MINI SENIOR PROJECT REPORT IN BUSINESS ANALYTICS

As a partial fulfillment of the degree of

Bachelor in Business Administration

Realized by

Majd Hamdi

Course Registration System for Tunis Business School

Academic Advisor

Dr. Sonia Rebai

Academic Year: 2024/2025

Approval

ACADEMIC ADVISOR

Name **Signature** **Date**

ACADEMIC EVALUATOR

Name **Signature** **Date**

Dedication

I hereby confirm that I am the sole author of this report. Any form of support or contribution I received during its preparation has been properly acknowledged and disclosed within the document. All sources of data, ideas, or wording, whether quoted or paraphrased have been appropriately cited. This report fully complies with the referencing and citation standards required at TBS and upholds the academic integrity policies outlined in the TBS Honor Code.

Additionally, no part of this work has been previously submitted for any other degree or academic qualification at this or any other institution.

Majd Hamdi
28/06/2025

Dedication

I dedicate my work to all those who contributed to it. Be they family, friends, and instructors. Thank you.

Acknowledgement

I express my most genuine gratitude towards certain individuals who contributed greatly in my academic journey. Their support and guidance have been fundamental in helping me reach this milestone.

Dr. Sonia Rebai - Thank you for your patience, your mentorship, and your invaluable input at every stage. Your insightful feedback and constant encouragement shaped this work into what it became.

Ms. Kaouther Ben Abdallah - Thank you for your time, support, and advice. Your suggestions and participation contributed greatly to the refinement and quality of this project.

To the esteemed jury members - Thank you for your time, your evaluation, and your role in ensuring the academic rigor of this work. Your presence and feedback are greatly appreciated.

Thank you all for your inputs.

Abstract

The primary focus of this project is the development of a web application for the TBS course registration system, designed to comply with the official rules and restrictions described in the school handbook. The system covers key processes such as course enrollment, schedule generation, and specialty selection. It begins by offering students an interactive interface that only presents the options they are eligible for, based on their individual academic profiles. Furthermore, to improve student decision making, all relevant information and guidance is included to help them make more informed decisions. A linear optimization model was also developed for scheduling enrolled courses to generate the best possible solution based on both system constraints and individual preferences. Finally, an admin interface was created to handle all administrative tasks related to the project, namely a settings panel to modify both global system parameters and individual configurations, multiple panels for handling students' requests and a dashboard displaying key statistics for the current and past semesters, all to ensure that the system remains adaptable to future changes.

Keywords: Course Registration System, Web Application Optimization Model, Decision Support

Contents

List of Figures

List of Tables

Abbreviations

TBS	Tunis Business School
ACCT	Accounting
BA	Business Administration
FIN	Finance
IT	Information Technology
MRK	Marketing
IBE	International Business and Economics
GPA	Grade Point Average
RDBMS	Relational Database Management System
SQL	Structured Query Language
HTTP	HyperText Transfer Protocol
REST	Representational State Transfer
API	Application Programming Interface
IACBE	International Accreditation Council for Business Education
SMTP	Simple Mail Transfer Protocol

Executive Summary

The course registration process at Tunis Business School (TBS) is often confusing for students, making it difficult for them to understand their options and plan effectively. On the administrative side, manual handling increases the risk of mistakes and inconsistencies. To address these challenges, a SPA was developed with two main goals: to support students in making informed decisions through better guidance and clarity, and to reduce administrative workload and minimize human error. The development began by identifying the most common questions students ask, the issues they encounter, and the mistakes they frequently make during course registration, along with a thorough understanding of the official rules and constraints outlined in the handbook. Any crucial missing information was addressed by contacting the responsible administrative staff to fill those gaps.Based on this groundwork, the essential components required for a comprehensive system were identified: course enrollment, schedule generation, and specialty selection. An interactive interface was designed for students, presenting all relevant information and restricting choices to only those for which they are eligible each semester. This ensures that students are well-informed, understand their current academic standing, and can plan ahead effectively. A scheduling model was then developed to generate a feasible timetable that aims to accommodate student preferences as much as possible, while also ensuring compliance with all academic constraints and requirements, and ultimately validating and authorizing the student's course selection. On the administrative side of the system, dedicated interfaces were built to streamline management tasks such as providing a settings panel where global system parameters and individual student configurations can be adjusted, or panels used to handle different kinds of requests. Finally, a dashboard was designed to present key statistics for the current semester, providing a clear overview of activity and academic trends.

General Introduction

Over the past decade, higher education has undergone a significant technological shift, strategically embracing digital tools to streamline administrative workflows and enhance the student experience. Leading institutions are moving beyond traditional, paper-based processes, such as manual registration and siloed student services, to fully integrated, technology-driven systems [1]. This shift is further supported by the adoption of workflow automation tools designed specifically for academic institutions [2].

In particular, course registration systems have become a focal point of this digital transformation. Recent surveys indicate that students frequently express frustration with opaque registration interfaces and hidden prerequisite requirements, underscoring the need for more transparent and guided enrollment platforms [3]. At the same time, educational technology providers highlight the benefits of centralized, one-stop service portals—which integrate registration, advising, billing, and academic alerts—resulting in more efficient processes and higher user satisfaction [4].

Yet at TBS, the course registration process remains a frustrating hurdle each semester. Students receive most of their updates on course enrollment periods, specialty selection, and deadlines through social media or word of mouth, which remains their main source of information. They are then required to fill out online forms which can result in inaccurate submissions and inconsistent processing of requests. On top of that, they must juggle prerequisite rules, and resolve schedule conflicts through basic trial and error, often discovering eligibility issues only after attempting to enroll. Meanwhile, administrative staff answer the same questions hundreds of times at the start of every semester and spend hours manually verifying requests and resolving errors, a process that slows down registration and leaves little room for proactive support.

Recognizing these issues, this project set out to build a web-based course registration system tailored to TBS's defined policies and regulation. At its core, the web app is a Single Page Application that guides students step by step, essentially transforming the lengthy, static content of the academic handbook into a more dynamic and digestible format that makes understanding requirements and planning decisions more intuitive. Students are presented with the courses and specialties for which they qualify and receive relevant advice on how their choices may affect their future academic situation. On the other hand, a linear optimization model checks session availability, avoids clashes, and respects individual preferences, producing a feasible timetable with a maximum satisfaction score, if one exists. On the administrative side, staff have full control over system settings through dedicated interfaces for managing requests and overseeing all registration components. Finally, a real-time dashboard displays key semester metrics, offering quick insight into trends and activity.

This report is divided into three main chapters. Chapter 1 introduces the project by framing

the academic problem and presenting the vision that drives the development. It also explores the hosting institution, details the shortcomings of the existing system, and outlines the proposed solution along with its academic foundation. Chapter 2 delves into the design phase, covering system requirements, flow modeling, and the scheduling optimization formulation. It also explains the development methodology adopted and the data collection and pre-processing efforts. Chapter 3 focuses on implementation, describing the technologies used, the developed features for both student and admin interfaces, and how academic logic was enforced.

Chapter 1

Problem Framing and Vision-Driven Development

This chapter provides an overview of the project, including its context, objectives, and the environment in which it was developed. It outlines the problem being addressed, the tasks carried out during development, and a critique of the existing system. The chapter also introduces the proposed solution, supported by relevant technical background and methodology.

1.1 Project Context

This mini senior project falls under the course code **BA 498** and serves as a compulsory end-of-studies requirement at TBS. It was conducted over a four-month period, from March 1st to June 30th, as the final step toward fulfilling the requirements for obtaining a Bachelor's degree in Business Administration, with a major in Business Analytics and a minor in Information Technology. The subject was chosen with the intention of developing a solution that offers practical value within my academic environment. Based on personal experience, a course registration web app seemed the perfect choice.

1.2 Host Organization

Tunis Business School (TBS), a distinguished faculty under the University of Tunis, opened its doors on October 25, 2010, under decree n°2755. It holds a unique place in the national academic landscape as the first public business school in Tunisia to teach entirely in English and to implement the American higher education model.

TBS was founded in alignment with national development goals, aiming to strengthen Tunisia's economic competitiveness, encourage innovation, and support offshore business initiatives. its mission is built upon three core pillars: Education, Research, and Community Engagement.

Accreditation

TBS has successfully met the IACBE Accreditation Principles for its business programs by undergoing a thorough self-assessment and an external peer-review process.

Education

TBS offers a forward-thinking academic environment that nurtures analytical reasoning, leadership capabilities, and international awareness. Its teaching philosophy is rooted in globally recognized methods and includes:
Case-Based Learning: Students explore authentic business problems, assess strategic options, and develop practical solutions.
Simulated Business Scenarios: Through role-playing and simulations, students face realistic challenges and sharpen their decision-making skills.
Practical Skills Development: Emphasis on learning by doing, with workshops, team projects, and real-world tasks.
Hands-On Experience: Integration of classroom knowledge with field internships and applied learning opportunities.

Research

TBS fosters a dynamic academic research culture, where faculty and students alike contribute to solving modern economic and business issues. Interdisciplinary research, innovation, and publication are actively promoted to enrich both academic and professional communities.

Community Engagement

Beyond the classroom, TBS builds strong ties with industries, policymakers, and civil society. Whether through tailored training programs, consultancy services, or collaborative projects, the school extends its impact to support social and economic advancement in Tunisia and beyond.

Vision and Mission

TBS aims to be a globally respected institution recognized for excellence in education and research. By fostering innovation, advancing knowledge, and maintaining strong partnerships with the business community, TBS is dedicated to preparing future leaders, experts, and decision-makers who will contribute meaningfully to both the economy and society.

1.3 Project Overview

1.3.1 Problem Statement

Course registration is one of the most critical academic processes in a student's university experience. It directly affects academic progression, specialization choices, and ultimately, graduation timeline. At TBS, this process has remained largely manual, decentralized, and unclear, leading to frequent confusion and inefficiencies for both students and administrative staff.

Each semester, students rely primarily on unofficial channels such as social media groups, forwarded messages, or word-of-mouth to learn about registration deadlines, specialization declarations, or procedural changes. These fragmented sources of information often result in missed deadlines or misinterpretations of academic rules. In parallel, the official academic handbook,

although comprehensive, is long, rigidly formatted, and difficult to consult quickly. This creates an environment where students either over-rely on staff for clarification or make incorrect assumptions, which can delay their academic plans or force unnecessary changes later on.

On the administrative side, staff are burdened with manually handling hundreds of registration forms, messages, and emails at the start of each semester. They repeatedly answer the same questions and verify eligibility on a case-by-case basis. This process is not only time-consuming but also prone to inconsistencies and errors.

Another important issue is the mismatch between the official system described in the handbook and the way registration is handled in practice. While the rules are mostly well-defined on paper, real-world decisions often involve informal exceptions and added flexibility, especially in areas like major selection or probation handling. This gap creates uncertainty for both students and system designers.

Some constraints, although officially required, are also difficult to apply without automation. For example, ensuring that students only enroll in courses whose sessions can be scheduled without conflicts is really challenging to manage manually. Without support tools, students may not be able to construct an optimal timetable since they rely on a trial and error process to navigate hundreds of possible combinations.

To address this, the use of an optimization model becomes essential. It can automatically generate feasible, conflict-free schedules that respect both institutional constraints and student preferences, reducing back-and-forth corrections and improving overall efficiency.

From a long-term perspective, the lack of a structured system also means that no institutional memory is preserved. Each semester repeats the same set of problems, without any way to analyze trends, measure student flow through programs, or anticipate common bottlenecks in the registration process. This limits the ability of TBS to optimize academic planning or make data-driven decisions.

