FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Assignment of bachelor's thesis

Title:	Web application to support departmental timetable administrators
Student:	Murad Mammadli
Supervisor:	Ing. Ondřej Guth, Ph.D.
Study program:	Informatics
Branch / specialization:	Software Engineering 2021
Department:	Department of Software Engineering
Validity:	until the end of summer semester 2024/2025

Instructions

Design and implement a web application based on Spring Boot to help a departmental timetable administrator manage the covering of classes by teachers.
1. Describe the current process of assigning teachers to courses and the timetable at FIT CTU.
2. Analyse and design the application to support some part of this process.
3. Implement the application as a prototype.
4. Verify the result by appropriate tests.

Electronically approved by Ing. Michal Valenta, Ph.D. on 28 January 2024 in Prague.

Bachelor's thesis

WEB APPLICATION TO SUPPORT DEPARTMENTAL TIMETABLE ADMINISTRATORS

Murad Mammadli

Faculty of Information Technology
Department of Software Engineering
Supervisor: Ing. Ondřej Guth , Ph.D.
June 27, 2024

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis: Mammadli Murad. *Web application to support departmental timetable administrators.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2024.

Contents

List of Figures

List of Tables

I would like to express my deepest gratitude to my supervisor, Ing. Ondřej Guth, Ph.D., for his invaluable assistance and guidance throughout the development of this thesis. His regular consultations, patience in answering my questions, and help in identifying and solving problems were crucial in completing this work. Above all, I appreciate his kindness and the pleasant discussions we had on various topics. His mentorship has been an really enjoyable experience.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis. I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as a school work under the provisions of Article 60 (1) of the Act.

In Prague on June 27, 2024

Abstract

The thesis encompasses the analysis, design, implementation, and testing of a prototype application developed to help a departmental timetable administrator manage the covering of classes by teachers. The application features a Spring Boot server and web interfaces for administrators and teachers, developed using Thymeleaf. The thesis starts with a comprehensive analysis of the current process of assigning teachers to classes, covering its main elements, functional and non-functional requirements, use cases, and the domain model. In the following chapters, the design and implementation of the prototype app are discussed. The thesis concludes with proposals for future enhancements, derived from the results of user testing. As a result of this thesis, a prototype system is developed that is intended to act as the base platform for future improvements.

Keywords Spring Boot, Java, Thymeleaf, timetable administration, prototype app

Abstrakt

Tato práce obsahuje analýzu, návrh, implementaci a testování prototypu aplikace, která má pomoci katedernímu rozvháři s pokrytím výukových paralelek učiteli. Aplikace využívá server Spring Boot a webové rozhraní pro rozvrháře i učitele vyvinuté pomocí Thymeleaf. Práce začíná s podrobnou analýzou současného procesu přiřazování učitelů k předmětům a pokrývá hlavní části, funkční a nefunkční požadavky,průzkum existujících řešení a doménový model. V následujících kapitolách jsou diskutované návrh a implementace. Práce je zakončena návrhy na budoucí vylepšení, které vychází z uživatelského testování. Výsledkem práce je vyvinutý prototyp, jehož záměrem je být základem pro budoucí vylepšování.

Klíčová slova Spring Boot, Java, Thymeleaf, rozvrhář, prototyp

List of abbreviations

API	Application Programming Interface
MVC	Model View Controller
RBAC	Role Based Access Control
SQL	Structured Query Language
UI	User Interface
WORA	Write Once Run Anywhere
XML	Extensible Markup Language

Introduction

Motivation

In today's educational world, things are always changing. Managing schools and colleges smoothly depends a lot on how well administrative tasks are handled and how easily teachers, timetable administrators, and the systems they use can communicate with each other. There's a growing need for new digital tools to make these interactions smoother. This project suggests creating a new user interface (UI), designed with teachers and timetable administrators in mind. The aim is to make it simpler to organize who teaches what classes. By doing this, we hope to make administrative work more efficient and make the educational environment more adaptable to everyone's needs.

Objectives

The primary objective of this thesis is to analyze the existing process of assigning teachers to classes and to develop an application that simplifies this process for both the timetable administrator and the teachers. This simplification will include the creation of a new user interface (UI). We also want to create the views listing teachers and courses and necessary functionality for these views. The person in charge (timetable administrator) should be able to modify the data in these views. Other objectives involve testing the prototype and proposing possible improvements to the system.

Chapter 1
Analysis

The main goal of this chapter is to analyze the existing process of Assigning Teachers to Classes and improve/extend it by creating an application to support some parts of this process. Additionally, it will outline the functional and nonfunctional requirements, use cases, and domain model for the application that is to be developed.

1.1 Assigning Teachers to Classes Process

Existing Process — also referred to as the "As Is" process, describes the existing processes within the product without the implementation of the intended system or suggested changes, focusing on understanding the activities of the client, identifying problematic processes and activities, selecting supported parts of processes, and serving as the basis for developing new or upgraded processes. [1]

To build a thorough understanding of the existing process, I held several consultations with my supervisor, where we discussed the process steps and details of the application to be developed. During these meetings, we divided the process into two phases and I drafted a brief summary for each to understand it thoroughly. These phases will be described in subsection 1.1.6

1.1.1 Process Introduction and Key Terms

I would like to start this subsection with the introduction to the **Assigning Teachers to Classes Process.** To fully understand the process and its steps, it's necessary first to define a few key terms. The process includes the departmental timetable administrator, referred to as the "admin" going forward, and the teachers. Each course is represented by a unique code. From this point forward, the term "course code" will be used to refer to course names (for example:BIE-AG1). Each course includes several "parallel types." There could be three parallel types in a course: lectures, seminars, and labs. Conversely, "class" refers to a specific instance, defined by a course code and its associated parallel type (example: BIE-AG1 Lecture). Additionally, each course can have multiple classes of the same parallel type (e.g., BIE-PA1 with 2 lectures). The ultimate goal of this process is to ensure that all classes of courses are covered by teachers. This process is manual and involves many details, and therefore it is important to clearly outline the roles and steps involved.

1.1.2 Roles

The process is split into two distinct roles: Admins and Teachers. Below, I will outline the behavior specific to each role.

Admin

A person who controls all the processes, including assessing the faculty's needs for managing course coverage by teachers. Responsibilities will be discussed in more detail in the subsequent subsections.

Teacher

A person who communicates their willingness to teach classes.

1.1.3 Course with many Lab classes

What is a course that contains many lab classes? This concept is crucial to the process and must be clearly understood. The main difference from a regular course arises when a specific course has many lab classes and, at the same time, many teachers. This situation necessitates distributing these lab classes among the teachers, which will be described in more detail in subsection 1.1.6, specifically in Phase 2. Later in the thesis, the terms "course with high number of lab classes" or "hasLabs" will be employed to differentiate between regular and courses that have many lab classes.

1.1.4 Data received from Teachers

In this section, the data submitted by teachers are described:

1. **Decision to Teach**
   Teachers send their decision "Yes" or "No" on their assigned classes. The default or non-submitted decision is denoted with the "Pending" keyword.

2. **Number of Lab Classes**
   Teachers submit the minimum and maximum number of lab classes they are willing to teach if they are assigned to a course with many lab classes.

1.1.5 Successfully Covered Course

How can the successfully covered course be described? It should meet the listed requirements:

1. Each class of the course has to be staffed with the required number of teachers.

2. The assignment of classes should be approved by the selected teachers.

3. For courses with many lab classes, the lab requirement must be satisfied. This means summing the minimum and maximum number of lab classes that teachers are willing to teach and ensuring that lab requirement falls within this range.

Additionally, refer to Figure 1.2 to see how the Google sheet looks like for a successfully covered course.

1.1.6 Process Steps

The section below outlines several steps in this process that need to be taken. The person in charge of assigning teachers to classes (admin) is the one performing this process. He creates Google sheets (please refer to Figure 1.2 to see how this Google sheet looks like for a particular

course) containing teachers and courses, and sends emails manually to the teachers to see if they agree with their current assignments.

Phase 1

Step 1: Pre-Enrollment The admin checks the study plan of the faculty to see what courses will be taught in the upcoming semester. He gathers information from the faculty website, including the course name, the number of registered students for each course, and the number of required teachers for each parallel type of course.

Step 2: Creating Google Sheets Based on the information gathered in Step 1, the admin creates Google Sheets for each course. These sheets will be used to store all the information regarding the course. See example of this sheet 1.2.

Step 3: Filling the Google Sheets The admin has information on the assignment of teachers to classes from the previous year or semester. Initially, he fills the classes with the teachers assigned to the course from the previous year or semester. Usually, teachers agree to their previous assignments, thus helping the admin avoid looking for additional (external) teachers.

Step 4: Communication for Confirmation The admin contacts the teachers via email, asking them to confirm their assignment to the class. Their responses are then stored in the Google Sheets. This step is iterated as necessary because the main goal of the Assigning Teachers to Classes process is to ensure that all classes of the course are covered by the required number of teachers. If some of the assigned teachers do not want to teach a class, the admin searches for new teachers and contacts them until all classes are covered.

Intermediate Step

Between Phase 1 and Phase 2, the timetable (schedule) is created. Information from this timetable will be used in Phase 2.

Phase 2

This phase is designed for handling courses with many lab classes, described in subsection 1.1.3

Step 1: Filling the Data Now that the timetable is available to the admin, he knows when the lab classes will take place. Based on this information, he first fills in the timeslots of the lab classes in the corresponding Google Sheets.

Step 2: Communication with Teachers In this step, he again sends emails to the teachers, presenting them with the timeslots of the lab classes. He then expects information from the teachers on the timeslots and the minimum and maximum number of lab classes they are willing to teach. He stores this information in the Google Sheets. This step is also iterated as needed.

