# Timetable generator project

Grigoriță Delia-Iustina, Cotiugă Ionela
Boboc Raul-Mădălin, Duțuc Paul

**Abstract**

Making a timetable takes a lot of effort and time and it takes a lot of patience and many hours to design. A timetable is made for a number of reasons, such as scheduling lectures at universities and schools, making timetables for bus and train schedules, and many more. A timetable takes a lot of time and labor to develop. This article provides a thorough analysis of the planning and execution of an application for creating timetables that can manage a variety of characteristics, including classes, students, teachers, subjects, resources, and classrooms. The program simplifies the preparation of timetables by utilizing a well-organized database and offers an intuitive user interface for effective maintenance. Aspect-Oriented Programming (AOP) and Monitoring-Oriented Programming (MOP) approaches are integrated into the system to optimize a variety of scheduling factors and provide a balanced and ideal schedule. Moreover, the application offers synchronization capabilities with personal calendars, facilitating enhanced coordination for teachers and students. This synchronization ensures that the generated timetable aligns with individual commitments, thus fostering improved time management.

## Introduction and problem presentation

Every academic institution faces the same scheduling challenge once or twice a year: class scheduling. It seems like a laborious task [8]. When time tables were first introduced, scheduling them required a lot of work and time, and could be completed manually by one person or by a group. Creating schedules is of the trickiest and most prone to mistakes applications [8].

The majority of college administrative tasks are now computerized, but because preparing lecture schedules is a complex task, it is still done by hand most of the time. The manual lecture schedule arranging takes a lot of time and work. The lecture schedule scheduling challenge is one in which we must find a solution to meet the specified limitations [4]. Since each institution has a unique timetabling issue, not all colleges will require the commercial software products [4]. As a result, we have created a workable method for creating a lecture course scheduling system that can be adjusted to any scheduling issues at institutions. The present study investigates the difficulties posed by dynamic scheduling in academic settings, with a particular emphasis on the requirement to integrate newly enrolled students into batches and to adapt to the addition of new courses. It becomes necessary to create thorough schedules as soon as possible, and before new courses start, the entire schedule has to be

effectively recalculated. The study also addresses the problem of scheduling exams for several batches on the same day, highlighting the need to take existing facilities and related logistical issues into account.

# State-of-the-art

Mei Rui in [10] proposes a mathematical model for the course scheduling system by means of analysis and summarizing of the current issues. Simultaneously, a new university course schedule system design program is designed and implemented, aiming at this mathematical model, via the application of artificial intelligence's pattern recognition technology [4]. This application is not only very versatile and simple to use, but it can also effectively address the shortcomings of the current course scheduling system.

Evolutionary approaches have been employed by Bhaduri A [2] to address the issue of timetable scheduling. Approaches such as Evolutionary Algorithms (EAs) and Genetic Algorithms (GAs) have been applied with varying degrees of success.

Shrinivasan Dipti [11] describe an ongoing challenge in educational institutions is coming up with a workable lecture/tutorial schedule in a major university department. In this work [11], an evolutionary algorithm (EA)-based method for resolving a very limited university timetabling difficulty. The method makes use of a chromosomal representation unique to the challenge. In a reasonable amount of processing time, workable schedules have been obtained through the use of heuristics and context-based reasoning. To expedite the convergence, an intelligent adaptive mutation technique has been implemented. This paper [11] presents a thorough course scheduling system that has been verified, examined, and tested using actual data from a major institution.

Anuja Chowdhary [4] presents a useful timetabling technique that may be utilized in an automatic timetabling system to handle both strong and weak restrictions so that each instructor and student may check their timetable after they are final for a specific semester but they do not modify them. The Timetable Generation System creates a schedule for each class and instructor based on the teachers convenient schedules, the availability and capacity of physical resources, and certain regulations that apply to certain courses, semesters, teachers, and topic levels.