All of these challenges point to the urgent need for a centralized, intelligent, and user-friendly platform that streamlines course registration, guides students through their academic choices, and provides administrators with tools to manage and oversee the process efficiently. Without such a system, both student satisfaction and institutional effectiveness will remain hindered.

1.3.2 Application Objectives

At the heart of this application is the goal of making academic life easier and more intuitive for both students and administrators. The objectives were shaped by real needs within the academic environment, focusing on clarity, efficiency, and better decision-making throughout the course registration process. The system aims to facilitate and streamline the course registration process for students while supporting essential academic planning functions throughout their educational journey. Additionally, it provides students with reliable guidance and tools for informed academic decision-making, and equips administrators with capabilities to manage registration workflows and adjust key academic settings as needed.

1.3.3 System Understanding

A thorough understanding of the course registration process was a necessary first step before beginning development. This involved analyzing the official academic handbook to extract all relevant rules, including but not limited to prerequisite structures, credit limits, and specialization criteria.

A focal point in this phase was identifying the differences between the theoretical system outlined in the official handbook and the version practiced in reality. This distinction was essential in determining the development direction and deciding which version to base the system on. The main discrepancies were the greater leniency often applied by the administration in practice, particularly in areas like major selection, probation handling, and the flexibility to allow course enrollment even when session scheduling is unfeasible. Ultimately, a balanced decision was made to build a the system adheres to the official rules as a foundation, with the hope that these standards will become the norm for future generations, while also offering configurable parameters and settings to accommodate administrative adjustments and ensure the system remains adaptable over time.

1.3.4 Study and Critique of The Existing

To build an effective solution, a thorough understanding of the existing system's deficiencies is essential. The current course registration process exhibits several critical weaknesses that impede efficient academic planning and create unnecessary burdens for both students and administrative staff.

Foremost among these weaknesses is the significant fragmentation of the registration process, which relies on disparate tools including Google Forms, Excel spreadsheets, and informal communication channels. This decentralized approach creates numerous opportunities for errors, miscommunication, and data inconsistency. The absence of a unified platform where students can comprehensively view, select, and confirm their courses within a rule-aware environment fundamentally undermines the registration experience.

Stemming directly from this fragmentation is a pervasive issue: the widespread lack of student understanding regarding registration procedures and academic requirements. Each semester, the grades department receives approximately 1,100 inquiries related to registration processes. The recurring nature of these questions indicates systemic gaps in guidance rather than isolated confusion. The most frequent inquiries include:

– "How does adding courses work?"

– "Which groups should I enroll in for my courses?"

– "Which courses do I have to add to be able to pick major X?"

These questions reveal that students struggle to comprehend how their current choices affect future eligibility, specialization options, and graduation timelines. Without systematic guidance, students make uninformed decisions whose consequences only become apparent in subsequent semesters, often when corrective options are limited or unavailable.

Compounding these information gaps, the current system also lacks tools for constructing conflict-free timetables. While the absence of enforced constraints against overlapping sessions might appear to offer flexibility, it creates deeper academic problems. Without assistance in

building personalized, conflict-free schedules, many students inadvertently register for courses with overlapping sessions. This scheduling conflict forces students to miss classes regularly, negatively impacting their academic performance. The consequences cascade into lower grades, increased course failures, and potential threats to academic standing.

Further exacerbating these challenges is the manual processing of registration requests by administrative staff, which introduces significant vulnerability to human error. Despite diligent efforts to interpret and apply academic policies consistently, the case-by-case approach inevitably leads to inconsistencies. Examples include students being granted higher-level status without fulfilling prerequisite credit requirements or being erroneously excluded from necessary courses. These administrative inconsistencies undermine the integrity of academic progression and create inequities in student experiences.

These systemic weaknesses collectively diminish the effectiveness of the registration process, create unnecessary administrative burden, and potentially compromise student academic success. Addressing these fundamental issues requires a comprehensive redesign rather than incremental adjustments to the existing system.

1.3.5 Proposed Solution

In light of the shortcomings of the existing system, the proposed solution addresses the current challenges through the development of a comprehensive web-based course registration system. This centralized platform will streamline the entire registration process while providing both students and administrators with intuitive interfaces tailored to their specific needs.

The system architecture will be built around a centralized web application that unifies all aspects of course registration into a single platform. This integration eliminates the fragmentation of the current process by consolidating course selection, schedule generation, and specialization selection into one cohesive workflow. The system will intelligently filter and present only eligible options based on their academic history and program requirements, thereby preventing errors at the source and reducing administrative burden.

For students, the platform will feature an interactive, guided interface that transforms the static academic handbook into an engaging, dynamic experience. This interface will walk them through each step of the registration process, highlighting course eligibility based on their academic record, providing immediate feedback on invalid selections, and ensuring that all choices align with institutional rules and program constraints.

At the core of the scheduling functionality, the system will implement a well defined linear optimization model that automatically generates conflict-free timetables. This model will balance both hard constraints set by the academic policies and student preferences for time slots and instructors to create optimal schedules. This automation eliminates the manual trial-and-error approach currently used by students and eliminates scheduling conflicts.

For administrative staff, a dedicated interface will provide comprehensive control over the registration process. This interface will include configurable system settings to adjust global parameters and the ability to implement individual exceptions when necessary. The administrative dashboard will deliver real-time analytics on registration progress, course statistics, and other key metrics. These insights will support data-driven decision-making for academic planning, resource

allocation, and curriculum development in the future.

The implementation will follow an iterative phased approach, beginning with core functionality and gradually incorporating advanced features. This methodology ensures that the most critical issues are addressed first while allowing for iterative refinement based on subsequent feedback. Throughout development, the system will be designed with scalability in mind to accommodate future enrollment growth and potential expansion of academic offerings.

1.4 Background

1.4.1 Linear Optimization in Academic Scheduling of Courses

Once a student has selected their courses, the next challenge is scheduling them into a conflict-free weekly timetable. This is a well-known NP-hard problem due to numerous constraints, such as avoiding time overlaps for students and instructors, satisfying room capacities, and respecting professor preferences [5]. Traditionally solved manually by administrators, the problem has been increasingly addressed through algorithmic solutions. Among them, constraint satisfaction programming (CSP) has proven effective, with constraint logic programming (CLP) approaches modeling institutional rules declaratively and yielding feasible timetables where manual methods struggle [6].

1.4.2 Score-Based Objective Function

A score-based objective function is a mathematical expression used to evaluate and compare different possible solutions by assigning them a numerical score based on how well they satisfy predefined criteria. Rather than focusing solely on minimizing cost or maximizing efficiency, this type of objective function aggregates multiple quality-related factors into a unified score. Each element of the solution, such as time slot assignments, preference fulfillment, or constraint satisfaction, is evaluated and weighted according to its importance. The overall score reflects the desirability of a solution, with higher scores typically indicating better outcomes. This approach is particularly effective in problems involving soft constraints, where certain preferences can be violated but with penalties. By using a score-based objective, the optimization model can guide the search toward solutions that offer the best balance between competing priorities. The flexibility of this method also allows for easy customization by adjusting the weights or components of the score function to align with institutional goals or user preferences.

In this approach, each possible choice or assignment is associated with a score value which can be derived from user preferences, performance metrics, costs, benefits, or other relevant factors. These scores are then aggregated into an objective function.

1.4.3 Rule-Based Prerequisite Checking and Curriculum Modeling

Rule-based prerequisite checking and curriculum modeling refer to the process of using clearly defined logical rules to enforce academic requirements and validate course eligibility within a structured curriculum. In this approach, prerequisite relationships, co-requisite dependencies, and program-specific progression rules are encoded as conditional statements. When a student selects a course, the system evaluates these rules against the student's academic history to determine

whether enrollment is permitted. This ensures that students follow the intended learning sequence and meet academic standards. Curriculum modeling complements this by representing the full structure of a degree plan, including core requirements, elective options, credit distributions, and allowable substitutions. Together, these rule-based mechanisms support automated validation, reduce administrative errors, and provide transparency to students and advisors. The approach is scalable and adaptable to different academic programs, allowing for consistent enforcement of policies while remaining flexible enough to accommodate curriculum changes.

1.4.4 Decision Support Systems

Decision Support Systems (DSS) in education are interactive software tools designed to assist students, advisors, and administrators in making informed academic decisions. These systems process large volumes of academic data to provide recommendations, detect potential issues, and support planning activities. In the context of course registration, a DSS can help students evaluate their course options based on prerequisites, scheduling conflicts, credit requirements, and personal preferences. For administrators, DSS tools can highlight enrollment trends, identify under- or over-capacity courses, and streamline academic policy enforcement [7]. By combining data retrieval, logical validation, and user-specific feedback, decision support systems enhance the overall efficiency, transparency, and accuracy of academic planning processes. Their integration into registration platforms ensures that users are guided toward valid, optimized, and goal-aligned academic pathways [8]. .

1.4.5 Error Prevention and Recovery in Registration Systems

Academic registration systems must be designed to minimize the risk of user errors and ensure that any mistakes can be detected, communicated, and corrected efficiently. Errors in course selection, exceeding credit limits, ignoring prerequisites, or enrolling in overlapping sessions can significantly impact a student's academic progress. A robust system should incorporate real-time validation mechanisms that immediately alert users when actions violate institutional policies or scheduling constraints. Furthermore, recovery features such as undo options, confirmation steps, and administrative override capabilities are essential to maintaining the system's reliability. By proactively preventing invalid actions and supporting easy correction of mistakes, such systems reduce the burden on academic staff and enhance the overall user experience. Error prevention and recovery are therefore key components of a dependable and student-centered course registration platform.

1.5 Conclusion

This chapter outlined the main issues with the current registration process and the motivation behind developing a smarter solution. It introduced a system that combines automation and academic logic to improve efficiency and clarity. The goal is to make registration more reliable for both students and administrators.

Chapter 2

Conception of the Web Application

In this chapter, the methodology and logic followed will be presented in depth to define the techniques and approach implemented in this project to develop all components of the web application.

2.1 Determining Needs

2.1.1 Actors Identification

The first step is to specify for whom this web application is made. In order to understand and map out our expected requirements later on. In this case, we define two main actors:

- **The administrative staff:** responsible for managing the platform's configuration and ensuring that academic and registration-related settings are properly maintained. They are the ones who will interact with the application frequently.
- **The students:** who interact with the platform to access its core functionalities according to the rules and constraints in place. The student will rely on this application the moment he enrolls in TBS all the way to graduation. Their use for this application generally will not exceed a few times per semester.

2.1.2 Functional Requirements

To meet the needs of the actors, the system must offer the following core functionalities that serve as the foundation upon which the entire application is built and the practical value is delivered.

All users, regardless of role, must be able to securely manage their account and identity within the system:

- Authentication using email and password.
- Editing of certain personal profile information.
- Resetting password.
- Account Creation.

Students must be empowered with tools that allow them to engage with the academic process and manage their academic path:

- Course selection during registration periods.
- Access to enrollment history and academic statistics.
- Ability to send requests within the rules set in place.
- Selection of specializations when eligible.
- Generation of personalized schedules based on preferences and enforced system constraints.

Administrators must have control over academic operations and oversight of student activity through specialized interfaces:

- Scheduling, initiation, and closure of academic periods (course registration, specialty selection, semester transitions, makeup session).
- The ability to access a dashboard for to view key summaries and statistics.
- Handling of various types of student requests.
- Manipulation of parameters at both global and individual levels.
- Access to student information.

2.1.3 Non-Functional Requirements

In addition to core functionalities, the system must also satisfy several non-functional requirements that ensure its reliability, usability, and overall performance.