Step 3: Finalizing the Work and Completing the Google Sheets After receiving the teachers' responses, he assigns the teachers to timeslots for the lab classes based on his best judgment. Phase 2 is completed when all the lab-classes are covered by teachers who agree with the assigned timeslots.

These steps are described in an activity diagram. Please see Figure **??** to understand these steps more thoroughly. Additionally, please refer to Figure 1.2 to see how this Google sheet looks like for a particular course.

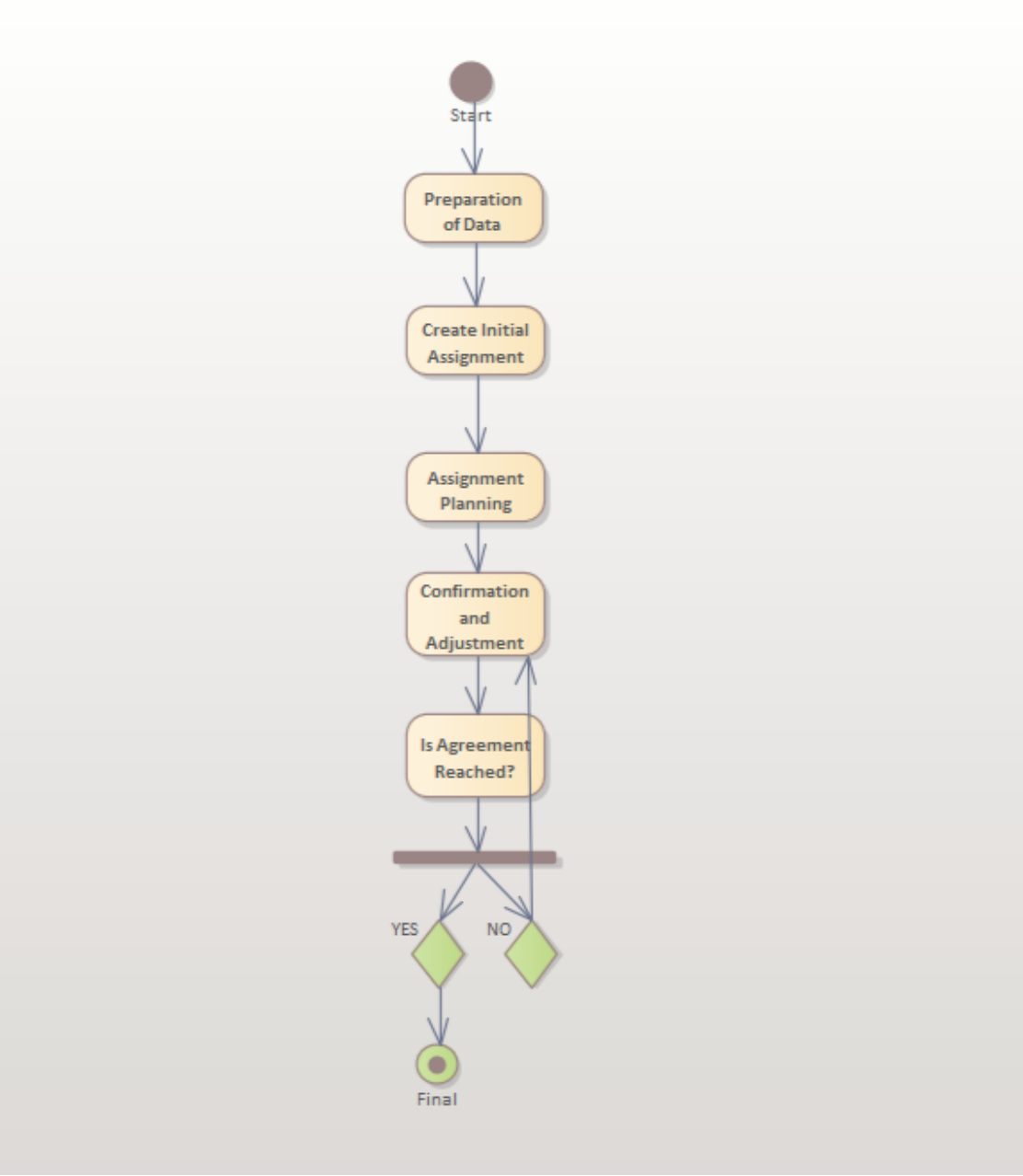

Figure 1.1 Flowchart describing Process of Assigning Teachers to Classes

1.1.7 Problems with Process

Projects often begin with problems. The challenge is to figure out the nature of the problem, and then to consider what kind of intervention might be required to resolve the problem. [1] In this section, I am going to outline several encountered difficulties in **Assigning Teachers to Courses Process**.

- Creating these sheets and manually entering records is a time-consuming process.
- Emailing each teacher individually and awaiting their responses also requires a significant amount of time.
- Keeping track of whether every course is fully staffed with the necessary teachers for lectures, seminars, and labs can be challenging.
- A dedicated user interface for performing this process is currently lacking.

1.2 Possible Solutions from the Author

The Future State Process — also called the "TO BE" model, represents the envisioning of process improvements, the introduction of new processes, and updates to existing processes, utilizing information systems for certain activities, which outlines the processes after the deployment of the system including when, how, and by whom the system is used. This model enables the comparison of benefits and prevents the support of inefficient, "bad" processes. [2]

In this subsection I will propose following solutions and automations to the encountered difficulties in **Assigning Teachers to Courses Process**

- A UI will be introduced, which will be user-friendly and visually appealing.
- In our application, administrators will create each teacher's profile (only once), simplifying the process of tracking the courses and parallels associated with each teacher by accessing their assigned courses directly.
- After assigning courses in the system, the administrator will email each teacher, instructing them to log into their accounts and make their selections regarding the assigned courses and parallel types.
- The application will include a page for courses that indicates if each course has the required number of teachers. On this page, admins can also view teachers' decisions and make changes to them if necessary.
- There will be a dedicated users and courses pages for administrators, providing them with the capability to access and modify any teacher's data as necessary.
- The app will differentiate between courses with low and high number of labs. For courses with a high number of labs, teachers will be required to submit not only their decision on their assigned course parallel but also the minimum and maximum number of labs they are willing to teach, thus helping admin with optimizing assignment process.

1.2.1 KosAPI

Initially, the application was intended to use KosAPI to gather data from the university's courses and teachers and further modify this data according to the scope of the thesis. KosAPI [3] is an API (Application Programming Interface) provided by the Czech Technical University (CTU) in Prague, specifically designed to access and manage data related to the university's educational systems. Using this API could have the following benefits:

Real-Time Data Access: It provides real-time access to data, ensuring that the information displayed in connected applications is always up-to-date. This feature could be really important in our application's context, as new teachers and courses might be registered every semester, and having this functionality would greatly simplify application maintenance.

Efficiency in Resource Management By automating and streamlining data management tasks, KosAPI could reduce the workload of timetable administrator.

Unfortunately, permission to use this API was not granted, and the author had to develop the application without using it.

Figure 1.2 Example Google sheet of some course

	B	C	D	E	F	G	H	I	J
1									
2			BIE-PA2						
3									
4		Lecture	Seminar	Lab					
5	Sarah Davis	0	0	"2-3"	YES				
6	Michael Brown	0	0	"1-3"	YES				
7	Matthew Anderson			?	PENDING				
8	Steven Doe			0	NO				
9	Chuck Norris	1	1	0	YES	garantor			
10									
11	covered in total	1	1	6					
12	required	1	1	6					
13									
14		B212	B222	B232					
15	preliminary register	73	89	142					
16	beginning of the Sem		67						

1.3 Requirements

An excellent software product results from a well-executed design based on excellent requirements. Excellent requirements result from effective collaboration between developers and customers.

There are two main rules of writing excellent software requirements :

- Anyone who reads the requirement comes to the same interpretation as any other reader.
- Each reader's interpretation matches what the author intended to communicate.

[4]

With these considerations in mind, I frequently consulted with my supervisor. Since this application is aimed at being utilized by departmental timetable administrators, extensive discussions with my supervisor, who also holds the role of an admin, led us to establish the following requirements:

1.3.1 Functional Requirements

Functional requirements specify the behaviors the product will exhibit under specific conditions. They describe what the developers must implement to enable users to accomplish their tasks (user requirements), thereby satisfying the business requirements.[4]

F1: **Manage Teachers** The system should allow the administrator to create, update, and delete teacher profiles. Additionally, it should display a list of all registered teachers and provide detailed information about their assigned classes.

F2: **Manage Courses** The system should allow the administrator to create, update, and delete courses. It should also display a list of all registered courses and provide detailed information about each course, including whether it has many lab classes, how many classes are required for each parallel type, and most importantly, for each parallel type of the course, the number of classes currently covered by teachers.

F3: **Handle Class-Teacher Assignments** The system should allow the administrator to assign teachers to classes and vice versa, and manage these assignments. The system should also enable teachers to manage their own class assignments by setting their decision to teach and adjusting their lab teaching limits (min and max), with all updates reflected in both the teacher's and the course's records.

F4: **Course Coverage** For each course, the system should allow the administrator to see whether all classes in this course are covered by teachers or not.

1.3.2 Non-functional Requirements

Non-functional Requirement is a description of a property or characteristic that a system must exhibit or a constraint that it must respect. [4]
The following non-functional requirements have been established for **Web Application to Support Departmental Timetable Administrators**.

N1: **Server** The application server must utilize Spring Boot, leveraging its embedded server capabilities for simplified deployment and server management.

N2: **Usability** The application should be easy to use for both administrators and teachers with minimal training. The user interface should be intuitive and user-friendly, enhancing user satisfaction.

N3: **Maintainability** The application should be easy to maintain and update. Furthermore, it should be flexible to accommodate the university's login/authorization page in the future.

N4: **Roles** The system must implement different roles to access various pages, and to accommodate that, security introduction is needed. Within security, we have to implement authentication and authorization to distinguish between admins and teachers.

