “example”}
					Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/dynamic_timetabling_algorithm/run
						Method: POST
						Body: {data: “example”}
						Response: {data: “example”}
							Pass	-

TC-58	Test getDynamicProgress method	Verify that the getDynamicProgress method sends request to the correct route & method and receives the response correctly.	-	Route: http://127.0.0.1:5000/dynamic_timetabling_algorithm/progress Method: GET Response: {data: “example”}	Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/dynamic_timetabling_algorithm/progress Method: GET Response: {data: “example”}	Pass	-

TC-59	Test postDynamicTimetable method	Verify that the postDynamicTimetable method sends request to the correct route & method and receives the response correctly.	{data: “example”}	Route: http://127.0.0.1:5000/dynamic_timetables
				Method: POST
				Body: {data: “example”}
				Response: {data: “example”}
					Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/dynamic_timetables
						Method: POST
						Body: {data: “example”}
						Response: {data: “example”}
							Pass	-

TC-60	Test getStaticTimetables method	Verify that the getStaticTimetables method sends request to the correct route & method and receives the response correctly.	ID: 1	Route: http://127.0.0.1:5000/static_timetables/1 Method: GET Response: {data: “example”}	Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/static_timetables/1 Method: GET Response: {data: “example”}	Pass	-
TC-61	Test getConsoleMessages method	Verify that the getConsoleMessages method sends request to the correct route &	-	Route: http://127.0.0.1:5000/dynamic_timetabling_algorithm/console_messages	Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/dynamic_timetabling_algorithm/console_messages	Pass	-

		method and receives the response correctly.		Method: GET Response: {data: “example”}		Method: GET Response: {data: “example”}		

Table 7.1.11: Test Cases and Results for Utils Function.

Project Name:	**STSAMS**	**Test Designed by:**	**Dennis Yap Jian Yuan**					
Module Name:	**Utils Functions**	**Test Designed date:**	**15th April 2024**					
Release Version:	**1**	**Test Executed by:**	**Dennis Yap Jian Yuan**					
		Test Execution date:	**22nd April 2024**					
Pre-condition	Input value is valid.							
Dependencies:								
Test Priority	**Medium**							
Test Case#	**Test Title**	**Test Summary**	**Test Data**	**Expected Result**	**Post-condition**	**Actual Result**	**Status**	**Notes**
TC-62	Test generateDigitCode method	Verify that the generateDigitCode method will generate 6-digit random code correctly.	1,2,3,4,5,6	Code Length: 6 Generated Code: 123456	Actual result is equals to Expected Result	Code Length: 6 Generated Code: 123456	Pass	-
TC-63	Test generatePassword method	Verify that the generatePassword method generate random password	1) Length: 20 2)	1)Generated Password Length: 20 2) Generated Password Length: 10	Actual result is equals to Expected Result	1)Generated Password Length: 20 2) Generated Password Length: 10	Pass	-

		with the correct length.	Length: 10 3) Length: 5	3) Generate Password Length: 5		3) Generate Password Length: 5		
TC-64	Test generatePassword method	Verify that the generatePassword method generate random password with the correct formatting.	1) Regex: /[a-z]/ 2) Regex: /[A-Z]/ 3) Regex: /[!$*+-]/	1) Matched: True 2) Matched: True 3) Matched: True	Actual result is equals to Expected Result	1) Matched: True 2) Matched: True 3) Matched: True	Pass	-
TC-65	Test generatePassword method	Verify that the generatePassword method generate random password each time.	Password1: generatePassword() Password2: generatedPassword ()	Password1 != Password2	Actual result is equals to Expected Result	Password1 != Password2	Pass	-
TC-66	Test randomNumberIn Range method	Verify that the randomNumberIn Range method will value between specified min and max integer correctly.	1) Min: 0, Max: 10 2) Min: -42, Max:-1 3) Min: 23, Max: 60	Random Number > Min Integer: True Random Number < Max Integer: True	Actual result is equals to Expected Result	Random Number > Min Integer: True Random Number < Max Integer: True	Pass	-

Table 7.1.12: Test Cases and Results for Venues.

Project Name:	**STSAMS**	**Test Designed by:**	**Dennis Yap Jian Yuan**
Module Name:	**Venues**	**Test Designed date:**	**15th April 2024**
Release Version:	**1**	**Test Executed by:**	**Dennis Yap Jian Yuan**
		Test Execution date:	**22nd April 2024**

Pre-condition	Input value is valid.
Dependencies:	
Test Priority	**Medium**

Test Case#	**Test Title**	**Test Summary**	**Test Data**	**Expected Result**	**Post-condition**	**Actual Result**	**Status**	**Notes**

TC-67	Test postVenue method	Verify that the postVenue method sends request to the correct route & method and receives the response correctly.	{data: “example”}	Route: http://127.0.0.1:5000/venues
				Method: POST
				Body: {data: “example”}
				Response: {data: “example”}
					Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/venues
						Method: POST
						Body: {data: “example”}
						Response: {data: “example”}
							Pass	-

TC-68	Test getVenues method	Verify that the getVenues method sends request to the correct route & method and receives the response correctly.	-	Route: http://127.0.0.1:5000/venues Method: GET Response: [{ id: 1, venue_name: 'Venue A' }, { id: 2, venue_name: 'Venue B' }]	Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/venues Method: GET Response: [{ id: 1, venue_name: 'Venue A' }, { id: 2, venue_name: 'Venue B' }]	Pass	-
TC-69	Test getVenue method	Verify that the getVenue method sends request to the correct route & method and receives the response correctly.	ID: 1	Route: http://127.0.0.1:5000/venues/1 Method: GET Response: {data: “example”}	Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/venues/1 Method: GET Response: {data: “example”}	Pass	-

TC-70	Test putVenue method	Verify that the putClass method sends request to the correct route & method and receives the response correctly.	ID: 1
			JSON: {data: “example”}
				Route: http://127.0.0.1:5000/venues/1
				Method: PUT
				Body: {data: “example”}
				Response: {data: “example”}
					Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/venues/1
						Method: PUT
						Body: {data: “example”}
						Response: {data: “example”}
							Pass	-

TC-71	Test deleteVenue method	Verify that the deleteVenue method sends	ID: 1	Route: http://127.0.0.1:5000/venues/1	Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/ venues /1	Pass	-

		request to the correct route & method and receives the response correctly.		Method: DELETE Response: {data: “example”}		Method: DELETE Response: {data: “example”}		
TC-72	Test handleSorting method	Verify that the handleSorting method swtiches sorting modes correctly.	Name: “venue_name” sortField: { name: “venue_name”, Sorts.Ascending }	{ name: “venue_name”, sort_type: “Sorts.Descending” }	Actual result is equals to Expected Result	{ name: “venue_name”, sort_type: “Sorts.Descending” }	Pass	
TC-73	Test handleFilterAndSort method	Verify that the handleFilterAndSort filter and sorts out the venues correctly.	Venues:[{ venue_name: 'Venue A' },{ venue_name: 'Venue B' },{ venue_name: 'Venue C' }] sortField: { name: “venue_name”, Sorts.Descending }	[{ venue_name: 'Venue C' },{ venue_name: 'Venue B' },{ venue_name: 'Venue A' }]	Actual result is equals to Expected Result	[{ venue_name: 'Venue C' },{ venue_name: 'Venue B' },{ venue_name: 'Venue A' }]	Pass	

Table 7.1.13: Test Cases and Results for Venue Settings.

Project Name:	**STSAMS**	**Test Designed by:**	**Dennis Yap Jian Yuan**					
Module Name:	**Venue Settings**	**Test Designed date:**	**15th April 2024**					
Release Version:	**1**	**Test Executed by:**	**Dennis Yap Jian Yuan**					
		Test Execution date:	**22nd April 2024**					
Pre-condition	Input value is valid.							
Dependencies:								
Test Priority	**Medium**							
Test Case#	**Test Title**	**Test Summary**	**Test Data**	**Expected Result**	**Post-condition**	**Actual Result**	**Status**	**Notes**
TC-74	Test getVenueUsages method	Verify that the getPreferences method sends request to the correct route & method and receives the response correctly.	ID: 1	Route: http://127.0.0.1:5000/venue_usages/1 Method: GET Response: {data: “example”}	Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/venue_usages/1 Method: GET Response: {data: “example”}	Pass	-
TC-75	Test putVenueUsages method	Verify that the putVenueUsages method sends	ID: 1	Route: http://127.0.0.1:5000/venue_usages/1	Actual result is equals to Expected Result	Route: http://127.0.0.1:5000/venue_usages/1	Pass	-

		request to the correct route & method and receives the response correctly.	JSON: {data: “example”}	Method: PUT
				Body: {data: “example”}
				Response: {data: “example”}
						Method: PUT
						Body: {data: “example”}
						Response: {data: “example”}		

7.2 API Testing

7.2.1 Overview

After ensuring that the helper functions in the frontend are sending requests to the API server with the correct route paths, HTTP request methods, and request bodies, it is also required that the logic of the route handler functions must be tested. This is because these functions are playing a crucial part in the system such as CRUD operations will be performed to manipulate the data in the database.

Test Method: Documentation-driven testing.

As Swagger, an API documentation tool is being used to document all the available routes in the API server, its “Try it out” function can be utilized to perform a comprehensive test on the route handler functions. By referring to each item in the API documentation, it can be certain that high testing coverage will be achieved. To illustrate, test data can first be prepared. Then, the test for each documented routes will be executed using the listed test data. Lastly, the returned results are captured and recorded to indicate the test outcomes.

To ease the testing, the API routes will be tested according to the order as such: posting, putting, getting, and deleting. The test data will be first filled into the parameters required, then screenshots of the returned response data will be screenshot as a prove for the test results. A new database will be created to isolate the API test from the real data. In this test, some of the sections of API routes will be omitted because of the infeasibility of sending request body to the route because the route handle function process does not support sending through Swagger or the request body is too complex. To recap, the documentations are modularized as follows:

Table 7.2.1.1 All the modules that will be tested in the API test.

Announcements	Classes	Subject Distributions
Dynamic Timetables	~~Dynamic Timetabling Algorithm~~	~~Preferences~~
Allotment Sets	~~Static Timeslots~~	Static Timetables
Static Timetabling Algorithm	Study Hours	Subjects
Teachers	Teaching Classes	Venue Usages
Venues	Admins	

7.2.2 Test Executions and Results

7.2.2.1 Admins

Since there is no creation function for admin accounts, a mock email and password for the admin will be created manually in the database to simulate the default admin account in the system.