According to A. Elkhyari [5], the suggested algorithm prioritizes instructor availability while assisting in the resolution of the scheduling issue. This program employs a heuristic methodology to provide a comprehensive fix for scheduling issues in schools. Initially, a temporary timetable is created using a randomly generated subject sequence. The subjects are shifted toward a Clash data structure if the teacher has assigned more lectures than the maximum permitted. Randomization of this variable selection criterion can help prevent cycling and enhance the search.
Timetable generators often address multiple conflicting objectives, such as minimizing clashes, optimizing resource utilization, and accommodating preferences of students and teachers si-

multaneously. AOP and MOP is used to enhance the modularity and maintainability of the codebase, allowing for the easy addition or modification of features without disrupting the entire system [3].

In contrast to alternative approaches addressing the intricacies of timetable generation, our emphasis resides in the strategic utilization of Aspect-Oriented Programming (AOP) and Monitoring-Oriented Programming (MOP) [3]. This distinctive focus is predicated on the notion that AOP and MOP offer a more efficacious and nuanced response to the multifaceted challenges inherent in the creation of a timetable generator.

# Architecture and proposed plan

In our application, the user flow is structured around a secure login system where individuals can register as either teachers (figure 1) or students (figure 2). The distinction in privileges between teachers and students is a fundamental aspect of our platform, reflecting the varying responsibilities associated with these roles. Teachers, granted elevated privileges, play a pivotal role in managing and updating timetables for their respective classes.

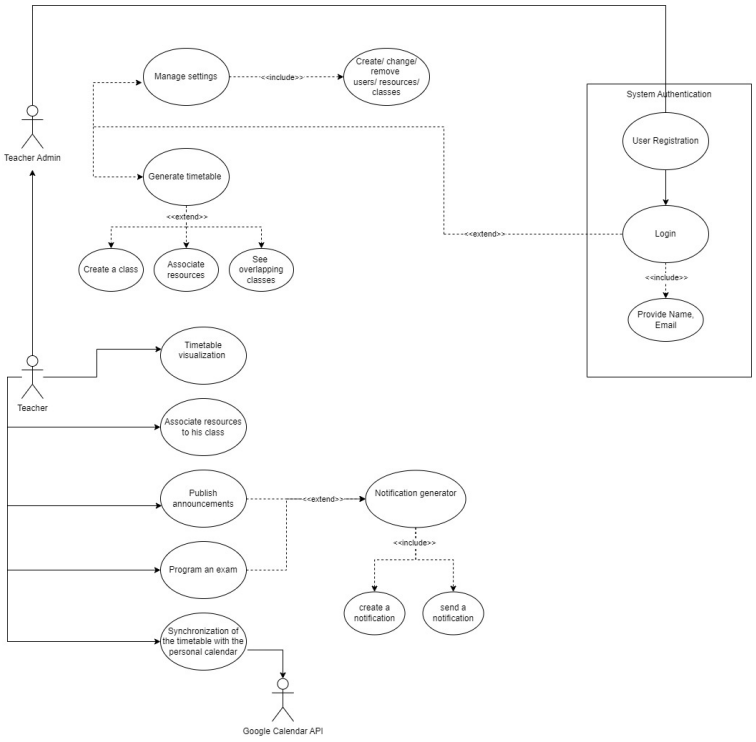


Figure 1: UML diagram for teacher

Upon logging in, a teacher gains access to a comprehensive set of features, empowering them to efficiently handle various administrative tasks. Teachers can create, modify, or remove users, resources, classes, schedules and classrooms. The capability to generate

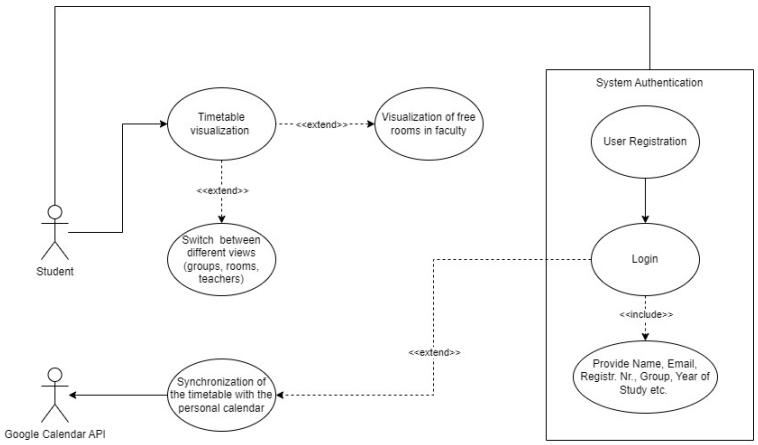


Figure 2: UML diagram for student

timetables is a significant feature, enabling teachers to orchestrate class schedules in line with the evolving needs of their educational programs. Moreover, teachers have the authority to associate resources with their classes, ensuring optimal utilization of available facilities. They can also publish announcements to communicate important information to their students and program examinations as part of their academic responsibilities.