.

1.4 Use Cases

A use case describes a sequence of interactions between a system and an external actor that results in the actor being able to achieve some outcome of value. An actor is a person (or sometimes another software system or a hardware device) that interacts with the system to perform a use case. [4] We don't always need a comprehensive use case specifcation. Cockburn [5] describes casual and fully dressed use case templates. A casual use case is simply a textual narrative of the user goal and interactions with the system. On the other hand, fully dressed use cases provide a comprehensive and detailed description of a use case. Fully dressed use cases are necessary when the to be developed application is complex and system failures carry a high risk. Or, use cases represent novel requirements with which the developer is not familiar.

Taking these options into account, I opted for a casual template. In some use cases, I used this template for simplicity, while in others, where I felt more details were needed, I provided additional information within the casual template.

Below are the specified use cases for **Web Application to Support Departmental Timetable Administrators**, categorized into two groups: **Admin and Teacher**

1.4.1 Admin Use Cases

Admin-The admin has the ability to create, update, and delete information related to teachers and courses.Additionally, admin can set decisions for teachers.

UC1: **Show List of Teachers**

Goal: Allows the admin to view a list of all registered teachers within the system, displaying their usernames.

Main Scenario:

1. Admin logs into the system and navigates to the "Teachers" section.
2. System retrieves and displays a list of all registered teachers, sorted by their usernames.

UC2: **Create New Teacher**

Goal: Enables the admin to create a new teacher's profile by entering details such as username and password.

Main Scenario:

1. Admin navigates to the "Teachers" section.
2. Admin selects the "Create Teacher" option.
3. Admin enters the teacher's details, such as username and password.
4. Admin reviews the information.
5. Admin submits the form.
6. System creates the teacher's profile.

Alternative Scenario: If the username is already in use, the system alerts the admin, who must select a different username and resubmit the form.

UC3: Update Teacher

Goal: Allows the admin to update a teacher's credentials, such as username.

Main Scenario:

1. Admin navigates to the "Teachers" section.
2. Admin selects a teacher from the list.
3. Admin chooses the "Update" option.
4. Admin modifies the teacher's credentials (e.g., username).
5. Admin submits the changes.
6. System processes the update.
7. System confirms the update to the admin.

Alternative Scenario: If the username is already in use, the system alerts the admin, who must select a different username and resubmit the form.

UC4: Delete Teacher

Goal: Enables the admin to permanently delete a teacher's profile from the system after confirmation.

Postcondition: The teacher's profile is permanently removed from the system.

Main Scenario:

1. Admin navigates to the "Teachers" section.
2. Admin selects a teacher from the list.
3. Admin chooses the option to delete the teacher.
4. The system asks for confirmation to ensure the deletion is intentional.
5. Admin confirms the deletion.
6. System permanently deletes the teacher's profile.

Alternative Scenario: If the teacher is assigned to any classes, the system warns the admin and denies the deletion until the teacher is unassigned from all classes.

UC5: Show List of Courses

Goal: Allows the admin to view a list of all registered courses, displayed by course codes.

Main Scenario:

1. Admin logs into the system and navigates to the "Courses" section.
2. System retrieves and displays all registered courses, listed by their course codes.

UC6: Create New Course

Goal: Enables the admin to create a new course by entering details such as the course code, information on whether this will be a course with many labs, the number of required classes for each type of parallel, and to submit these details for system confirmation.

Main Scenario:

1. Admin navigates to the "Courses" section.
2. Admin selects the "Create Course" option.

3. Admin enters the course details such as course code and information whether this course contains many labs. Additionally, he sets the number of classes for each parallel type of the course.
4. Admin submits the information.
5. System processes the new course data.
6. System confirms the creation of the new course to the admin.

Alternative Scenario:

1. Admin attempts to register a new course using a code that already exists in the system.
2. System checks the existing course codes.
3. If a duplicate code is found, the system alerts the admin.
4. Admin modifies the course code.
5. Admin resubmits the updated course information.
6. System processes and confirms the creation of the updated course.

UC7: Update Course

Goal: Allows the admin to update existing course details such as the course code and lab requirements, with system verification and confirmation.

Main Scenario:

1. Admin navigates to the "Courses" section.
2. Admin selects a course from the list.
3. Admin opts to update the course.
4. Admin changes the course details, such as the course code, information on whether this course has many lab classes and number of classes for each parallel type of the course.
5. Admin submits the changes.
6. System updates the course information.

Alternative Scenario:

1. Admin updates a course with a new course code.
2. System checks for conflicts with existing course codes.
3. If a conflict is detected, the system notifies the admin of the code duplication.
4. Admin chooses a different course code.
5. Admin resubmits the updated information.
6. System processes and confirms the changes.

UC8: Delete Course

Goal: Allows the admin to permanently delete a course from the system after receiving confirmation.

Main Scenario:

1. Admin navigates to the "Courses" section.
2. Admin selects a course from the list.
3. Admin chooses to delete the selected course.
4. The system requests confirmation to ensure the deletion is intentional.

5. Admin confirms the deletion.
6. System permanently deletes the course.

Alternative Scenario:

1. Admin attempts to delete a course that is currently assigned to a teacher.
2. System checks if the course is assigned to any teacher.
3. If the course is assigned, the system provides a warning.
4. If the warning is ignored, the course is permanently deleted.

UC9: Assign Class to Teacher

Goal: Allows the admin to assign a class to teacher.

Main Scenario:

1. Admin navigates to the teacher's profile.
2. Admin selects the course from the available list.
3. Admin sets the desired parallel type for the course.
4. Admin confirms the assignment.
5. System updates both the teacher's and the course's records accordingly.

Alternative Scenario:

1. Admin attempts to assign a class to a teacher.
2. System checks if the selected teacher is already assigned to a lab class for this course and whether the course has many labs.
3. If the assignment already exists and the course has many labs, the system alerts the admin to the duplication.
4. Admin cancels the action or chooses a different course, parallel type, or teacher.

Exception

1. Admin attempts to add a class.
2. System checks the required number of classes for the selected parallel type.
3. If the required number of classes for the selected parallel type is less than the number the admin is trying to add, the system displays an error message indicating that no additional classes are needed for this parallel type.
4. The system prevents the creation of the class.

UC10: Remove Class from Teacher

Goal: Enables the admin to remove a teacher's assigned class.

Precondition: The teacher has an assigned class.

Main Scenario:

1. Admin navigates to the teacher's profile.
2. Admin selects a class from the teacher's listed classes.
3. Admin chooses to remove the selected class.
4. The system requests confirmation to ensure the removal is intentional.
5. Admin confirms the removal.

6. System processes the removal and updates the records of both the teacher and the course.

UC11: Assign Teacher to Class

Goal: Allows the admin to assign a teacher to a class.

Main Scenario:

1. Admin navigates to the "Courses" section.
2. Admin selects a course and chooses to assign a teacher.
3. Admin selects the appropriate teacher and sets the parallel type.
4. Admin confirms the assignment.
5. System updates the course and teacher records accordingly.

Alternative Scenario:

1. Admin attempts to assign a teacher to a class.
2. System checks if the selected teacher is already assigned to a lab class for this course and whether the course has many labs.
3. If the assignment already exists and the course has many labs, the system alerts the admin to the duplication.
4. Admin cancels the action or chooses a different course, parallel type, or teacher.

Exception

1. Admin attempts to assign a teacher to a class.
2. System checks the required number of classes for the selected parallel type of the selected course.
3. If the required number of classes for the selected parallel type is less than the number the admin is trying to add, the system displays an error message indicating that no additional classes are needed for this parallel type.
4. The system prevents the creation of the class.

UC12: Remove Teacher from Class

Goal: Enables the admin to remove a teacher's assigned class.

Main Scenario:

1. Admin navigates to the "Courses" section.
2. Admin selects a course from the list.
3. The system displays all classes of the course.
4. Admin chooses to remove one or more classes of the desired teacher.
5. The system requests confirmation.
6. Admin confirms the removal.
7. System updates the records accordingly.

UC13: Set Class Teaching Decision for Teachers

Goal: Admin sets the decision status for a teacher's engagement with assigned class .

Main Scenario:

1. Admin accesses the teaching assignments for a specific teacher.
2. Admin selects a class.

3. Admin sets the decision status (YES, NO, PENDING).
4. System updates the status in both the teacher's and course's section.

UC14: Set Min/Max for Labs for Teachers

Goal: Enables the admin to set the minimum and maximum number of lab classes for courses that "haveLabs" for each teacher.

Main Scenario:

1. Admin navigates to a teacher's profile and accesses the lab management section.
2. Admin selects the lab teaching assignments.
3. Admin adjusts the minimum and maximum number of lab classes each teacher can handle.
4. The system updates and confirms the new settings.

Exception:

1. If the admin enters a maximum number of lab classes that is less than the minimum number, the system automatically sets the maximum number to be equal to the minimum number.
2. The system updates and confirms the new settings with the maximum number equal to the minimum number.

UC15: Show Teacher Info

Goal: Allows the admin to access detailed information about a teacher, including assigned classes and information about them.

Main Scenario:

1. Admin navigates to the "Teachers" section within the application.
2. Admin clicks on the username of a teacher.
3. The system directs the admin to the selected teacher's page, displaying detailed information.

UC16: Show Course Info

Goal: Allows the admin to access detailed information about a course, including assigned teachers and their respective classes.

Main Scenario:

1. Admin navigates to the "Courses" section within the application.
2. Admin clicks on the code of a course.
3. The system directs the admin to the Course Info page, showing detailed information about the course and its assigned teachers. Additionally, for each parallel type of the course, it displays the number of current and the number of required classes.

UC17: View Course Coverage

Goal: Enables the admin to view the coverage status of courses, specifically if they are fully staffed by teachers or not.

