

Faculty of Natural and Mathematical Sciences

# Department of Informatics

Faculty of Natural and Mathematical Sciences

# Department of Engineering/Informatics

Faculty of Natural and Mathematical Sciences

# Department of Engineering/Informatics

Faculty of Natural and Mathematical Sciences

# Department of Engineering/Informatics

Faculty of Natural and Mathematical Sciences

# Department of Engineering/Informatics

Faculty of Natural and Mathematical Sciences

# Department of Engineering/Informatics

Faculty of Natural and Mathematical Sciences

# Department of Engineering/Informatics

Faculty of Natural and Mathematical Sciences

# Department of Engineering/Informatics

Faculty of Natural and Mathematical Sciences

# Department of Engineering/Informatics

Faculty of Natural and Mathematical Sciences

# Department of Engineering/Informatics

Faculty of Natural and Mathematical Sciences

# Department of Engineering/Informatics

Faculty of Natural and Mathematical Sciences

# Department of Engineering/Informatics

Faculty of Natural and Mathematical Sciences

# Department of Engineering/Informatics

Faculty of Natural and Mathematical Sciences

# Department of Engineering/Informatics

Faculty of Natural and Mathematical Sciences

# Department of Engineering/Informatics

Faculty of Natural and Mathematical Sciences

# Department of Engineering/Informatics

Faculty of Natural and Mathematical Sciences

# Department of Engineering/Informatics

Faculty of Natural and Mathematical Sciences

# Department of Engineering/Informatics

Faculty of Natural and Mathematical Sciences

# Department of Engineering/Informatics

Faculty of Natural and Mathematical Sciences

# Department of Engineering/Informatics

Faculty of Natural and Mathematical Sciences

# Department of Engineering/Informatics

Faculty of Natural and Mathematical Sciences

# Department of Engineering/Informatics

Faculty of Natural and Mathematical Sciences

# Department of Engineering/Informatics

Faculty of Natural and Mathematical Sciences

# Department of Engineering/Informatics

Faculty of Natural and Mathematical Sciences

# Department of Engineering/Informatics

Faculty of Natural and Mathematical Sciences

# Department of Engineering/Informatics

Faculty of Natural and Mathematical Sciences

# Department of Engineering/Informatics

Faculty of Natural and Mathematical Sciences

# Department of Engineering/Informatics

Faculty of Natural and Mathematical Sciences

# Department of Engineering/Informatics

Faculty of Natural and Mathematical Sciences

# Department of Engineering/Informatics

Faculty of Natural and Mathematical Sciences

# Department of Engineering/Informatics

Faculty of Natural and Mathematical Sciences

# Department of Engineering/Informatics

King's College London, Strand Campus, London, United Kingdom

King's College London, Strand Campus, London, United Kingdom

King's College London, Strand Campus, London, United Kingdom

King's College London, Strand Campus, London, United Kingdom

King's College London, Strand Campus, London, United Kingdom

King's College London, Strand Campus, London, United Kingdom

King's College London, Strand Campus, London, United Kingdom

King's College London, Strand Campus, London, United Kingdom

King's College London, Strand Campus, London, United Kingdom

King's College London, Strand Campus, London, United Kingdom

King's College London, Strand Campus, London, United Kingdom

King's College London, Strand Campus, London, United Kingdom

King's College London, Strand Campus, London, United Kingdom

King's College London, Strand Campus, London, United Kingdom

King's College London, Strand Campus, London, United Kingdom

King's College London, Strand Campus, London, United Kingdom

King's College London, Strand Campus, London, United Kingdom

King's College London, Strand Campus, London, United Kingdom

King's College London, Strand Campus, London, United Kingdom

King's College London, Strand Campus, London, United Kingdom

King's College London, Strand Campus, London, United Kingdom

King's College London, Strand Campus, London, United Kingdom

King's College London, Strand Campus, London, United Kingdom

King's College London, Strand Campus, London, United Kingdom

King's College London, Strand Campus, London, United Kingdom

King's College London, Strand Campus, London, United Kingdom

King's College London, Strand Campus, London, United Kingdom

King's College London, Strand Campus, London, United Kingdom

King's College London, Strand Campus, London, United Kingdom

King's College London, Strand Campus, London, United Kingdom

King's College London, Strand Campus, London, United Kingdom

King's College London, Strand Campus, London, United Kingdom

7CCSMPRJ

Individual Project Submission 2022/23

Name: Tangsheng Geng

Student Number: 19032422

Degree Programme: MSc Advanced Software Engineering

Project Title: A booking system for teaching assistants’ office hours with a dynamic resource management mechanism

Supervisor: Dr Leonardo Magela Cunha

Word count: 14900

RELEASE OF PROJECT

Following the submission of your project, the Department would like to make it publicly available via the library electronic resources. You will retain copyright of the project.

I **agree** to the release of my project

I **do not** agree to the release of my project

**Signature:**

文本, 信件

描述已自动生成

Date: 2023/08/14



Department of Informatics

King’s College London

United Kingdom

7CCSMPRJ MSc Project

A booking system for teaching assistants’ office hours with a dynamic resource management mechanism

Name: Tangsheng Geng

Student Number: 19032422

Degree Programme: MSc Advanced Software Engineering

Supervisor’s Name: Dr Leonardo Magela Cunha

This dissertation is submitted for the degree of MSc in MSc Advanced Software Engineering

Acknowledgement

Many thanks to my supervisor, Dr Leonardo Magela Cunha, for his feedback, assistance and suggestions on this project.

Abstract

The teaching assistants: students who concurrently take on an additional role of supporting and aiding in educational activities within the same institution where they are pursuing their studies are widely hired in computer science higher education. The fact that teaching assistants are students means that they could more easily understand the problems and they are more approachable to students than professors. It is proven that employing teaching assistants could improve students’ experiences and grades.

However, the student identity of teaching assistants also gives rise to problems. According to research, teaching assistants face various issues when conducting their duties. This report aims to discover the difficulties teaching assistants face when performing one of their major duties: holding office hours, and develops a piece of software that solves these problems[1][2].

This report summarises the problems that teaching assistants face when holding office hours and studies existing solutions to similar problems. Based on the requirements gathered, a booking system with a dynamic resource management mechanism is designed and implemented as a solution to the problems.

The designs and algorithms used in the system have been thoroughly discussed and compared. The implementation has been compared with the functional requirements gathered. The system has been evaluated and concluded as theoretically capable of solving the problems. The system should be deployed and tested in real-life scenarios before a statistical conclusion on whether the system actually solves the problems.

Nomenclature

*TA* Teaching Assistant.

*DAO* Data Access Objects, which are objects responsible for interacting with tables in the database.

*PO* Persistence Objects, which are objects whose properties match the column of a certain table. One PO matches one row in a specific table.

*DTO* Data Transfer Objects, which are objects being used to encapsulate and transfer data among different layers of a system.

*CRUD* Create, Read, Update, Delete.

Contents

[1 Introduction 7](#_Toc142950196)

[1.1 Background 7](#_Toc142950197)

[1.2 Project aims and objectives 8](#_Toc142950198)

[1.3 Discovering the problems 8](#_Toc142950199)

[1.4 Review of solutions to similar problems 9](#_Toc142950200)

[1.4.1 A booking system 9](#_Toc142950201)

[1.4.2 A resource management system 11](#_Toc142950202)

[1.5 Dissertation structure 13](#_Toc142950203)

[2 Background Theories 14](#_Toc142950204)

[3 Requirements and Design 16](#_Toc142950205)

[3.1 System requirements 16](#_Toc142950206)

[3.2 The choice of the form of the software 17](#_Toc142950207)

[3.3 General system designs 17](#_Toc142950208)

[3.3.1 Software structure design 18](#_Toc142950209)

[3.3.2 Resource group design 18](#_Toc142950210)

[3.3.3 Resource allocation design 21](#_Toc142950211)

[3.3.4 Request queue design 22](#_Toc142950212)

[3.3.5 Office hours design 22](#_Toc142950213)

[3.3.6 Use case diagrams 23](#_Toc142950214)

[4 Methodology and Implementation 25](#_Toc142950215)

[4.1 Overview of the system 25](#_Toc142950216)

[4.1.1 Database overview 26](#_Toc142950217)

[4.1.2 Persistence objects overview 27](#_Toc142950218)

[4.1.3 Data access objects overview 27](#_Toc142950219)

[4.1.4 Dependency management overview 28](#_Toc142950220)

[4.1.5 Frontend and controller overview 29](#_Toc142950221)

[4.2 Subsystems 30](#_Toc142950222)

[4.2.1 Identity authentication and management subsystem 30](#_Toc142950223)

[4.2.2 Project properties subsystem 31](#_Toc142950224)

[4.2.3 Resource group subsystem 32](#_Toc142950225)

[4.2.4 Appointments and requests subsystem 32](#_Toc142950226)

[4.2.5 Student management subsystem 33](#_Toc142950227)

[4.2.6 Student priority mechanism and reflections 34](#_Toc142950228)

[4.2.7 Teaching Assistant Management subsystem 36](#_Toc142950229)

[4.2.8 Work flow of the use case: student make an appointment 37](#_Toc142950230)

[4.2.9 Automated Appointment and Request Update subsystem 38](#_Toc142950231)

[4.2.10 Automated Teaching Assistants Update subsystem 42](#_Toc142950232)

[4.2.11 Reflections on the Automated Resource Allocation Mechanism 46](#_Toc142950233)

[5 Results, Analysis and Evaluation 48](#_Toc142950234)

[5.1 Testing of the system 48](#_Toc142950235)

[5.2 Results 48](#_Toc142950236)

[5.2.1 Technical result 48](#_Toc142950237)

[5.2.2 Project Aims and Objectives 49](#_Toc142950238)

[5.2.3 Research question 49](#_Toc142950239)

[5.3 Evaluation and discussion 49](#_Toc142950240)

[6 Professional and Ethical Issues 51](#_Toc142950241)

[7 Conclusion and Future Works 52](#_Toc142950242)

[8 References 53](#_Toc142950243)

[9 Appendices 55](#_Toc142950244)

[Appendix A: Source code structure 55](#_Toc142950245)

[Appendix B: Screenshots of the user interface 57](#_Toc142950246)

[Admin portal screenshots: 57](#_Toc142950247)

[Student portal screenshots 60](#_Toc142950248)

[Teaching assistant portal 63](#_Toc142950249)

[Appendix C: Source code of core services 65](#_Toc142950250)

[AutomatedRequestsAndAppointmentsUpdateService 65](#_Toc142950251)

[AutomatedTeachingAssistantsUpdateService 67](#_Toc142950252)

List of Figures

Figure 1: the LibCal booking system[7]

Figure 2: the Alaska Airlines seat map[8]

Figure 3: motorway stretch with a contraflow buffer zone[9]

Figure 4: the Moscow list

Figure 5: software structure design

Figure 6: resource group design

Figure 7: design without resource group

Figure 8: resource allocation when demand surges

Figure 9: request queue

Figure 10: administrator use cases

Figure 11: student use cases

Figure 12: teaching assistant use cases

Figure 13: automated service use cases

Figure 14: system architecture

Figure 15: database architecture

Figure 16: a table and its corresponding PO

Figure 17: a data access object and its corresponding implementation

Figure 18: comparison of dependency management

Figure 19: front end JavaScript file and corresponding controller

Figure 20: identity authentication and management subsystem tables

Figure 21: project properties subsystem tables

Figure 22: resource group subsystem tables

Figure 23: appointments and requests subsystem tables

Figure 24: student management subsystem tables

Figure 25: an example of access to office hours

Figure 26: teaching assistant management subsystem tables

Figure 27: the steps for a student to book an appointment

Figure 28: an example aspect class

Figure 29: pseudocode on removeObsoleteRequestsAndAppointments

Figure 30: pseudocode on checkAndUpdateRequestQueue

Figure 31: comparison with memory allocation

Figure 32: pseudocode on updateTeachingAssistantAvailabilityStatus

Figure 33: pseudocode on updateUnreachableTeachingAssistantAvailableTimes

Figure 34: pseudocode on refreshTeachingAssistantAvailabilityStatus

Figure 35: pseudocode on checkAndAddTeachingAssistantResourceGroup

Figure 36: pseudocode on checkAndRemoveTeachingAssistantResourceGroup

Figure 37: review of requirements

1 Introduction

## 1.1 Background

Teaching assistants are students who concurrently take on an additional role of supporting and aiding in educational activities within the same institution where they are pursuing their studies. They are extensively employed in computer science courses as a means to manage high enrolment while being able to offer students individual tutoring experiences. Teaching assistants have been identified as a contributing element to students’ achievements in their studies because of the following facts:

1. TAs themselves are also students, and they are more approachable than professors or lecturers.

2. TAs have studied similar courses, and they understand what problems or misconceptions the student might face based on their experience[1].