- Easy-to-use interface: The system should have a clear, user-friendly interface that administrators and students can use with little assistance or training.
- Quick website loading and response time: All supported features should have low latency, and the application should respond quickly to user input.
- Browser compatibility: The system needs to work and be accessible on a variety of browsers, including Chrome, Opera, Mozilla Firefox...
- Security: The web app needs to protect user data from common online threats and properly authenticate users.

2.2 Registration System Flow

After determining our users' expectations from this application, it is also imperative to understand and define the general flow and the components of the existing course registration system. The purpose of this step is to determine the best fitting methodology to adopt during development. Since the policies and rules are pre-established by the institution, the challenge lies in translating them into clear system logic and constraints that can be enforced automatically.

As illustrated in Figure 2.1, the course registration process can be broken down into a series of tasks carried out by either students or administrators.
From the **student**'s perspective, the process consists of the following key steps:

- **Course selection:** The student selects from eligible courses, based on their academic progress and program requirements.The selection of some courses could be automatic depending on the case.
- **Gets initial feedback:** Under certain conditions and based on the current course selection, the student will receive some insights on long-term implications (e.g. delaying key prerequisites for specialty selection), helping them avoid common mistakes.
- **Schedule validation:** A conflict-free schedule is generated and verified to ensure feasibility, if no feasible schedule is found, the student must change his current course selection.
- **Submission of choices:** The selected courses are submitted, with the ability to make changes during the registration period or, if permitted, after it with administrative approval.
- **Specialization selection:** Based on eligibility criteria and program structure, the student decide on their major-minor combination. This decision can often be revised later if the requirements allow.

On the **administrative** side, the responsibilities include the interventions in the form of different tasks such as initializing system parameters, setting up deadlines for specific actions, and handling various types of student requests.

It is apparent that the course registration have intertwined points between the student and the admin.

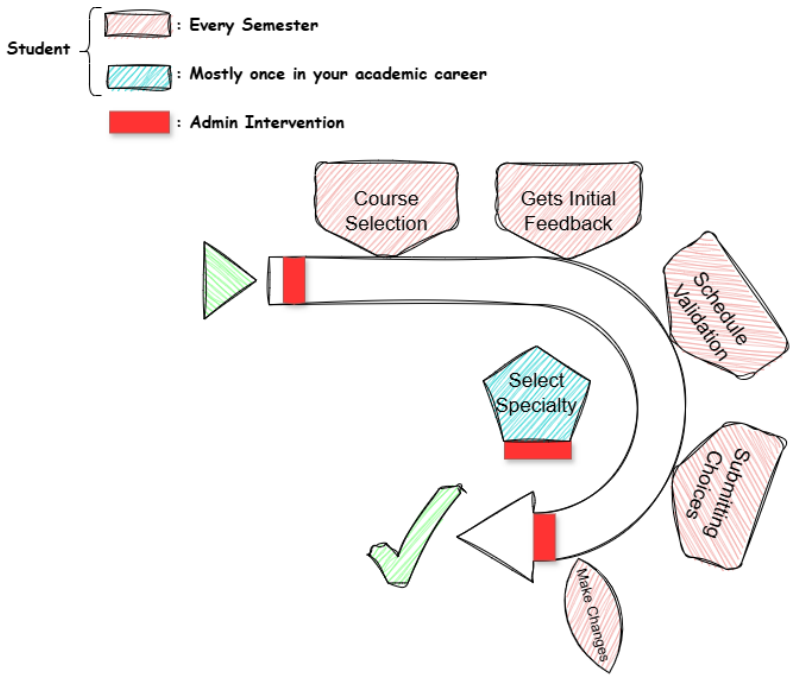


Figure 2.1: Course Registration Flow

2.3 Methodology Used

An Agile methodology, as shown in Figure 2.2, was adopted for the development of this project, with the workload divided into smaller, iterative cycles. These iterations were not defined arbitrarily; rather, they were closely aligned with the natural sequence of academic activities experienced by both students and administrators. This scenario-driven structure allowed the system to grow in sync with real-world institutional workflows, making each development cycle meaningful and contextually grounded.

After each iteration, rigorous testing and debugging were conducted to validate the accuracy and stability of the implemented features. The development road map was anchored around functional milestones that corresponded to distinct phases of the academic journey. It began with the foundational administrative task of defining the semester structure and opening course registration periods. This was followed by implementing course selection logic for first-year students in their initial semester. As the student life cycle progressed, subsequent cycles introduced more complex features such as prerequisite validation, academic probation handling, failed course logic, specialty selection, all the way to graduation and handling of special cases.

Each feature was carefully introduced in the order it would become relevant in a student's academic progression. This ensured that dependencies,whether administrative, logical, or rule-based,were respected and that every layer of functionality was constructed upon a coherent and realistic foundation.

In parallel with the web application's development, the theoretical formulation of the linear optimization model for schedule generation was also underway. However, the practical integration of this model into the system was deferred until the application development took shape. This ensured that the optimization logic could be seamlessly embedded into a stable, fully contextualized environment.

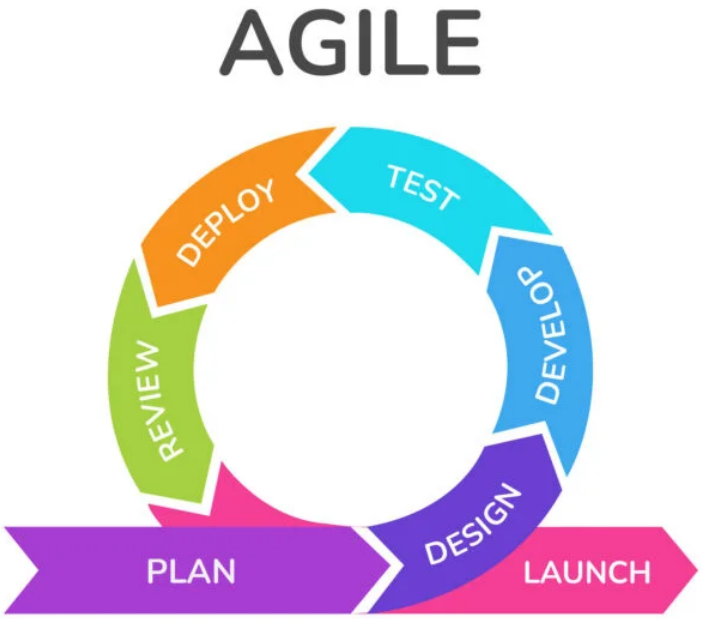


Figure 2.2: Agile Methodology Steps

2.4 Course Selection

Course selection is one of the core components of the system, making it essential to thoroughly understand the rules it is based on.

2.4.1 Course Selection Logic

By examining Figure 2.3, we can identify seven possible distinct categories of courses throughout a student's academic journey:

- **New Semester Courses**: These are the set of courses officially offered during the upcoming semester, as determined by the program's curriculum structure. They form the baseline of available options for students during course selection and vary depending on the student's academic level and major progression.
- **Illegible Courses**: This category includes any course for which the student has not yet fulfilled the prerequisite conditions. Such courses remain inaccessible and cannot be registered for until all required prerequisites are successfully completed. They represent future opportunities that are currently locked due to academic dependencies.
- **Skipped Courses**: These are mandatory courses for graduation that the student has not taken at the time they were originally offered. This may be due to one of two reasons: either the course was unavailable to the student because prerequisite conditions had not yet been met, or the student made the decision not to enroll in it during its scheduled offering. Regardless of the reason, these courses must be completed before graduation.
- **Retake Courses**: These are previously passed courses in which the student received a letter grade lower than C but higher than F. They do not fall under the category of failed courses, but they may be repeated to improve the student's cumulative GPA. (The default grade threshold used is C-, as defined by the student handbook, although this may be updated by administrative policy). The is no limit set on how many times a student can retake a course.
- **Extra Courses**: This category concerns junior- and senior-level courses only. It includes courses that are usually related to academic majors other than the student's own. Although they are rarely selected, students may enroll in extra courses as a strategic measure to increase their CGPA, particularly in cases where the minimum CGPA required for graduation has not been achieved.
- **Failed Courses**: These are courses in which the student has earned a letter grade of F on their first attempt. They must be retaken in the very next academic year. A student cannot graduate without passing all failed courses, making their timely retake and completion a strict academic requirement.
- **Completed Courses**: Includes courses in which the student earned a grade of C or higher. These courses are considered finalized.

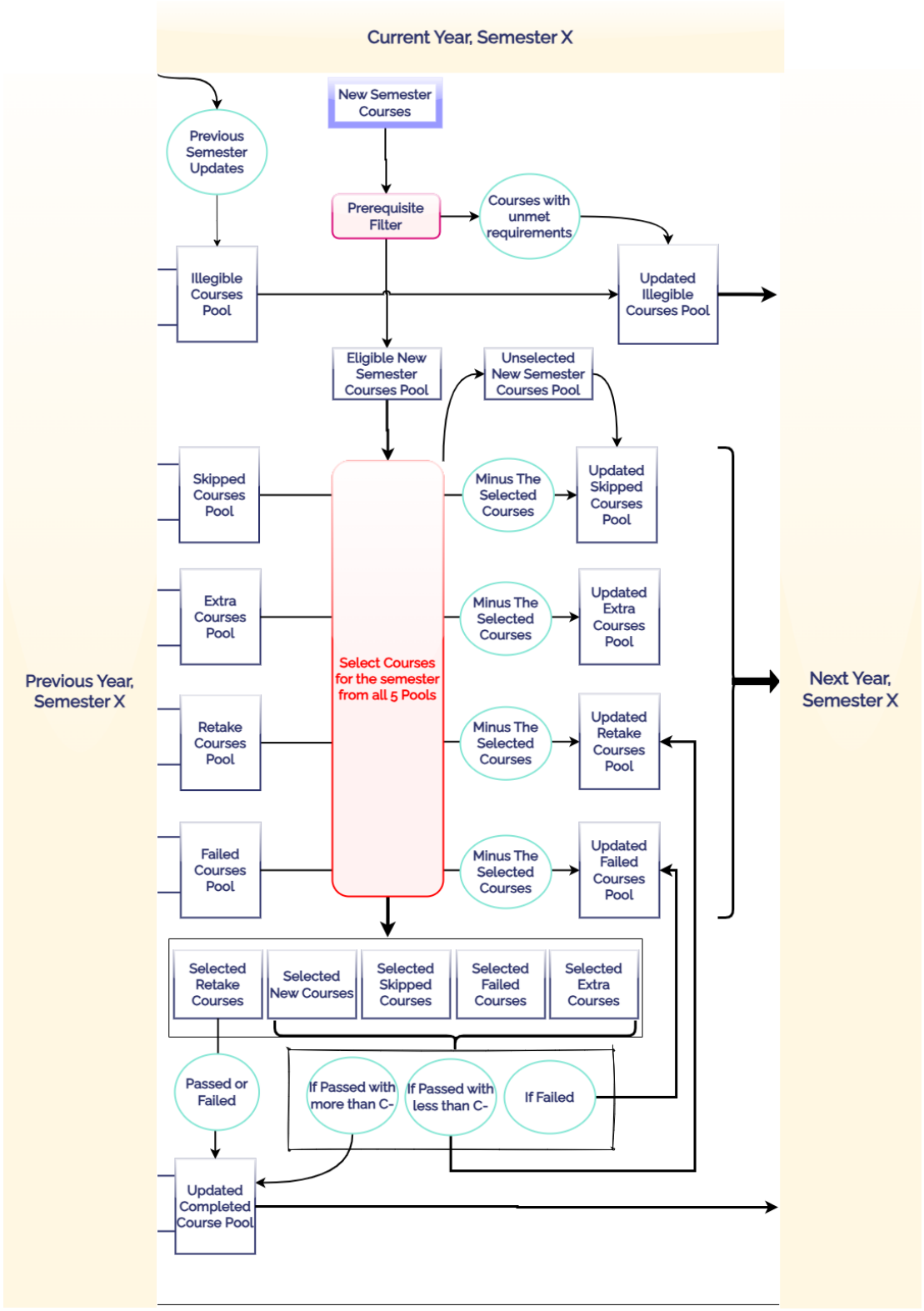


Figure 2.3: General Flow of Courses

2.4.2 Course Hierarchy

To properly guide the student, determining the official hierarchy of courses is essential to build the app with sound logic.As shown in Figure 2.4, there are two specific cases in which the course selection hierarchy changes:

- **Case 1 – Presence of Failed Courses:** If the student has failed courses at the start of a semester, those courses will be selected by default. After that, the student is free to choose any number of additional courses from any available category, provided that the total number of enrolled courses does not exceed the maximum course limit.
- **Case 2 – No Failed or Skipped Courses:** If the student has neither failed not skipped courses at the beginning of a semester, they are required to enroll in all courses of the current semester. After this requirement is fulfilled may the student select additional courses, with a limit of one or two extra courses to remain within the maximum allowed course load.