Main Scenario:

1. Admin navigates to the "Courses" section.
2. Admin selects a specific course.
3. The system displays information on whether it is fully covered with teachers or not.

1.4.2 Teacher Use Cases

Teacher-The teacher has the ability to set decisions for assigned classes. Additionally, teachers can set number min and max on how many lab classes they can teach.

UC18: Set Decisions for Assigned Classes

Goal: Teacher sets the decision status for their engagement with an assigned class. **Main Scenario:**

1. Teacher accesses their teaching assignments.
2. Teacher selects a class.
3. Teacher sets the decision status (YES, NO, PENDING).
4. System confirms the changes.

UC19: Set Min/Max for Assigned Lab Class

Main Scenario:

1. Teacher navigates to their profile and accesses the lab management section.
2. Teacher selects the lab teaching assignments.
3. Teacher adjusts the minimum and maximum number of lab classes they can handle.
4. The system updates and confirms the new settings.

Exception:

1. If the teacher enters a maximum number of lab classes that is less than the minimum number, the system automatically sets the maximum number to be equal to the minimum number.
2. The system updates and confirms the new settings with the maximum number equal to the minimum number.

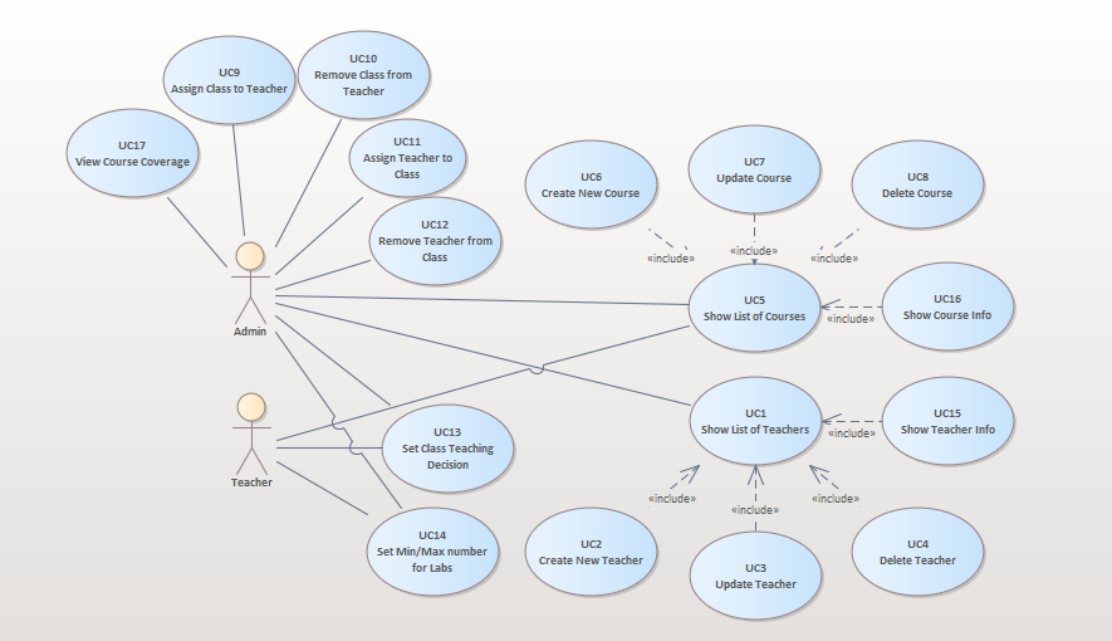

Figure 1.3 Use Case Diagram Describing Actors and their Use Cases

1.4.3 Realization Matrix

Requirements Satisfaction: We should check that every functional requirement is realized by at least one use case. For this purpose, a realization matrix was created:

Table 1.1 Use Cases corresponding to Functional Requirements.

Functional Requirements	Use Cases
F1	UC1—UC4, UC16
F2	UC5—UC8, UC15
F3	UC9—UC14, UC18, UC19
F4	UC17

1.5 Domain Model

This chapter explores the system's requirements, both functional and non-functional, analyzes the problems it aims to address, and reviews use cases to guide the construction of the domain model.

User Entity The User entity represents individuals who can access the application, encompassing both administrators and teachers.

Course Entity The Course Entity helps us to create and store courses with course code. Additionally, it stores information if this course "hasLabs".

Class Entity The Class entity represents individual classes that users can enroll in. Each class is based on a specific course and is categorized by a particular parallel type (such as lecture, seminar, or lab). A class can have multiple users enrolled in it, and each class is associated with one course and one parallel type.

Parallel Type Enumeration The Parallel Type is structured to associate parallel types (LECTURE, SEMINAR, LAB) with courses.

Decision Enumeration The Users (teachers) decide whether they wish to teach certain classes, for which three states have been established: the default state (PENDING), and two additional states (YES, NO) to facilitate this decision-making process.

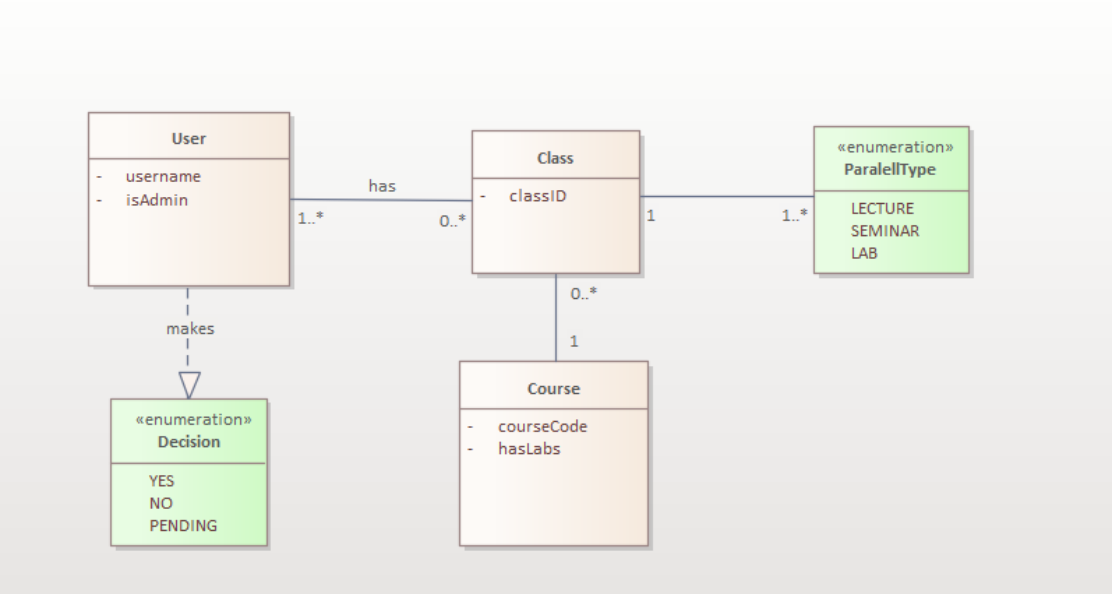

Figure 1.4 Domain Model

Chapter 2
Design

This chapter will outline the design of the "Web Application To Support Departmental Timetable Administrators" application, covering the system architecture and its key functional elements. It will delve into the server-side architecture and the user interface layer, including a thorough discussion of the wireframes.

2.1 System Architecture

The software architecture of a computing system is the set of structures needed to reason about the system, which comprise software elements, relations among them, and properties of both. [6]

In this section, we will discuss the system architecture, including its types and the functionality of various components. However, I will first outline several key points and provide definitions.

2.1.1 Overview of Different Architecture Types

This subsection offers general information about different types of system architecture and describes their advantages and disadvantages. After reviewing these architecture types, the final decision will be made to choose one of them for development.

Monolithic Architecture

Monolithic architecture is a traditional model of software development where all components of an application are integrated into a single, self-contained unit. This single unit typically includes the user interface, business logic, and data access layers. [7]

Advantages:

- *Simplicity*: Easier to develop, test, and deploy since all parts of the application are within a single codebase.
- *Performance*: Direct calls within a single process can be faster than calls between services.

Disadvantages:

- *Scalability*: Difficult to scale individual components; the entire application must be scaled.
- *Maintenance*: Large codebases become cumbersome and difficult to manage over time.
- *Flexibility*: Limited flexibility in technology choices as the entire application must be written in the same language and framework.

MicroServices Architecture
A microservices architecture is a type of application architecture where the application is developed as a collection of services. It provides the framework to develop, deploy, and maintain microservices architecture diagrams and services independently. [7]

Advantages:

- *Scalability*: Each service can be scaled independently, optimizing resource usage.
- *Flexibility*: Different services can use different technologies, which allows teams to choose the best tool for each job.
- *Resilience*: Failure in one service does not necessarily impact the others, improving overall system reliability.
- *Speed of Deployment*: Individual services can be deployed independently, enabling continuous deployment practices and faster iteration.

Disadvantages:

- *Complexity*: Increases the complexity of the system due to the need for service coordination and communication.
- *Data Consistency*: Maintaining data consistency across distributed services can be challenging.
- *Operational Overhead*: Requires sophisticated infrastructure for monitoring, logging, and managing multiple services.

2.1.1.1 Decision

In the context of our application, the monolithic architecture was chosen for the following reasons:

Simplicity and Quick Development: The monolithic approach allowed me to quickly set up the application and begin development. Given the project's requirements and my familiarity with the technology stack, this was the most straightforward choice.

Ease of Maintenance: For the initial phase of the project, having a single codebase made it easier to manage and understand the interactions between different components.

Testing: With a monolithic application,manual testing was more straightforward. I did not have to deal with the complexities of managing multiple services.