On the other hand, students, while lacking the extensive administrative privileges of teachers, benefit from a simplified yet essential set of features. Students can view the timetables created by their respective teachers, providing them with easy access to class schedules. Additionally, students have the convenience of synchronizing these timetables with their personal calendars, promoting efficient time management and organization. This dichotomy in user roles ensures that the application caters to the specific needs and responsibilities of both teachers and students, fostering a user-friendly and collaborative environment within educational institutions.

Embedded within our system architecture is the adoption of Business Process Modeling Notation (BPMN), as we can see in figure 3, a sophisticated approach rooted in a flowcharting technique akin to activity diagrams in the Unified Modeling Language (UML). BPMN stands as a robust framework, aligning with the primary objective of offering a standardized notation that is universally comprehensible across diverse business stakeholders [9].

# Implementation

Our implementation creates a unified and dynamic user experience by integrating frontend and backend technologies. Python [7] is the preferred programming language for the backend because of its adaptability and efficiency, which allow it to manage the complex logic and data manipulation that are intrinsic to our application. At the same time, we utilize the potent combination of HTML, CSS, and JavaScript on the front end, enhanced

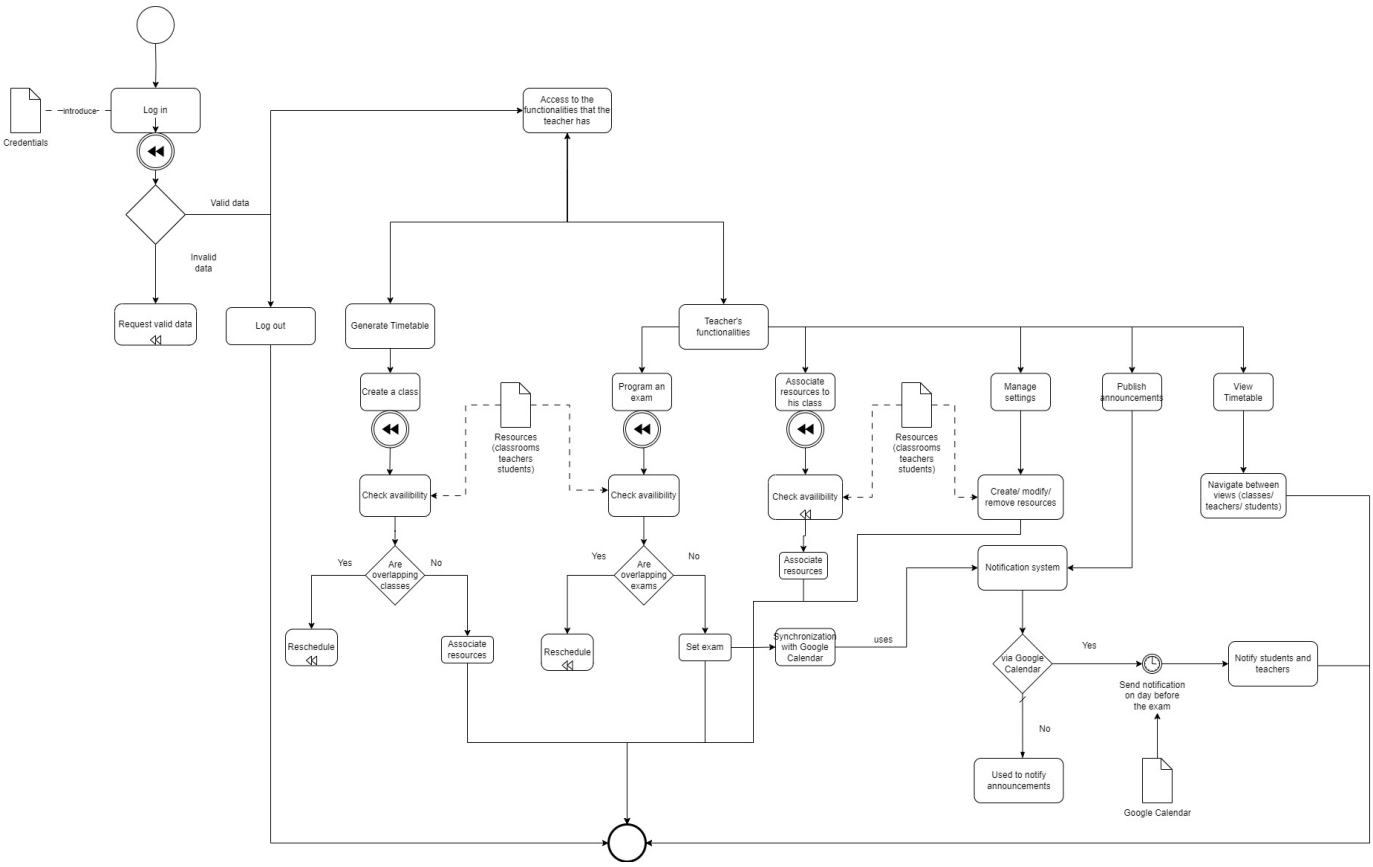


Figure 3: Business Process Modeling Notation (BPMN)

by the strong features of Bootstrap, to create a user experience that is both logical and aesthetically pleasing.

Our application's robust data management is the utilization of Django [6] , a high-level web framework for Python that empowers us to create a structured and efficient database. Within this Django-powered database, we store and organize a wealth of critical information, encompassing classes, teachers, students, resources, schedules, and more.

Python powers the backend by coordinating the numerous procedures and managing the complexity of data management. Because of its large library and natural readability, our backend implementation is strong and provides a scalable and maintainable base for our application.