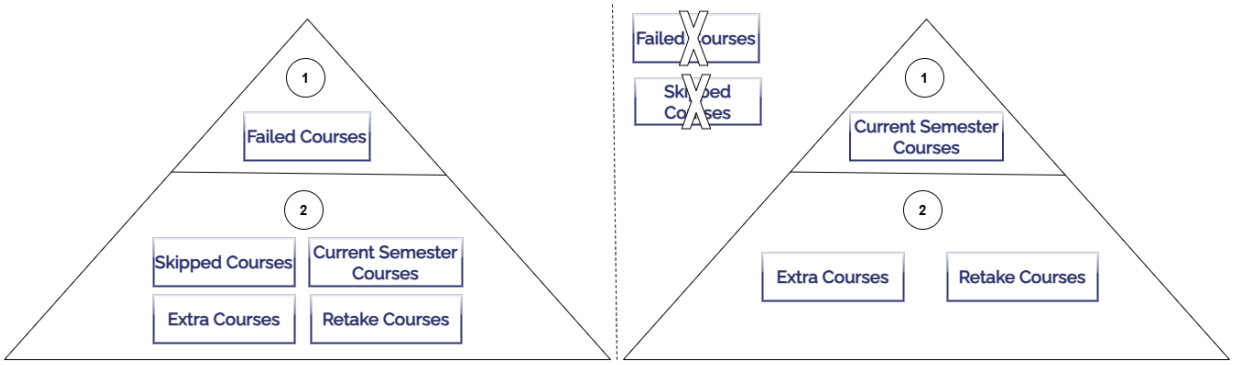


Figure 2.4: Hierarchy of course selection

2.4.3 Advisory Feedback

Since the student has total freedom in selecting their courses within the system constraints, the definition of the "best" decision is subjective and varies greatly from one student to another. For that reason, our aim is not to prescribe a single optimal path but rather to provide students with insight into the ramifications of their current selections. This includes highlighting potential issues such as unmet prerequisites, unfulfilled specialized GPA for majors, and falling into an almost certain probation status. Hence helping students make more informed and strategic choices and avoid making mistakes that they would regret. The Assessment Unit, as depicted in Figure 2.5 consists of a collection of code-based functions that are triggered under specific conditions to perform evaluations related to a student's academic status and provide suggestions in some instances.

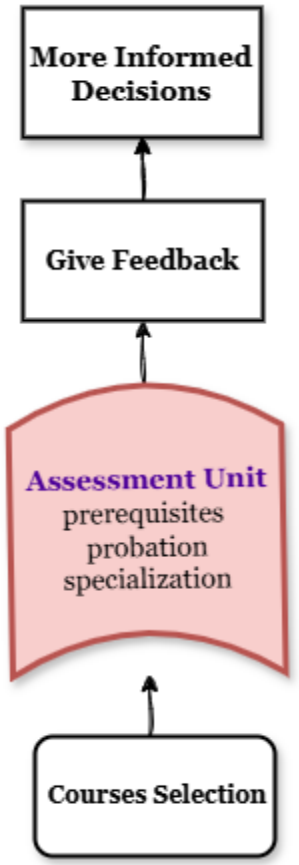


Figure 2.5: Assessment Unit Goal

2.5 Schedule Optimization Model

This model aims to automate and optimize the process of building a customized student schedule within the limits of what is allowed. It accounts for various academic rules and constraints while seeking to maximize a satisfaction score that reflects how well the generated schedule aligns with the student's stated preferences.The problem is formulated as a linear optimization model, allowing for efficient computation and scalable implementation.

2.5.1 Preparations

For the model to function correctly, preliminary work must be carried out to ensure that the official schedules published by the administration are properly cleaned, indexed, and inserted into the corresponding table in the database. This table is accessed each time the model is executed.

2.5.2 Model Flow

The scheduling model serves two key roles:

- **Validation of Course Selection:** The model is first executed once using default preferences and the student's current selection of courses to ensure it can yield at least one feasible solution. This step confirms the viability of the student's intended enrollment and if no feasible solution is found, changes to the set of selected courses must be made.

- **Finalization of Course Registration:** Once preferences are provided, along with the validated set of enrolled courses, the model generates feasible schedule options. The student can then review these options, select a preferred schedule, and complete the course registration process by confirming the group assignments for each course.

Both processes will be performed in different sections of the web application.

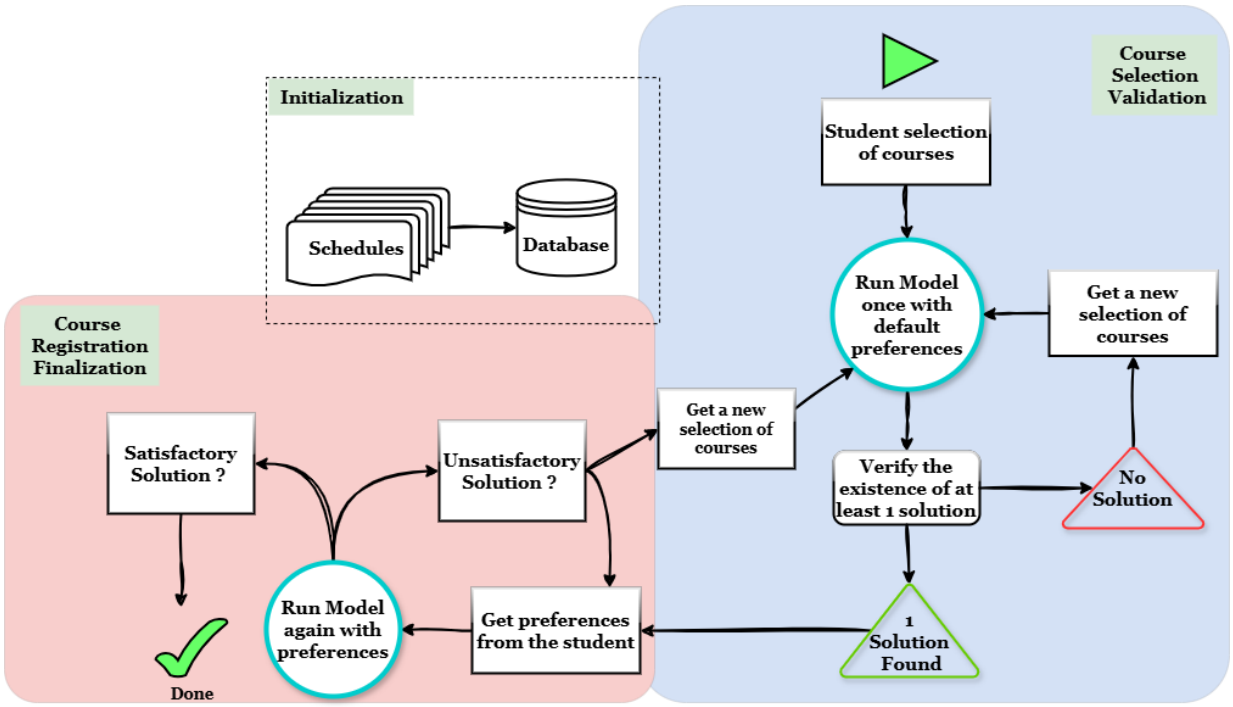


Figure 2.6: Generating Schedule Flow

2.5.3 Model Logic

The model is constructed using a set of hard constraints that ensure compliance with academic policies and regulations. While strictly enforcing these constraints, the model aims to maximize an objective function that serves as a satisfaction score, reflecting the degree to which personalized student preferences are met.

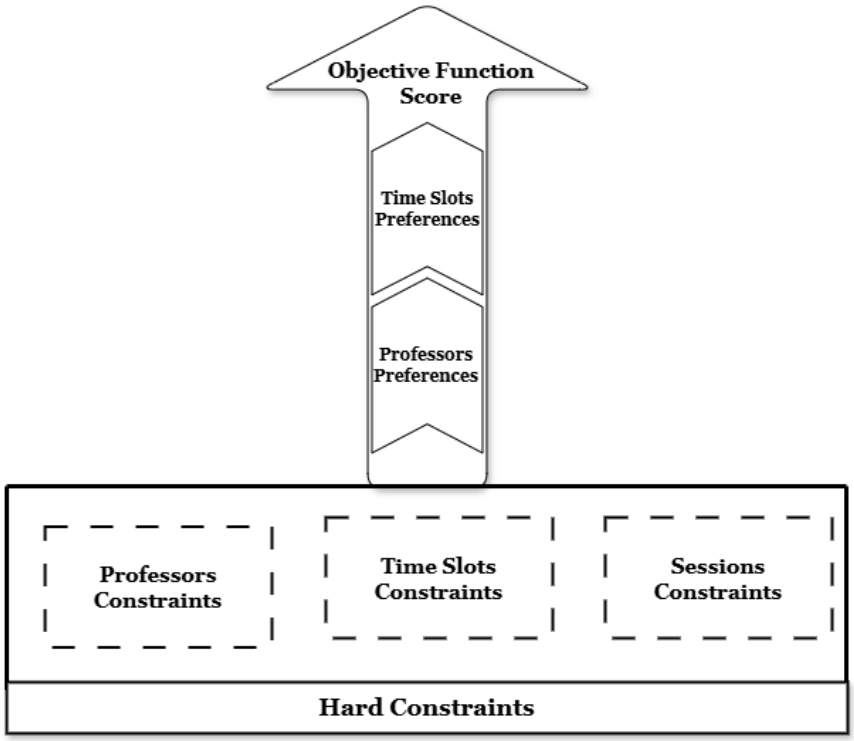


Figure 2.7: Schedule Model Flow

2.5.4 Model Development

We begin by defining:

Sets

- I: Set of a student's selection of courses
- I_1: Subset of I containing courses that have tutorials
- I_2: Subset of I containing courses that do not have tutorials
- Prof_i: Set of professors who teaches course $i \in I$
- $L_{i,k}$: Set of time slots for session number k, $k \in \{1, \ldots, K_i\}$ for course $i \in I$
- $TutProf_{i,q}$: Set of tutorial professors who can be assigned when professor q is selected to teach lecture sessions for course i
- $S_{i,k,p}$: Set of time slots where professor p teaches session number k, for course $i \in I$
- LectProf_i: Set of lecture professors who teaches course i
- FixSlot: Set of fixed combinations of course, session number, and time slot, denoted as (i, k, l).
- FixProf: Set of fixed combinations of course, session number, and professor, denoted as (i, k, p).