2.2 Server Design

Three-layered-application — is a software design pattern that separates an application into three distinct layers: the presentation layer, the business logic layer, and the data access layer. Each layer is responsible for specific functionalities and communicates with the other layers to provide a cohesive application. This separation enhances modularity, scalability, and maintainability.

Presentation Layer This layer is responsible for creating routes that expose the server's functionality via a REST API, using the HTTP protocol. It essentially provides an interface for users and other systems to interact with the application.

Business Layer Situated between the presentation and data layers, the business layer encapsulates the core application logic of the server. This includes processing incoming data, applying business rules, and preparing data for storage or dissemination.

Data Layer

The underlying layer is the data layer, whose primary function is data persistence. In this architecture, it interfaces with an SQL Server database, ensuring that all data manipulated by the business layer is accurately stored and readily available for retrieval.

2.3 Data Layer

2.3.1 Data Managament

SQL Databases:

- **Strengths**:
  - Structured schema offers robust data organization and integrity.
  - Well-established security frameworks like Spring Security provide formidable defenses against breaches.
  - Transactions ensure data consistency and reliability.
- **Weaknesses**:

  NOSQL Databases:
  - Rigid schema architecture may pose challenges in adapting to evolving requirements.
  - Scaling can be complex and may require additional resources.
  - May incur higher costs for licensing and maintenance.
- **Strengths**:
  - Flexible schema allows for seamless integration of new data elements without schema modifications.
  - Horizontal scalability enables handling of large volumes of data with ease.
  - High performance and speed, especially for unstructured or semi-structured data.
- **Weaknesses**:
  - Lack of strict schema can lead to data inconsistency and integrity issues.
  - Limited support for complex queries and transactions.
  - May require more expertise and effort to set up and manage effectively.

In conclusion, the decision to select a database language was influenced by several critical factors, including simplicity, ease of setup, reliability, the presence of a well-established community, and prior experience. Among the various database languages I have tried over the years, SQL consistently stood out due to its extensive resources and simplicity. Despite its difficulties, such as the rigidity of its schema, the advantages of using SQL, particularly its reliability and community support, far outweigh these potential drawbacks. Thus, SQL was the most suitable choice for my database management needs.

2.3.2 Roles

To fulfill the system requirements detailed in N4, which necessitate the implementation of distinct roles to manage access to various pages, Role-Based Access Control (RBAC) [8] is utilized as a strategic approach. Specifically, the Flat RBAC model is employed, where each role is defined with a specific set of permissions without any hierarchical structure. This model involves creating a Role entity that encapsulates essential details such as the role's name and its precise permissions. By assigning users to these predefined roles, the system enables tailored access control, allowing users to interact only with parts of the system relevant to their roles. This method streamlines permission management by centralizing modifications at the role level, rather than adjusting individual user settings, thereby enhancing system security and ensuring consistency.

2.4 Presentation Layer

2.4.1 Wireframes

Wireframes are a fundamental component in the user experience (UX) design process, often likened to the architectural blueprints used in building construction. They serve as a visual guide that outlines the skeletal framework of a website or application, focusing on layout, content arrangement, and functionality rather than aesthetic details like color and typography.
The author chose to display the wireframes for the application's user interface. These were designed based on the analysis from the previous chapter, focusing mainly on the system's functional requirements and use cases.

- Figure 2.1 displays the administrator's interface related to managing teachers, illustrating the use cases UC1, UC2, UC3 and UC4. The interface features a navigation bar at the top with options for "Teachers," "Courses," "Log Out," and "Admin," highlighting that these are admin functionalities. Below the navigation bar, the interface provides a button labeled "Create a new Teacher," which allows administrators to add new teachers to the system. Beneath this button, the main section of the screen lists all currently registered teachers in a table format. Each teacher's name appears alongside two action buttons: 'Edit' and 'Delete.' These buttons enable the administrator to modify details about a teacher or remove a teacher from the system, respectively.

  This interface facilitates easy management of teacher records, providing essential functions in a clear and organized layout to ensure efficient administrative operations. The design aims to keep user interactions straightforward, enabling quick access to teacher details.

  For the purpose of this documentation, it should be noted that this navigation bar is consistent across all administrative interfaces and will not be described in detail in subsequent wireframe figures unless there are specific modifications relevant to a particular discussion.

- Figure 2.2 displays the administrator's interface related to managing teachers, encompassing the use cases UC5, UC6, UC7, UC8, UC17. At the top of the interface, just below the standard navigation features, there is a button labeled "Create a new Course." This button enables administrators to add new courses to the system. The main section of the screen presents a table listing each course.

- Figure 2.3 displays the interface for assigning a course to a teacher, illustrating the use case UC9.

  At the top of the interface, 'dovsteven' is displayed, identifying the username of teacher to whom the course will be assigned. Below this, there are two dropdown menus involved in the course assignment process. The first dropdown allows administrators to choose a course by its code. Adjacent to it, the second dropdown menu is used to select the parallel type such as

"Lecture". An 'Add' button is located below these menus to complete the course assignment to the teacher with username, 'dovsteven.'

- Figure 2.4 illustrates the interface for assigning a teacher to class, supporting the use case UC11. The main section starts with the course code "BIE-PA1" prominently displayed. Below this, the section labeled "Assigning Teacher to Course" contains two dropdown menus. The first dropdown allows administrators to select a teacher from a list, and the second dropdown enables the selection of the parallel type, such as "Lecture." An "Add" button is located below these dropdowns to finalize the assignment of the selected teacher to the chosen course parallel.
- Figure 2.5 displays the interface for removing a course parallel from a teacher, illustrating the use case UC10. The interface displays the username of the teacher whose course parallels are being managed. Below this, there is a table listing the registered courses for the teacher, including the course code, the type of parallel (e.g., LAB, LECTURE, SEMINAR), and a "Delete" button next to each course parallel. This delete button allows the admin to remove the selected course parallel from the teacher's assignments.
- Figure 2.6 displays the administrator's interface for managing teachers, which includes the use case UC10. In the main section of the screen, the administrator is presented with a table listing each teacher registered for a course. In this case, the course is "BIE-PA1". The table includes columns for "Username," the type of parallel session (LAB, LECTURE, SEMINAR), and a button labeled "Delete" for each teacher entry. Administrators can remove a teacher from the course parallel by using this button.
- Figure 2.7 displays the administrator's interface related to managing courses, illustrating the use cases UC11.
- Figure 2.8 displays the teacher's interface for managing their assigned courses, encompassing the use cases UC9, UC10, UC11, and UC12. The screen shows the designated courses for the logged-in teacher, in this case, the username "dovsteven". In the main section, there is a table with columns for "Course," "Parallel," and "Decision" (with options such as YES, NO, PENDING), and a "Delete" button for each course entry. This gives the teacher the option to confirm or decline their assignments and remove them if needed. Below the table, there's a section to save changes and manage registered LAB classes. In the section under it, the user specifies the minimum and maximum required lab classes.