The frontend is sculpted by the seamless collaboration of HTML, CSS, and JavaScript, which results in an aesthetically beautiful and responsive design. Strong front-end framework bootstrap makes our application even more aesthetically pleasing and responsive, offering a unified and intuitive user experience across a range of screens and devices.

Most importantly, we extend traditional programming paradigms by implementing both Monitoring-Oriented Programming (MOP) and Aspect-Oriented Programming (AOP) in our technique [3]. AOP makes it easier to modularize cross-cutting issues, which improves our codebase's extensibility and maintainability. MOP adds another degree of control and flexibility at the same time, making it possible to monitor and modify system behavior in

real time [3]. Our application is strengthened by this dual programming paradigm approach, which promotes a harmony between user-centric design, system observability, and code modularity. Our application, which combines several technologies and programming paradigms, is proof of an inventive and comprehensive approach to software development.

## Aspect-Oriented Programming

Using the aspectlib library's capabilities [7], our project has carefully integrated aspects into each of the 12 different views that make up our application: admin, class, classroom, student, resource, schedule, subject, teacher, authentication, google calendar, index and timetable. Our system is guaranteed to be more than simply a static information repository thanks to this thorough integration; rather, it is a dynamic and responsive entity that is highly aware of changes occurring in real time.
Whether a user(teacher) interacts with the system by adding a new student, changing a class, or updating the schedule, the aspect-oriented approach makes it possible to invoke pertinent aspects instantly. This in turn sets off notifications that function as dynamic alerts, giving a clear and immediate picture of the changes that are taking place. Our capacity to watch over, control, and react to the changing dynamics inside each part of our application is much improved by this real-time connection.
Essentially, the addition of aspects—made possible by the aspectlib library—gives our project an increased degree of responsiveness and consciousness.