Parameters

- K_i: total number of sessions of course i
- l: time slot number in $\{1, \ldots, 30\}$

Decision Variables

$$x_{i,k,l} = \begin{cases} 1, & \text{if session number } k \text{ for course } i \text{ is scheduled in time slot } l \\ 0, & \text{otherwise} \end{cases} \tag{2.1}$$

$$i \in I, \quad k \in \{1, \ldots, K_i\}, \quad l \in L_{i,k}$$

$$y_{i,k,p} = \begin{cases} 1, & \text{if professor } p \text{ is teaching session number } k \text{ for course } i \\ 0, & \text{otherwise} \end{cases} \tag{2.2}$$

$$i \in I, \quad k \in \{1, \ldots, K_i\}, \quad p \in Prof_i$$

Preference Weight:

We define three weights, two of them ($\alpha_{i,p}$ and β_l) vary depending on student's preferences and the last one (λ) is fixed.

$$\alpha_{i,p} = \begin{cases} R - r + 1, & \text{if professor } p \text{ is the } r\text{-th preferred for course } i \\ & \text{out of R ranked professors} \\ 1, & \text{if there are no preferences (R = r)} \end{cases}$$

$$i \in I, \quad p \in Prof_i$$

$$\beta_l = \begin{cases} b, & \text{if time slot } l \text{ is marked as preferred by the student, with } b \in \{1, 10\} \\ 0, & \text{otherwise} \end{cases}$$

$$l \in L_{i,k}$$

b=1 if the student prioritizes professor preferences over time slot preferences. b=10 if the student prioritizes time slot preferences over professor preferences. The student have to prioritize one or the other.

λ : a large fixed penalty for gaps between same-day sessions

Objective Function

The objective value z is to maximize the total score, which is the sum of the following terms:
- Professor Preference Satisfaction term that quantifies how well the assignment of professors to course sessions aligns with their preference rankings.
- Time Slot Preference Satisfaction term that captures the extent to which student-preferred time slots are utilized in scheduling course sessions.
-The final term, $-\lambda \cdot \text{TotalGaps}$, imposes a penalty proportional to the total number of gaps or idle periods between sessions occurring on the same day. The parameter λ is a large positive constant that emphasizes the importance of minimizing these gaps. By subtracting this value, the model encourages schedules that reduce unnecessary breaks, resulting in more compact and efficient daily timetables.

$$\max z = \left(\sum_{i,k,p} \alpha_{i,p} \cdot y_{i,k,p} + \sum_{i,k,l} \beta_l \cdot x_{i,k,l} - \lambda \cdot \text{TotalGaps} \right)$$

Constraint Formulations

1. Each Required Session is Scheduled Exactly Once

These constraints ensures that each required session of any selected course is scheduled exactly once.

$$\sum_{l \in L_{i,k}} x_{i,k,l} = 1 \quad \forall i \in I,\ k = 1..K_i \tag{2.3}$$

2. No Overlapping Sessions

These constraints ensures that for any time slot, at most one session is scheduled in it.

$$\sum_{i \in I} \sum_{k=1}^{K_i} x_{i,k,l} \leq 1 \quad \forall l \in \{1, \ldots, 30\} \tag{2.4}$$

3. Respecting the Chronological Ordering of Lecture Sessions

These constraints ensures that the time slot number used for lecture session number k for a course i is smaller than the time slot number used for lecture session number $k+1$.
By setting the upper bound to $K_i - 2$ in (2.3) , the model accounts for the structural distinction between lecture and tutorial sessions. Since K_i represents the total number of sessions (both lectures and tutorial) and courses in set I_1 have exactly one tutorial session positioned at the final session number, the constraint deliberately excludes this tutorial session from the ordering requirement.

$$1 + \sum_{l \in L_{i,k}} l \cdot x_{i,k,l} \leq \sum_{l \in L_{i,k}} l \cdot x_{i,k+1,l} \quad \forall i \in I_1, \quad k = 1 \ldots K_i - 2 \tag{2.5}$$

$$1 + \sum_{l \in L_{i,k}} l \cdot x_{i,k,l} \leq \sum_{l \in L_{i,k}} l \cdot x_{i,k+1,l} \quad \forall i \in I_2, \quad k = 1 \ldots K_i - 1 \tag{2.6}$$

4. Only One Professor is Assigned to Each Lecture Session

These constraints ensure that exactly one professor is selected for each lecture session of course i.

$$\sum_{p \in \text{LectProf}_i} y_{i,k,p} = 1 \quad \forall i \in I_1, \quad k = 1 \dots K_i - 1 \tag{2.7}$$

$$\sum_{p \in \text{Prof}_i} y_{i,k,p} = 1 \quad \forall i \in I_2, \quad k = 1 \dots K_i \tag{2.8}$$

5. Same Professor is Assigned to Both Lecture Sessions

These constraints ensure that the same professor is teaching both lecture sessions of course i.

$$y_{i,k,p} = y_{i,k',p} \quad \forall i \in I_1, \quad \forall p \in \text{LectProf}_i, \quad \forall k, k' = 1 \dots K_i - 1, \quad k \neq k' \tag{2.9}$$

$$y_{i,k,p} = y_{i,k',p} \quad \forall i \in I_2, \quad \forall p \in \text{Prof}_i, \quad \forall k, k' = 1 \dots K_i, \quad k \neq k' \tag{2.10}$$

6. Respect Existing Professor Pairs

These constraints ensures that the tutorial professor is selected only from the valid options associated with the lecture professor that teaches course i.

$$\sum_{p \in \text{TutProf}_{i,q}} y_{i,k,p} = 1 \quad \forall i \in I_1, \quad k = 1 \dots K_i, \quad \forall q \in \text{LectProf}_i \tag{2.11}$$

7. Professors Teach in The Right Time Slot

These constraints ensures that a professor can not teach in time slots outside his schedule, in other words,

$$y_{i,k,p} \leq \sum_{l \in S_{i,k,p}} x_{i,k,l} \quad \forall i \in I, \quad k = 1 \dots K_i, \quad p \in \text{Prof}_i \tag{2.12}$$

8. Fixed Course Sessions

These constraints enforce that sessions predefined with fixed time slots or assigned professors must strictly adhere to those fixed values during the scheduling process. They cover two cases: - When a student has neither failed nor skipped courses (currently), both the assigned professors and time slots are predetermined to align with the group the student was placed in at the beginning of the academic year.

- Junior and Senior students who have failed courses are required to retake them exclusively with groups that corresponds to their declared specialization combination.

$$x_{i,k,l} = 1 \quad \forall (i,k,l) \in \text{FixSlot} \tag{2.13}$$

$$y_{i,k,p} = 1 \quad \forall (i,k,p) \in \text{FixProf} \tag{2.14}$$

2.6 UML Diagrams Used

Unified Modeling Language (UML) offers a range of diagrams for visualizing different aspects of a system. In this project, two diagrams were particularly important:

- **Use Case Diagram:** This diagram focuses on the system's external behavior by mapping out how users (actors) interact with the system. It was useful in capturing the core functional requirements and establishing user-related scenarios.

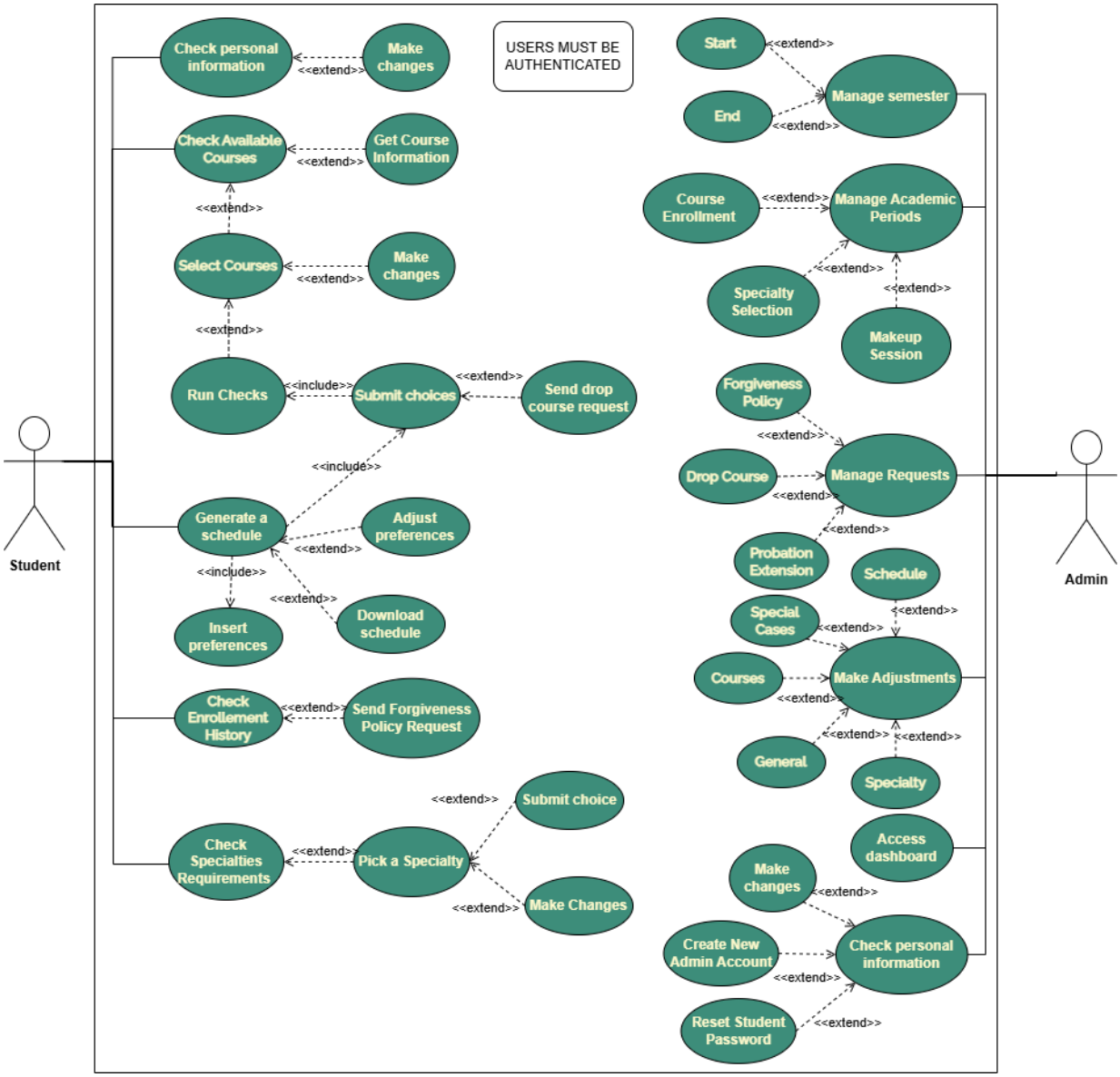


Figure 2.8: Use Case Diagram

- **Class Diagram:** This diagram provides a structural snapshot of the system by illustrating its key classes and how they are connected. It was instrumental in shaping the database design and guiding code development.