The conclusion that TAs can be of positive impact on students’ studies is agreed by Pivkina, who conducted research that compares students’ performances in different semesters of the same course in the field of computer science with or without peer learning assistants(In Pivkina’s article, undergraduate students who work as teaching assistants are defined as peer learning assistants. The definition of peer learning assistant in Pivkina’s report is similar to the definition of teaching assistant in this dissertation). The research concluded that:

1. With the presence of peer learning assistants, student scores in the course increased

2. With the presence of peer learning assistants, the occurrence of failing grades decreased

3. Students are more willing to use peer learning assistants’ office hours than graduate teaching assistants’ or instructors’ office hours.

4. Students reported positive experiences when interacting with peer learning assistants.

Aside from the positive impact on students, the research also discovered that the use of peer learning assistants benefits peer learning assistants themselves, graduate teaching assistants and instructors[2]. Based on the above research, it can be concluded that the application of teaching assistants is of great importance in computer science higher education.

However, the fact that the teaching assistants are students leads to many challenges and difficulties when they are fulfilling their duties, which commonly involve holding office hours, conducting labs, and grading students’ assignments[2]. According to the research conducted by Riese, TAs reported challenges in becoming professional TAs, interacting with students, making assessments, and adopting best practices[1]. The author of this dissertation had experience working as a teaching assistant during his undergraduate and had experienced some of the challenges in person. The project aims to develop a piece of software to tackle the obstacles TAs face to improve the overall experience in tutoring with TAs for both students and TAs. Considering the time and resource constraints, the project focuses on challenges and problems relevant to one of the TA’s primary duties: holding office hours. The research questions of this dissertation are:

1. In computer science higher education, what are the problems that TAs face when holding their office hours?

2. What functionalities should the software developed possess to solve these problems?

## 1.2 Project aims and objectives

**Project aim:**

In computer science higher education, discover the problems TAs face when holding office hours, and develop a piece of software that solves these problems.

**Project objectives:**

Read articles to discover the problems TAs in computer science higher education face when holding office hours and summarise these problems.

After concluding the problems, read articles and research how similar problems were solved.

Determine the requirements and the design of the software. Develop a piece of software based on the information gathered.

After development work is finished, analyse the software to check whether the specified requirements have been achieved, then perform testing on the software.

## 1.3 Discovering the problems

Based on relevant research and articles, three common problems that TAs face when holding office hours have been concluded:

**Teaching assistants may find it challenging when trying to answer more personalised questions from students**

According to research conducted by Malysheva et al. on TAs in a computer science course, TAs are generally able to provide satisfying results when the students require generalised assistance, with the success rate of these sessions being 87.5%. For example, when students ask about a homework solution, the TAs could easily assist them by guiding them through a well-prepared solution. However, many TAs failed to offer the students clarification when they required debugging their own code or providing a conceptual explanation. In these scenarios, a success rate of only 57% is witnessed[3].

**Teaching assistants may feel time pressure when students are queueing for support, especially when a course approaches its deadlines**

During office hours, there may be several students queueing to be seen, and the TAs may feel the time pressure of trying to help all the students. These pressures probably heightened the cognitive burden on TAs, pushing them to opt for the quickest and easiest approach to tackle the student's problems, which may have a negative impact on the quality and outcome of a session[3]. This phenomenon is agreed by research on TAs’ lab sessions(TAs’ workflow of lab sessions is similar to office hours, where in both scenarios, the TAs wait for students to visit with questions), which reported TAs feel rushed when the queue of students become too long especially when deadline days approach[4]. An analysis of the students’ visits to undergraduate teaching assistants' office hours in computing courses by Ko and Stephens-Martinez further verified that the number and percentage of students' interactions with teaching assistants surged three days before the due date of a course[5].

**Teaching assistants may find it difficult to allocate time to different students fairly**

In the research by Riese et al., teaching assistants reported challenges in balancing time allocation, prioritising students requiring assistance, evaluating students who prefer presenting their solutions, and providing attention to advanced students[1]. It has also been discovered that teaching assistants might provide more assistance to female students than male students[4]. An analysis of computing classes also showed that a small group of highly active students, comprising 10-20% of the total, consumed approximately 50% of the TAs’ available office hours[5].

## 1.4 Review of solutions to similar problems

Based on the summarised problems, the author researched and discovered that two forms of solutions could effectively solve the problems: A booking system and a resource management system. While these systems might be the same thing in some contexts, the author would like to clarify the difference between them in this dissertation: The booking system allows the user or consumer to lock on some resources that are available to them, while a resource management system specifies which user, or a group of users have access to which resources. Different applications of these systems are presented and discussed in this section.

### 1.4.1 A booking system

With the presence of booking systems, TAs no longer need to worry about time pressure incurred from students waiting to be seen or try to allocate time fairly to students with different needs because bookings determine their schedules. The system could also possess the functionality of informing the TAs about what to expect in their sessions, thus allowing them to be more prepared prior to the actual meeting.

**KCL LibCal**

This library booking system allows students to book study spaces in KCL libraries. Students can select from a range of available rooms and times and use their KCL email addresses to finish the booking.

表格

描述已自动生成

Figure 1: the LibCal booking system[7]

The figure 1 shows a typical time slot booking system that allows students to make bookings on a first-come, first-serve basis. The TAs’ office hour resources are similar to the study spaces’ availability. The advantage of this booking system is that all the available rooms and hours are clearly presented to the users, making it easy to use. But in systems like these, where all users are treated equally, students’ personal needs are ignored as there may be students with a disability or other illnesses that need some priority. Moreover, if a booking system like this is implemented on office hours, TAs may not be aware of the problems the students are encountering before meeting in person and may be less prepared, thus less likely to offer students satisfying support.

**Alaska Airlines seat selection**

The seat selection system is commonly used when travellers are booking flights, and it allows the travellers to click on the seat in the seat map and lock the seat.

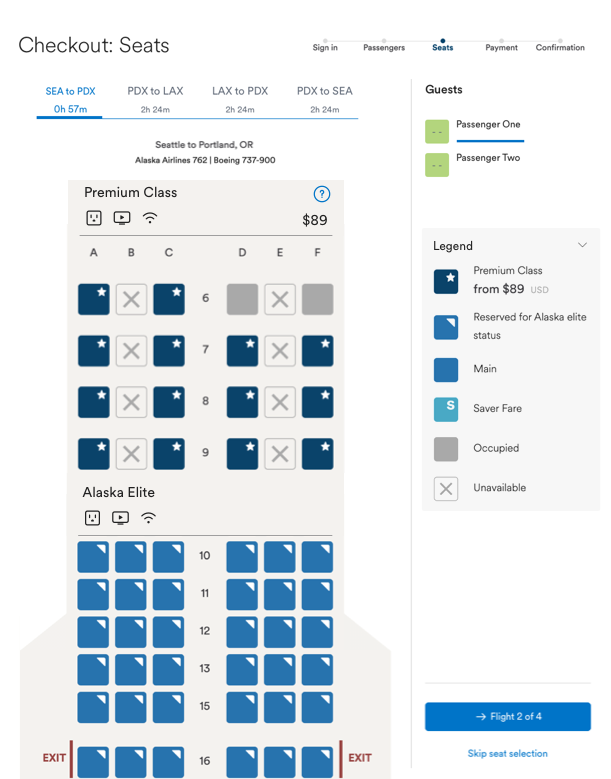


Figure 2: the Alaska Airlines seat map[8]

As we can see from the figure 2, some seats belong to premium classes, which the user can only select after paying extra money. There are also seats reserved for Alaska elite status members. In long-distance international flights, users could also choose their meal preferences or specify their personal needs when booking.

Such booking systems take into account of personal needs and implement prioritisation mechanisms for different users. However, this prioritisation mechanism may not directly apply to an office hour booking system. The booking of office hours is a non-profit process, so there must not be any mechanism where students pay some money to get seen by the TAs early. Regarding students requiring some priority, it should be remembered when designing the systems that we are trying to offer all students equal support rather than giving some students privileges. Neither students with priority nor other students should feel the difference in their statuses when making bookings.

### 1.4.2 A resource management system

The problem of surging needs for TAs’ office hours near the deadline is the reflection of Parkinson’s law, which states that “work expands so as to fill the time available for its completion”. According to research by Gutierrez and Kouvelis, the most effective solution to this problem would be establishing suitable and reasonable deadlines to reduce project activity completion delays[6]. While software cannot persuade professors to set appropriate deadlines(we cannot reduce the number of needs from students or eliminate the surge in demands near deadlines), this problem can be characterised as the allocation of limited resources(TAs’ office hours) on consumers(students), whose needs change drastically in different periods.

**Linux file permission system**

In the Linux operating system, each file in the system may be accessed by three types of users: file owners, file owner groups, and others. Each type of user may or may not be privileged to read, write, or execute the file. The user type “group” is a collection of users. Establishing and handling groups in Linux is a straightforward method to manage multiple users simultaneously, especially when managing file permissions.

The author discovered significant similarities between the Linux file permission system and the structure of many higher education courses. The files in Linux and the TAs’ office hours can be considered resources; the users in Linux and the students can be regarded as resource-consumer. In Linux, each user belongs to several groups, and some files are only accessible to specific groups. In universities, students enrolled in a course belong to various modules, and each module contains TAs that are only accessible to students within the same module. We may adopt Linux’s permission system by managing students’ access to TAs based on their modules.

**Motorway tidal flow control**

****

Figure 3: motorway stretch with a contraflow buffer zone[9]

The above figure 3 shows a bi-directional motorway with a buffer zone in the middle. During morning rush hours, suppose most people travel from left to right to get to work; the buffer zone could be allocated to the upper region and becomes a rightward lane. During evening rush hours, when the majority of people travel from right to left, the buffer zone then becomes a leftward lane. It has been shown that lanes with control mechanisms like this see an average of 57.3% improvement in speed and a 48.4% drop in total travel time[9].

The travellers in this scenario correspond to the students in our problem, and the lanes correspond to the TAs’ time resources. Assuming the total number of teaching assistants in the computer science department remains the same, when one specific module in a particular year comes close to a deadline and the number of students seeking help from TAs surges, we may allocate additional TAs from other similar modules, or more preferably TAs from different years to cope with this increase in demand.

Other possible strategies include hiring temporary TAs for students before deadlines or dynamically allocating the TAs’ working hours so that they work less during term time and work more when close to deadlines.