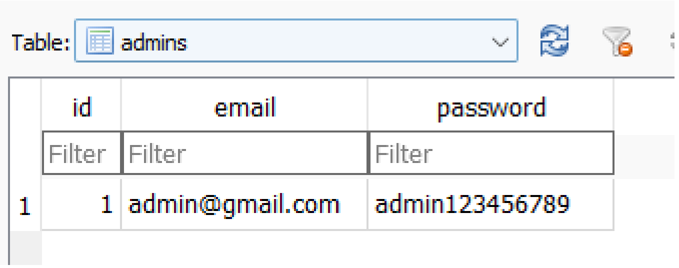


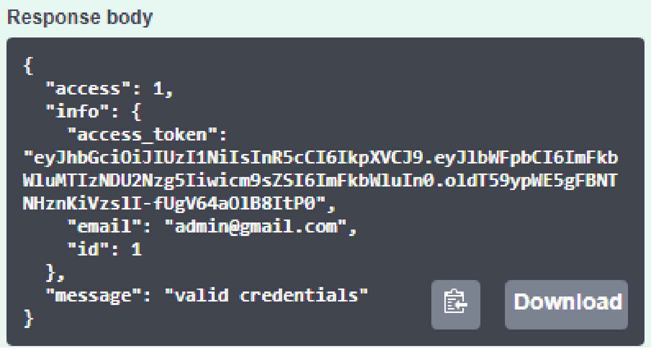
Table: admins

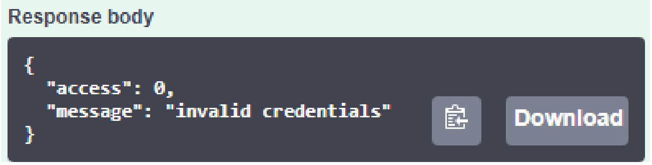
	id	email	password
	Filter	Filter	Filter
1	1	admin@gmail.com	admin123456789

Figure 7.2.2.1.1: Created Admin Account in the Database.

Table 7.2.2.1.1: API Request Information for Admins Category.

No.	Route	Method	Test Data	Expected Result	Actual Result	Status

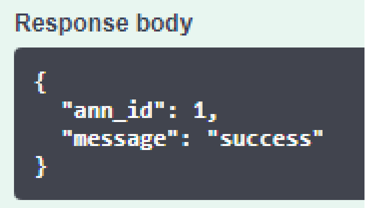
1	/admins/login	POST	{
			"credentials": {
			"admin_email": "admin@gmail.com",
			"admin_password": "admin123456789"
			}
			}
				Login successfully, then return admin info and access token.	
					Response body
					{
					"access": 1,
					"info": {
					"access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6ImFkbWluMTIzNDU2Nzg5Iiwicm9sZSI6ImFkbWluIn0.oldT59ypWE5gFBNTNHznKiVzslI-fUgV64aO1B8ItP0",
					"email": "admin@gmail.com",
					"id": 1
					},
					"message": "valid credentials"
					}
					Figure 7.2.2.1.2: Testing result for successful login.
						Pass

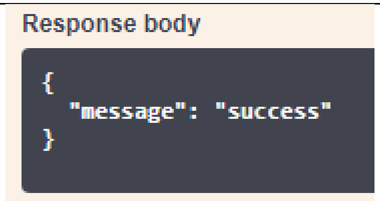
2	/admins/login	POST	{
			"credentials": {
			"admin_email": "user@gmail.com",
			"admin_password": "admin123456789"
			}
			}
				Login failed, then return failing message.	
					Response body
					{
					"access": 0,
					"message": "invalid credentials"
					}
					Figure 7.2.2.1.3: Test result for failed login.
						Pass

7.2.2.2 Announcements

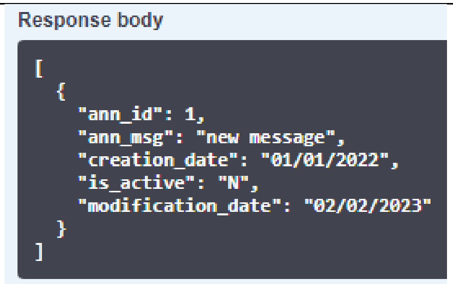
Table 7.2.2.2.1: API Request Information for Announcements Category.

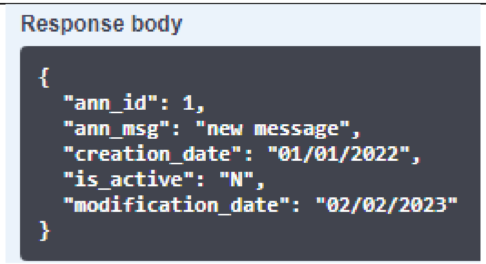
No.	Route	Method	Test Data	Expected Result	Actual Result	Status

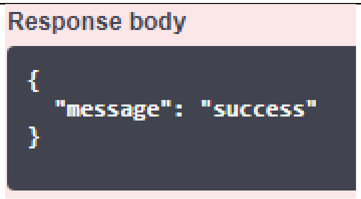
1	/announcements	POST	{
			"ann_msg": "example message",
			"creation_date": "01/01/2022",
			"is_active": "Y",
			"modification_date": "01/01/2022"
			}
				Announcement added to announcements database table successfully.	 Response body
					{
					"ann_id": 1,
					"message": "success"
					}
					Figure 7.2.2.2.1: Testing result upon successful announcement creation.
						Pass

2	/announcements/{ann}	PUT	ann: 1
			JSON: {
			"ann_msg": "new message",
			"creation_date": "02/02/2023",
				Update class in database successfully.	 Response body
					{
					"message": "success"
					}
					Figure 7.2.2.2.2: Test result upon successful announcement update.
						Pass

			"is_active": "N", "modification_date": "02/02/2023" }			

3	/announcements	GET	-	Get all announcements successfully.	
					Response body
					[
					{
					"ann_id": 1,
					"ann_msg": "new message",
					"creation_date": "01/01/2022",
					"is_active": "N",
					"modification_date": "02/02/2023"
					}
					]
					Figure 7.2.2.2.3: Test result upon reading all announcements successfully.
						Pass

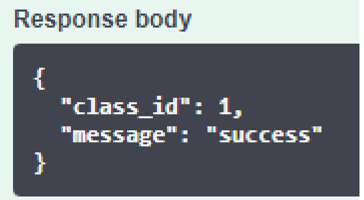
4	/announcements/{ann}	GET	ann: 1	Get announcement with ID of 1 successfully.	
					Response body
					{
					"ann_id": 1,
					"ann_msg": "new message",
					"creation_date": "01/01/2022",
					"is_active": "N",
					"modification_date": "02/02/2023"
					}
					Figure 7.2.2.2.4: Test result upon reading announcement with id of 1 successfully.
						Pass

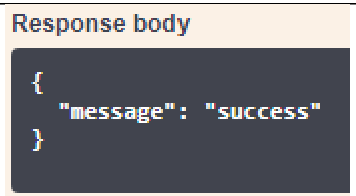
5	/announcements/{ann}	DELETE	ann: 1	Delete announcement with ID of 1 successfully.	 Response body { "message": "success" } Figure 7.2.2.2.5: Test result upon deleting announcement with id of 1 successfully.	Pass

7.2.2.3 Classes

Table 7.2.2.3.1: API Request Information for Classes Category.

No.	Route	Method	Test Data	Expected Result	Actual Result	Status

1	/classes	POST	{
			"class_name": "1A"
			}
				Class added to classes database table successfully.	
					Response body
					{
					"class_id": 1,
					"message": "success"
					}
					Figure 7.2.2.3.1: Testing result upon successful class creation.
						Pass

2	/classes/{cls}	PUT	cls: 1
			JSON: {
			"class_name": "2B"
			}
				Update class in database successfully.	
					Response body
					{
					"message": "success"
					}
					Figure 7.2.2.3.2: Test result upon successful class update.
						Pass

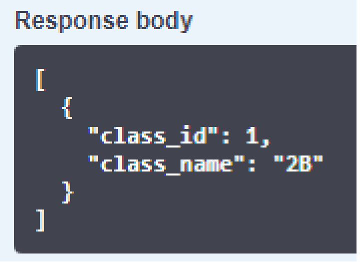
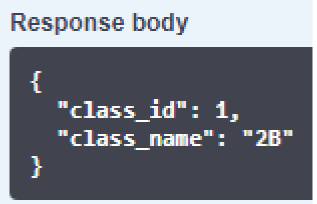
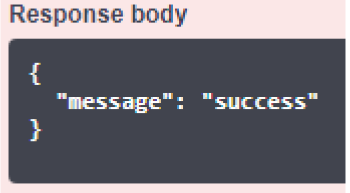
3	/classes	GET	-	Get all classes successfully.	
					Response body
					[
					{
					"class_id": 1,
					"class_name": "2B"
					}
					]
						Pass

					Figure 7.2.2.3.3: Test result for reading all classes successfully.	

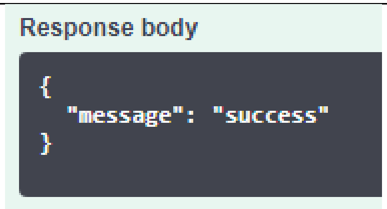
4	/classes/{cls}	GET	cls: 1	Get class with ID of 1 successfully.	
					Response body
					{
					"class_id": 1,
					"class_name": "2B"
					}
					Figure 7.2.2.3.4: Test result of reading class with id of 1 successfully.
						Pass

5	/classes/{cls}	DELETE	cls: 1	Delete class with ID of 1 successfully.	
					Response body
					{
					"message": "success"
					}
					Figure 7.2.2.3.5: Test result of deleting class with id of 1 successfully.
						Pass

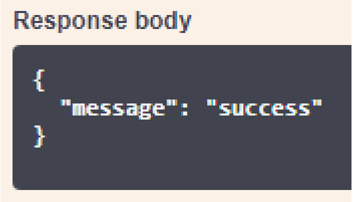
7.2.2.4 Subject Distribution

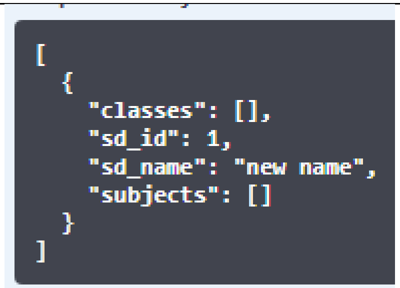
Table 7.2.2.4.1: API Request Information for Subject Distributions Category.

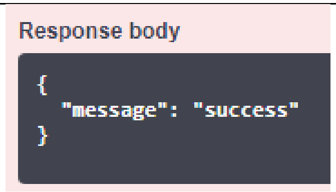
No.	Route	Method	Test Data	Expected Result	Actual Result	Status

1	/distributions	POST	{
			"as_id": 1,
			"classes": [
			{
			"class_id": 1
			}
			],
			"sd_name": "distribution 1",
			"subjects": [
			{
			"double_qty": 1,
			"single_qty": 0,
			"sub_id": 1,
			"triple_qty": 0
			}
			]
				Distribution added to database successfully.	 Response body
					{
					"message": "success"
					}
					Figure 7.2.2.4.1: Testing result upon successful distribution creation.
						Pass

			}				

2	/distributions/{sd_id}	PUT	sd_id: 1
			JSON: {
			"classes": [
			{
			"class_id": 2
			}
			],
			"sd_name": "new name",
			"subjects": [
			{
			"double_qty": 2,
			"single_qty": 0,
			"sub_id": 2,
			"triple_qty": 0
			}
			]
			}
				Update distribution in database successfully.	
					Response body
					{
					"message": "success"
					}
					Figure 7.2.2.4.2: Test result upon successful distribution update.
						Pass

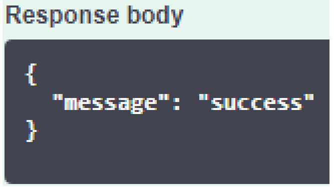
3	/distributions/{as_id}	GET	as_id: 1	Get all distribution in an allotment set successfully. If class_id or subject_id is empty, it will return empty list.	
					[
					  {
					    "classes": [],
					    "sd_id": 1,
					    "sd_name": "new name",
					    "subjects": []
					  }
					]
					Figure 7.2.2.4.3: Test result for reading all distributions successfully.
						Pass

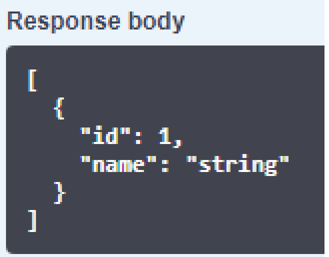
4	/distributions/{sd_id}	DELETE	sd_id: 1	Delete distribution with ID of successfully.	
					Response body
					{
					  "message": "success"
					}
					Figure 7.2.2.4.4: Test result of deleting distribution with id of 1 successfully.
						Pass

7.2.2.5 Dynamic Timetables

Table 7.2.2.5.1: API Request Information for Dynamic Timetables Category.