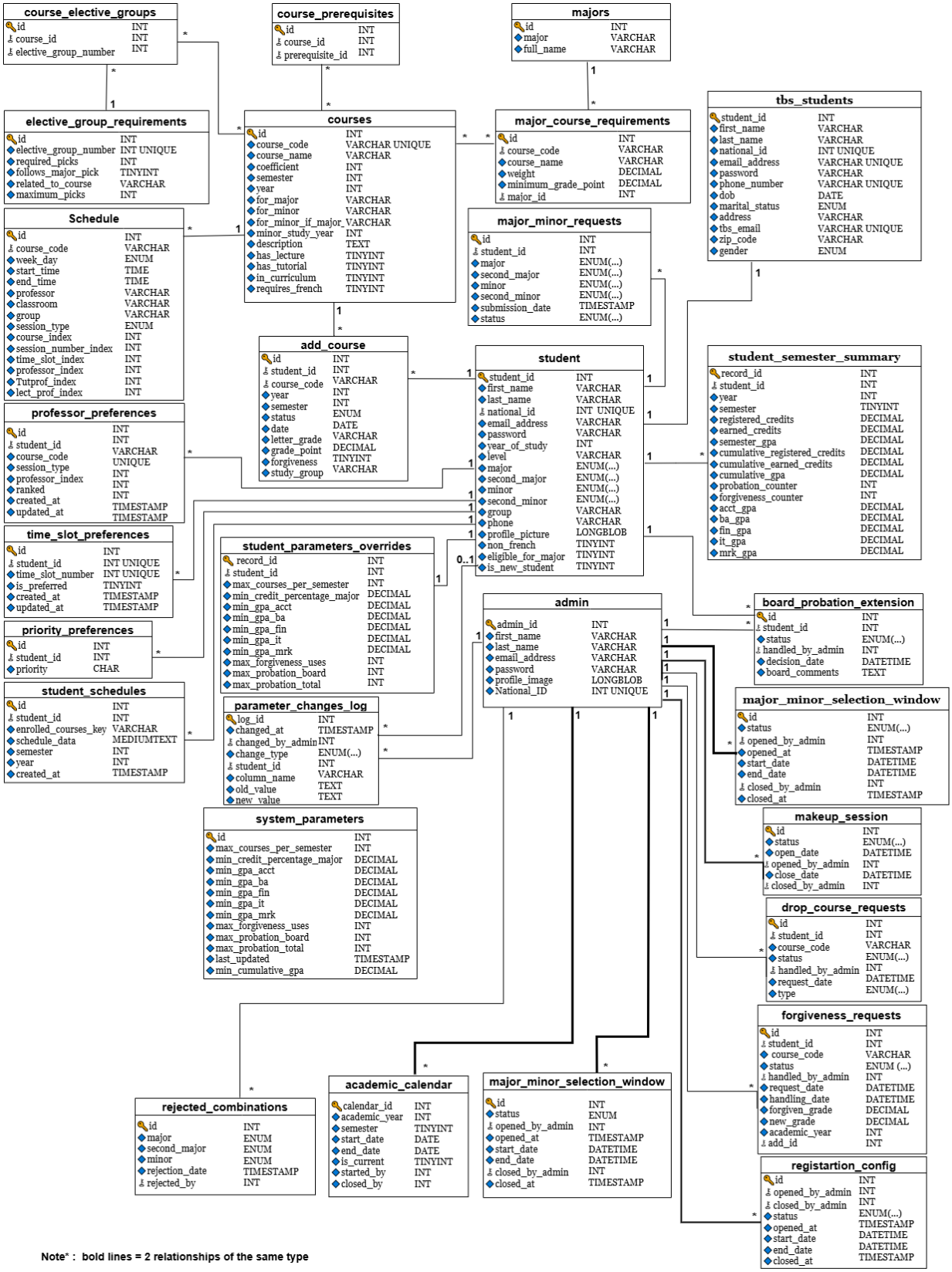


Figure 2.9: Class Diagram of The Course Registration System

2.7 Data Acquisition and Preprocessing

2.7.1 Data Collection

Given the nature of the project and its significant impact on the academic trajectory of the students, it was essential to ensure that the collected data was accurate and comprehensive. Accuracy in this context is critical to guarantee that the system functions without flaws or inconsistencies. To achieve this, two primary sources of information were considered:

- **Official Handbooks:** Multiple versions of the official student handbook were reviewed, with particular emphasis on the most recent edition to ensure that the up-to-date policies and academic rules were reflected in the system.
- **Administrative Interviews:** Direct, face-to-face interviews were conducted with the representative of the grade department at TBS, the main authority responsible for managing all student course registration processes. These discussions were essential to clarify the practical details of implementation and identify any undocumented rules or exceptions.
- **Other Sources:** Certain specific data points were obtained from various official and unofficial sources, including the TBS student archive, the official website, and social media platforms.

2.7.2 Data Preparation

The data preparation phase was a time-intensive and ongoing process that extended throughout the project. It was not limited to a single step but evolved continuously as new issues were discovered and new requirements emerged. From initial collection to final validation, the process involved multiple iterations of data retrieval, inspection, correction, and standardization. Ensuring data consistency and completeness was essential, particularly because even small inconsistencies could severely impact data sensitive processes like generation of schedules.

2.7.3 Data Cleaning

Data cleaning was primarily performed using SQL queries directly within the database. In certain cases, manual corrections were applied for convenience. It is imperative that the data be as clean and consistent as possible, particularly for schedule generation where even minor discrepancies can result in infeasible or incorrect solutions. For this reason, the majority of the cleaning efforts focused on the schedule data, which initially exhibited several inconsistencies, including:

- **Missing Data:** Some schedule entries lacked essential values such as professor names or time slots. These were addressed either by inserting placeholder values or, when possible, by retrieving accurate information from alternative reliable sources.
- **Inconsistent Naming Conventions:** Identical entities (e.g., professors or courses) were referred to using different names or formats (e.g., "Dr. Smith" vs "Smith, D."). This inconsistency was resolved by standardizing entries using mapping tables and unifying them under a single, consistent naming convention or identifier.

- **Redundant Entries:** Some sessions were duplicated in the schedule table with slight variations. These duplicates were identified using a combination of fields (course, time, professor) and eliminated to avoid confusion or conflicts during optimization.

Data Insertion

After the data cleaning process was completed, the structured and validated data was systematically inserted into the 30 database tables that were defined as part of the system architecture. This insertion process occurred progressively throughout the development of the project, ensuring that each module of the system had access to the necessary and consistent data. The data population was closely aligned with the evolving logic of the application, allowing for continuous testing, validation, and refinement.

2.8 Conclusion

This chapter detailed the systematic process followed to conceive, structure, and develop the course registration web application. From identifying actors and functional requirements to building the course selection logic and optimization model, each component was designed to reflect real academic policies and user needs. The scenario-driven agile methodology ensured practical alignment with the student life cycle and administrative operations. By combining rigorous data preparation with well-defined constraints and models, the system delivers a foundation that supports both student decision-making and institutional control in an efficient, scalable way

Chapter 3

Implementation

This chapter presents the technical foundation of the project, outlining the tools and technologies used throughout its development. It discusses the essential requirements, the development environment, and the rationale behind major technological decisions. Additionally, the chapter provides an overview of the implementation process and showcases the final deliverables.

3.1 Prerequisites

This section outlines the software setup and selected technologies that enabled the successful execution of the project.

3.1.1 Hardware Environment

Table 3.1 summarizes the key hardware and system specifications of the development machine.

Model	Dell
Processor	Intel Core (TM) i7-8750H
RAM	8 GB
Operating System	Windows 10 SE

Table 3.1: Computer Specifications

3.1.2 Software Environment

MySQL: a RDBMS used for handling structured data in web applications. It supports SQL and provides robust tools for data storage, retrieval, and consistency. MySQL is known for its reliability, speed, and compatibility with various platforms, making it a popular choice for both small-scale and enterprise-level systems. In this project, MySQL was used as the core database engine to store and manage academic data, including but not limited to information concerning students, courses, multiple parameters and schedule. Complex queries and database triggers and procedures were also implemented to enforce academic rules and automate various operations such as grade updates and credit calculations.

Visual Studio Code: is a lightweight yet powerful source-code editor developed by Microsoft. It supports a wide range of programming languages and provides built-in features such

as intelligent code completion, debugging tools, Git integration, and an extensive marketplace of extensions. VS Code is highly customizable, allowing developers to tailor the environment to specific project needs. For this project, Visual Studio Code was the main development environment used to build both the frontend interface and backend logic. It facilitated efficient management of the codebase, seamless navigation between files, and integration with version control systems during the development lifecycle.

Insomnia: is a powerful REST client designed to help developers test and debug APIs. It offers an intuitive interface for crafting HTTP requests, managing authentication, inspecting responses, and organizing requests into workspaces. It is especially useful during API development and testing phases due to its simplicity and flexibility. In this project, Insomnia was used extensively to test all backend endpoints built using Flask. It allowed for simulating various student and admin requests, ensuring that data was correctly handled, authentication was working, and error handling was properly implemented before integrating the front-end.

3.1.3 Technology Choices

This section provides an overview of the frameworks and libraries utilized in the project, along with the rationale behind their selection.

Programming Language

Python: is a high-level, interpreted programming language known for its clarity, concise syntax, and broad applicability across domains. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming, and offers a vast ecosystem of libraries and frameworks. Python is commonly used in web development, data science, scripting, automation, and software prototyping due to its flexibility and strong community support. In this project, Python was used as the sole programming language for backend development. It enabled the creation of RESTful APIs, implementation of business logic, and integration with the MySQL database. Its simplicity and powerful libraries facilitated rapid development and easy maintenance throughout the project lifecycle.

Framework and Libraries

Flask: is a lightweight Python web framework built on the Werkzeug toolkit and Jinja2 templating engine. It provides essential tools for building scalable web applications with minimal setup, including support for routing, request handling, and templating. In this project, Flask served as the foundation of the backend web application. It was used for URL routing, API endpoint creation, and HTML rendering via templates such as `index.html`.The application was modularized using Flask blueprints to maintain clean separation of concerns. Additionally, Flask was integrated with MySQL using `Flask-MySQLDB` for database communication, and handled environment configuration such as debug mode, secret keys, and database credentials.

PuLP: is an open-source Python library used for modeling and solving linear and integer programming problems. It offers an intuitive interface to define decision variables, constraints, and objective functions, and can work with a variety of solvers. PuLP is widely applied in

fields such as operations research, logistics, and resource allocation. In this project, PuLP was employed to handle scheduling optimization tasks. It was used to construct a mathematical model representing scheduling decisions and constraints, with an objective to generate feasible and preference-aware schedules. The library provided the flexibility to encode business rules and solve them efficiently using a suitable solver backend.

datetime: is a built-in Python module used for manipulating date and time objects. It offers functionality for formatting, parsing, arithmetic, and timezone-aware time handling. In this project, `datetime` played a key role in managing time-related logic. It was used for handling registration deadlines, calculating differences between time intervals, formatting timestamps for display, and managing course start and end times. Additionally, it supported time conflict detection in the scheduling algorithm and helped log operations with precise time records for tracking and debugging purposes.

Chart.js: is an open-source JavaScript library designed for creating responsive and interactive data visualizations using HTML5 canvas. It supports a wide range of chart types such as bar, line, pie, and radar, and is known for its simplicity and ease of integration with modern frontend frameworks. In this project, Chart.js was used to build visual components within the administrative dashboard. It enabled the display of key metrics and summaries through dynamic charts, helping to present complex data in a clear, accessible, and visually engaging manner.

3.2 Login and Registration

3.2.1 Home Page

The Figure 3.1 represents the simple home interface that the user will see initially.