**Resource allocation algorithms**

After the structure and the working pattern of a resource management system are determined, the actual algorithms on how the system dynamically allocates the resources require investigation and comparison. That is, when the time approaches a module’s deadline and the need for office hours from that module surges, how should the system allocate resources from other modules to produce an optimal outcome? They will be presented in the later sections of this dissertation. The system should also possess the functionality for the administrator or instructor to manage access to resources manually, as the course administrator may have custom resource allocation strategies.

Following the above research and discussions, the author decided to implement a system that consists of booking and resource management functionalities. Recalling the project's aim of discovering and solving problems faced by TAs, the author named the project a booking system with a resource management mechanism, as from the TA’s perspective, the software developed and presented is a booking system.

## 1.5 Dissertation structure

The length of this report is 14547 words.

Chapter 1 states the background, aim and objectives of this project, followed by the general introduction of the problems that TAs are facing and the presentation of existing solutions to similar problems.

Chapter 2 introduces the background theories used in this dissertation.

Chapter 3 covers the requirements gathered on the system to develop, along with the design

choices on some of the subsystems.

Chapter 4 explains the implementations and methodologies of the system in detail.

Chapter 5 presents the results achieved by this project and the evaluations on the results.

Chapter 6 summarises the professional and ethical issues related to this project.

Chapter 7 concludes the project and presents future work to be done.

2 Background Theories

**Builder design pattern**

The Builder design pattern is a design pattern that separates the construction of complex objects into sub-procedures. Classes may consist of many parameters, resulting in complex constructors. Builder pattern solves this problem by offering methods to construct the object step-by-step, each constructing a part of the object and providing a method that returns the final Object[10].

**Proxy design pattern**

The Proxy design pattern is a design pattern that allows one class to act as a substitute for another class. It provides a way to control access to the actual object, add additional behaviour to the objects’ methods, or delay the creation of the real object until it is needed. The pattern involves a proxy class that implements the same interface as the original class. The developer may define extra behaviour in the proxy class’s methods in addition to the invocation of the original class’s methods[11].

**Spring and Spring Boot**

The Spring Framework is an application framework for Java. It provides a broad range of features and functionalities to facilitate the development of loosely coupled, modular, and maintainable applications. The core features of the Spring framework include IOC and AOP, which are explained below[12].

Spring Boot is an extension of the Spring Framework designed to simplify the configuration and set-up of Spring applications. Spring Boot follows “convention over configuration, " relying on predefined conventions and assumptions rather than requiring explicit configuration for every aspect. Applications built with SpringBoot include an embedded web server, making deployment and distribution simple[13].

**Spring Inversion of Control(IoC)**

Spring IoC (Inversion of Control) is a core feature of the Spring Framework. It is also known as Dependency Injection (DI), which helps manage the flow of control and dependency resolution in an application. In traditional programming, when an object relies on another object or requires specific resources, it explicitly creates or looks up those dependencies. This leads to tightly coupled programs, making the system difficult to maintain, test, or modify. With the application of Spring IoC, the control over the creation and management of objects or dependencies is inverted or handed over to the Spring container. Instead of objects explicitly creating or finding their dependencies, the Spring container provides the required dependencies to the objects during their instantiation of the application[14].

**Spring Aspect Oriented Programming(AOP)**

Spring AOP (Aspect-Oriented Programming) is another core feature of the Spring Framework that enables developers to implement cross-cutting concerns in a modular and reusable manner. Cross-cutting concerns, such as logging, security, transaction management, and caching, are required in multiple places throughout the application. Spring AOP addresses this concern by allowing developers to separate these cross-cutting concerns from the main business logic, thereby achieving better code modularity and maintainability. The location where the cross-cutting concerns are inserted in the program is called “pointcut”, and cross-cutting logic is encapsulated in “aspect”. Spring AOP uses the Proxy design pattern to achieve aspect-oriented programming. It dynamically creates proxy objects that wrap the target objects and apply aspects before, after, or around their method invocations[15].

**MyBatis**

MyBatis is a Java persistence framework that simplifies database access for Java applications. It provides a way to interact with relational databases using SQL queries without requiring developers to write low-level JDBC code explicitly. Developers can use XML configuration files or annotations to map Java objects to database tables[16].

**ACID properties of transactions**

**Atomicity**: The transaction is treated as a single unit. The operations in the transaction are executed only when no single failure or error are detected, else no operation is executed.

**Consistency**: The state before and after a transaction of a database should be consistent.

**Isolation**: One transaction should not be affected by any other transaction, even when multiple transactions are happening simultaneously.

**Durability**: After the transaction is committed, any change made to the data are permanent and not affected by any subsequent system failure.

**jQuery**

jQuery is a JavaScript library. It contains many useful methods that could be used when dealing with data manipulation, event handling, showing animations, and sending ajax requests (which allows updating the webpage without refreshing the page)[17].

**Bootstrap**

Bootstrap is a CSS framework that provides pre-designed and customizable styles and components[18].

**Lombok**

Lombok is a Java library that automatically generates the methods that are repetitive, for example the getters, setters, and constructor of a Java class, with the use of annotations. The use of Lombok produces cleaner and more concise code, and allows the programmers to focus on the business logic.

Some of the Lombok annotations used in the project include:

**@AllArgsConstructor**: generate a constructor that contains all the attributes of this class.

**@NoArgsContructor**: generate a constructor that contains no attributes.

**@Data**: generate a series of methods, which includes getters on all fields, setters on all non-final attributes, and the “toString” method for this class[19].

3 Requirements and Design

## 3.1 System requirements

The following Moscow list of requirements of the system is gathered:

文本

中度可信度描述已自动生成

Figure 4: the Moscow list

Most of the requirements in figure 4 are either basic requirements on a booking or resource management system or have been discussed and justified in section 1 of this dissertation. Some of the requirements choices are further explained in this section:

**Why there isn’t a functionality requirement to allow TAs to manage or cancel their appointments?**

As mentioned in section 1, holding office hours is one of the primary duties of TAs, and we assume TAs have signed contracts and agreed to hold office hours for a certain time every week. Moreover, with the booking system, the TAs can now view the contents of an appointment. If the TAs possess the privilege to cancel an appointment, they could reject any appointment that contains complex questions and only accept the easy ones. This contradicts the purpose of the functionality of displaying appointment information to the TAs, which is to allow them to become more prepared when answering students’ questions rather than refusing to answer them at all.

**Allow users to specify personal information such as age, gender, and others:**

It is common in commercial booking systems such as restaurant or transport bookings to allow users to create accounts and specify their personal information. However, this is not a major concern in building a functional booking system. The storage of personal data also raises ethical issues that require careful handling.

**An evaluation functionality for students to give feedback to their teaching assistants after sessions:**

According to the research by Riese, TAs reported that they rarely receive feedback from students or course managers on their performances as TAs[4]. Sending TAs feedback on their office hours allows TAs to understand what they have done well and what could be improved. Again, this invokes the handling of ethical issues, such as whether the feedback from students should be anonymous. This functionality cannot fit into a booking system and may require extra work to build an evaluation system.

**A caching framework to store contents that are less likely to change frequently, such as usernames:**

The application of caching frameworks such as Redis could significantly improve the system's performance, as it reduces the number of queries to the database. This is categorised as a “won’t have” request as it is complicated to implement, and it is not essential in a booking or resource management system.

## 3.2 The choice of the form of the software

The system could be implemented in two primary forms; one form is a web application, and the other form is a Moodle plugin.

Systems built as web applications have clear structures, as the front end and back end are clearly separated, and each file or program has a single responsibility. Web applications are highly scalable, and it can be easy to modify, maintain or extend functionalities.

Recalling that the project aims to solve the problems in higher education, as several universities use Moodle-based portals for course management and student services, a system built as a Moodle plugin could be directly integrated into many universities’ existing frameworks. It would be effortless for these universities to incorporate this system seamlessly with their other existing services.

From the usability and software distribution perspective, it may be more reasonable to develop a Moodle plugin rather than a web application. However, this is a project with time and resource constraints. The author had no experience in coding with PHP or creating Moodle plugins, and studying this knowledge occupies the time that could be spent on developing fully functional software. Even if the author had successfully acquired relevant knowledge after efforts, it is unlikely that the knowledge the author had gained on Moodle plugins would allow anything more than an elementary implementation. The author has experience in building complex Java web applications and is familiar with production-grade functionalities, designs, and frameworks, all of which could be of benefit in satisfying the gathered requirements more efficiently and effectively.

Therefore, the author decided that building a web application is more likely to result in functional, scalable, and maintainable software, thus better achieving the project aim of solving the problems TAs face when holding office hours in computer science higher education.

## 3.3 General system designs

In this part, the general designs of some core system features are illustrated and explained. The detailed implementations are introduced in chapter 4 of this dissertation.

### 3.3.1 Software structure design

图示

描述已自动生成

Figure 5: software structure design

The above figure 5 shows the structure of the software, which implements a typical Java web application three-layer design. The Data Access Layer consists of classes that are responsible for interacting with the database, who are able to manage data from the database and instantiate objects whose properties match tables’ columns. The Service Layer consists of classes in which the business logic is written. The Controllers Layer contains classes whose responsibilities are to receive requests from the front end (webpage) and respond to requests. This design ensures that each class undertakes a single responsibility, and changes in one layer’s functionalities do not affect any other layer, making the system modular, maintainable, and extensible.

As different education organisations may have distinct functional and non-functional requirements for the implementation of the software, extensibility and scalability are concerns that should be addressed when developing the system. Interfaces are widely used in the design, particularly in the service and data access layers. With the use of interfaces, the concepts and behaviours of a class are abstracted. Aside from the implementations provided by the author, future developers can develop their own implementations of the interfaces. Enumerable is also used when declaring the category of some objects, allowing users to add custom types without modifying any existing types or functionalities.

### 3.3.2 Resource group design

As mentioned in the introduction, the Linux file permission system and the structure of many higher education courses are very similar. Thus the following consumer-resource group-resources structure is formed:

图示

描述已自动生成

Figure 6: resource group design

As we can see from the above figure 6, each student belongs to one or more modules, and each module has hired several teaching assistants. At regular times, each student can only access the TAs that belongs to the modules they are studying. For example, student S1 only studies module A, so he may only access the office hours from teaching assistants T1 and T2. This design takes into consideration of the module system in universities, and by grouping students and TAs into resource groups, both the consumers’ and teaching assistants’ belongings and access rights can be easily managed.

A design without the resource group would directly consider and manage the relationships between resources(TAs’ office hours) and consumers(students):

图示

描述已自动生成

Figure 7: design without resource group

This design(figure 7) may be more straightforward and allows a simpler system structure. But as the number of students and TAs increases, this design needs to take into account of much more relationships than the design with resource groups.

Consider a scenario where we need to add a group of students into the system; we set the number of these students as , the number of existing TAs as , the number of resource groups as . Suppose these students are studying all the modules and we need to map them to all the teaching assistants.

The number of relationships we need to create in the first design with resource groups is

The number of relationships we need to create in the design without resource groups is

Since the number of modules in a course is likely to be far less than that of TAs, we conclude:

Thus although the first design increases system complexity, it allows much higher maintainability and scalability as the number of TAs or students increases. The first design also addresses concerns such as adding students to specific modules, which is far more complicated in a design without resource groups.

### 3.3.3 Resource allocation design

Recalling one of the major problems TAs face on office hours is when the deadline of a course approaches, the number of students queueing for office hours surges, and it becomes almost impossible to help all the students. The following resource allocation mechanism is implemented to solve the problem:

图示

描述已自动生成

Figure 8: resource allocation when demand surges

In the above scenario(figure 8), suppose course A has coursework that approaches its deadlines and the surge in the number of requests in group A is detected. The system automatically allocates teaching assistants T3 and T4 to group A (see orange arrows). Students in group A can now access more teaching assistants and their office hours. When the system decides that there is no need to allocate extra TAs to group A, T3 and T4’s resource groups are modified so that they are no longer accessible by students from group A (orange arrows removed).