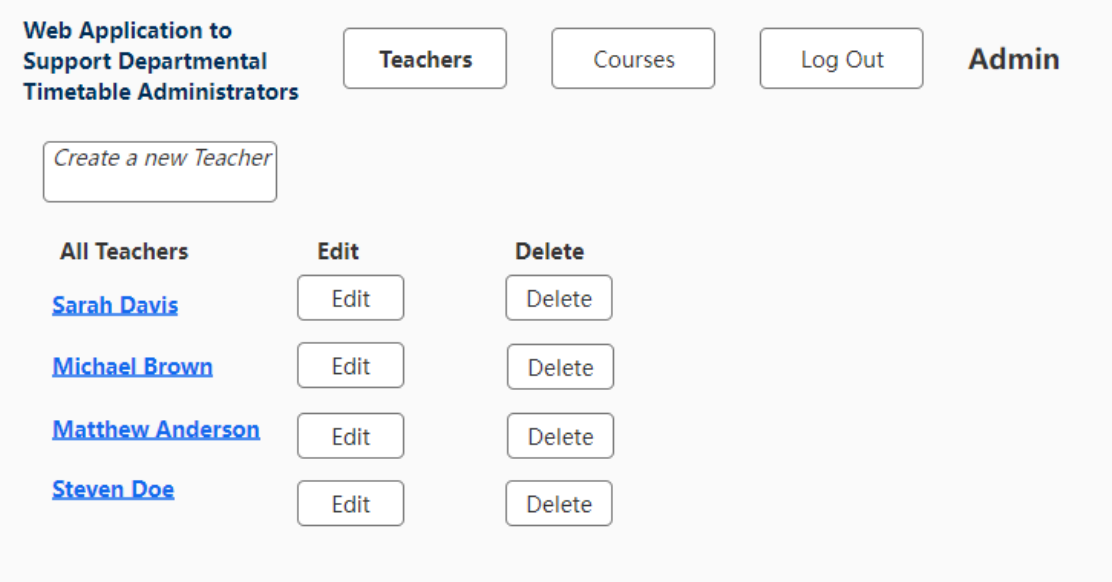

Figure 2.1 List of All Teachers

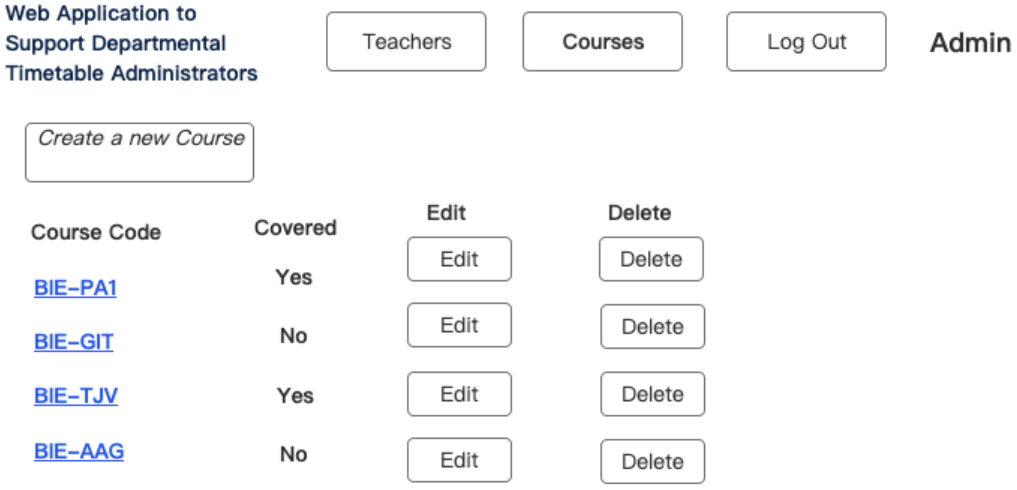

Figure 2.2 List of All Courses

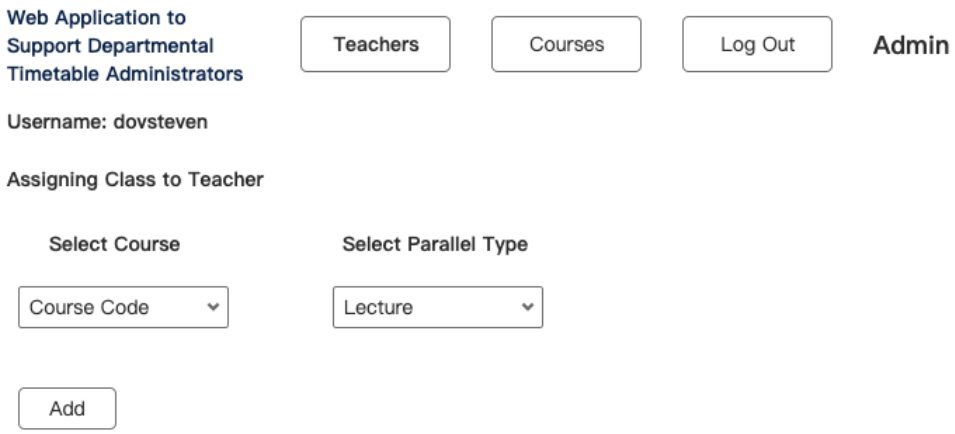

Figure 2.3 Assigning Class to Teacher

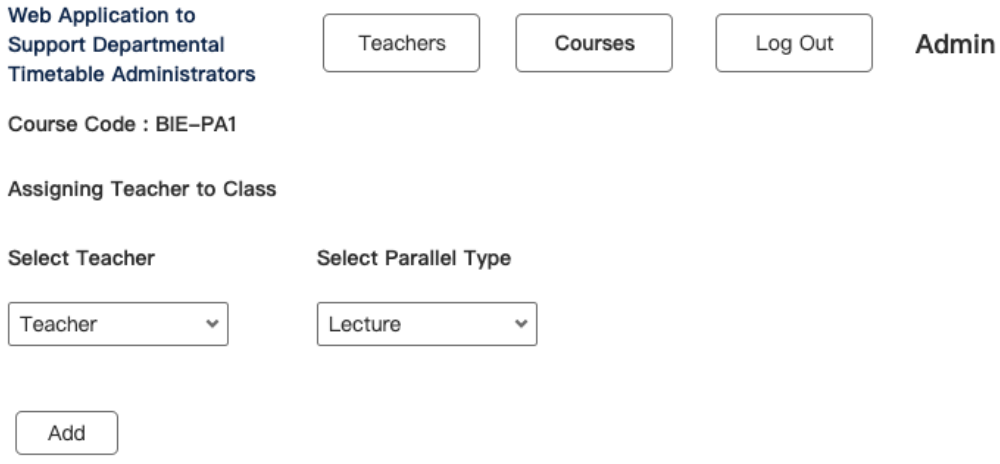

Figure 2.4 Assigning Teacher to Class

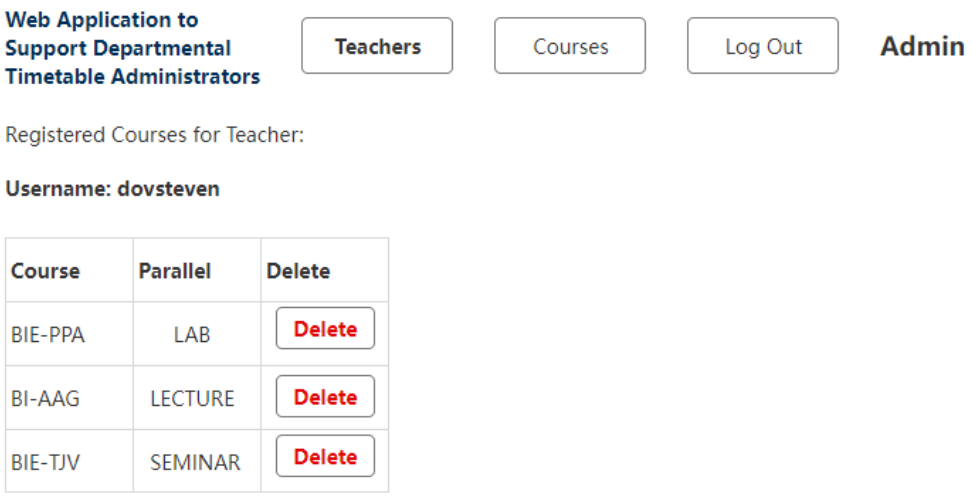

Figure 2.5 Removing Class from Teacher

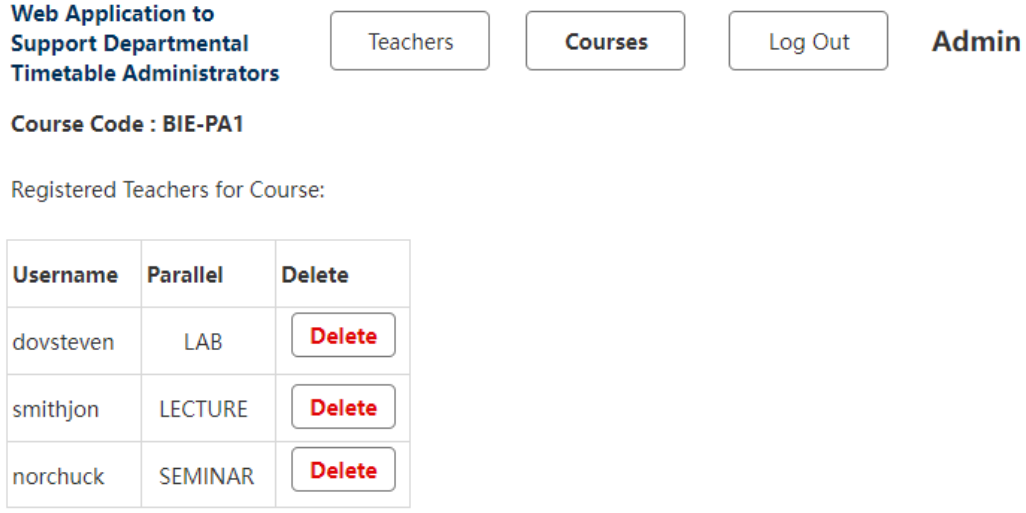

Figure 2.6 Removing Teacher from Class

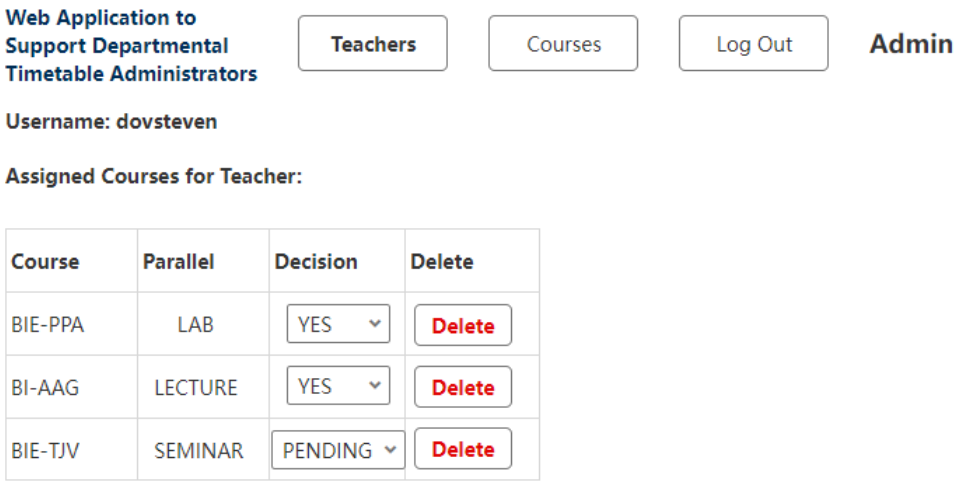

Figure 2.7 Single Teacher View for Admin

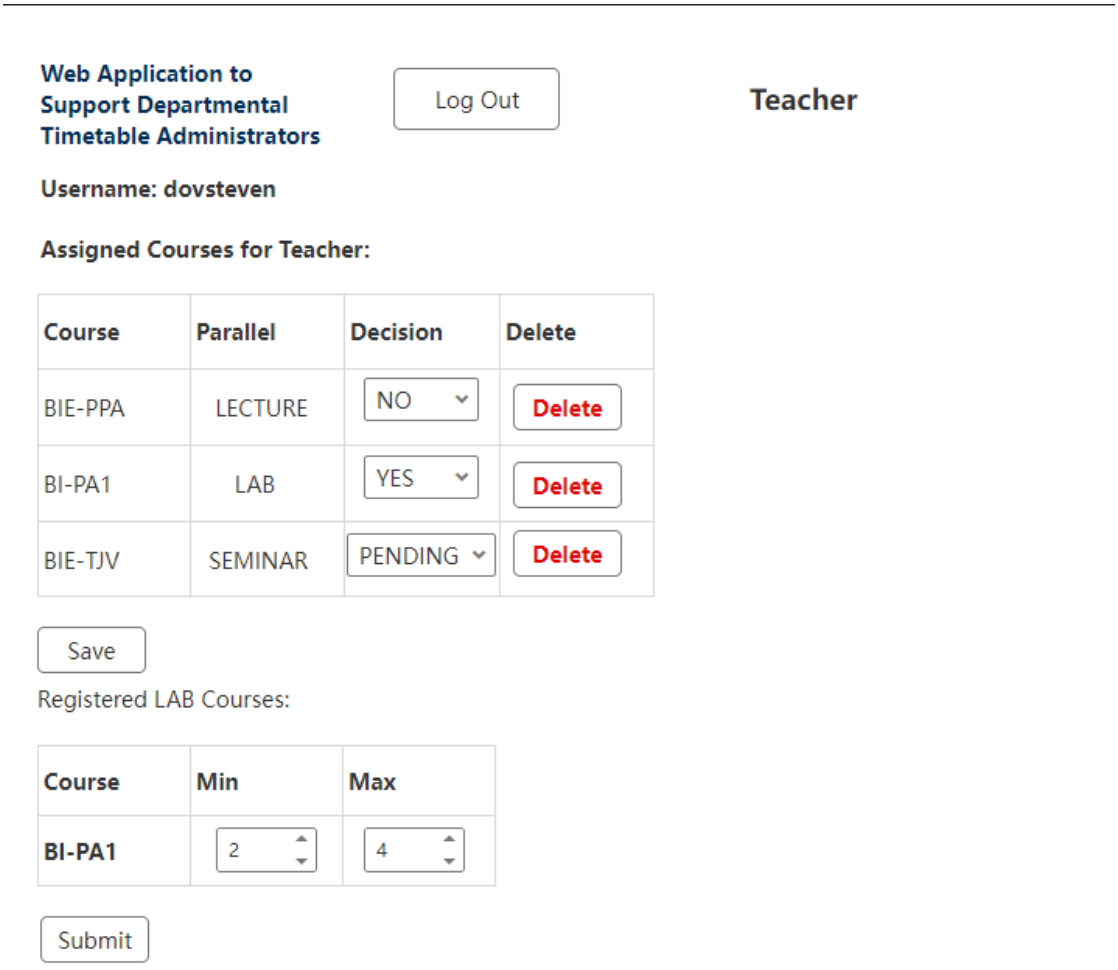

Figure 2.8 Teacher Accessing App

Chapter 3
Implementation

This chapter aims to demonstrate the implementation of the "Web Application to Support Departmental Timetable Administrators" prototype using the design discussed in the previous chapter.

3.1 Server Implementation

3.1.1 Spring Boot

Spring Boot is an open-source framework based on the Spring Framework, designed to simplify the development of production-ready applications. It reduces the need for extensive boilerplate code and configuration by providing default settings and built-in functionalities. Spring Boot allows developers to build stand-alone, production-grade applications that can be easily started and configured using embedded servers such as Tomcat or Jetty. It also includes features for security, data access, and dependency management. [9]

According to the project requirements and NF1, Spring Boot was selected as the development framework.

3.2 Libraries of Spring Boot

3.2.1 Spring Data JPA

Spring Data JPA, part of the larger Spring Data family, makes it easy to easily implement JPA-based (Java Persistence API) repositories. It makes it easier to build Spring-powered applications that use data access technologies. Spring Data JPA aims to significantly improve the implementation of data access layers by reducing the effort to the amount that's actually needed. [10]

3.2.2 Spring Security

Spring Security is a powerful and customizable authentication and access-control framework for Java applications. It is the de-facto standard for securing Spring-based applications, providing comprehensive security services including authentication, authorization, and protection against common security vulnerabilities. [11]

There was a need to distinguish between admin and teacher roles, ensuring that each user type sees different views and has access to different functionalities. Spring Security allowed me

to implement role-based access control described in Chapter 2.3.1, providing the necessary tools to secure our application by defining distinct access rules for admins and teachers.

3.3 Choice of Programming Language

When selecting the programming language for your Spring Boot application, it's essential to consider the advantages and disadvantages of various options such as Java, Kotlin, and Apache Groovy. Each language has its unique features and trade-offs.

3.3.1 Java

Java is a high-level, class-based, object-oriented programming language designed to be platform-independent. It is a general-purpose programming language intended to let application developers write once, run anywhere (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. [12]
Advantages:

- **Mature Ecosystem:** Java has a mature and extensive ecosystem with a vast array of libraries, frameworks, and tools.
- **Performance:** Java's performance is robust, and it benefits from years of optimization in the JVM.
- **Static Typing:** Static typing helps catch errors at compile-time, improving code reliability and maintainability.
- **Community and Support:** Java has a large and active community, providing resources, tutorials, and support.

Disadvantages:

- **Verbose Syntax:** Java tends to be more verbose compared to modern languages like Kotlin, leading to more boilerplate code.

3.3.2 Kotlin

Kotlin is a statically typed, object-oriented programming language that is interoperable with the Java virtual machine (JVM), Java Class Libraries and Android. [13]
Advantages:

- **Conciseness:** Kotlin reduces boilerplate code, making the codebase more concise and readable.
- **Null Safety:** Built-in null safety features help prevent NullPointerExceptions.

Disadvantages:

- **Learning Curve:** Learning Kotlin might require additional training and adaptation.

3.3.3 Decision

Given my extensive experience with Java, its mature ecosystem, and seamless integration with Spring Boot, I chose Java over Kotlin. While Kotlin offers modern features and flexibility, my lack of experience with this language and the project's need for a stable, maintainable prototype application made Java the most practical and reliable choice.

3.4 Java Libraries

This section highlights some of the key Java libraries, demonstrating their importance and how they were integrated into project to achieve optimal results.