### Example

An example of how aspects are integrated into the view that adds students is provided here, taken from our source. This implementation acts as a template for comparable modifications for every perspective.The "add student" function in this sample represents the aspect-oriented methodology. The "aspectlib" is the essential component of this strategy.Aspect decorator, which changes the function into an aspect, allowing us to inject behavior dynamically into the view's functionality.

```
@aspectlib.Aspect
\texttt{def add_student(*args, **kwargs):}
    print("Adding a student...")
    result = yield aspectlib.Proceed
    print("Student added successfully.")
```

The aspect begins with a print statement upon invocation, indicating that the process has begun — in this example: "Adding a student...". The next line, "result = yield aspectlib.Proceed", opens the door for the aspect to smoothly become part of the view's current flow, "aspectlib.Proceed" basically means that the element should permit the initial operation to continue uninterrupted.
After the first action (adding a student) is completed, the aspect re-engages to print a success message and provide immediate feedback on the task's completion. This process improves the openness of our project by giving us immediate insights into how important activities are being carried out.

Our other views (admin, class, classroom, resource, schedule, subject, teacher, authentification and timetable) all have comparable implementations of this aspect-oriented paradigm.

### AOP for security

The deployment of strong security measures has received top priority in our efforts to improve the efficiency and safety of our learning management system. A noteworthy advancement in this is the incorporation of a strict security feature that reinforces user access and rights. This security feature, which is especially designed for teachers, allows authorized users to quickly add resources to their own schedules after logging in. Concurrently, malicious efforts encounter a strong resistance in the shape of an error status "401," guaranteeing a stringent control system over access to confidential features.

```
import aspectlib
from django.http import JsonResponse

@aspectlib.Aspect
def secure_resource_access(*args, **kwargs):
    view_function = args[0]

    request = next((arg for arg in args if getattr(arg, 'user', None)), None)

    if not request or not request.user.is_authenticated:
        print("Unauthorized access")
        return JsonResponse({'error': 'Unauthorized access'}, status=401)

    result = yield aspectlib.Proceed
    return result
```

This code defines the security aspect using aspect-oriented programming to enforce security measures for resource access in a Django web application. The aspect checks if the user making the request is authenticated, and if not, it responds with a 401 error indicating unauthorized access.

## Monitoring-Oriented Programming

To improve the scheduling functionality, our project has included monitors in addition to using aspect-oriented programming ideas [3]. Our system's specialized monitor is made to deal with the problem of courses overlapping while creating schedules. When this monitor notices any instances of course overlap, it acts pro-actively by sending out notifications instantly. Additionally, it makes use of dynamic schedule adjustment techniques, which make dispute resolution easier. The monitor employs adaptive rescheduling to make sure that courses that clash are intelligently changed within the schedule in accordance with

user preferences and established criteria. This harmonious combination of aspect-oriented programming and monitors strengthens our scheduling system's resilience and shows how a multipronged approach may effectively tackle challenging software engineering problems [3].

# Testing

The testing approach is distinguished by a careful blending of functional and non-functional testing techniques. This two-pronged strategy guarantees a comprehensive assessment of the software's functionalities and performance characteristics. Functional testing is used to examine features and components in detail and confirm that it complies with design standards and requirements. Simultaneously, non-functional testing is essential for evaluating the software's responsiveness, scalability, and dependability in many scenarios.

## Functional testing

We have tested all of the important features of the system as part of our thorough review, paying special attention to the login process. Examining the login mechanism was necessary because of the obvious differences in the permissions granted to users according to their assigned roles: teachers and students. Our research showed that users are granted additional rights, such the ability to easily add resources, when they login as teachers. This unique feature is closely related to our security system (AOP), which requires the user to be registered as a teacher in order to access some operations (like adding resources).

## Non-Functional testing

Within the context of non-functional testing, our investigation made use of the Locust tool [1], a modern load testing framework. This tool made it easier to conduct a thorough assessment of our system's scalability and performance in a variety of operating circumstances. The main goal was to determine whether the system could handle a large number of users in order to provide insight into important aspects like responsiveness, stability, and other non-functional characteristics. The Locust tool [1] was crucial in providing empirical insights necessary to validate the resilience of our project and ensure that it could continue to provide optimal performance and stability even in the face of increased user demand. This was achieved by submitting the system to a range of scenarios.

### Statistics

Throughout our non-functional testing, the simulation was carefully set up with the addition of a pre-selected group of aspirant users, as we can see in Figure 4. Their involvement was intended to simulate real-world situations in which they attempted to interact with our program.