This allocation process can also be done manually by the administrator(the course manager), who also has the option to disable this automated allocation process at all.

The modules or groups in a system should be modules within a specific department, and in this project, the scope of the modules is within the computer science department. The point to consider is when dynamically allocating TAs, they should understand the knowledge of or have studied the module that they are being assigned to.

### 3.3.4 Request queue design

An essential requirement to implement the above resource allocation mechanism is the successful detection of a surge in demands. A request queue is implemented for this detection:

图片包含 表格

描述已自动生成

Figure 9: request queue

When students cannot find any available TAs’ office hours to book an appointment successfully, they can submit a request, which contains information on the appointment and the group(module)in which they are trying to make an appointment(figure 9). The status of this request queue is automatically checked when a new request is added to the queue. When the number of requests of a particular group becomes higher than a threshold, the automated resource allocation mechanism is triggered. The status of this request queue is also viewable by the administrator, which helps them decide which groups may request extra TAs when they are manually managing the resources.

The request queue also contains other automated mechanisms. For example, when a new teaching assistant is added to the system, it checks all the requests in the queue to determine if any existing request could be satisfied and turned into an appointment with that TA.

A possible alternative design in the detection of requirement fluctuations would be only allowing students to report which module fails to provide enough teaching assistants and recording only the module names and the number of cases in each module. Obviously, this design allows the detection of changes in demands and reduces system complexity. But considering the system is a combination of a booking system and a resource management system, this design only satisfies the functional requirements of a resource management system and completely ignores the needs of the booking system. The first design of a request queue provides the functionalities that meet the needs of both systems.

### 3.3.5 Office hours design

The TAs’ office hours are stored, managed, and refreshed on a weekly basis. This design is based on the United Kingdom’s Tier 4 visa regulation that the number of hours international students are allowed to work is monitored and managed by week. As most universities have international students studying and working as TAs, the universities usually specify the number of hours a TA works per week in the contract[20][21][22].

The TAs can specify their office hours with the format: weekday, which hour of that weekday, and which minutes in that hour. The value stored in the system is a string with the format "weekday\_hour\_tenMinuteIntervalOfAnHour", where weekday is an integer between 1 and 7 inclusive, hour is an integer between 1 and 24 inclusive, and tenMinuteIntervalOfAnHour is an integer between 1 and 6 inclusive. For example, 01\_08\_02 represents the office hour on Monday, 8 o’clock, with the time interval of 8:10 to 8:20. The minimum unit of office hours is set to ten minutes. This is based on the research by Malysheva, Allen, and Kelleher on TAs’ office hours, which discovered the average session time is 10 minutes and 20 seconds[3].

In the current design, TAs don’t need to manage their office hours every week after setting them up for the first time, and they are automatically handled and refreshed by the system. The current system implementation specifies that the end of Friday represents the end of a week, and performs relevant actions at that moment. This design eases the TAs' workloads in scenarios where TAs weekly office hours are fixed. If any office hour needs to be changed, TAs can add, update, remove their office hours, or mark them as unavailable at any time.

### 3.3.6 Use case diagrams

There are four types of “users” for this system, which are administrator, student, teaching assistant and automated service. Their use cases are described below:

图示

描述已自动生成

Figure 10: administrator use cases

图示

描述已自动生成

Figure 11: student use cases

图示

描述已自动生成

Figure 12: teaching assistant use cases

图示

描述已自动生成

Figure 13: automated service use cases

4 Methodology and Implementation

## 4.1 Overview of the system

图示

描述已自动生成

Figure 14: system architecture

The above figure 14 shows the structure of the system.

At the bottom of this figure are the database tables. The database used in this system is MySQL database.

Above the tables, we can see the data access objects, each responsible for interacting with their corresponding tables. These DAOs are created with the support of the MyBatis framework.

They are invoked by service layers, in which the business logic is written. There are two automated services on the right that invokes other services and are not called by any controller.

On top of the service layer are controllers. The controllers receive and respond to requests sent from the front end. As can be seen from the controllers’ names, each is responsible for one or more use cases of a type of user. These controllers invoke relevant services to achieve these tasks.

Spring Boot framework is used when building the data access layer, service layer, and controller layer of the system. The framework offers support in creating and assigning objects, allowing objects from each layer to cooperate closely with each other but not tightly coupled in code.

The system’s front end is achieved using HTML, CSS, and JavaScript, with the extensive use of two libraries: Bootstrap and jQuery.

The detailed implementation of the system shall be explained in the following parts:

In part 4.1, an overview of the system will be given, with explanations of core concepts and technologies.

In part 4.2, the subsystems and relevant discussions are presented.

Appendix A shows the structure of the source code.

Appendix B shows the screen shots of all the user interfaces.

Appendix C is the source code of two major services of this system, that is AutomatedRequestsAndAppointmentsUpdateService and

AutomatedTeachingAssistantsUpdateService.

### 4.1.1 Database overview

图形用户界面, 图示

描述已自动生成

Figure 15: database architecture

The above figure 15 shows the database design for this system. Orange key icons represent primary keys, blue key icons represent foreign keys, and the arrows indicate the keys that these foreign keys have referenced.

The database of this system is implemented as a MySQL relational database. A relational database has the following major advantages that are beneficial to be applied in this project.

1. It is able to handle complex data relationships and ensure data integrity through the use of foreign keys. As is seen from the above figure, different types of data are highly relevant to each other, and a relational database is ideal in this scenario.

2. It guarantees the ACID properties of transactions. There are several cases in the system where multiple tables are changed within the same method; transactions are used in these scenarios to ensure data integrity(see methods decorated with annotation @Transactional).

3. It is highly scalable and maintainable. Relational databases’ storing engines allow storage, query, and update of large amounts of data, which is suitable for universities.

### 4.1.2 Persistence objects overview

Each table in the database(except for project\_properties) has a persistence object in the system that maps all the table’s columns. An instance of a persistence object equals a column in a table. All the persistence objects are placed in the “po” package and follow the naming convention: The PO’s name capitalizes the first character of the table and any character after an underscore, then removes all the underscores and the plural form. The following figure shows one of the persistence objects: Request.

图形用户界面, 文本, 应用程序

描述已自动生成

Figure 16: a table and its corresponding PO

As shown in figure 16, the type and the name of a column are “translated” into corresponding Java properties. The Lombok annotations placed at the top of the Request class also provide getters, setters, and constructors for this class. In the rest of this dissertation, the author will not mention persistence objects explicitly unless in scenarios requiring extra clarification.

### 4.1.3 Data access objects overview

The objects responsible for retrieving or updating data in tables are DAOs (data access objects), which offer various CRUD methods for their corresponding tables. Each table has a corresponding DAO. The DAOs in this system are interfaces whose implementations are provided by MyBatis. The implementations of DAO interfaces are written in XML files, stored in the “src/main/resources/mapper” folder.

The following figure 17 shows the RequestsDAO and its implementation.

图形用户界面, 文本, 应用程序

描述已自动生成

Figure 17: a data access object and its corresponding implementation

As we can see, there is a @Mapper annotation on top of the RequestsDAO; this tells MyBatis to look for implementations of this interface. In the XML files, the developer needs to specify the method name of the interface, the type of SQL operation to perform, and the SQL query itself. Any method parameter could be directly used in the query, even if the parameter is not a primitive data type(We can see that the addRequest query uses the title and many other fields, but the parameter for this method in the interface is a Request. This is because MyBatis allows the query to use any fields of the objects when passed as parameters). The results of selection queries from databases are automatically constructed as corresponding Java persistence objects when correctly configured.

The application of the MyBatis framework allows programmers to focus on writing correct and efficient SQL queries without worrying about complex methods incurred from traditional ways of accessing data with Java.

The author will not introduce implementations of DAOs or SQL queries explicitly for the rest of this dissertation unless in scenarios requiring extra clarification.

### 4.1.4 Dependency management overview

With the Spring IoC(Inversion of control) feature, developers no longer need to create and maintain tightly coupled relationships. Consider the following case: The controller StudentRequestsController is dependent on RequestsService, as the controller requires relevant methods from the service to achieve some tasks. The traditional way of instantiating the service would be using the keyword “new” to instantiate an implementation of RequestsService and assign it to a field. We can see from below figure 18 that we must declare and assign an actual implementation of the service. In the scenario where we want to assign another implementation, we have to go back to the place where we instantiate the object and rewrite the code.

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

Figure 18: comparison of dependency management

With the Spring IoC mechanism, the Spring container takes account of the instantiation and allocation of objects. Objects that are to be used by other objects are decorated with annotations such @Repository(denoting it is a DAO), @Service(denoting it is a service), and @RestController(denoting it is a controller). The Spring container creates instances of classes with such decorations and assigns these instances to places requiring them.

In figure 18 above, Spring detects that the constructor of StudentRequestsController is decorated with annotation @Autowired, which means it is asking Spring to find the instances of the parameter of the constructor. Spring then iterates all the instances it has created to check if any of them match the class RequestsService. It finds the instance and then assigns it as the constructor’s parameter.

We can see with the use of Spring, developers no longer need to hard code any concrete implementations, and by only using interfaces, the coupling of code is reduced. This practice allows a maintainable and modular system.

### 4.1.5 Frontend and controller overview

图形用户界面, 文本, 应用程序

描述已自动生成

Figure 19: front end JavaScript file and corresponding controller

In the scenario where the student wants to submit a request to the request queue(This is not the request in the context of network communication, but the request for office hours), the student clicks on a button in the user interface to trigger an event(figure 19). An Ajax request is sent to the relevant controller, where corresponding methods capture this Ajax request and responds with messages. All the files for the front end are placed in the resources folder, and the author will not elaborate on the front end implementation of this project unless in scenarios requiring extra clarification.

## 4.2 Subsystems

### 4.2.1 Identity authentication and management subsystem

图形用户界面, 图示

描述已自动生成

Figure 20: identity authentication and management subsystem tables

There are three types of human users of this system: students, TAs, and administrators. The system needs to have a mechanism to verify the identities of each type of user and ensure the users are only accessing services corresponding to their identities.

As we can see from the above figure 20, all these users have two columns in their tables: username and password. The passwords stored in the database are not plain text. They are encrypted with a component PasswordManager, which provides two methods: An encode method that takes the input of the original password and returns the encoded password, and a verify method that returns a boolean on whether the password to verify matches the existing record in the database. The DAOs of these three users all implement a LoginAble interface, which contains methods to retrieve and update passwords.

When a user sends a login request, it is captured by relevant controllers and handled by the LoginService. The login request contains the username, password, and identity the user declares. The LoginService looks into the corresponding table to find out if the user detail matches. If so, the system returns the corresponding portal page to the user.

When the user successfully logins, a session is created on the server: the key of that session is a constant string “user”, and the value is a class UserDTO which contains the username, password, and the identity of the user. The ID of that session is sent back to the user’s browser and stored in cookies.

The next time when the user sends a request to access some resources, this request is first intercepted by an interceptor UserInterceptor before being sent to any controller. The system checks if the user has logged in by checking whether a session has been created for that user. If so, the system further checks if the user’s user details are correct and if they are accessing the resource corresponding to their identities. Only when these checks pass does the request gets passed on to controllers. Any mismatch will result in the user being redirected back to the login screen. The session is removed after a certain time or when the user clicks on the logout button.

The system also allows users to update their passwords. Requests for updating passwords are sent to corresponding controllers and are handled by the AccountManagementService.