No.	Route	Method	Test Data	Expected Result	Actual Result	Status

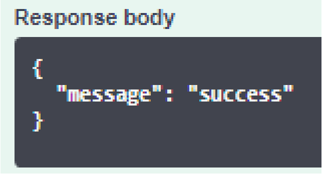
1	/dynamic_timetables	POST	{
			"allotment": {"1": {"0_0": {"teacher": "1", "subject": "1", "venue": "1"}}},
			"as_id": 0,
			"name": "string"
			}
				Dynamic timetable added to database successfully.	
					Response body
					`{ "message": "success" }`
					Figure 7.2.2.5.1: Testing result upon successful dynamic timetable creation.
						Pass

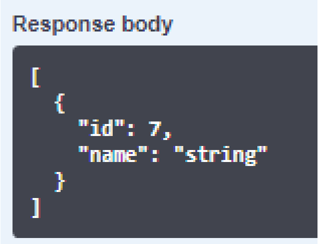
2	/dynamic_timetables/{as_id}	GET	as_id: 1	Get all dynamic timetable details in an allotment set successfully.	 Response body `[ { "id": 1, "name": "string" } ]` Figure 7.2.2.5.2: Test result for reading all dynamic timetable details in specific allotment set successfully.	Pass

7.2.2.6 Static Timetables

Table 7.2.2.6.1: API Request Information for Static Timetables Category.

No.	Route	Method	Test Data	Expected Result	Actual Result	Status

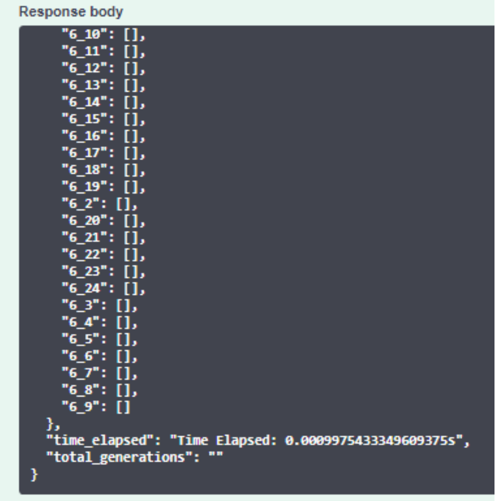
1	/static_timetables	POST	{
			"allotment": {},
			"as_id": 0,
			"name": "string"
			}
				Static timetable added to database successfully.	 Response body
					{
					"message": "success"
					}
					Figure 7.2.2.6.1: Testing result upon successful static timetables creation.
						Pass

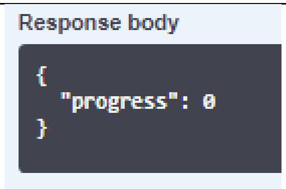
2	/static_timetables/{as_id}	GET	as_id: 0	Get static timetables with ID of 0 successfully.	 Response body
					[
					{
					"id": 7,
					"name": "string"
					}
					]
					Figure 7.2.2.6.2: Test result of reading static timetables with as_id of 0 successfully.
						Pass

7.2.2.7 Static Timetabling Algorithm

Table 7.2.2.7.1: API Request Information for Static Timetabling Algorithm Category.

No.	Route	Method	Test Data	Expected Result	Actual Result	Status

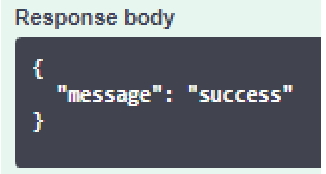
1	/static_timetabling_algorithm/run	POST	{
			"as_id": 0,
			"generation_limit": 0
			}
				Static timetabling algorithm ran successfully.	
					Response body
					`"6_10": [],`
					`"6_11": [],`
					`"6_12": [],`
					`"6_13": [],`
					`"6_14": [],`
					`"6_15": [],`
					`"6_16": [],`
					`"6_17": [],`
					`"6_18": [],`
					`"6_19": [],`
					`"6_2": [],`
					`"6_20": [],`
					`"6_21": [],`
					`"6_22": [],`
					`"6_23": [],`
					`"6_24": [],`
					`"6_3": [],`
					`"6_4": [],`
					`"6_5": [],`
					`"6_6": [],`
					`"6_7": [],`
					`"6_8": [],`
					`"6_9": []`
					`},`
					`"time_elapsed": "Time Elapsed: 0.0009975433349609375s",`
					`"total_generations": ""`
					`}`
					Figure 7.2.2.7.1: Testing result upon successful running the algorithm.
						Pass

2	/static_timetables/{as_id}	GET	-	Get progress of a static timetabling algorithm successfully.	
					Response body
					`{`
					`"progress": 0`
					`}`
					Figure 7.2.2.7.2: Test result of reading the progress of a running algorithm
						Pass

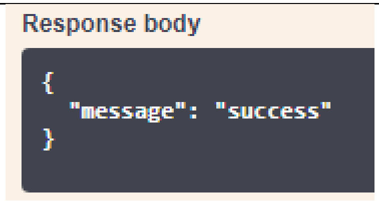
7.2.2.8 Study Hours

Table 7.2.2.8.1: API Request Information for Study Hours Category.

No.	Route	Method	Test Data	Expected Result	Actual Result	Status

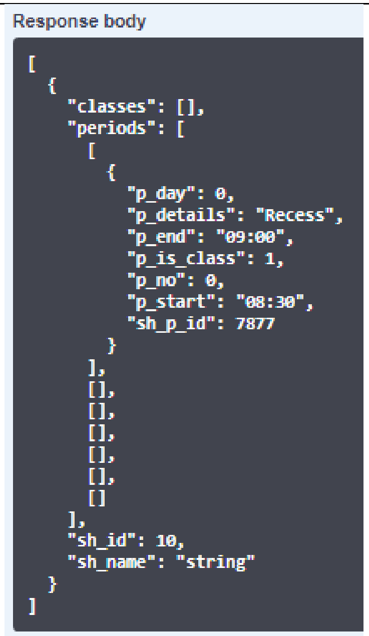
1	/study_hours	POST	{
			"as_id": 0,
			"classes": [
			{
			"class_id": 1
			}
			],
			"periods": [
			[
			{
			"p_day": 0,
			"p_details": "Rollcall",
			"p_end": "07:30",
			"p_is_class": true,
				Study hour added to database successfully.	
					{
					"message": "success"
					}
					Figure 7.2.2.8.1: Testing result upon successful study hour creation.
						Pass

			"p_no": 0, "p_start": "07:00" }]], "sh_name": "name" }			

2	/study_hours/{sh_id}	PUT	sh_id: 10
			{
			"classes": [
			{
			"class_id": 3
			}
			],
			"periods": [
			[
			{
			"p_day": 0,
			"p_details": "Recess",
				Update study hour in database successfully.	
					Response body
					{
					"message": "success"
					}
					Figure 7.2.2.8.2: Test result upon successful study hour update.
						Pass

			"p_end": "09:00", "p_is_class": true, "p_no": 0, "p_start": "08:30" }]], "sh_name": "string" }			

<table>
<tr>
<td>3</td>
<td>/study_hours/{as_id}</td>
<td>GET</td>
<td>-</td>
<td>Get all study hours successfully.</td>
<td>

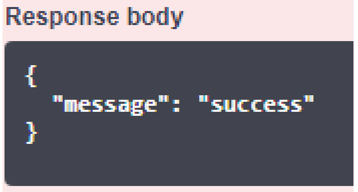


Response body

```
[
  {
    "classes": [],
    "periods": [
      [
        {
          "p_day": 0,
          "p_details": "Recess",
          "p_end": "09:00",
          "p_is_class": 1,
          "p_no": 0,
          "p_start": "08:30",
          "sh_p_id": 7877
        }
      ],
      [],
      [],
      [],
      [],
      [],
      []
    ],
    "sh_id": 10,
    "sh_name": "string"
  }
]
```

Figure 7.2.2.8.3: Test result upon reading all study hours successfully.

</td>
<td>Pass</td>
</tr>
<tr>
<td>4</td>
<td>/study_hours/{sh_id}</td>
<td>DELETE</td>
<td>sh_id: 10</td>
<td>Delete study hour with ID of successfully.</td>
<td>



Response body

```
{
  "message": "success"
}
```

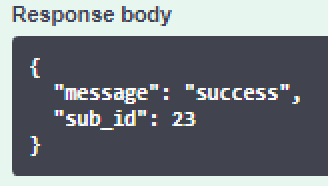
Figure 7.2.2.8.4: Test result upon deleting study hour with id of 10 successfully.

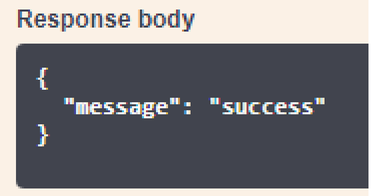
</td>
<td>Pass</td>
</tr>
</table>

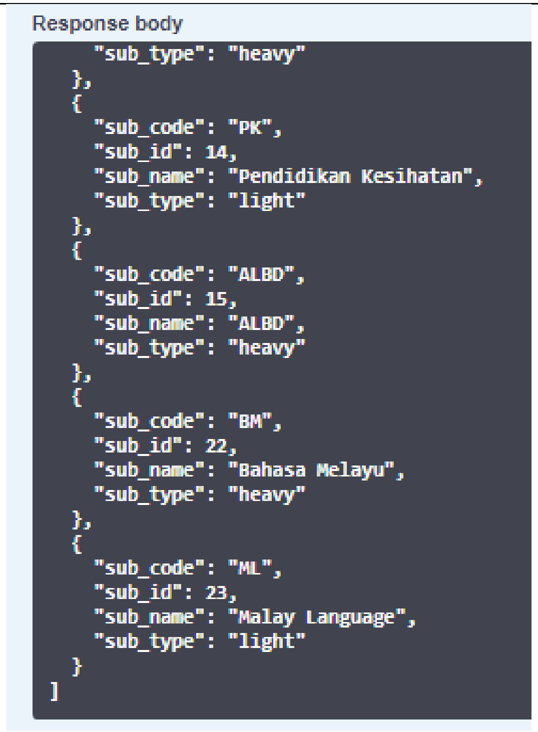
7.2.2.9 Subjects

Table 7.2.2.9.1: API Request Information for Subjects Category.