The utilization of this experimental framework was crucial in gathering relevant statistical data that offered insightful information on the behavior and interactions of users in our application (Figure 5).

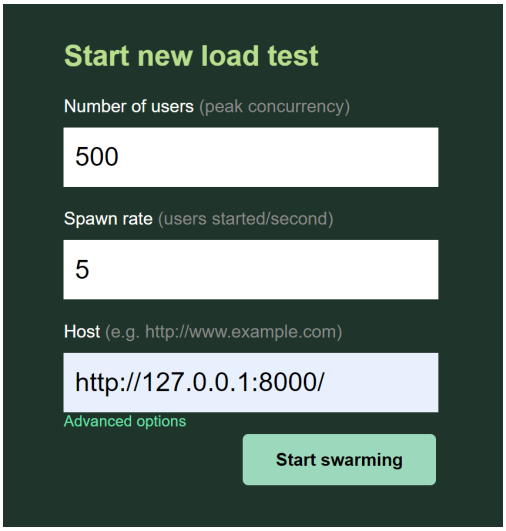


Figure 4: Setting data to obtain relevant statistics

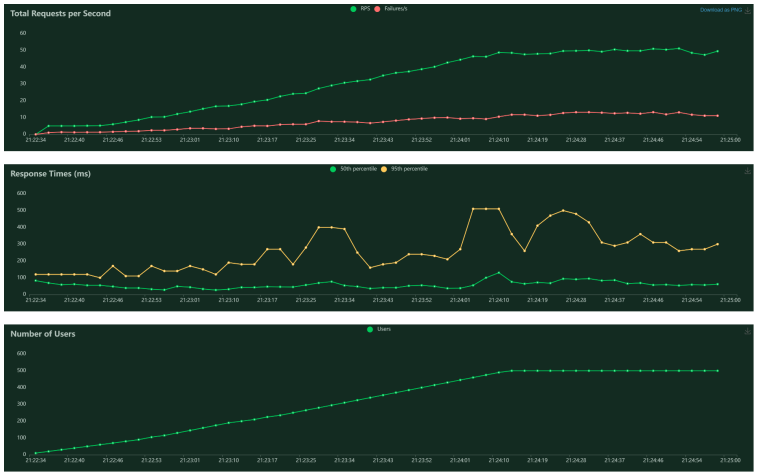


Figure 5: Statistics

In Figure 5, our observations reveal the acquisition of three distinct categories of statistical metrics. Firstly, the total requests per second exhibit a consistent upward trajectory, corresponding to the users endeavors to log in. Concurrently, the response time and the count of users demonstrate incremental increments at 5-second intervals. Notably, upon reaching a user count of 500, a discernible linearity manifests, signifying a plateau in login attempts. This inflection point suggests that users cease their login efforts, contributing to the stabilization of the observed parameters.

### Selenium

For our NFT, we also used Selenium, made especially for building browser-based regression automation suites and tests, to do extensive testing to guarantee the robustness and dependability of our application. The features of Selenium were quite helpful in verifying

that our login method worked as intended. We were able to replicate actual user interactions with the program by utilizing Selenium, which made sure that the login process was not only smooth but also resilient. This code sample, which highlights the thorough steps required to ensure a user experience and the general reliability of our program, demonstrates the painstaking testing procedure applied to our login feature.

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

email = "deliagrigorita@yahoo.com"
password = "1234"

urls = ['http://127.0.0.1:8000/auth/']

chrome_options = webdriver.ChromeOptions()

chrome_options.add_experimental_option('excludeSwitches', ['enable-logging'])
chrome_options.add_argument('--log-level=3')

driver = webdriver.Chrome(options=chrome_options)

wait = WebDriverWait(driver, 10)

try:
    for url in urls:
        driver.get(url)

        email_field = wait.until(EC.presence_of_element_located((By.ID, "id_email")))
        email_field.send_keys(email)

        password_field = driver.find_element(By.ID, "id_password")
        password_field.send_keys(password)

        login_button = driver.find_element(By.XPATH, "//button[text()='Login']")
        login_button.click()