### 4.2.2 Project properties subsystem

图形用户界面, 图示

描述已自动生成

Figure 21: project properties subsystem tables

As figure 21 above shows, there is one table in this subsystem, project\_properties, which contains some parameters that the system uses. These parameters can be configured by the administrator. Currently, there are three parameters:

**amountToTriggerAutoAllocation**: An integer that specifies how many requests on a resource group could trigger the automated TA allocation mechanism. The default value is 5, which means if there are 5 requests on the same resource group in the request queue, the TA auto allocation mechanism will be triggered.

**autoTeachingAssistantAllocationEnabled**: A boolean that specifies whether the automated TA allocation mechanism is specified.

**defaultTime:**An integer specifying how many hours can default students book an office hour prior to its time. See later sections for clarification.

There is a corresponding DAO and service to manage this table.

### 4.2.3 Resource group subsystem

图形用户界面, 应用程序

描述已自动生成

Figure 22: resource group subsystem tables

The adoption of resource groups is one of the core features of this system, as it allows the administrator to manage access rights in batches and an automated resource allocation mechanism. However, the implementation of the resource group subsystem itself is rather simple. As we can see from figure 22 above, there is only one column in the resource\_groups table, which is the name of the resource group. This name can be considered the name of a module in the university.

The group names are referenced by many other tables, for example, appointments, as appointments on office hours must be created within a certain module. The DAO and service for resource groups are basic database operations. The current single-column design ensures the simplicity of the system in the meantime, not inhibiting any vital mechanism. Future developers may wish to add more columns to this table, such as module codes, module managers, module levels.

### 4.2.4 Appointments and requests subsystem

图形用户界面, 应用程序

描述已自动生成

Figure 23: appointments and requests subsystem tables

As figure 23 shows: two tables are involved in this subsystem, appointments, and requests. These tables can actually be divided into two subsystems. As they are of similarity and a request could be transferred to an appointment(see later section), they are introduced together. The table appointments stores all the active appointments in the system, and the table requests stores all the active requests in the system.

For the table appointments, the content of each column can be inferred from column names. The start\_time and end\_time of an appointment follow the format mentioned in section 3.3.5. The appointment\_type corresponds to the Enum AppointmentTypeEnum(placed in the “constants” package along with other Enums). Currently, there are five types of Enums, DEFAULT, QUERY, COURSEWORK, PERSONAL, OTHER. Future users of this system may wish to add other custom types.

For the table requests, the only column that requires clarification is the column time\_intervals. It is an integer indicating the amount of time the request needs on office hours. For example, if that value is 2, it means that the request requires 20 minutes of consecutive office hours for the request to be satisfied.

Builder design patterns are used in both persistence objects, Appointments and Requests, as both are classes with multiple fields. Each table has corresponding DAOs and services.

### 4.2.5 Student management subsystem

Two tables are involved in this student management subsystem, the table students and the table student\_resource\_groups. Administrators are able to manage students except for their passwords through this subsystem.

图形用户界面, 应用程序

描述已自动生成

Figure 24: student management subsystem tables

As we can see from the above figure 24, the student\_resource\_groups table indicates the resource groups(modules) a student belongs to. Each table row’s username indicates the student’s username, and the group\_name represents the student’s resource group. Students can belong to multiple resource groups as they can study several modules in universities.

The students table has four columns. Username, password, and identity are fields used for identity verification. The column priority\_status indicates the priority status of the current student. The value of this column corresponds to the Enum PriorityStatusEnum. In the current implementation of the system, there are only two types of priority status, a DEFAULT representing common students and a PRIORITY representing students requiring special assistance.

There are two DAOs responsible for interaction with these tables, StudentsDAO and StudentResourceGroupsDAO. The corresponding Persistence objects are Student and StudentResourceGroup.

When the administrator is managing the students, they need to view and manage the students along with their resource groups together; a service that is able to return an entity that contains both information in table students and table student\_resource\_groups is required.

The two DAOs are invoked and managed by one service StudentManagementService, which contains all the CRUD methods on these two tables and an additional method selectAllStudentDTOs that return a list of StudentDTO objects.

The StudentDTO is a data transfer object only used when transferring data between front end and back end. It is placed in package “dto” along with other data transfer objects. It contains all four properties of a student, along with a list of strings storing the name of resource groups that this student belongs to.

### 4.2.6 Student priority mechanism and reflections

When students are trying to make appointments within some module, they are able to view all the teaching assistants’ office hours in that module. Students with status PRIORITY are able to book any office hour presented to them. When students with status DEFAULT try to select any office hour and book an appointment, the system checks if the current time and the time of that office hour are within some threshold(This is defined in the project\_properties table, and the default value is 48 hours). These students are only able to book an appointment 48 hours prior to any office hours’ time. Any attempts on later office hours are declined by the system. Figure 25 offers an example to illustrate.

表格

描述已自动生成

Figure 25: an example of access to office hours

The priority mechanism is also inspired by the Linux access control system. In Linux, a file(resource) opens three kinds of access rights(read, write, execute) to different types of users. In this system, an office hour(resource) opens two kinds of access rights(view, book) to students with different priorities.

When designing the priority system, the author considered the following rules:

1. We must guarantee students requiring special assistance have access to office hours.

2. We should not let any students think that there is another group of students who are more/less privileged than them.

While the first rule can be easily satisfied in any priority system, it is difficult to design a system where priority exists but is hardly felt. This is done by inherently allowing students requiring special assistance to book all the office hours while only allowing other students to make appointments 48 hours before any office hour. As a result, students in both groups would not realize that there is actually another group who are accessible to more/less office hours, at least in the context of using the system.

The author also considered another mechanism, which is not allowing DEFAULT students to book or even view the office hours more than 48 hours later, and they are only able to view the office hours that they can book. Comparing these two mechanisms:

With the first mechanism, there could be possible scenarios where a DEFAULT student sees an office hour that they are willing to book 72 hours before; they waited a day and logged into the system only to find out the office hour is no longer there.

With the second mechanism, the DEFAULT students may have to log in to the system frequently to find if any office hour shows up in their portal.

The author decided to adopt the first one, as no matter what, it gives DEFAULT students more privilege to become aware of all the office hours.

The administrator could improve the general students’ experience by tightly controlling the number of students with priorities. The existing priority system could also be modified to limit the number or duration of appointments each student can book. This mechanism is used by the KCL LibCal system, which only allows each student to book 2 hours of study space each time[7]. This modification further improves fairness to different types of students.

The author has also studied other commercial and non-commercial priority systems and concludes that any priority system would unavoidably impair fairness to general users. The author argues that the problem on how to offer students priorities is more of an ethical issue, and cannot be solved merely through any algorithm or technical implementation. The administrator or course manager needs to carefully consider how to implement or modify a priority system based on the universities’ policies on students requiring assistance, and ensure that certain balance is achieved between normal students and students requiring assistance.

### 4.2.7 Teaching Assistant Management subsystem

图形用户界面, 应用程序

描述已自动生成

Figure 26: teaching assistant management subsystem tables

As we can see from figure 26, there are three tables in this subsystem.

The table teaching\_assistant\_resource\_groups indicates the resource groups that the TA belongs to. There is another column creation time, which ensures that the TA’s original resource group is never removed due to the resource allocation mechanism. The algorithms for the allocation of TAs are introduced in later sections.

The table teaching\_assistant\_available\_times indicates the office hours of TAs. The time follows the format mentioned in section 3.3.5. The available is a boolean indicating whether this time is available to be booked.

The table teaching\_assistants contains all the TAs in the system. The column “available” indicates whether the current TA is available. The column “adjustable” indicates whether the current TA is willing to be reallocated to another resource group. This design considers the fact that there could be TAs who are not willing to be assigned to other modules, even their weekly working hours remain the same.

Each table has a corresponding persistence object and DAO. When administrators are managing TAs and when students are querying for all the TAs’ office hours, they require information on TAs along with their resource groups or office hours. There needs to be an entity that contains information on all three tables. A service TeachingAssistantManagementService is created, which offers CRUD methods to all three tables and additional methods to return a list of TeachingAssistantDTOs. The TeachingAssistantDTOs is a data transfer object that contains all the fields of a TeachingAssistant, a list of strings indicating the groups that this TA belongs to, and a list of TeachingAssistantAvailableTime indicating this TAs’ office hours.

### 4.2.8 Work flow of the use case: student make an appointment

The majority of human users’ use cases can be achieved inside one or two subsystems and wouldn’t need much explanation. Due to the limitation on the length of the report, they are not introduced in detail.

However, there is one use case that involves multiple subsystems: Students make a new appointment. If we have a look at figure 14, we can see that the controller invokes four services. The author would like to elaborate on this use case separately, as it cannot be covered in any single subsystem. All the requests relevant to this use case are placed inside StudentAddNewAppointmentController.

图形用户界面, 文本, 应用程序

描述已自动生成

Figure 27: the steps for a student to book an appointment

As we can see from figure 27, after a student has logged in to the portal, he needs to perform three actions to make an appointment.

**1. Click on the “make new appointment” button.**

This shall send a “/resourceGroupNames” request to the controller. The controller receives this request, retrieves the username from the current users’ session, and then invokes the studentsManagmentService to get this students’ resource groups. These resource groups are sent back to the front end, and placed inside the drop-down bar at step 2.

**2. Select the resource group(module) from the drop-down bar to book an appointment in, and click on confirm.**

This shall send a “/teachingAssistantAvailableTimes” request to the controller, along with the name of the group that has been selected. The controller receives the request, and then returns all the available office hours back to the front end.

**3. Select some office hours, fill in appointment details, and click on “Submit”.**

This shall send a “/addAppointment” request to the controller, along with all the ids of the selected office hours and the appointment information. The controller retrieves the user’s username from the session and checks with StudentManagementService if he is a priority student. If not, the controller then performs a check on whether the chosen office hour can be booked by him.

If everything goes on well, an appointment is created and inserted into appointments table, and relevant office hours are set to unavailable and updated in the database. All the actions involved in this step are treated as a single transaction, so no data will be changed if any error occurs. The controller then returns “success” or error message to the front end.

### 4.2.9 Automated Appointment and Request Update subsystem

This subsystem is the service AutomatedRequestsAndAppointmentsUpdateService. The reason that this service’s name starts with “automated” is because methods in this service cannot be invoked directly by any request from users, and cannot be invoked by any controller. There are two situations in which such methods are triggered.

**When a certain time is reached**

The Spring framework allows certain methods to be executed at certain time periods with the use of an annotation @Scheduled. For example, the annotation @Scheduled(cron = "0 0 1 \* \* SAT") means that this method is executed at 1 am every Saturday.

**When certain methods are executed**

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

Figure 28: an example aspect class

Recalling that in the background theories section, we introduced the Spring AOP mechanism. The above figure 28 is one of the aspect classes in this system. All the aspect classes are placed in the “aspects” folder.

The top annotations @Aspect and @Component each tell the Spring framework that it is an aspect class, and it should be handled by the Spring container.

The @Pointcut annotation states that the scenario to apply this aspect is at the execution of the method addTeachingAssistantResourceGroup, which is a method to add a new row to the table teaching\_assistant\_resource\_groups.

The @After annotation states that the method it is decorating should be executed after the execution of a certain method, which is addTeachingAssistantResourceGroup in this scenario.

There are also other types of annotations, such as @Before, which are used when the developer wants certain methods to be executed before the execution of another method.

The above class have the effect that: After the execution of the method addTeachingAssistantResourceGroup in TeachingAssistantResourceGroupsDAO, immediately invoke the method checkAndUpdateRequestQueue in AutomatedRequestsAndAppointmentsUpdateService.