No.	Route	Method	Test Data	Expected Result	Actual Result	Status

1	/subjects	POST	{
			"sub_code": "SN",
			"sub_name": "Science",
			"sub_type": "heavy"
			}
				Subject added to database successfully.	
					Response body
					{
					"message": "success",
					"sub_id": 23
					}
					Figure 7.2.2.9.1: Testing result upon successful subject creation.
						Pass

2	/subjects/{sub}	PUT	sub: 23
			{
			"sub_code": "ML",
			"sub_name": "Malay Language",
			"sub_type": "light"
			}
				Update subject in database successfully.	
					Response body
					{
					"message": "success"
					}
					Figure 7.2.2.9.2: Test result upon successful subject update.
						Pass

3	/subjects	GET	-	Get all subjects successfully.	 Response body
					"sub_type": "heavy"
					},
					{
					"sub_code": "PK",
					"sub_id": 14,
					"sub_name": "Pendidikan Kesihatan",
					"sub_type": "light"
					},
					{
					"sub_code": "ALBD",
					"sub_id": 15,
					"sub_name": "ALBD",
					"sub_type": "heavy"
					},
					{
					"sub_code": "BM",
					"sub_id": 22,
					"sub_name": "Bahasa Melayu",
					"sub_type": "heavy"
					},
					{
					"sub_code": "ML",
					"sub_id": 23,
					"sub_name": "Malay Language",
					"sub_type": "light"
					}
					]
					Figure 7.2.2.9.3: Test result upon reading all subjects successfully.
						Pass

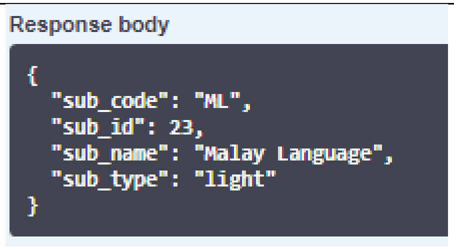
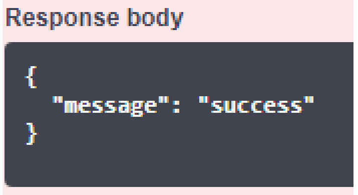
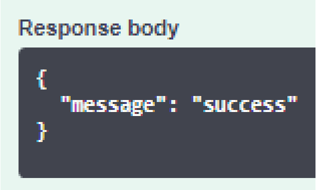
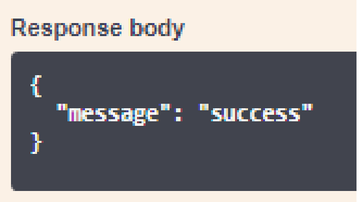
4	/subjects/{sub}	GET	sub: 23	Get subject details with specific ID successfully.	 Response body
					{
					"sub_code": "ML",
					"sub_id": 23,
					"sub_name": "Malay Language",
					"sub_type": "light"
					}
						Pass

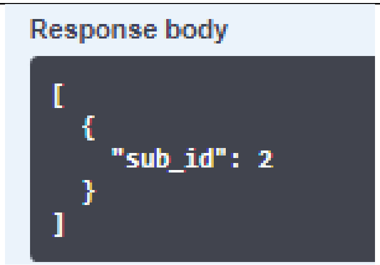
					Figure 7.2.2.9.4: Test result upon reading all subjects successfully.	

5	/subjects/{sub}	DELETE	sub: 23	Delete subject with ID of successfully.	
					Response body
					{
					"message": "success"
					}
					Figure 7.2.2.9.5: Test result upon deleting subject with id of 23 successfully.
						Pass

6	/teaching-subjects	POST	{
			"teacher_id": 1,
			"teaching_subjects": [
			{
			"sub_id": 1
			}
			]
			}
				Teaching subjects added to the database successfully.	
					Response body
					{
					"message": "success"
					}
					Figure 7.2.2.9.6: Test result upon added teaching subjects successfully.	

7	/teaching-subjects/{teacher_id}	PUT	teacher_id: 1
			{
			"teaching_subjects": [
				Update teaching subjects in database successfully.	
					Response body
					{
					"message": "success"
					}	

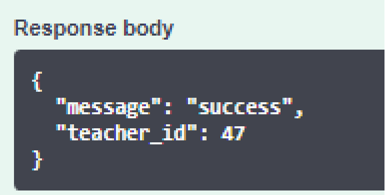
			{
			"sub_id": 2
			}
			]
			}
					Figure 7.2.2.9.7: Test result upon updating teaching subjects successfully.	

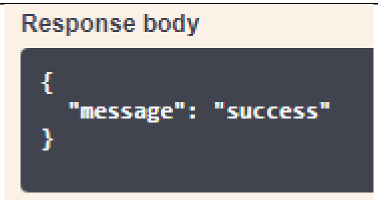
8	/teaching-subjects/{teacher_id}	GET	teacher_id: 1	Get teaching subjects with specific ID successfully.	
					Response body
					[
					{
					"sub_id": 2
					}
					]
					Figure 7.2.2.9.8: Test result upon reading teaching subjects successfully.	

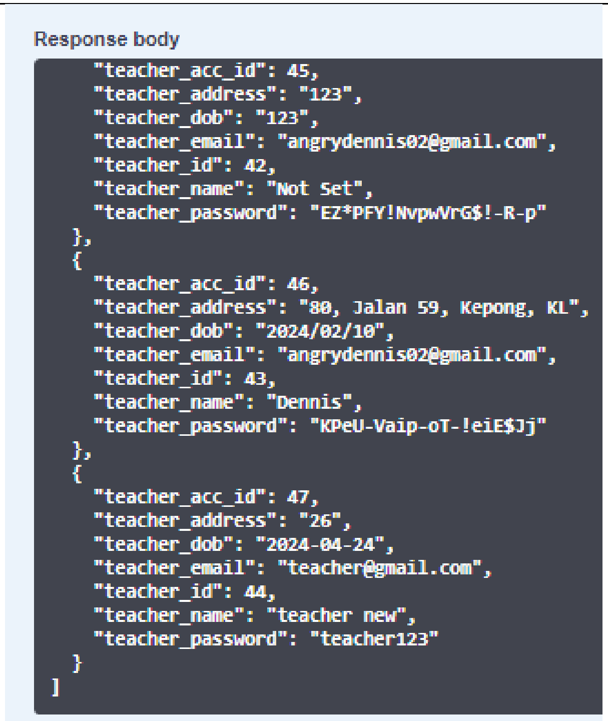
7.2.2.10 Teachers

Table 7.2.2.10.1: API Request Information for Teachers Category.

No.	Route	Method	Test Data	Expected Result	Actual Result	Status

1	/teachers	POST	{
			"teacher_details": {
			"teacher_address": "26, Jalan 64, KL",
			"teacher_dob": "2024-04-24",
			"teacher_email": "teacher@gmail.com",
			"teacher_name": "teacher",
			"teacher_password": "teacher123"
			}
			}
				Teacher added to database successfully.	
					Response body
					{
					"message": "success",
					"teacher_id": 47
					}
					Figure 7.2.2.10.1: Testing result upon successful teacher creation.
						Pass

2	/teacher/{teacher_id}	PUT	teacher_id: 44
			{
			"teacher_details": {
			"teacher_acc_id": 47,
			"teacher_address": "26",
			"teacher_dob": "2024-04-24",
			"teacher_email": "teacher123@gmail.com",
			"teacher_name": "teacher new",
			"teacher_password": "teacher123"
			}
			}
				Update teacher details in database successfully.	
					Response body
					{
					"message": "success"
					}
					Figure 7.2.2.10.2: Test result upon successful teacher update.
						Pass

3	/teachers	GET	-	Get all teachers successfully.	
					Response body
					"teacher_acc_id": 45,
					"teacher_address": "123",
					"teacher_dob": "123",
					"teacher_email": "angrydennis02@gmail.com",
					"teacher_id": 42,
					"teacher_name": "Not Set",
					"teacher_password": "EZ*PFY!NvpwVrG$!-R-p"
					},
					{
					"teacher_acc_id": 46,
					"teacher_address": "80, Jalan 59, Kepong, KL",
					"teacher_dob": "2024/02/10",
					"teacher_email": "angrydennis02@gmail.com",
					"teacher_id": 43,
					"teacher_name": "Dennis",
					"teacher_password": "KPeU-Vaip-oT-!eiE$Jj"
					},
					{
					"teacher_acc_id": 47,
					"teacher_address": "26",
					"teacher_dob": "2024-04-24",
					"teacher_email": "teacher@gmail.com",
					"teacher_id": 44,
					"teacher_name": "teacher new",
					"teacher_password": "teacher123"
					}
					]
					Figure 7.2.2.10.3: Test result upon reading all teachers successfully.
						Pass

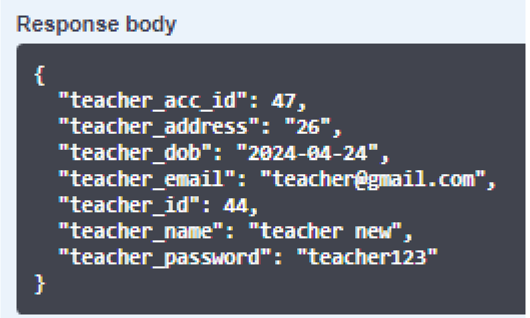
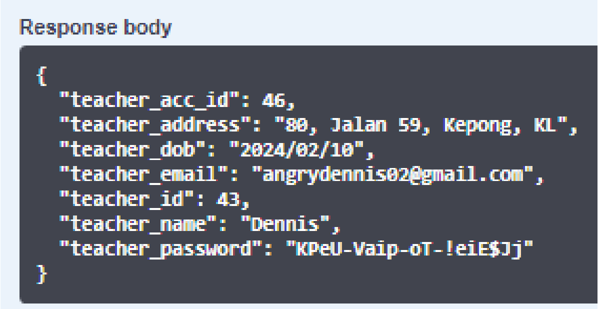
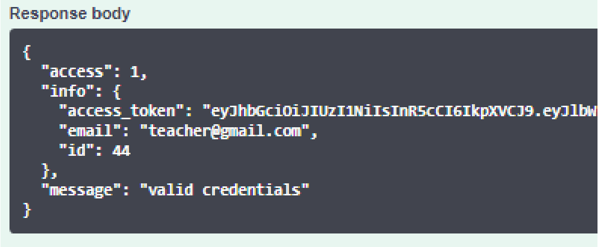
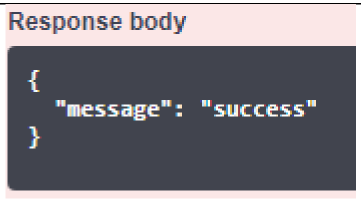
4	/teachers/{teacher}	GET	teacher: 44	Get teacher details with specific ID successfully.	
					Response body
					{
					"teacher_acc_id": 47,
					"teacher_address": "26",
					"teacher_dob": "2024-04-24",
					"teacher_email": "teacher@gmail.com",
					"teacher_id": 44,
					"teacher_name": "teacher new",
					"teacher_password": "teacher123"
					}
						Pass

					Figure 7.2.2.10.4: Test result upon reading a teacher’s details successfully.	