3.4.0.1 Lombok

Lombok is a Java library that automatically plugs into your editor and build tools, spicing up your Java with methods like getter/setter, equals, hashCode, and toString, and handling non-null checks, constructors, and much more with annotations.

3.5 Server-side View Implementation with Thymeleaf

Thymeleaf is a modern server-side Java template engine for web and standalone environments capable of processing HTML, XML, JavaScript, CSS, and even plain text. It is well-suited for serving XHTML/HTML5 in web applications, allowing seamless integration with Spring Boot to create dynamic web pages. Thymeleaf is designed to be a replacement for JSP (JavaServer Pages) and provides an elegant and highly maintainable way of creating templates. Thymeleaf has following advantages:

Why I chose Thymeleaf?

- **Seamless Integration with Spring Boot:** As my project is built on Spring Boot, choosing Thymeleaf was a natural fit. Thymeleaf integrates seamlessly with Spring Boot, allowing for easy configuration and smooth operation within the Spring ecosystem. This tight integration simplifies development by providing consistent and straightforward ways to handle template rendering, form submissions, and data binding directly within the Spring framework.
- **Simplified Development:** Both the server-side logic (Spring Boot) and the view rendering (Thymeleaf) are managed in the same codebase, simplifying development and maintenance.
- **Dynamic Content Injection:** Thymeleaf enables dynamic content injection into HTML pages directly from the server, reducing the need for complex client-side JavaScript for rendering dynamic data.
- **Reduced Client-Side Complexity** By leveraging Thymeleaf for server-side rendering, I can focus more on building the core functionality of the application without getting bogged down by complex client-side frameworks. This approach not only simplifies the development process but also enhances the performance of the application. Since the server generates the HTML, the client receives a complete and ready-to-display page, leading to faster initial load times and a more responsive user experience.

3.6 UI Implementation

3.6.1 Bootstrap

Bootstrap is a popular front-end framework used for developing responsive and mobile-first websites. Originally created by Twitter, it provides a collection of reusable HTML, CSS, and JavaScript components. These components simplify the process of building web pages by offering pre-designed elements such as buttons, forms, navigation bars, and modals, which can be easily customized to fit specific design needs. [14]

Combining Bootstrap and Thymeleaf in web application development offers numerous benefits from my perspective. Bootstrap provides a comprehensive library of pre-designed UI components

that seamlessly integrate with Thymeleaf templates. This integration allows me to swiftly create visually appealing web pages. Moreover, Bootstrap ensures that web pages are optimized for mobile devices and maintain a consistent, professional appearance across various browsers and devices.

3.6.2 JavaScript and AJAX

JavaScript is a versatile programming language used to enhance the interactivity and dynamic behavior of web pages. It enables the creation of interactive elements, efficient handling of user events, and manipulation of the Document Object Model (DOM) to update the web page content in response to user actions.

AJAX (Asynchronous JavaScript and XML) is a technique that allows web applications to communicate with the server asynchronously. This means data can be sent and retrieved from the server without needing to reload the entire web page, thereby providing a smoother and faster user experience [15].

Benefits

Improved User Experience JavaScript and AJAX enable dynamic content updates and interactive elements on the web page, creating a more responsive and seamless experience for the user.

Efficiency Using AJAX, web applications can fetch and send data incrementally, reducing the need for full page reloads. This decreases load times and reduces the overall strain on the server.

Security By implementing security measures such as CSRF tokens, AJAX requests can be secured and authenticated, ensuring that data transactions between the client and server are protected

I chose JavaScript and AJAX for their ability to create highly interactive and responsive web applications. JavaScript's versatility allows for real-time updates and event handling directly in the browser, while AJAX enhances the efficiency and user experience by enabling asynchronous communication with the server.

Chapter 4
Testing

In this chapter, the implemented system is discussed in terms of testing, as well as outlining possible improvements for future development.