There are many situations to invoke the automated services in this system, and these situations are scattered across different methods in different classes. Traditional ways of invoking these services explicitly in code produce tightly coupled systems, and they are hard to maintain when modification or removal of services are required. The use of AOP allows the unified management of where these automated services are invoked and which methods to call within a standalone individual class.

There are two methods in the AutomatedRequestsAndAppointmentsUpdateService.

#### 4.2.9.1 removeObsoleteRequestsAndAppointments

This method removes any request or appointment in the database that is no longer valid.

This method is executed at the beginning of every week, that is when time reaches Saturday 1 a.m. every week.

|  |
| --- |
| Algorithm: remove obsolete requests and appointments |
| **Input:** void  **Output:** void   1. requests = requestsService.selectAllRequests() 2. appointments = appointmentService.selectAllAppointments() 3. currentTimestamp = system.currentTime() 4. **for** request **in** requests **do** 5. **if** request.getCreationTime() is earlier than currentTimestamp **then** 6. requestsService.removeRequest(request.getRequestId()) 7. **for** appointment **in** appointments **do** 8. **if** appointment.getCreationTime() is earlier than currentTimestamp **then** 9. appointmentService.removeAppointment(appointment.getAppointmentId()) |

Figure 29: pseudocode on removeObsoleteRequestsAndAppointments

The figure 29 shows the algorithm of the method. The method removes any appointment or unsatisfied request that is created earlier than the current time.

#### 4.2.9.2 checkAndUpdateRequestQueue

This method iterates through all the requests in the request table to determine if any request could be satisfied with the existing available resources.

This method is executed when a new request is added to the request table, or additional resources(TAs’ office hours) are added to the system. That is:

1. When addTeachingAssistantResourceGroup method inside TeachingAssistantResourceGroupsDAO is executed.

2. When addRequest method inside RequestsDAO is executed.

3. When addTeachingAssistantAvailableTime inside TeachingAssistantAvailableTimesDAO is executed.

|  |
| --- |
| Algorithm: check and update requests queue |
| **Input:** void  **Output:** void   1. requests = requestsService.selectAllRequests() 2. **for** request **in** requests **do** 3. groupName = request.getGroupName() 4. requiredAmountOfContiguousIntervals = request.getTimeIntervals() 5. teachingAssistantDTOs = TAManagementService. 6. selectAllAvailableTeachingAssistantDTOsByGroupName(groupName); 7. **loopOnTA:** 8. **for** dto **in** teachingAssistantDTOs **do** 9. times = dto.getTimes() 10. appointTimes = new List 11. previousTimeInterval = “” 12. **for** currentTime **in** times **do** 13. **if** currentTime is not available **then** 14. **continue** 15. **if** previousTimeInterval = “” **or** !isContiguous(previousTimeInterval, currentTime) **then** 16. appointmentTimes.clear 17. appointmentTimes.add(currentTime) 18. previousTimeInterval = currentTime 19. **if** appointmentTimes.size() == requiredAmountOfContiguousIntervals **then** 20. **for** appointmentTime **in** appointmentTimes **do** 21. appointmentTime.setAvailable(false) 22. TeachingAssistantManagementService.updateTAAvailableTime(appointmentTime) 23. appointment = new Appointment(request, appointmentTimes.get(0), 24. appointmentTimes.get(appointmentTimes.size() - 1)) 25. appointmentService.addAppointment(appointment) 26. requestsService.removeRequest(request.getRequestId()) 27. **break loopOnTA** |

Figure 30: pseudocode on checkAndUpdateRequestQueue

Figure 30 displays the pseudocode of the method. This method first iterates through all the requests in the requests table(line 1). When the RequestsDAO is querying the data from the requests table, it orders the requests based on their creation time, and a queue is simulated. We now deal with each element in the queue; we first get the name of the resource group(module) that the request belongs to and how many 10-minute intervals are needed(lines 3, 4). We then retrieve all the available teaching assistants in this resource group along with their office hours(lines 5, 6). For each teaching assistant, we get all their office hours(line 9). We also create a list to store the office hours that this request might occupy(line 10) and a variable to record the previous time interval(line 11). We need to instantiate these variables because requests may require longer than average time (recalling section 3.3.5), such as 20 minutes or 30 minutes. In these scenarios, we need to ensure we are providing consecutive office hours with the same TA for these requests.

We then iterate through all the office hours of the current TA. If an office hour is marked as unavailable, we skip this office hour(lines 12-14). If the current office hour is the first office hour we have iterated, or it is not continuous with the previous office hour, we clear the list appointmentTimes and add the currentTime to the list(lines 15-17). We then assign the previousTimeInterval with the currentTime and continue the next iteration.

Before we continue the next iteration, if we have found enough contiguous office hours that the request requires, we can create an appointment(line 19). We mark all the office hours in the appointmentTimes list as unavailable and update them in the database(lines 20-22). We construct a new appointment with the information we have on the request and the office hours in the list appointmentTimes(lines 23-24). We then update relevant databases and look at the next request(lines 25-27).

图表, 条形图

描述已自动生成

Figure 31: comparison with memory allocation

The problem that we try to find available office hours for a request is very similar to memory allocation: When a process needs to be executed, it asks the system to allocate memories for it. The situation that the process asks for memories is similar to the situation where the request asks for office hours. And both allocated memories and allocated office hours need to be contiguous(figure 31).

There are various major algorithms for memory allocation:

First Fit, which iterates the memory to find out the first space that the process can fit.

Best Fit, which looks for the smallest possible hole in the memory that fits the process.

Worst Fit, which looks for the largest possible hole in the memory that fits the process.

According to research comparing the performance of these allocation mechanisms, the Best Fit algorithm has the best performance in memory utilization, as it produces the least percentage of fragmentation of 30%(memories that could not be used by any process), and allocates all the processes. This is followed by First Fit which produces 31.76% of fragmentation and allocates 69.72% of processes. From the perspective of utilizing office hours and satisfying more requests, we might adopt the Best fit algorithm[23].

But there are two vital differences between the memory allocation and the request handling scenarios:

1. The average time of office hour sessions is around 10 minutes[3], so it is unlikely there will be many requests requiring a large amount of continuous time. Most requests are likely to be 10 minutes, which is the minimum unit of office hours in the system, resulting in few office hours being wasted.

2. The Best Fit algorithm is much more time-consuming than the First Fit in our scenario, because there is only one stack to iterate in the memory allocation scenario and many TAs to iterate in ours. In a memory allocation scenario, let’s assume the total number of holes is , while a First Fit algorithm needs to iterate times to find the first space a process can fit().

Suppose it takes constant level time to check any hole in the stack, then the difference between the time used by Best Fit and the time used by First Fit is

When coming to our automated request handling system, the Best Fit algorithm needs to iterate through all the TAs’ available office hours within the module to come up with the best solution; While a First Fit algorithm stops immediately after it finds one satisfying time interval on any TA. Let’s assume the number of holes in one TA’s office hour is , and it takes iterations for a First Fit algorithm to find a space to satisfy the request(). The number of TAs in the module is set as . The time difference now becomes

Which becomes much higher.

Based on the discussions above, despite the fact that the Best Fit guarantees less waste on TAs’ office hours, the author decided to adopt the First Fit algorithm as it is more efficient and more suitable for the scenario.

### 4.2.10 Automated Teaching Assistants Update subsystem

This subsystem is the service AutomatedTeachingAssistantsUpdateService. It contains five methods:

#### 4.2.10.1 updateTeachingAssistantAvailabilityStatus

This method scans all the TAs’ available times availability statuses and updates TA’s availability. If all the TA’s available times are unavailable, we mark the TA as unavailable. This action ensures the system doesn’t perform queries on unavailable TAs and relevant tables when displaying office hours to students, thus improving system efficiency.

This method is executed when any change happens to TA available time, that is when any method (except select) inside TeachingAssistantAvailableTimesDAO is executed.

|  |
| --- |
| Algorithm: update teaching assistant availability status |
| **Input:** void  **Output:** void   1. teachingAssistants = teachingAssistantsManagementService.selectAllTeachingAssistants(); 2. **for** teachingAssistant **in** teachingAssistants **do** 3. originalAvailability = teachingAssistant.isAvailable() 4. TAAvailableTimes = 5. TAManagementService.selectTAAllTimesByTAUsername(teachingAssistant.getUsername()); 6. available = false 7. **for** time **in** TAAvailableTimes **do** 8. **if** time.isAvailable() **then** 9. available = true 10. **break** 11. **if** available != originalAvailability **then** 12. teachingAssistant.setAvailable(available) 13. teachingAssistantsManagementService.updateTeachingAssistant(teachingAssistant); |

Figure 32: pseudocode on updateTeachingAssistantAvailabilityStatus

Figure 32 displays the algorithm. We loop through all the TAs. For each TA, we record their original availability and loop through all the office hours of this TA. If all the current TAs’ office hours are not available, then the current TA’s availability should be false, else it is true. We compare the TA’s original availability and current availability; if there is a difference, we update the database.

It is obvious that this method also performs querying to the database and incurs costs. This method is triggered in scenarios when TAs update their office hours or when appointments are created. The author assumes the querying of office hours happens more frequently than the update of office hours.

#### 4.2.10.2 updateUnreachableTeachingAssistantAvailableTimes

This method marks any TA's time that had gone past the current weekday as unavailable.

This method is executed at the beginning of every day, that is the 1 a.m. of every day. This method ensures that students don’t get to see or book office hours in the past.

|  |
| --- |
| Algorithm: update unreachable teaching assistant available times |
| **Input:** void  **Output:** void   1. weekday = System.getWeekday() 2. teachingAssistants = teachingAssistantsManagementService.selectAllTeachingAssistants() 3. **for** teachingAssistant **in** teachingAssistants **do** 4. times = TAManagementService.getTAAllTimesByUsername(teachingAssitant.getUsername()) 5. **for** time **in** times **do** 6. **if** time.weekday is earlier than weekday **then** 7. time.setAvailable(false) 8. teachingAssistantsManagementService.updateTeachingAssistantAvailableTime(time) |

Figure 33: pseudocode on updateUnreachableTeachingAssistantAvailableTimes

As we can see from figure 33, this method simply loops through all TAs’ office hours and mark any office hour that is earlier than the current time as unavailable.

#### 4.2.10.3 refreshTeachingAssistantAvailabilityStatus

This method marks all teaching assistants and their office hours as available at the beginning of every week.

This method is executed at the beginning of every week, that is Saturday 1 a.m.

|  |
| --- |
| Algorithm: refresh teaching assistant availability status |
| **Input:** void  **Output:** void   1. teachingAssistants = teachingAssistantsManagementService.selectAllTeachingAssistants() 2. **for** teachingAssistant **in** teachingAssistants **do** 3. teachingAssistant.setAvailable(true) 4. teachingAssistantsManagementService.updateTeachingAssistant(teachingAssistant); 5. times = TAManagementService.getTAAllTimesByUsername(teachingAssitant.getUsername()) 6. **for** time **in** times **do** 7. time.setAvailable(true) 8. teachingAssistantsManagementService.updateTeachingAssistantAvailableTime(time) |

Figure 34: pseudocode on refreshTeachingAssistantAvailabilityStatus

As we can see from figure 34, this method simply loops through all TAs and TAs’ office hours and mark all as available.

#### 4.2.10.4 checkAndAddTeachingAssistantResourceGroup

This method adds more TAs to resource groups whenever it detects a surge in needs from any resource group.