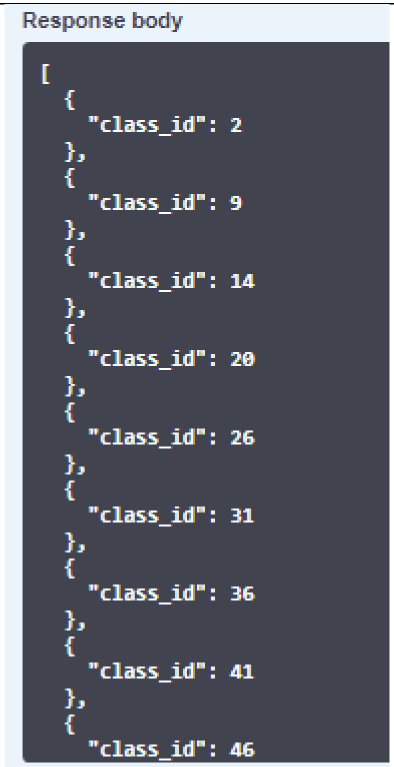
5	/teachers/get-started/{access_token}	GET	access_token: KPeU-Vaip-oT-!eiE$Jj	Get details of new teacher successfully.	
					Response body
					{
					"teacher_acc_id": 46,
					"teacher_address": "80, Jalan 59, Kepong, KL",
					"teacher_dob": "2024/02/10",
					"teacher_email": "angrydennis02@gmail.com",
					"teacher_id": 43,
					"teacher_name": "Dennis",
					"teacher_password": "KPeU-Vaip-oT-!eiE$Jj"
					}
					Figure 7.2.2.10.5: Test result upon getting details of new teacher.
						Pass

6	/teachers/login	POST	{
			"credentials": {
			"teacher_email": "teacher@gmail.com",
			"teacher_password": "teacher123"
			}
			}
				Log into a teacher account successfully.	
					Response body
					{
					"access": 1,
					"info": {
					"access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbW
					"email": "teacher@gmail.com",
					"id": 44
					},
					"message": "valid credentials"
					}
					Figure 7.2.2.10.6: Test result upon logging into a teacher account.	

7	/teachers/{teacher_id}	DELETE	teacher_id: 44	Delete teacher with ID of 44 successfully.	
					Response body
					{
					"message": "success"
					}
					Figure 7.2.2.10.7: Test result upon deleting teacher with id of 44 successfully.
						Pass

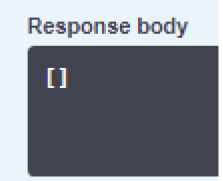
7.2.2.11 Teaching Classes

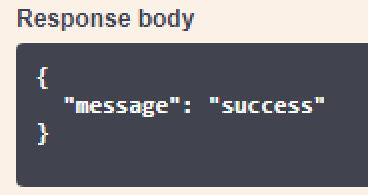
Table 7.2.2.11.1: API Request Information for Teaching Classes Category.

No.	Route	Method	Test Data	Expected Result	Actual Result	Status
1	/teaching_classes/{teacher_id}	GET	teacher_id: 2	Get teaching classes of a teacher in database successfully.	 Response body [{ "class_id": 2 }, { "class_id": 9 }, { "class_id": 14 }, { "class_id": 20 }, { "class_id": 26 }, { "class_id": 31 }, { "class_id": 36 }, { "class_id": 41 }, { "class_id": 46 Figure 7.2.2.11.1: Test result upon reading teaching classes of a teacher successfully.	Pass

7.2.2.12 Venue Usages

Table 7.2.2.12.1: API Request Information for Venue Usages Category.

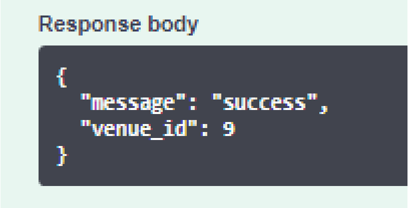
No.	Route	Method	Test Data	Expected Result	Actual Result	Status
1	/venues_usages/{as_id }	GET	as_id: 0	Get venue usages successfully from database.	 Response body [] Figure 7.2.2.12.1 Test result upon getting venue usages successfully.	Pass

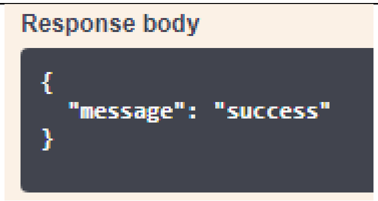
2	/venue_usages/{as_id}	PUT	as_id: 2
			{
			"venue_usages": [
			{
			"class_id": 33,
			"sub_id": 10,
			"venue_ids": [
			3
			]
			}
			]
			}
				Update venue usages in database successfully.	
					Response body
					{
					"message": "success"
					}
					Figure 7.2.2.12.2: Test result upon updating venue successfully.
						Pass

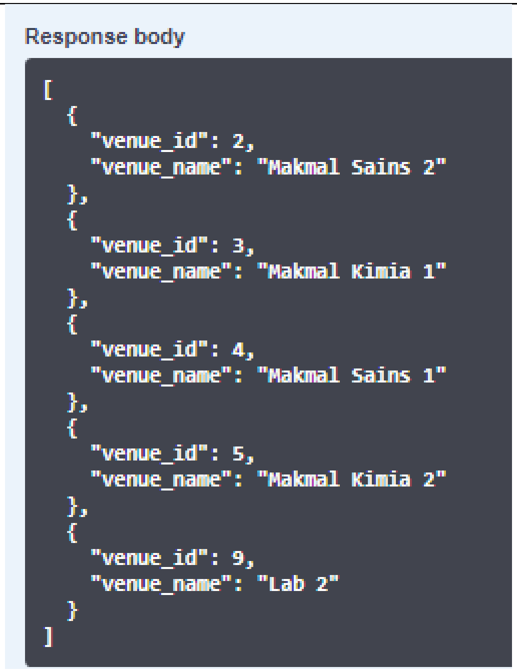
7.2.2.13 Venues

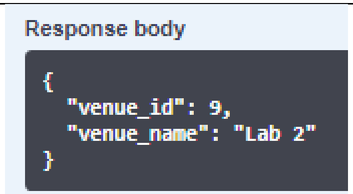
Table 7.2.2.13.1: API Request Information for Venues Category.

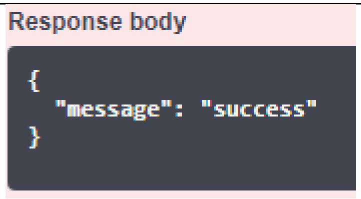
No.	Route	Method	Test Data	Expected Result	Actual Result	Status

1	/venues	POST	{
			"venue_name": "Lab 1"
			}
				Venue added to venues database table successfully.	
					Response body
					{
					"message": "success",
					"venue_id": 9
					}
					Figure 7.2.2.13.1: Testing result upon successful venue creation.
						Pass

2	/venues/{venue}	PUT	venue: 9
			{
			"venue_name": "Lab 2"
			}
				Update venue in database successfully.	
					Response body
					{
					"message": "success"
					}
					Figure 7.2.2.13.2: Test result upon successful venue update.
						Pass

3	/venues	GET	-	Get all venues successfully.	Response body
					[
					{
					"venue_id": 2,
					"venue_name": "Makmal Sains 2"
					},
					{
					"venue_id": 3,
					"venue_name": "Makmal Kimia 1"
					},
					{
					"venue_id": 4,
					"venue_name": "Makmal Sains 1"
					},
					{
					"venue_id": 5,
					"venue_name": "Makmal Kimia 2"
					},
					{
					"venue_id": 9,
					"venue_name": "Lab 2"
					}
					]
					Figure 7.2.2.13.3: Test result upon reading all venues successfully.
						Pass

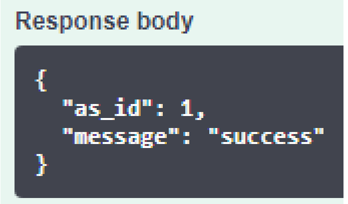
4	/venues/{venue}	GET	venue: 9	Get announcement with ID of 19successfully.	Response body
					{
					"venue_id": 9,
					"venue_name": "Lab 2"
					}
					Figure 7.2.2.13.4: Test result upon reading venue with id of 9 successfully.
						Pass

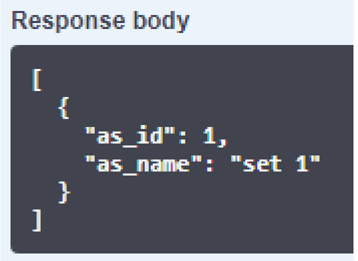
5	/venues/{venue}	DELETE	venue: 9	Delete venue with ID of 9 successfully.	
					Response body
					`{`
					`  "message": "success"`
					`}`
					Figure 7.2.2.13.5: Test result upon deleting venue with id of 9 successfully.
						Pass

7.2.2.14 Allotment Sets

Table 7.2.2.14.1: API Request Information for Allotment Sets Category.

No.	Route	Method	Test Data	Expected Result	Actual Result	Status

1	/sets	POST	{
			"as_name": "set 1"
			}
				Allotment set added to database successfully.	 Response body
					{
					"as_id": 1,
					"message": "success"
					}
					Figure 7.2.2.14.1: Testing result upon successful allotment set creation.
						Pass

2	/sets	GET	-	Get all allotment sets successfully.	 Response body
					[
					{
					"as_id": 1,
					"as_name": "set 1"
					}
					]
					Figure 7.2.2.14.2: Test result for reading all allotment sets successfully.
						Pass

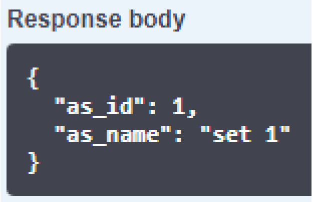
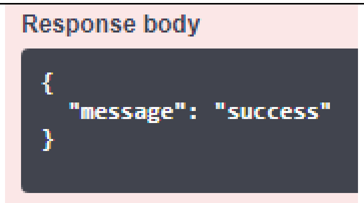
3	/sets/{as_id}	GET	as_id: 1	Get allotment set with ID of 1 successfully.	 Response body
					{
					"as_id": 1,
					"as_name": "set 1"
					}
						Pass

					Figure 7.2.2.14.3: Test result of reading allotment set with id of 1 successfully.	

4	/sets/{as_id}	DELETE	as_id: 1	Delete allotment set with ID of 1 successfully.	
					Response body
					{
					"message": "success"
					}
					Figure 7.2.2.14.3: Test result of deleting allotment set with id of 1 successfully.
						Pass

7.3 User Acceptance Test

In this section, the UAT done on two types of users will be explained. To illustrate, an experienced user who is an educator and an inexperienced user who never use the system before. But first, a set of 10 questions will be prepared to evaluate customer satisfaction on the system.

7.3.1 Customer Satisfaction Survey

Table 7.3.1.1: List of Question Used in the Customer Satisfaction Survey.

No.	Question	Scoring
1	How satisfied are you with the overall user interface of the system. (Easy to use?)	1 to 5
2	How well do you able to understand the functionalities in the system? (Easy to learn?)	
3	On a scale of 1 to 5, how much is system able to fulfill your objective to allocate timetables for a school?	
4	How satisfied are you with the data management module in the system? (Add, delete, sort, and filter subjects, classes, ...)	
5	How much do you think that the system is able to reduce the time taken to allocate timetables compared to the manual approach? (Automated vs Manual)	
6	Rate your overall experience while using the system to achieve your objectives. (Happy? Annoyed?)	
7	If you want to rate STSAMS, what is the score that you will give?	
8	How well do you think the teacher's satisfaction level calculated after each dynamic timetable generations are able to improve system admin's decision to create better quality timetables? (Teacher's satisfaction fulfillment reflected from the preference settings in teacher's module)	
9	Are you able clearly identify the separation between the system admin's module and the teacher's module?	
10	What is your reliability level on the authentication and authorization of the system? (Login modules, prevent unauthorized access)	

7.3.2 Test Results

7.3.2.1 Inexperienced User

The inexperienced user in a school would be a student. So, a student has been chosen as a respondent to evaluate STSAMS. The gotten scores of the evaluation are stated in Table 7.3.2.1.1.