    print("Before wait - Current URL:", driver.current_url)
    wait.until(EC.url_contains('http://127.0.0.1:8000'))
    print("After wait - Current URL:", driver.current_url)

    print("Login Successful")
```

```
except Exception as e:
    import traceback
    traceback.print_exc()

finally:
    driver.quit()
```

### Google Calendar

As a scheduling application, a primary external integration revolves around interfacing seamlessly with calendar applications. Predominantly, the integration with Google Calendar stands out as the most widely adopted. Consequently, we have established the foundational architecture to support this functionality, encompassing the creation of a class event and its subsequent synchronization with Google Calendar.
Each event(Figure 6) encapsulates pertinent class-related information, including the Class Name (course name), mandatory date and time details, and the designated Class Location (classroom name).

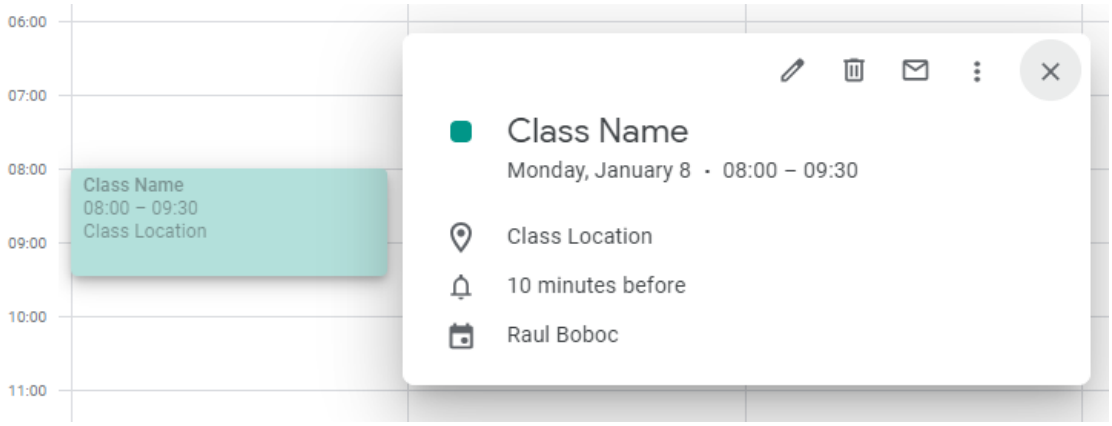


Figure 6: Google Calendar

As depicted in Figure 6, the core information is distinctly presented within the event square, alongside highlighting several advantages inherent to Google Calendar, such as notifications and calendrical features. While these attributes are configured as default values within the calendar system, users have the flexibility to tailor them to their preferences within the Google Calendar client. This customization empowers individuals, whether students or educators, to align the tool with their distinct organizational preferences. For instance, users may opt to assign events to specific calendars, facilitating streamlined filtering and sharing of class-related information.

# Future work

The Timetable Generator project has several opportunities to expand its capabilities and adapt to changing requirements. The following are some possible areas for further development:
1. **Machine Learning Algorithm Integration**: Investigate using machine learning techniques to examine past scheduling information, user preferences, and patterns of resource usage. This might result in recommendations for scheduling that are more adaptable and intelligent, increasing the overall efficiency of creating timetables.
2. **Enhanced Experience and User Interface**: Make constant improvements to the user interface to increase its intuitiveness and usability. To improve the user experience overall, think about adding features like real-time collaboration, customizable dashboards, and drag-and-drop capability.
3. **Mobile Application Development**: Create a specific mobile interface (our application is responsive) to provide consumers with the freedom to see and control their schedules when they're on the road. This can increase teachers and students accessibility and convenience.
4. **Sophisticated Techniques for Allocating Resources**: Examine cutting-edge techniques for allocating resources in order to maximize the use of classrooms and other resources. To identify the most effective allocation patterns, this may include applying optimization methods.
5.**Metrics and User Input**: Include a feedback system so that users can provide comments on the timetables that have been created. Examine user comments and use trends to find areas that need work, then adjust the scheduling algorithms appropriately.
6. **Optimizing Scalability and Performance**: To make sure the system can manage an increasing number of users, classes, and resources, perform further scalability testing. Enhance the application's performance to ensure that it remains responsive as the dataset grows.
7. **Improvements in Security**: To protect important scheduling data, increase security measures. To improve data safety, incorporate more authentication methods and encryption procedures.

The schedule Generator can develop into a more complex and all-encompassing solution for effective schedule administration in educational institutions by solving these issues in subsequent iterations.