This method is executed when a new request is added to the request queue, that is when addRequest method inside RequestsDAO is executed.

|  |
| --- |
| Algorithm: check and add teaching assistant resource group |
| **Input:** void  **Output:** void   1. enabled = projectPropertiesService.getAutoTeachingAssistantAllocationEnabled() 2. **if** !enabled **then** 3. **return** 4. threshold = projectPropertiesService.getAmountToTriggerAutoAllocation() 5. requests = requestsService.selectAllRequests() 6. amountOfRequests = requests.size() 7. **if** amountOfRequests < threshold **then** 8. **return** 9. map = new Map<String, Integer> 10. **for** request **in** requests **do** 11. groupName = request.getGroupName() 12. map.put(groupName, map.getOrDefault(groupName, 0) + 1) 13. groupNeedsHelp = “” 14. **for** entry **in** map **do** 15. **if** entry.getValue() == threshold **then** 16. groupNeedHelp = entry.getKey() 17. **break** 18. **if** groupNeedsHelp == “” **then** 19. **return** 20. groupProvidesHelp = selectGroupNameWithMostAmountOfAdjustableOfficeHours() 21. teachingAssistantDTOs = 22. TAManagementService.selectAllAvailableTADTOsByGroupName(groupProvidesHelp) 23. **for** dto **in** teachingAssistantDTOs **do** 24. **if** !dto.isAdjustable() **or** dto.getResourceGroupNames().contains(groupNeedsHelp) **then** 25. **continue** 26. TAManagementService.addNewTAGroup(dto.getName(), groupNeedsHelp, currentTime) 27. currentAmountOfRequests = requests.size() 28. **if** currentAmountOfRequests == amountOfRequests **then** 29. notificationService.sendMessage(groupNeedsHelp + “needs more TAs”) |

Figure 35: pseudocode on checkAndAddTeachingAssistantResourceGroup

The above figure 35 shows the algorithm of this method. We first check if the administrator has allowed the auto TA allocation in the system. If it has been disabled, we stop(lines 1-3). We then retrieve the threshold from the projectPropertiesService and all the requests from the requests queue. We record the current number of requests and check if the number is smaller than the threshold. If the total number of requests is smaller than the threshold, then it is impossible any group’s amount of requests would meet the threshold(lines 6-8), and we stop.

We now iterate through all the requests to calculate the number of requests by group name and store it in a map(lines 9-12). We then iterate through the map to find out if any group’s amount of requests is equal to the threshold. The author would like to point out that, as this method gets executed each time a new request is added, there will only be one group whose amount of requests is exactly equal to the threshold. If we have found such a group, we record it, or else we stop(lines 13-19).

Now we have found a group that requires help; we now find the group with the most amount of available office hours as the group to provide help(line 20). We iterate through all the TAs in groupProvidesSupport and assign all of them to the groupNeedsHelp unless they are not adjustable or already belong to this resource group(lines 21-26).

Recalling section 4.2.9.2, with the support of AOP, after line 26 is executed, the method checkAndUpdateRequestQueue is immediately invoked implicitly to try to remove requests from the request queue. Then at line 27, after all the TAs have been added to the resource group, we record the current amount of requests and compare it with the number of requests before TA allocation. If the amount of requests is reduced, then we confirm we have satisfied some of the requests in the group that needs help. Else we report to the administrator that the current TA auto-allocation mechanism has failed and needs manual handling.

The current algorithm is an implementation of the greedy algorithm. That is, we make locally optimal choices at each step with the hope that these choices will lead to a globally optimal solution. Each time we detect a rise in demand in a resource group, we allocate the TAs from the resource group with the most amount of vacant resources(office hours) to that resource group because it is most likely that we could satisfy some requests and reduce the amount requests in the requests queue below the threshold. There are other possible algorithms that have been considered, such as allocating one TA from every other resource group to improve fairness(so that any resource group will not lose many resources).

When this system is being used by universities, it is likely that there will be multiple surges within a semester, concurrent surges in different modules during the same period of time, or other complex scenarios. The current report cannot prove whether the current algorithm or any other forms of algorithms will achieve an optimal result globally. This can only be done after the system is deployed in real life, and we monitor and compare the performance of different types of algorithms.

#### 4.2.10.5 checkAndRemoveTeachingAssistantResourceGroup

This method removes a resource group from TAs when the system detects no appointment had been made in that resource group in the past week, unless it is the original resource group of a TA. This method ensures that we remove redundant resource groups from TAs. Without such a mechanism, the checkAndAddTeachingAssistantResourceGroup method will eventually result in every TA (except for unadjustable ones) being allocated to every resource group, making the system meaningless.

This method is executed at the end of every week, that is Friday 11 p.m. of every week.

|  |
| --- |
| Algorithm: check and remove teaching assistant resource group |
| **Input:** void  **Output:** void   1. resourceGroupNamesToRemove = resourceGroupsService.selectAllResourceGroupNames() 2. appointments = appointmentsService.selectAllAppointments() 3. **for** appointment **in** appointments **do** 4. resourceGroupNamesToRemove.remove(appointment.getGroupName()) 5. **if** resourceGroupNamesToRemove.size() == 0 **then** 6. **return** 7. teachingAssistantDTOs = TAManagementService.selectAllTeachingAssistantDTOs() 8. **for** dto **in** teachingAssistantDTOs **do** 9. teachingAssistantGroupNames = dto.getResourceGroupNames(); 10. **if** teachingAssistantGroupNames.size() <= 1 **then** 11. **continue** 12. teachingAssistantGroupNames.remove(0) 13. **for** name **in** teachingAssistantGroupNames **do** 14. **if** resourceGroupNamesToRemove.contains(name) **then** 15. TAManagementService. 16. deleteTAResourceGroup(new TAResourceGroup(dto.getUsername(), groupName)); |

Figure 36: pseudocode on checkAndRemoveTeachingAssistantResourceGroup

Figure 36 shows the algorithm of this method. We first select all resource groups’ names and store them in a list. Then we remove those groups which have been made an appointment with in the past week. Now the list contains the names of resource groups to remove. If the size of the list is zero, we stop(lines 1-6).

Now we iterate through all the TAs. First, we retrieve all the resource groups this TA belongs to. If the TA is only belonging to only one resource group or no resource groups, we continue(lines 8-11). This is because TAs should always belong to their original modules, that is, the first resource group the TA had been allocated to when being added to the system.

If a TA belongs to more than one resource group, then we remove the first resource group from the list. This is because the TAs’ resource groups are ordered by creation time when being queried from the database, and the first element in the list is the TAs’ original resource group that should not be removed(line 12).

For the rest of the group names, we iterate through all of them. If any of them belongs to the list resourceGroupNamesToRemove, we remove that row from the table(lines 13-16).

This method revokes the additional TAs allocated to other resource groups when these TAs are no longer needed in those resource groups. The current assessment criteria to withdraw TAs from a resource group is when no appointments have been made in a resource group, and this criterion can be modified to when there are fewer than a certain number of appointments in a resource group.

### 4.2.11 Reflections on the Automated Resource Allocation Mechanism

The initial purpose of developing a resource allocation mechanism is to tackle one of the three major problems faced by TA. That is, when a module approaches its deadlines, the number of students queueing for office hours surges and TAs cannot cope with the demand.

The author argues that the resource allocation mechanism is likely to be able to dramatically improve this situation, as it dynamically allocates spare TAs from other modules. Such an allocation mechanism does not harm TAs’ rights by forcing them to work longer, as each TAs office hours per week are not changed during the allocation.

However, during the author’s discussion with a professor, the author was reminded that some TAs might not be willing to be adjusted to other modules. As a solution, the adjustable column is added to the teaching\_assistant table to allow TAs to specify whether they are willing to be allocated. The author further developed the thoughts that this resource allocation mechanism might implicitly incur additional workloads on TAs, as they might get allocated to a module that they are not familiar with, even if all the modules are within the computer science department. They might need to spend additional time learning or revising some of the concepts relevant to appointments from that module.

A possible solution is to create a table, teaching\_assistant\_adjustable\_modules, which allows TAs to specify which modules they could be adjusted to. This impairs the liquidity of the resources, the time efficiency of the system, and, eventually, the efficacy of the resource allocation mechanism. But it guarantees that TAs do not take additional burdens incurred from the system. This solution is not implemented in the system due to time constraints.

Another point the author needs to stress is that this mechanism might improve the majority of members’ experience in the system by impairing some of the members’ rights, despite the fact that the overall improvement incurred from the adoption of the system far outweighs any damage to a small group of users.

Let’s consider the following scenario. There are three modules in a university, modules A, B, and C. Each module has several TAs that are only accessible to students within that module. Module A and module B’s office hours are always fully booked, and only one student is using office hours in module C. One day, the system that the author designed is placed in this university, and it detects the surge in needs in both groups A and B. It then allocates all TAs in module C to module A and module B. These TAs get fully booked immediately, and the student in module C now loses access to any office hours. As he is the only person to submit a request for group C, the number of requests does not meet the threshold to trigger any allocation or notification system. Globally the number of students who are able to access office hours increases significantly, but that number could drop within certain modules.

This scenario is unlikely to happen as we have the option to mark some TAs as unadjustable, so that they are only accessible by students within certain modules. It only demonstrates the possible ethical issues caused by the resource allocation mechanism. The administrator of the system needs to ensure the minority group’s rights are protected and make modifications where necessary.

5 Results, Analysis and Evaluation

## 5.1 Testing of the system

The system has been tested in the following ways.

**1. Unit testing on services and DAOs**

All the methods in services and DAOs have been tested to ensure they produce the expected results. These tests are placed in the “test” folder.

**2. Using Postman to test controllers**

A software, Postman, is used for sending requests to controllers. The returned responses are monitored and compared with the expected results[24].

## 5.2 Results

### 5.2.1 Technical result

The major technical result of this project is a functional Java web application. It is a booking system with a dynamic resource allocation mechanism. Its functionalities have been tested to ensure it operates as expected.

文本

描述已自动生成

Figure 37: review of requirements

Figure 37 shows whether the requirements gathered before the development of the system have been achieved. All the must-have and should-have functionalities are implemented, along with one could-have functionality.

### 5.2.2 Project Aims and Objectives

Recalling the project aims and objectives:

**Project aim:**

In computer science higher education, discover the problems TAs face when holding office hours, and develop a piece of software that solves these problems.

**Project objectives:**

Read articles to discover the problems TAs in computer science higher education face when holding office hours and summarise these problems.

After concluding the problems, read articles and research how similar problems were solved.

Determine the requirements and the design of the software. Develop a piece of software based on the information gathered.

After development work is finished, analyse the software to check whether the specified requirements have been achieved, then perform testing on the software.

The author concludes that the aims and objectives of this project have been achieved.

### 5.2.3 Research question

The research questions of this project are:

1. In computer science higher education, what are the problems that TAs face when holding their office hours?

2. What functionalities should the software developed possess to solve these problems?

The author answered the first research question in part 1 of this report, and discovered the three major problems; The author then answered the second question by presenting a list of requirements.

## 5.3 Evaluation and discussion

The problems discovered by the author are:

1. Teaching assistants may find it challenging when trying to answer more personalised questions from students.

2. Teaching assistants may feel time pressure when students are queueing for support, especially when a course approaches its deadlines.

3. Teaching assistants may find it difficult to allocate time to different students fairly.

The first problem is solved by allowing TAs to view the information of the appointment, which involves appointment type, appointment student, appointment time and appointment description. TAs can make preparations in advance according to this information.

The second problem is solved by implementing a booking system with a resource allocation mechanism. With this system, TAs no longer feel time pressure because all the office hours are reserved and allocated to designated students in advance. The resource allocation mechanism can dynamically allocate TAs from other groups if it detects a surge in demand.

The third problem is solved by the booking system with a priority mechanism. The administrator and the system now take the job of allocating time to students, especially for students requiring special assistance.

The advantages of this system are:

1. It uses production-grade frameworks, technologies, and libraries to implement a clearly structured, maintainable and scalable system that provides the functionalities required to solve the problems.