Table 7.3.2.1.1: Evaluation Scores given by a Student to STSAMS.

No.	Question	Score Given
1	How satisfied are you with the overall user interface of the system. (Easy to use?)	3
2	How well do you able to understand the functionalities in the system? (Easy to learn?)	3
3	On a scale of 1 to 5, how much is system able to fulfill your objective to allocate timetables for a school?	4
4	How satisfied are you with the data management module in the system? (Add, delete, sort, and filter subjects, classes, ...)	5
5	How much do you think that the system is able to reduce the time taken to allocate timetables compared to the manual approach? (Automated vs Manual)	3
6	Rate your overall experience while using the system to achieve your objectives. (Happy? Annoyed?)	3
7	If you want to rate STSAMS, what is the score that you will give?	3
8	How well do you think the teacher's satisfaction level calculated after each dynamic timetable generations are able to improve system admin's decision to create better quality timetables? (Teacher's satisfaction fulfillment reflected from the preference settings in teacher's module)	3
9	Are you able clearly identify the separation between the system admin's module and the teacher's module?	5
10	What is your reliability level on the authentication and authorization of the system? (Login modules, prevent unauthorized access)	5

To calculate the average score or average satisfaction level given by the student, the formula below will be used: -

$$Average\ Satisfaction\ Percentage = \frac{Total\ Score\ Given}{\#Questions * Max\ Scoring} * 100\%$$

$$Average\ Satisfaction\ Percentage = \frac{37}{10 * 5} * 100\%$$

$$Average\ Satisfaction\ Percentage = \frac{37}{50} * 100\%$$

$$Average\ Satisfaction\ Percentage = 74\%$$

7.3.2.2 Experienced User

The experienced user in a school would be a teacher. So, a teacher has been chosen as a respondent to evaluate STSAMS. The gotten scores of the evaluation are stated in Table 7.3.2.2.1.

Table 7.3.2.2.1: Evaluation Scores given by a Teacher to STSAMS.

No.	Question	Score Given
1	How satisfied are you with the overall user interface of the system. (Easy to use?)	3
2	How well do you able to understand the functionalities in the system? (Easy to learn?)	3
3	On a scale of 1 to 5, how much is system able to fulfill your objective to allocate timetables for a school?	3
4	How satisfied are you with the data management module in the system? (Add, delete, sort, and filter subjects, classes, ...)	5
5	How much do you think that the system is able to reduce the time taken to allocate timetables compared to the manual approach? (Automated vs Manual)	4
6	Rate your overall experience while using the system to achieve your objectives. (Happy? Annoyed?)	3
7	If you want to rate STSAMS, what is the score that you will give?	3
8	How well do you think the teacher's satisfaction level calculated after each dynamic timetable generations are able to improve system admin's decision to create better quality timetables? (Teacher's satisfaction fulfillment reflected from the preference settings in teacher's module)	4
9	Are you able clearly identify the separation between the system admin's module and the teacher's module?	5
10	What is your reliability level on the authentication and authorization of the system? (Login modules, prevent unauthorized access)	5

To calculate the average score or average satisfaction level given by the teacher, the formula below will be used: -

$$Average\ Satisfaction\ Percentage = \frac{Total\ Score\ Given}{\#Questions * Max\ Scoring} * 100\%$$

$$Average\ Satisfaction\ Percentage = \frac{38}{10 * 5} * 100\%$$

$$Average\ Satisfaction\ Percentage = \frac{38}{50} * 100\%$$

$$Average\ Satisfaction\ Percentage = 76\%$$

In conclusion, if the average satisfaction percentage of both users is found, the overall user’s satisfaction of STSAMS is 75%.

CHAPTER 8

CONCLUSION & RECOMMENDATIONS

8.1 Achievements

It was stated in the project objectives that the developed system must be prepared alongside the design documents to give a high-level view of the STSAMS. This objective was achieved through preparing the use case diagram and use case descriptions to describe the involvement of users in different STSAMS's functionalities. Not just that, the design for data persistence was also shown using the entity relationship diagram (ERD) equipped with the description explaining what kind of data exactly for each column in database table. As for API routes, every parameter or request body and the response body is shown to get better understanding of the routes in API server. Before the development of the genetic algorithms, the genetic encodings for both algorithms were sketched out with a clear explanation on the execution steps. By doing so, the STSAMS was able to be developed into a web-based application within the specified period which is 12 weeks for the development phase. To solve the problems from existing software that is like STSAMS, the system including functions like calculating teacher's satisfaction level with constraints like preferred teaching class, subjects and time period for each timetable generation, hence the considering teacher's preferences in timetabling.

Moreover, the development methodologies for the project were also assessed, resulting in applying the Scrum methodology for the project executions. The Scrum methodology recommended the practitioners to separate the development process into iteration. So, unit tests were able to be created for the developed components to evaluate the correctness of the system. During the early iteration of the project, different types of allocation algorithms such as PSO, GA, Bipartite Matching, and Brute Force Approach were compared to select the suitable algorithm for timetabling which is the Genetic Algorithm that is meta-heuristic in nature. So, it could solve the problem of clashing better than the deterministic algorithms. As for requirements of STSAMS, questionnaires were sent to the public-school teachers and interview were conducted with a system administrator to gather insight of the teacher-subject allocation in the school's context. Using this requirement seeking approach, the important requirements can be focused on, so that the functionalities of the STSAMS won't be too vague but still able to the achieve the objectives of generating a timetable.

8.2 Limitations & Recommendation for Future Development.

STSAMS will be able to help system administrators in public schools to generate their timetable yearly without the need of manually assigning teachers to teach specific subjects or have lessons in different classes. This is because STSAMS provided the data management module to the system administrators to manage data like subjects, classes, venues, and teachers. Not just that, the system administrator can also generate the timetable automatically. By doing so, STSAMS can save the admins' time because they will not need to compare different timetable combinations and finding clashing manually. The time saved can be used in other more productive activities in the school.

Nonetheless, every coin has two sides. STSAMS is still immature in terms of usage as most of the modules like user interfaces, API server, and the algorithms were built from scratch. The system users may encounter usability issue in the user interface of STSAMS, since usability testing is not performed during this project execution. Not just that, STSAMS's user cases are only tailored to a specific public school. So, system administrators from other public schools may discover that only some of the functionalities can be used by them, resulting in finding other better solution for their use case. For example, functionalities that may pose problems are the algorithm constraints as different schools may treat their timetabling process differently. Due to time constraints, the STSAMS is still not completely teacher centered due to teacher preference settings. This is because the current version of STSAMS is only able to calculate the satisfaction level of teachers by mapping the preferences with a timetable result, hence calculating the number of satisfied constraints set by the teachers. Although this function can provide the satisfaction level, it is still depended to the system administrators to decide whether they want to ignore the low satisfaction count. However, the teacher preferences were seen as a soft constraint for the algorithm, thus it is optional that it need to be achieved.

So, there are future improvements that can be done to eliminate the limitations. To solve usability problem, usability testing can be conduct consistently to find out the root cause that will increase retention and system learnability. Not just that, the user interface design that may found out to be troublesome to the users can be improve by satisfying their feedback during the usability testing. Moreover, requirement seeking scope can also be increased to find out the constraints used by different school. So, the STSAMS functionalities can be more generalized and are able cater to more public schools in Malaysia. As for a teacher centered STSAMS, the teacher preferences can also be evaluated in a way that the algorithms will have a history of satisfied teachers in previously timetable generations. Based on the history, the previously satisfied will have a lower chance to get their preferred constraints, hence making way for the unsatisfied teacher to have a chance to teach their preferred subjects, classes, and time periods. This will ensure fairer evaluation of teacher preference, hence making the constraints more dynamic.

REFERENCES

Adams, N. E., 2015. Bloom’s taxonomy of cognitive learning objectives. *Journal of the Medical Library Association: JMLA*, [online] 103(3), pp.152–153. doi: <https://doi.org/10.3163/1536-5050.103.3.010>.

Akbar, R., 2019. Tailoring Agile-Based Software Development Processes. *IEEE Access*, 7, pp.139852–139869. doi: <https://doi.org/10.1109/access.2019.2944122>.

Alutbi, M., 2020. *Work Breakdown Structure (WBS)*. [online] ResearchGate. Available at: <https://www.researchgate.net/publication/342163727>.

Alsaqqa, S., Sawalha, S. and Abdel-Nabi, H., 2020. Agile Software Development: Methodologies and Trends. *International Journal of Interactive Mobile Technologies*, 14(11), pp.246–270.

Angelidakis, H., 2020. Bipartite graph. Available at: <https://www.cantorsparadise.com/matchings-in-bipartite-graphs-and-the-kőnig-egerváry-theorem-via-lp-duality-ff943454431> [Accessed 1 Sept. 2023].

Asano, T. and Asano, Y., 2000. RECENT DEVELOPMENTS IN MAXIMUM FLOW ALGORITHMS. *Journal of the Operations Research Society of Japan*, 43(1), pp.2–31. doi: https://doi.org/10.15807/jorsj.43.2.

Awad, F.H., Al-kubaisi, A. and Mahmood, M., 2022. Large-scale timetabling problems with adaptive tabu search. *Journal of Intelligent Systems*, 31(1), pp.168–176. doi: <https://doi.org/10.1515/jisys-2022-0003>.

Ayeni, A. J. and Amanekwe, A. P., 2018. Teachers' Instructional Workload Management and Students' Academic Performance in Public and Private Secondary Schools in Akoko North-East Local Government, Ondo State, Nigeria, *American International Journal of Education and Linguistics Research*, 1(1), pp. 9–23. doi: <https://10.46545/aijelr.v1i1.135>.

Bascia, N., Chindalo, P., Connelly, C., Faubert, B., Flessa, J., Fredua-Kwarteng, E., Leung, J., Mascall, B. and Rottmann, C., n.d. *Reducing Class Size: What Do We Know?* [online] Available at: <https://www.classsizematters.org/wp-content/uploads/2012/11/Reducing-Class-Size-What-do-we-Know.pdf>.

Baturu, C. and Naufal abdi., 2020. Brute Force Algorithm Implementation Of Dictionary Search. *Jurnal Info Sains: Informatika dan Sains*, 10(1), pp.24–30. doi: <https://doi.org/10.54209/infosains.v10i1.29>.

Botti, S., Iyengar, S.S. and McGill, A.L., 2023. Choice freedom. *Journal of Consumer Psychology*. doi: <https://doi.org/10.1002/jcpy.1325>.

Boehm, B. and Turner, R., 2004. *Balancing agility and discipline: evaluating and integrating agile and plan-driven methods*. [online] IEEE Xplore. doi: <https://doi.org/10.1109/ICSE.2004.1317503>.

Black, E, P., 2009."deterministic algorithm", in *Dictionary of Algorithms and Data Structures* [online]. Available from: <https://www.nist.gov/dads/HTML/deterministicAlgorithm.html>

Brownlee, A., 2005. An application of genetic algorithms to university timetabling. Available at: <https://www.cs.stir.ac.uk/~sbr/files/Report3.pdf>

Chan, C.K., Gooi, H.B. and Lim, M.H., 2006. An evolutionary algorithm based subject allocation system. *Journal of the Chinese Institute of Engineers*, 29(3), pp.415–422. doi: <https://doi.org/10.1080/02533839.2006.9671137>.