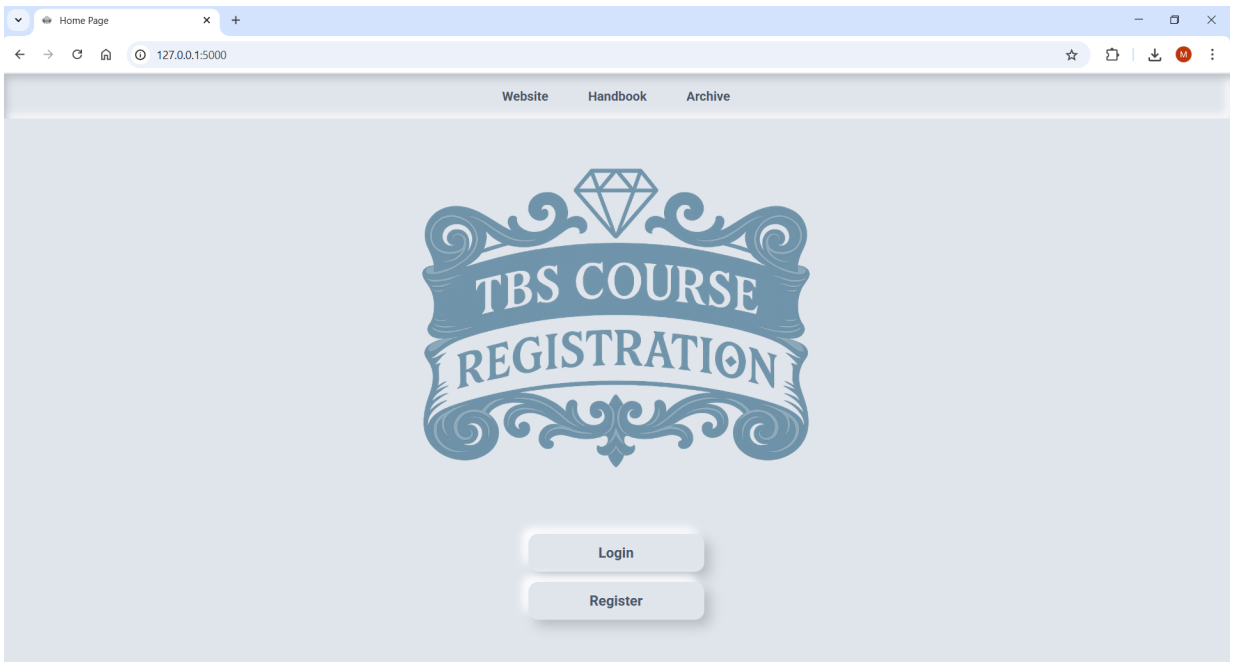


Figure 3.1: Home Page

It includes the following elements from top to bottom:

- A favicon inspired by the TBS mantra *"Diamonds are made under pressure"*

Figure 3.2: Website Favicon

- A utility bar with useful links
- An illustrative image representing the application's purpose
- *Login* and *Register* buttons for user access

3.2.2 Login Window

The login window is shared by both students and administrators. It includes two input fields: one for the email and one for the password, along with an option to reset the password.

Figure 3.3: Login Window

3.2.3 Reset Password Window

In case a student forgets their password, they can click on the *Forgot Password* option, which opens a new window prompting them to enter their email address and national ID, just as shown in Figure 3.4.

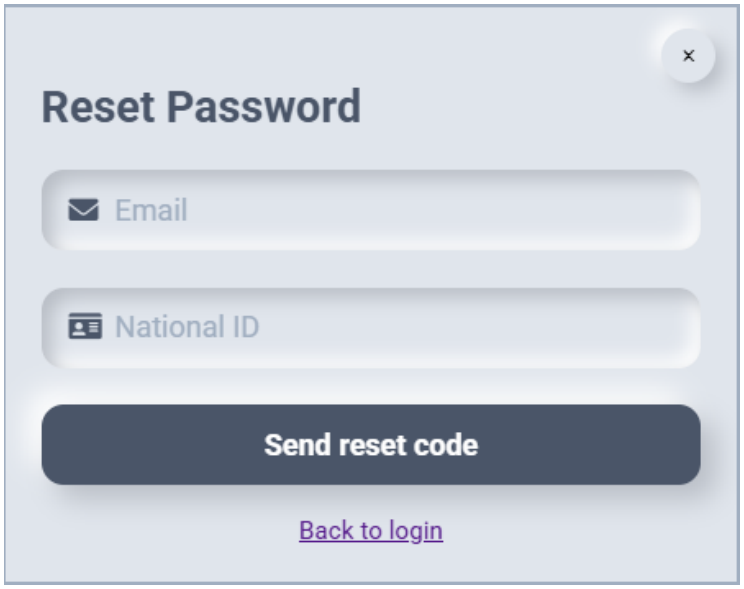


Figure 3.4: Password Reset Interface

After completing this step, a verification email containing a six-digit code is sent to the student's registered email address with a lifespan of 10 min just as illustrated in Figure 3.5.

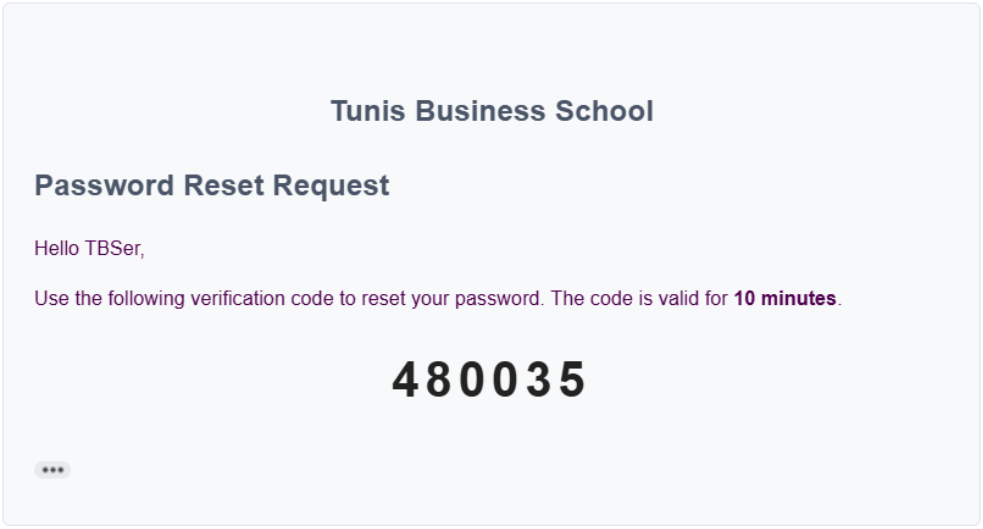


Figure 3.5: Email Verification Code Sample

The student must then enter the verification code along with their new password in the next window shown in Figure 3.6

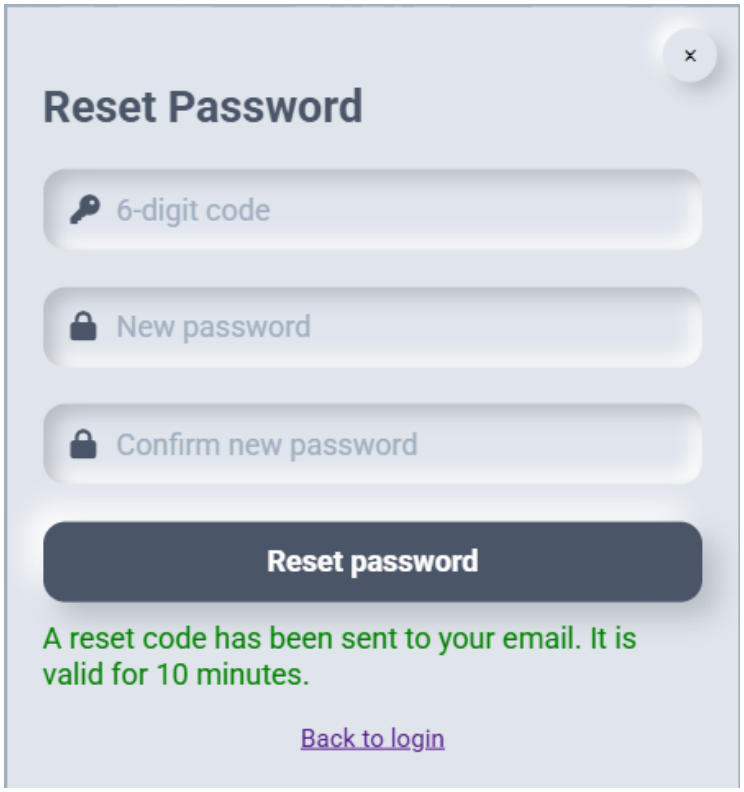


Figure 3.6: New Password Entry Screen

If the password inputs match and the verification code is valid, the system accepts the new password.

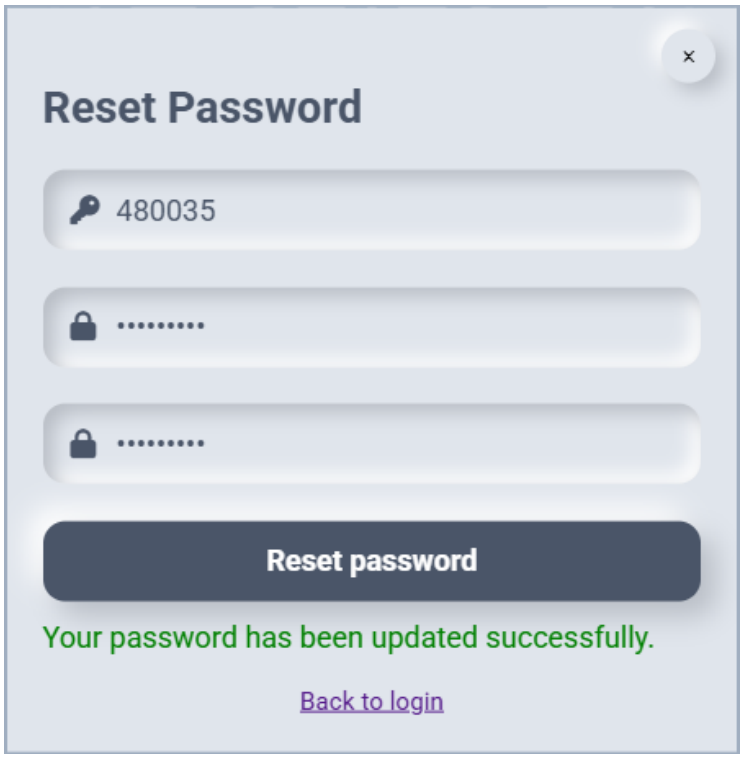


Figure 3.7: Confirmation of Successful Password Reset

3.2.4 Registration Window

The registration window is intended for student use only; administrator accounts can only be created by existing admin accounts. To ensure that only TBS students can register, the entered email and national ID are cross-checked against the information of currently enrolled students in the TBS database.