2. It uses considered designs and algorithms on request handling, resource allocation, and student priority mechanism to guarantee system efficiency and effectiveness.

The disadvantages of this system are:

1. All the designs, implementations, mechanisms, or algorithms have not been tested or deployed in real life to provide concrete evidence that these problems are solved. The system should be deployed in a computer science department of a university in the future. The performance of the system and its algorithms should be monitored, questionnaires on TAs’ experiences should be handed out, and interviews with different types of users of this system should be conducted.

2. The user interface of the system could be improved. The aesthetic and systematic designs on the front end of this system are not acceptable for production-grade software that is going to be deployed in universities.

The project is initiated with the goal of dealing with the problems faced by TAs in the computer science department, while all the systems’ developed functionalities and features can be of help to any other department whose TAs face similar problems. This system can be used by any department in higher education organisations to improve the experience of students, TAs, and any other participants.

6 Professional and Ethical Issues

**Professional Issues:**

The principles of the BSC Code of Conduct are carefully followed in this project[25].

All the third-party and open-source libraries, frameworks, or algorithms used in this project are clearly cited.

The source code is uploaded as a public repository on Git Hub so that it is accessible to any viewers.

This project pays close attention to places where the users’ rights could be harmed, and explicitly addresses them.

**Ethical Issues:**

If the current system is implemented in real life, it may give rise to some ethical issues. They have been discussed in this report, and they are summarised here:

The prioritization system may harm general students’ rights.

The resource allocation mechanism may give TAs extra workloads, or harm students’ rights within certain modules.

The majority of methodologies and designs within this project are developed and migrated from academic or technical concepts. Most of the systems that arise from these concepts are designed and behave to achieve a global optimal result that maximizes efficiency and output. The sacrifice or loss from some of the components is considered acceptable and necessary as long as the overall outcome is optimal.

When migrating these concepts into real-world scenarios where the participants are humans, we must be aware of the fact that humans are not like components, and each individual user’s rights should be respected and protected when designing and implementing any system.

7 Conclusion and Future Works

This project started with two research questions:

1. In computer science higher education, what are the problems that TAs face when holding their office hours?

2. What functionalities should the software developed possess to solve these problems?

The report answered the first research question by summarising three major problems that TAs face based on relevant research and then answered the second research question with a list of requirements gathered by presenting and comparing solutions to similar problems.

A Java web application is implemented as a solution to the problems, where relevant functionalities and algorithms are applied to address different concerns.

The project aim has been met, as the problems TAs face when holding office hours have been found, and the system is implemented as a solution. All four project objectives have been met in different stages of this project.

Based on the above comparisons and achievements, the author concludes that the research questions have been answered, the project aim and objectives have been met, and theoretically, the system is capable of solving the problems faced by TAs.

Due to the time constraints of the project, there are many works that are yet to be carried out. They have been listed in the report and summarised here.

1. Improve the implementation of the front end. The systematic design of the front end could be improved with the usage of some frameworks, and the aesthetic design could be more elegant, beautiful and user-friendly.

2. Implement the unachieved could-have and won’t-have functionalities. This involves allowing users to specify personal information such as age, gender, and other information in the system, providing an evaluation functionality for students to give feedback to their teaching assistants after sessions, providing a notification system to the students/teaching assistants to confirm/notify the booking, and implementing a caching framework to store contents that are less likely to change frequently.

3. Add a table to the system, teaching\_assistant\_adjustable\_modules, to allow TAs to specify the modules they could be adjusted to.

4. Deploy the system in a university’s computer science department.

5. Monitor the performance of the system to adopt optimal algorithms. Gather and analyse feedback from users to discover if the system solves the problems as expected. Make further functional improvements based on the data gathered.

8 References

[1] E. Riese, M. Lorås, M. Ukrop, and T. Effenberger, “Challenges faced by teaching assistants in Computer Science Education across Europe,” Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1, 2021. doi:10.1145/3430665.3456304

[2] I. Pivkina, “Peer learning assistants in Undergraduate Computer Science Courses,” 2016 IEEE Frontiers in Education Conference (FIE), 2016. doi:10.1109/fie.2016.7757658

[3] Y. Malysheva, J. Allen, and C. Kelleher, “How do teaching assistants teach? characterizing the interactions between students and TAS in a computer science course,” 2022 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), 2022. doi:10.1109/vl/hcc53370.2022.9832962

[4] E. Riese, “Teaching assistants’ experiences of lab sessions in introductory computer science courses,” 2018 IEEE Frontiers in Education Conference (FIE), 2018. doi:10.1109/fie.2018.8659243

[5] S.-H. Ko and K. Stephens-Martinez, “What drives students to office hours: Individual differences and similarities,” Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1, 2023. doi:10.1145/3545945.3569777

[6] G. J. Gutierrez and P. Kouvelis, “Parkinson’s law and its implications for Project Management,” Management Science, vol. 37, no. 8, pp. 990–1001, 1991. doi:10.1287/mnsc.37.8.990

[7] “Space availability - maughan library,” LibCal, https://libcal.kcl.ac.uk/spaces?lid=760 (accessed Aug. 6, 2023).

[8] Alaska Airlines, “Seat selection - easybiz user guide,” Alaska Airlines, https://www.alaskaair.com/content/easybiz/help/user-guide/choose-a-seat (accessed Aug. 6, 2023).

[9] K. Ampountolas, H. S. Furtado, and R. C. Carlson, “Motorway Tidal Flow Lane Control,” IFAC-PapersOnLine, vol. 51, no. 9, pp. 279–284, 2018.

[10] Pankaj, “Builder design pattern in Java,” DigitalOcean, https://www.digitalocean.com/community/tutorials/builder-design-pattern-in-java (accessed Aug. 7, 2023).

[11] Pankaj, “Proxy design pattern,” DigitalOcean, https://www.digitalocean.com/community/tutorials/proxy-design-pattern (accessed Aug. 7, 2023).

[12] “Spring framework6.0.11,” Spring Framework, https://spring.io/projects/spring-framework (accessed Aug. 7, 2023).

[13] “Spring Boot3.1.2,” Spring Boot, https://spring.io/projects/spring-boot (accessed Aug. 7, 2023).

[14] the IOC Container, https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/beans.html (accessed Aug. 7, 2023).

[15] “Aspect Oriented Programming with spring,” Aspect Oriented Programming with Spring :: Spring Framework, https://docs.spring.io/spring-framework/reference/core/aop.html (accessed Aug. 7, 2023).

[16] C. Begin, MyBatis, https://mybatis.org/mybatis-3/ (accessed Aug. 7, 2023).

[17] J. F.- js.foundation, JQuery, https://jquery.com/ (accessed Aug. 10, 2023).

[18] J. T. Mark Otto, “Bootstrap,” Bootstrap · The most popular HTML, CSS, and JS library in the world., https://getbootstrap.com/ (accessed Aug. 10, 2023).

[19] Project Lombok, https://projectlombok.org/ (accessed Aug. 10, 2023).

[20]A. Dangerfield, “Working on a student visa,” International Students, https://www.internationalstudents.cam.ac.uk/student-visa-responsibilities/working-student-visa (accessed Aug. 12, 2023).

[21] “Right to work for student/ tier 4 visa holders,” Staff Immigration, https://staffimmigration.admin.ox.ac.uk/employingstudents#:~:text=For%20information%20on%20acceptable%20evidence,not%20considered%20to%20have%20terms. (accessed Aug. 12, 2023).

[22] “Learning and teaching,” 12. Pay and conditions for GTAs - Staff home, University of York, https://www.york.ac.uk/staff/teaching/develop/graduate-teaching-assistants/university-policy-gta/12-gta-pay-conditions/ (accessed Aug. 12, 2023).

[23] A. K. Manda, D. K. Baruah, J. Medak, N. Gogoi, and P. Gogoi, “CRITICAL SCRUTINYOF MEMORY ALLOCATION ALGORITHMS: FIRST FIT, BEST FIT AND WORST FIT,” Turkish Journal of Computer and Mathematics Education , vol. 11, no. 03, pp. 2185–2194, 2020.

[24] Postman, https://www.postman.com/ (accessed Aug. 12, 2023).

[25]“BCS Code of conduct,” BCS, https://www.bcs.org/membership-and-registrations/become-a-member/bcs-code-of-conduct/ (accessed Aug. 12, 20

9 Appendices

## Appendix A: Source code structure

图片包含 图形用户界面

描述已自动生成

The above figure is the structure of the source code.

All the java srouce code are placed inside src/main/java folder. The majority of packages in this folder are introduced in the report. The ReservationApplication class is the main and starter class of a Spring Boot application.

Inside resources folder, there are three folders:

mapper, which contains the MyBatis XML files. These files contain the SQL queries of the DAOs.

static.js, which contains all the JavaScript files.

templates, which contains all the html files.

All the unit tests are placed inside test folder.

Aside from the directories above, there are some files that are of importance in the project:

application.yml: All the settings of the Spring Boot application is configured inside this file. Users may change the data source to connect to their corresponding database.

kcl.sql: This is the sql file to create tables to be used in the system.

pom.xml: This is the file to manage all the dependencies used in this application.

This repository is available at: <https://github.com/gts333/KCL>

**Instructions on running this application locally:**

1. Ensure you have Java 8 or later, maven, MySQL installed and configured on your machine. It is recommended that you also install Intellij Idea, which is an IDE allowing you to view all the source codes and run the program easily.

2. Create the database “kcl”, and run the script “kcl.sql”, to create tables being used by this application.

3. Configure the database properties at “application.yml”

4. Ensure you have installed all the dependencies in “pom.xml”. If you have installed Intellij then it will do it automatically for you.

5. Open the terminal, and enter “mvn spring-boot:run”. If you have installed Intellij, open src/main/java/ReservationApplication and click on the little green triangle.

6. The Spring Boot has a default Tomcat server incorporated, so the default port is 8080. You can go to <http://localhost:8080/loginPage.html> to see if the system is up and running.

Remember to insert some data in relevant tables. Notice that password insert directly in databases without being encrypted by the program will not pass user authentication check. You may want to disable the functionality by modifying src/main/component/impl/PasswordManagerBCryptImpl.

## Appendix B: Screenshots of the user interface

图形用户界面, 应用程序

描述已自动生成

### Admin portal screenshots:

图形用户界面, 应用程序

描述已自动生成

电脑屏幕截图

描述已自动生成

图形用户界面, 应用程序

描述已自动生成

图形用户界面, 应用程序, Word

描述已自动生成

图形用户界面, 应用程序

描述已自动生成

电脑网站的截图

描述已自动生成

### Student portal screenshots

图形用户界面, 应用程序

描述已自动生成

图形用户界面, 应用程序

描述已自动生成

电脑屏幕截图

描述已自动生成

图形用户界面, 应用程序, Word

描述已自动生成

图形用户界面, 应用程序

描述已自动生成

### Teaching assistant portal

图形用户界面, 应用程序

描述已自动生成

图形用户界面, 应用程序

描述已自动生成

图形用户界面

描述已自动生成

图形用户界面, 应用程序

描述已自动生成

## Appendix C: Source code of core services

### AutomatedRequestsAndAppointmentsUpdateService

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

图形用户界面, 文本, 应用程序

描述已自动生成

文本, 应用程序

描述已自动生成

### AutomatedTeachingAssistantsUpdateService

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

图片包含 背景图案

描述已自动生成

图形用户界面, 文本, 应用程序

描述已自动生成

图形用户界面, 文本, 应用程序

描述已自动生成

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

图形用户界面, 文本, 应用程序

描述已自动生成

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

图形用户界面, 应用程序

描述已自动生成