Cohen, D. & Lindvall, M. & Costa, Pierpaolo., 2003. Agile Software Development A DACS State-of-the-Art Report. A DACS State-of-the-art Report.

Engelbrecht, A. P., 2013. *Particle Swarm Optimization: Global Best or Local Best?* [online] IEEE Xplore. doi: <https://doi.org/10.1109/BRICS-CCI-CBIC.2013.31>.

Guerriero, S., 2017. *Teachers' Pedagogical Knowledge and the Teaching Profession Background Report and Project Objectives*. [online] Available at: <https://www.oecd.org/education/ceri/Background_document_to_Symposium_ITEL-FINAL.pdf>.

Hussain, A., Mkpojiogu, E.O.C. and Kamal, F., 2015. *Eliciting User Satisfying Requirements for an e-Health Awareness System Using Kano Model*. [online] Available at: <http://www.wseas.us/e-library/conferences/2015/Malaysia/COMP/COMP-20.pdf>.

Hamid, S.A., Abdulrahman, R.A. and Khamees, D.R.A., 2020. What is Client-Server System: Architecture, Issues and Challenge of Client -Server System (Review). *Recent Trends in Cloud Computing and Web Engineering*, [online] 2(1), pp.1–6. doi: <https://doi.org/10.5281/zenodo.3673071>.

Howley-Rouse, A., 2021. *An introduction to cognitive load theory*. [online] THE EDUCATION HUB. Available at: <https://theeducationhub.org.nz/an-introduction-to-cognitive-load-theory/>.

Jacqueline., 2022. *Everything To Know About Semi-Automated Material Handling Systems*. [online] REB Storage Systems International. Available at: <https://rebstorage.com/articles-white-papers/semi-automated-material-handling-systems/>.

Jain, N., 2019. *Subject-Teacher-Allocation*. [online] ResearchGate. Available at: <https://www.researchgate.net/publication/330290028_Subject-Teacher-Allocation> [Accessed 6 Jul. 2023].

Jeschke, C., Kuhn, C., Heinze, A., Zlatkin-Troitschanskaia, O., Saas, H. and Lindmeier, A. M. (2021). Teachers' Ability to Apply Their Subject-Specific Knowledge in Instructional Settings—A Qualitative Comparative Study in the Subjects Mathematics and Economics. *Frontiers in Education*, [online] 6. doi: <https://doi.org/10.3389/feduc.2021.683962>.

Jomuad, P., Leah, M., Cericos, E., Bacus, J., Vallejo, J., Dionio, B., Bazar, J., Cocolan, J. and Clarin, A., 2021. Teachers' workload in relation to burnout and work performance. *International Journal of Educational Policy Research and Review*, 8(2), pp.48–53. doi: <https://doi.org/10.15739/IJEPRR.21.007>.

Lavrijsen, J., Tracey, T. J. G., Verachtert, P., De Vroede, T., Soenens, B. and Verschueren, K., 2021. Understanding school subject preferences: The role of trait interests, cognitive abilities and perceived engaging teaching. *Personality and Individual Differences*, 174, p.110685. doi: <https://doi.org/10.1016/j.paid.2021.110685>.

Likourezos, V., 2021. *An introduction to cognitive load theory*. [online] THE EDUCATION HUB. Available at: <https://theeducationhub.org.nz/an-introduction-to-cognitive-load-theory/>.

Lazarides, R., Gaspard, H. and Dicke, A.-L., 2019. Dynamics of classroom motivation: Teacher enthusiasm and the development of math interest and teacher support. *Learning and Instruction*, [online] 60, pp.126–137. doi: <https://doi.org/10.1016/j.learninstruc.2018.01.012>.

Minellim, M., Chambers, M. and Dhiraj, A., 2013. *Big Data, Big Analytics: Emerging Business Intelligence and Analytic Trends for Today's Businesses*. [online] *Google Books*. John Wiley & Sons. Available at: <https://books.google.com.my/books?hl=en&lr=&id=HYYaX9dsZsYC&oi=fnd&pg= PR13&dq=Big+Data> [Accessed 9 Aug. 2023].

MESHDS, 2021. *The Disadvantages of Manual Document Filing Processes*. [online] blog.mesltd.ca. Available at: <https://blog.mesltd.ca/the-disadvantages-of-manual-document-filing-processes-1>.

Mkpojiogu, E.O.C. and Hashim, N. L., 2016. Understanding the relationship between Kano model's customer satisfaction scores and self-stated requirements importance. SpringerPlus, 5(1). doi: <https://doi.org/10.1186/s40064-016-1860-y>

Matzler K, Hinterhuber HH, Bailom F, Sauermein E., 1996. How to delight your customer. J Product Brand Manag 5(2):6–18

Mnkandla, Ernest & Dwolatzky, Barry., 2004. A Survey of Agile Methodologies. Transactions of the South African Institute of Electrical Engineers. 95. 236-247.

Mohammad, A., Saleh, O. and Abdeen, R.A., 2006. Occurrences algorithm for string searching based on brute-force algorithm. *Journal of Computer Science, 2*(1), pp.82-85.

Nakamura, Y. and Dev, S., 2022. Effects of Class-Size Reduction on Students' Performance. *Pertanika Journal of Social Sciences and Humanities*, 30(2), pp.797–812. doi: <https://doi.org/10.47836/pjssh.30.2.20>.

Oluwatosin, H.S., 2014. Client-Server Model. *IOSR Journal of Computer Engineering*, 16(1), pp.57–71. doi: <https://doi.org/10.9790/0661-16195771>

Osorio A, Esquivel M. A., 2020. A solution to the university course timetabling problem using a hybrid method based on genetic algorithms. DYNA, 87(215), pp.47–56. doi: <https://doi.org/10.15446/dyna.v87n215.85933>.

Oosthuizen, R. M., 2022. The Fourth Industrial Revolution – Smart Technology, Artificial Intelligence, Robotics and Algorithms: Industrial Psychologists in Future Workplaces. *Frontiers in Artificial Intelligence*, 5. doi: <https://doi.org/10.3389/frai.2022.913168>.

Palupi Rini, D., Mariyam Shamsuddin, S. and Sophiyati Yuhaniz, S., 2011. Particle Swarm Optimization: Technique, System and Challenges. *International Journal of Computer Applications*, 14(1), pp.19–27. doi: <https://doi.org/10.5120/1810-2331>.

Pambudi, R.A., Lubis, W., Saputra, F.R., Maulidina, H.P. and Wijayaningrum, V.N., 2019. Genetic Algorithm for Teaching Distribution based on Lecturers' Expertise. *Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control*, [online] pp.297–304. doi: <https://doi.org/10.22219/kinetik.v4i4.859>.

Patel, G.M., Jora, A., Singh, A., Arya, J. and Vashistha, B., 2021. Fault Tolerant Classroom Allocation System. *SSRN Electronic Journal*. doi: https://doi.org/10.2139/ssrn.3833703.

Rauf, F., Kalai, Y. and Adnan, Z., 2018. Course Allocation System for Lecturers. *International Journal of Computer Applications*, [online] 180(22), pp.9–14. doi: <https://doi.org/10.5120/ijca2018916344>.

Peter, B. and Ligembe, N., 2022. Impact of class size and students' academic performance in public secondary schools in Kwimba district Council , Mwanza – Tanzania. *Direct Research Journal of Education and Vocational Studies*, [online] 4(3), pp.109–122. Available at: <https://www.mendeley.com/catalogue/bf020591-e4ed-3256-94dd-c0f190e69442/> [Accessed 18 Aug. 2023].

Rouse, M., 2019. *What is Deterministic Algorithm? - Definition from Techopedia*. [online] Available at: <https://www.techopedia.com/definition/18830/deterministic-algorithm>.

Rossi-doria, O., Sampels, M., Birattari, M., Chiar, M., Dorigo, M., Gambardella, L. M. Knowles, J., Manfrin, M., Mastrolilli, M., Paechter, B., Paquete, L., Stützle, T., 2003. A Comparison of the Performance of Different Metaheuristics on the Timetabling Problem. Available at: <https://www.researchgate.net/publication/2561838_A_Comparison_of_the_Performance_of_Different_Metaheuristics_on_the_Timetabling_Problem>.

Safi, N., Almohawes, M. and Mohd Jamail, N.S. (2021). The impact of user involvement in software development process. *Indonesian Journal of Electrical Engineering and Computer Science*, 21(1), p.354. doi: <https://doi.org/10.11591/ijeecs.v21.i1.pp354-359>.

Shu-Chuan Chu, Yi-Tin Chen and Jiun-Huei Ho., 2006. Timetable Scheduling Using Particle Swarm Optimization. *First International Conference on Innovative Computing, Information and Control - Volume I (ICICIC'06)*. doi: <https://doi.org/10.1109/icicic.2006.541>.

SearchSoftwareQuality., n.d. *What is Web Application (Web Apps) and its Benefits*. [online] Available at: <https://www.techtarget.com/searchsoftwarequality/definition/Web-application-Web-app>.

Stacey, M., McGrath-Champ, S. and Wilson, R., 2023. Teacher attributions of workload increase in public sector schools: Reflections on change and policy development. *Journal of Educational Change*. doi: <https://doi.org/10.1007/s10833-022-09476-0>.

Tassopoulos, I. X. and Beligiannis, G. N., 2012. Solving effectively the school timetabling problem using particle swarm optimization. *Expert Systems with Applications*, 39(5), pp.6029–6040. doi: <https://doi.org/10.1016/j.eswa.2011.12.013>.

Trauth-Goik, A., 2020. Repudiating the Fourth Industrial Revolution Discourse: A New Episteme of Technological Progress. *World Futures*, pp.1–24. doi: <https://doi.org/10.1080/02604027.2020.1788357>.