# Conclusions

In conclusion, the Timetable Generator project is a noteworthy development in the realm of scheduling and project planning. The system provides a dynamic and responsive user experience by utilizing cutting-edge technology and programming paradigms like Monitoring-Oriented Programming (MOP) and Aspect-Oriented Programming (AOP). The application's overall safety is improved by the incorporation of security measures using AOP, which guarantees controlled access to critical functions.
The project's design, which combines front-end and back-end technologies, demonstrates

an organized and effective system. The integration of Business Process Modeling Notation (BPMN) enhances the system's comprehensibility and uniformity, guaranteeing shared comprehension across heterogeneous stakeholders.

Through examples, the Aspect-Oriented Programming technique improves the codebase's modularity and maintainability while simultaneously offering real-time feedback on crucial processes. The project's dedication to reliable and safe software development is demonstrated by the inclusion of security-related features.

Additionally, the use of Monitoring-Oriented Programming presents a proactive approach to conflict resolution by addressing the issue of scheduling overlap for courses. The project's testing phase, which includes both functional and non-functional components, guarantees a comprehensive assessment of the features, scalability, and performance of the system in many scenarios.

Non-functional testing yielded statistical insights that show the project can support a large user load and yet remain responsive. The Locust tool is useful for modeling real-world situations and yields useful information for evaluating the resilience of the system.

In the long run, the initiative creates the foundation for improvements in the creation of timetables for educational establishments. The schedule production process may be further optimized by the incorporation of sophisticated approaches like genetic algorithms and constraint fulfillment, as well as further research of AOP and MOP paradigms.

In summary, the Timetable Generator project not only accelerates the process of project scheduling but also provides unique programming paradigms to boost security, modularity, and responsiveness. With its rigorous testing and future-oriented approach, the project lays a solid basis for enhancing the efficiency and precision of timetable development in educational contexts.

# References


[1] *Locust documentation.*

[2] Antariksha Bhaduri. University time table scheduling using genetic artificial immune network. In *2009 International Conference on Advances in Recent Technologies in Communication and Computing*, pages 289–292. IEEE, 2009.

[3] Feng Chen and Grigore Roşu. Mop: an efficient and generic runtime verification framework. In *Proceedings of the 22nd annual ACM SIGPLAN conference on Object-oriented programming systems, languages and applications*, pages 569–588, 2007.

[4] Anuja Chowdhary, Priyanka Kakde, Shruti Dhoke, Sonali Ingle, Rupal Rushiya, and Dinesh Gawande. Timetable generation system. *International Journal of Computer Science and Mobile Computing*, 3(2):410–414, 2014.

[5] Abdallah Elkhyari, Christelle Guéret, and Narendra Jussien. Solving dynamic timetabling problems as dynamic resource constrained project scheduling problems using new constraint programming tools. In *Practice and Theory of Automated Timetabling IV*, pages 39–59. Springer-Verlag, 2003.

[6] Jeff Forcier, Paul Bissex, and Wesley J Chun. *Python web development with Django.* Addison-Wesley Professional, 2008.

[7] Martin Matusiak. Strategies for aspect oriented programming in python, 2009.

[8] Dipesh Mittal, Hiral Doshi, Mohammed Sunasra, and Renuka Nagpure. Automatic timetable generation using genetic algorithm. *International Journal of Advanced Research in Computer and Communication Engineering*, 4(2):245–248, 2015.

[9] Martin Owen and Jog Raj. Bpmn and business process management. *Introduction to the new business process modeling standard*, pages 1–27, 2003.

[10] Mei Rue. Computer and automation engineering. *The 2nd International Conference*, 4(2), 2010.

[11] Dipti Srinivasan, Tian Hou Seow, and Jian Xin Xu. Automated time table generation using multiple context reasoning for university modules. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*, volume 2, pages 1751–1756. IEEE, 2002.