4.1 Manual Testing

Automated testing can save time and increase accuracy, but it requires significant setup and maintenance. Due to the rapid changes and limited scope of this project, I focused primarily on manual testing, with plans to incorporate automated tests if time permits later on. Manual Testing is widely used in software development because it doesn't necessitate extra effort and resources. The manual testing process was straightforward. Each time a new feature was implemented, I retraced all the steps leading to that feature to ensure it functioned as expected. Any bugs found during testing were documented, and the scenarios causing these bugs were added to the testing steps for re-verification after fixing.

My supervisor also assisted in testing to confirm that the applications met his expectations. He reported any bugs or unexpected behavior for resolution.

4.1.1 Usability Testing

Usability testing is the type of manual testing. The purpose of usability testing is to evaluate user interfaces and to ensure the quality of the system. It helps in identifying the design issues from the user perspective. In a usability-testing session, a researcher (called a "facilitator" or a "moderator") asks a participant to perform tasks, usually using one or more specific user interfaces. While the participant completes each task, the researcher observes the participant's behavior and listens for feedback. [16]

The results I will obtain through testing will help me identify issues with the user interface design, discover opportunities for improvement, and understand the behavior and preferences of the intended audience.

4.1.1.1 Process

A diverse group of teachers from universities, holding various academic degrees, were selected as users to test the functionality of "Web Application to Support Departmental Timetable administrators". Additionally, since this application was specifically designed for use by university staff, we were not interested in users who did not meet these criteria.

Initially, the author took on the role of administrator while the teachers acted as teachers. Subsequently, the author switched to the role of a teacher, and the teachers alternated as the

administrator. Each participant tested the prototype independently, without real-time communication. The prototype was operated on a single laptop during these tests.

To make the testing process more engaging, the database was pre-populated with dummy data (please find attached SQL script in zip) such as sample teachers, courses and their assigned classes. This allowed testers to experience the app more realistically and provide more relevant feedback.

Established tasks for each role to test the common functionality that covers the use cases were following:

Admin:

1 Open the application and log in using the provided credentials.

2 Try to list all teachers registered in the app

3 Try to list all courses registered in the app.

4 Try to create teachers and update or delete one of them.

5 Try to create courses and update or delete one of them.

6 Try to assign class to teacher and vice versa.

7 Try to remove assigned class from teacher and vice versa.

9 Set decision for teacher's assigned class.

10 Check if course has been covered by teachers.

Teacher:

1 Open the application and log in using the provided credentials.

2 Try to see assigned classes.

3 Try to set decision on assigned classes.

4 Try to input desirable number min and max for assigned lab classes.

Once the tasks were completed by the users, I asked the following questions to conclude the testing:

1 How would you describe your overall experience with the applications?

2 Was the location of the UI elements or the flow of the actions intuitive or sometimes additional instructions were needed?

3 Did you encounter any unexpected behaviour?

4 Do you feel like features or some functionality were missing and what do you think could be changed or improved?

The responses provided by the users were as follows:

User1

1 Overall good experience with the app.

2 The location of UI elements and flow of actions were intuitive, I don't have any negative feedback regarding these.

3 I didn't experience any unexpected behaviour.

4 The app could benefit from a search function to find teachers and courses, instead of manually looking for them.

User2

1 The app was user-friendly and overall, I had a positive experience.

2 Most UI elements were easy to find, but a few actions needed additional instructions.

3 No.

4 Adding a notification system for updates or changes would improve usability for both admin and teacher.

User3

1 Using the app was an average experience.

2 The interface was mostly intuitive, though some features could use clearer labels.

3 I found it quite unexpected for admin to perform most of the tasks in one interface.

4 It would be useful to have a feature for exporting data to spreadsheets.

User4

1 Good.

2 The UI was mostly intuitive, but some features were not where I expected them to be.

3 Yes.

4 Adding functionality to highlight which interface the admin is currently using (users, courses) would greatly enhance the user experience.

User5

1 My experience with the app was mixed.

2 Some UI elements were intuitive, but others were confusing and took time to understand.

3 I didn't understand the lab parallel min-max assignment process, it was unexpected for me.

4 I did not like creating teachers and courses manually. It would be nice to have a functionality to incorporate university's login system and get the data directly from university's API's.

4.1.2 Summary

The user testing of the application yielded a range of responses, mostly leaning towards a positive experience. Users found the app to be generally user-friendly with an intuitive interface. They appreciated the logical flow of actions and the ease of finding UI elements. However, there were some notable areas for improvement. Occasional lags were reported, especially during the saving of changes. A few users found the labels on certain features unclear and felt that performing multiple tasks in one interface was unexpected. Additionally, the lab class min-max assignment for courses with many labs was confusing for some users.

4.2 Future Improvements

Based on the feedback from testers, several enhancements can be made to improve the application.

- **Connecting with university login systems** In the future, it would be great accommodate the university's login/authorization page.
- **Additional Constraints** Implementing a search function to easily find teachers and courses would be highly beneficial. Adding a notification system for updates and changes can improve usability for both administrators and teachers. To make the interface more user-friendly, clearer labels should be provided for all features. Additionally, a feature to export data to spreadsheets would be useful. Highlighting the current interface in use (e.g., users, courses) would help navigate the app more efficiently. Lastly, improving the UI design, including colors and overall aesthetics, would make the application more pleasant to use. These improvements can significantly enhance the user experience and functionality of the app.

Conclusion

The primary objective of this thesis was to analyze the existing process of assigning teachers to classes and develop an application that simplifies this process for both timetable administrator and teachers. The resulting prototype is proficient in showing course coverage, assigning teachers to classes, and overall handling all expected tasks from the timetable administrator. The first step was to analyze the assigning creation process, including its key steps and processes. Additionally, this part formulated detailed functional and non-functional requirements and use cases, as well as introduction of the domain model.

The design phase was comprehensively detailed, focusing on the robust system architecture that fulfills the application's functional needs. This included a thorough analysis of the server-side architecture. Furthermore, the user interface layer was created to enhance user interaction and accessibility, integrating responsive wireframes that provide the user with a simple and efficient experience.

Following the design phase, the implementation of the prototype was carried out.This chapter showed how the designs were turned into a working model, using Spring Boot to ensure smooth backend operations. The user interface was made intuitive and user-friendly to help administrators and teachers manage tasks efficiently.

User testing was crucial for evaluating the system's usability and accuracy. Although the prototype showed promise, feedback highlighted areas for improvement, like enhancing the user interface and adding features like data export and integration with university login systems.

In conclusion, while the prototype is not yet ready for deployment due to its limited scope, it provides a strong foundation for future improvements. This work sets the stage for developing a complete system that can significantly simplify the timetable management process, ultimately benefiting educational institutions by reducing administrative work and improving efficiency.

Bibliography

1. HARMON, Paul. *Business Process Change: A Business Process Management Guide for Managers and Process Professionals*. 4th ed. Morgan Kaufmann, 2019. ISBN 0128158476.
2. ARLOW, Jim; NEUSTADT, Ila. *UML 2.0 and the Unified Process: Practical Object-Oriented Analysis and Design*. 2nd ed. Addison-Wesley Professional, 2005. ISBN 0321321278.
3. CZECH TECHNICAL UNIVERSITY IN PRAGUE. *KosAPI Documentation*. 2023. Available also from: `https://kosapi.fit.cvut.cz/projects/kosapi/wiki`. Accessed: 2023-11-10.
4. WIEGERS, Karl; BEATTY, Jay. *Software Requirements*. 3rd ed. Microsoft Press, 2013. ISBN 0735679665.
5. COCKBURN, Alistair. *Agile Software Development*. Addison-Wesley Professional, 2001. ISBN 0201699699.
6. LEONARD J. BASS, Felix Bachmann and. *Documenting Software Architectures: Views and Beyond*. 2nd. Boston, MA: Addison-Wesley Professional, 2010. ISBN 9780321552686.
7. NEWMAN, Sam. *Building Microservices: Designing Fine-Grained Systems*. Sebastopol, CA: O'Reilly Media, 2021. ISBN 9781492047841.
8. AUTH0. *Role-Based Access Control (RBAC)*. 2024. Available also from: `https://auth0.com/docs/manage-users/access-control/rbac`. Accessed: 2024-05-20.
9. BOOT, Spring. *Spring Boot* [`https://spring.io/projects/spring-boot`]. 2024. Accessed: 2024-05-14.
10. SPRING.IO. *Spring Data JPA*. 2024. Available also from: `https://spring.io/projects/spring-data-jpa`. Accessed: 2024-04-11.
11. SPRING.IO. *Spring Security*. 2024. Available also from: `https://spring.io/projects/spring-security`. Accessed: 2024-05-20.
12. ORACLE CORPORATION. *Java Documentation*. 2024. Available also from: `https://docs.oracle.com/en/java/`. Accessed: 2024-05-10.
13. LUTKEVICH, Ben. *What is Kotlin and Why Use it?* 2022. Available also from: `https://www.techtarget.com/whatis/definition/Kotlin`. Accessed: 2024-05-10.
14. TEAM, Bootstrap. *Bootstrap 5.3.0 — Bootstrap Blog*. 2024. Available also from: `https://blog.getbootstrap.com/2024/01/01/bootstrap-5-3-0/`.
15. DOCS, MDN Web. *Fetching data from the server - Learn web development — MDN*. 2023. Available also from: `https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Fetching_data`. Accessed: 2024-06-24.

16. MORAN, Kate. Usability Testing 101. *Nielsen Norman Group*. 2019. Available also from: https://www.nngroup.com/articles/usability-testing-101/. Accessed: 2024-05-14.

Contents of enclosed media

- `readme.txt` ..... file which contains detailed instructions and information about the project
- `src`
  - `impl` ..... implementation of the code
  - `Sample-data` ..... Sql script to test the app
- `text` .....
  - `thesis.pdf` ..... the text of the work in the PDF format