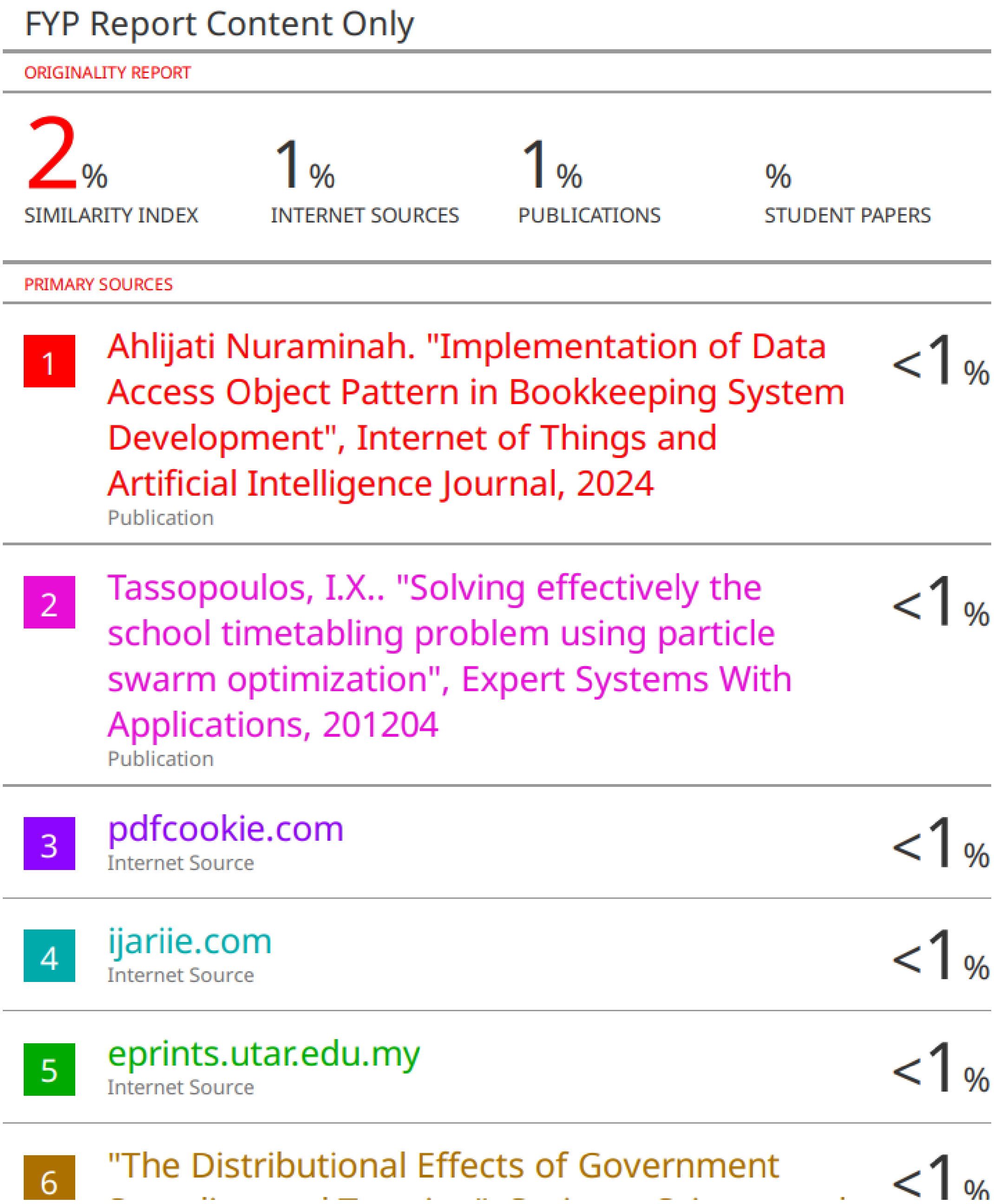
WinJaws5. (n.d.). 'winjaw5manual' [PowerPoint presentation]. Available at: <https://www.winjaws.com/Content/documents/winjaws5manual.pdf> [Accessed: 5 September 2014].

Williams, L., 2010. "Agile software development methodologies and practices". Advances in Computers (Vol. 80, pp. 1-44). Elsevier.

Xu, P. and Ramesh, B., 2008. Using Process Tailoring to Manage Software Development Challenges. *IT Professional*, 10(4), pp.39–45. doi: <https://doi.org/10.1109/mitp.2008.81>.

APPENDICES

Appendix A: Plagiarism Check Report Page 1



FYP Report Content Only

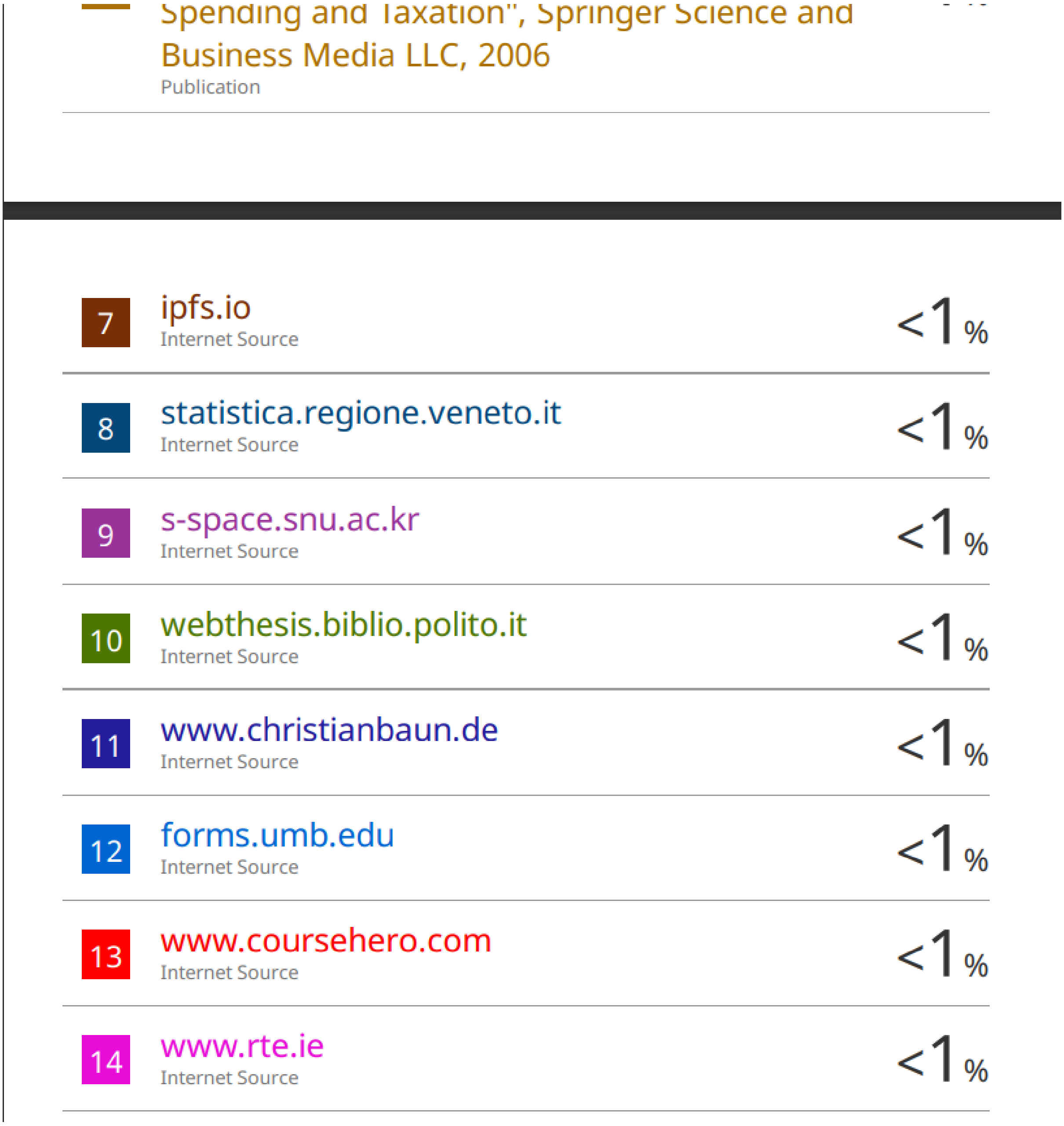
ORIGINALITY REPORT

2%	1%	1%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Ahlijati Nuraminah. "Implementation of Data Access Object Pattern in Bookkeeping System Development", Internet of Things and Artificial Intelligence Journal, 2024 Publication	<1%
2	Tassopoulos, I.X.. "Solving effectively the school timetabling problem using particle swarm optimization", Expert Systems With Applications, 201204 Publication	<1%
3	pdfcookie.com Internet Source	<1%
4	ijariie.com Internet Source	<1%
5	eprints.utar.edu.my Internet Source	<1%
6	"The Distributional Effects of Government	<1%

Appendix: B Plagiarism Check Report Page 2



Spending and Taxation", Springer Science and Business Media LLC, 2006
Publication

7	ipfs.io Internet Source	<1 %
8	statistica.regione.veneto.it Internet Source	<1 %
9	s-space.snu.ac.kr Internet Source	<1 %
10	webthesis.biblio.polito.it Internet Source	<1 %
11	www.christianbaun.de Internet Source	<1 %
12	forms.umb.edu Internet Source	<1 %
13	www.coursehero.com Internet Source	<1 %
14	www.rte.ie Internet Source	<1 %

Appendix C: FYP Poster

UNIVERSITI TUNKU ABDUL RAHMAN
LEE KONG CHIAN FACULTY OF ENGINEERING AND SCIENCE

SCHOOL TEACHER-SUBJECT ALLOTMENT MANAGEMENT SYSTEM (STSAMS)

PREPARED BY DENNIS YAP JIAN YUAN | SUPERVISED BY DR. TOO CHIAN WEN

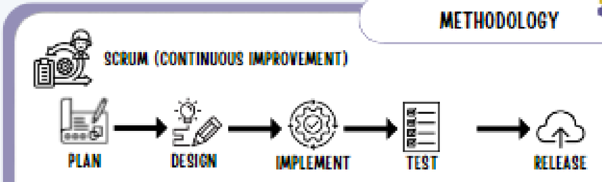
INTRODUCTION

As technology continues to evolve, the timetable systems for allocating school teachers to subjects have progressed from simply digitalizing the allotment process to fully automating it. However, despite these advancements, the timetabling solutions available in the market still fall short of perfection, leading to various issues that require addressing. In essence, the primary objective of this project is to identify the existing problems with the School Teacher Subject Allocation Management Systems (STSAMS) used in public schools. Then, the project aims to propose optimal solutions to enhance these systems, addressing any weaknesses or issues they may currently face.

OBJECTIVES

- To identify existing issues in the timetabling systems in the market and come out with solutions for them.
- To develop a web-based allotment system for teachers and subjects that solves the identified problems.
- To implement a meta-heuristic algorithm to complete tasks related to teacher-subject allotments.

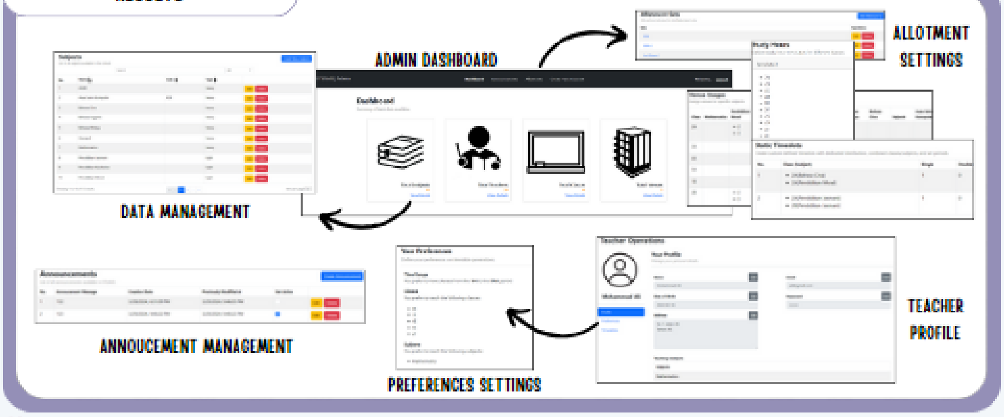
METHODOLOGY



TOOLS

React
MySQL
Bootstrap
Flask
draw.io
git
Resend

RESULTS



CONCLUSION

In a nutshell, the project transformed identified problems into improvements for a more robust STSAMS. However, it is also vital to take note of the opportunities for improvement, including capturing teaching scopes by utilizing digital image processing on resumes, adding more constraints to the allotment algorithm such as teacher's performance peer-to-peer rating system or considering teachers' travel distance between venues to ensure a higher quality of generated timetables.

UTAR
LKC