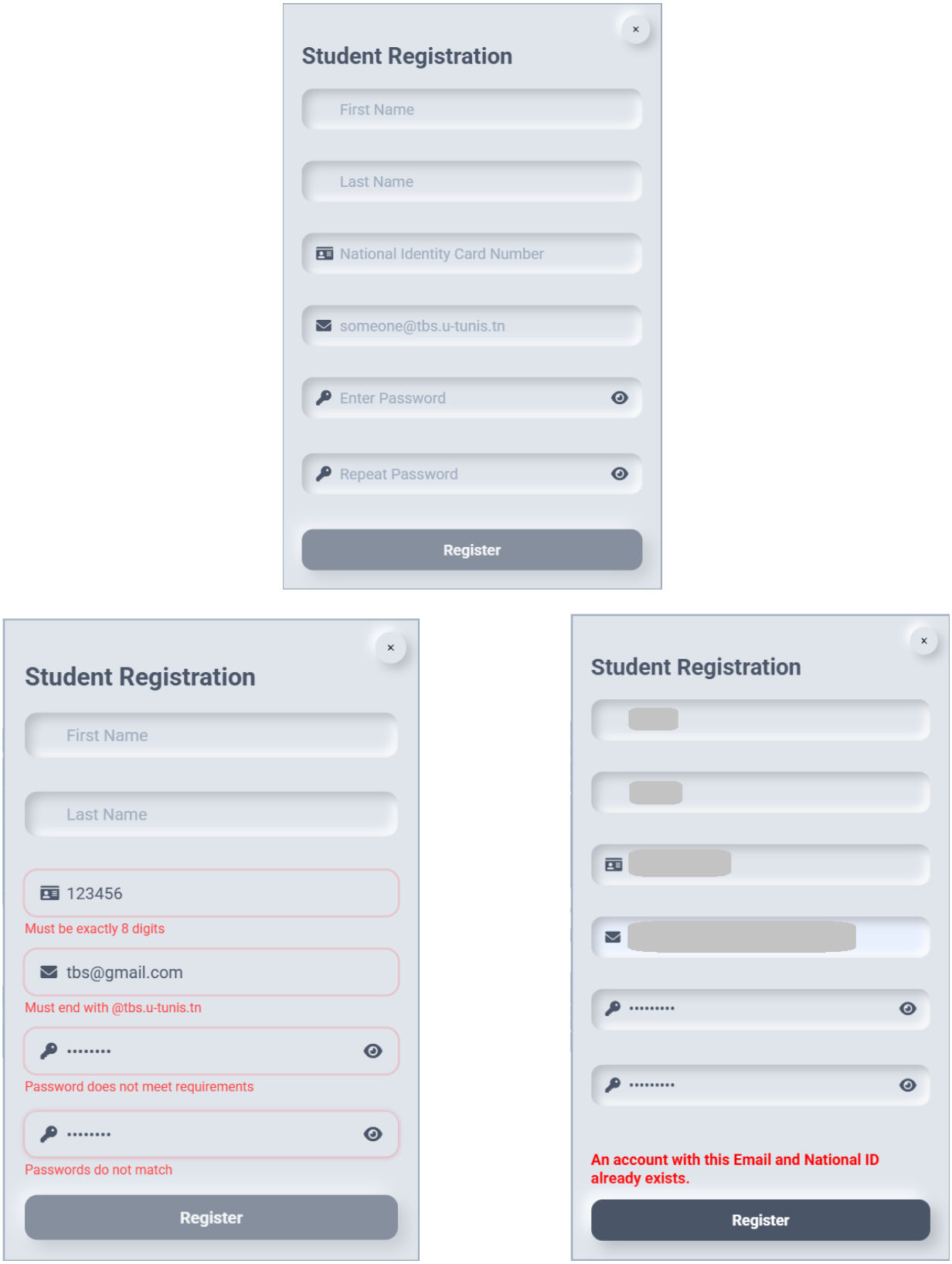


Figure 3.8: Registration window

Upon successful registration, a confirmation email is sent to the user's address, containing a clickable link to validate the account. Gmail Simple Mail Transfer Protocol (SMTP) was used to provide the email delivery service for this functionality. A sample of the email is presented in

Figure 3.9

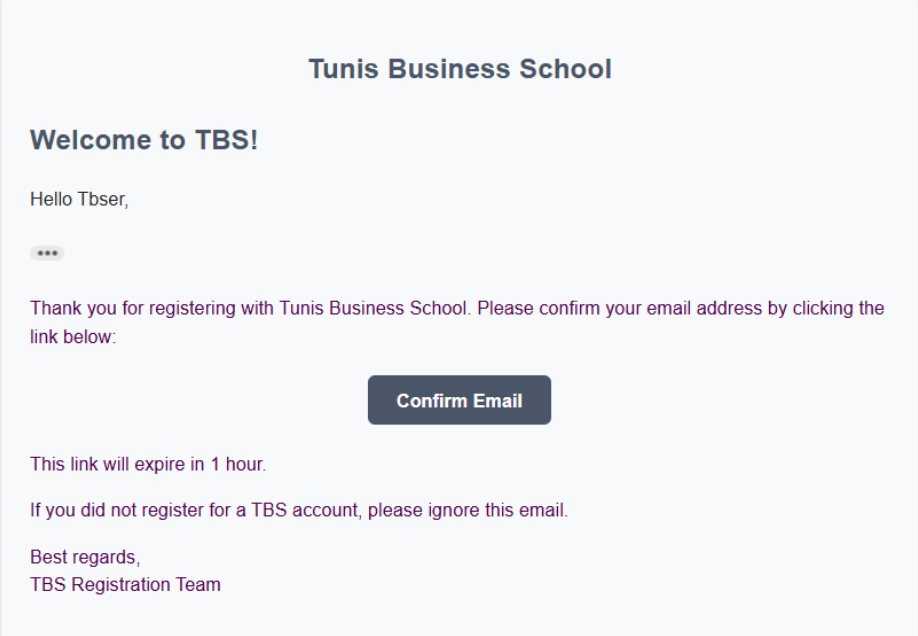


Figure 3.9: Sample confirmation email

3.3 Student Portal

Once logged in, the student gets access to the side menu displayed in Figure 3.10.A brief overview of the elements from top to bottom:

- **A clickable profile picture:** that gives access to personal information and the ability to edit them.
- **Course Enrollment section:** it is the landing section at login, where the student can select courses for the semester and where most of the notifications and alerts will be displayed.
- **Enrollment History section:** it acts as an unofficial academic transcript that provides a comprehensive record of the student's academic performance.
- **Major/Minor section:** it tracks major requirements and allows the student to select their specialization once elligible.
- **Schedule section:** This is where the final step of registration takes place. The student can generate their schedule and confirm the groups they will enroll in.
- **Makeup section:** A hidden section that will only be visible to eligible students.

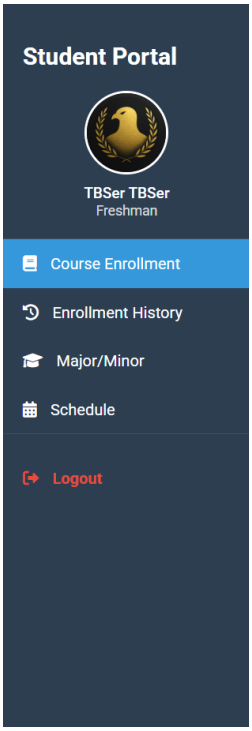


Figure 3.10: Student's side menu

3.3.1 Personal Information

The student can access their personal information by clicking on their profile picture. They are then able to freely edit all non-fixed fields.

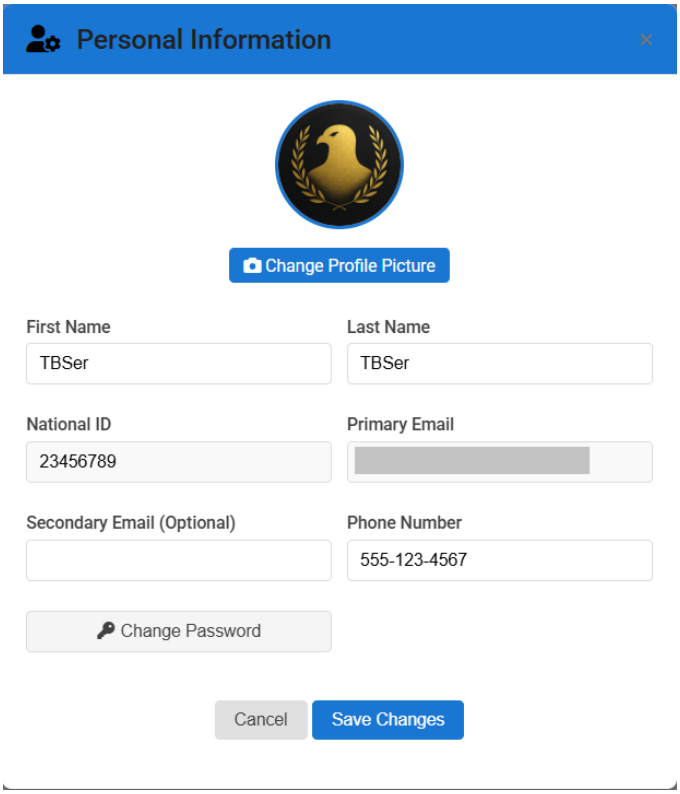


Figure 3.11: Student Personal Information Interface

3.3.2 Course Enrollment

The content and available actions in this section depend on the current state of the course registration period, which can be opened or closed by the admin.

1. No Active Semester

Shown in Figure 3.12 is the interface when there is no active semester yet. It will change once the admin starts a new one.

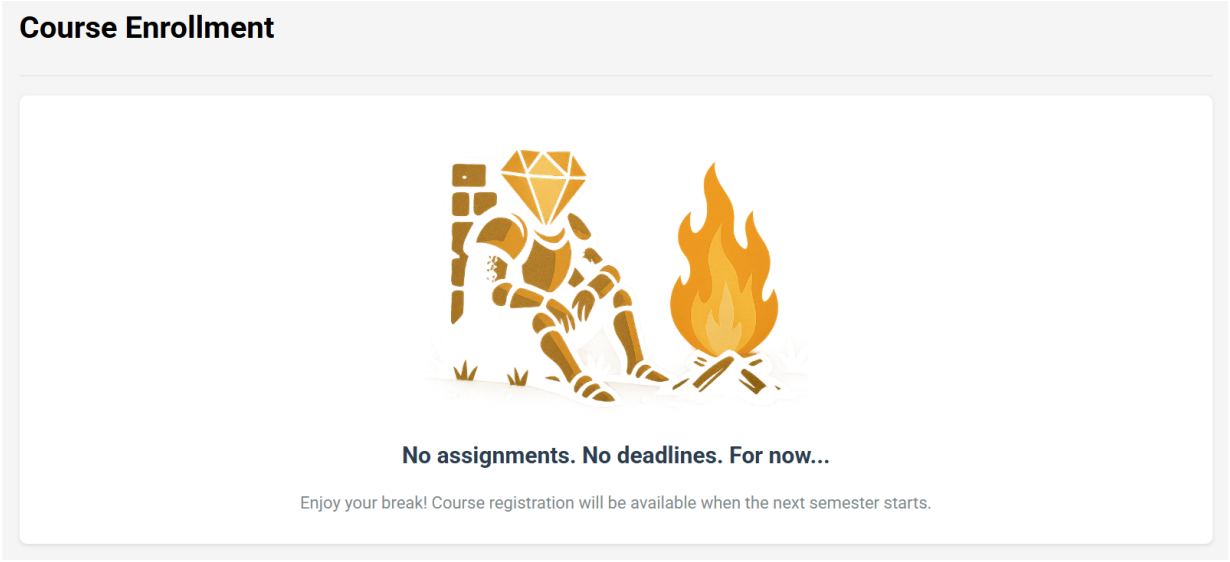


Figure 3.12: Closed Course Enrollment Section

2. Active Semester and Opened Registration Period

Once the admin starts a new semester and opens the course registration period, the student gains access to a similar interface as shown in Figure 3.13 but the displayed options will vary based on the individual's academic standing and eligibility. The one shown in the figure is for a newly enrolled student.

Example 1:

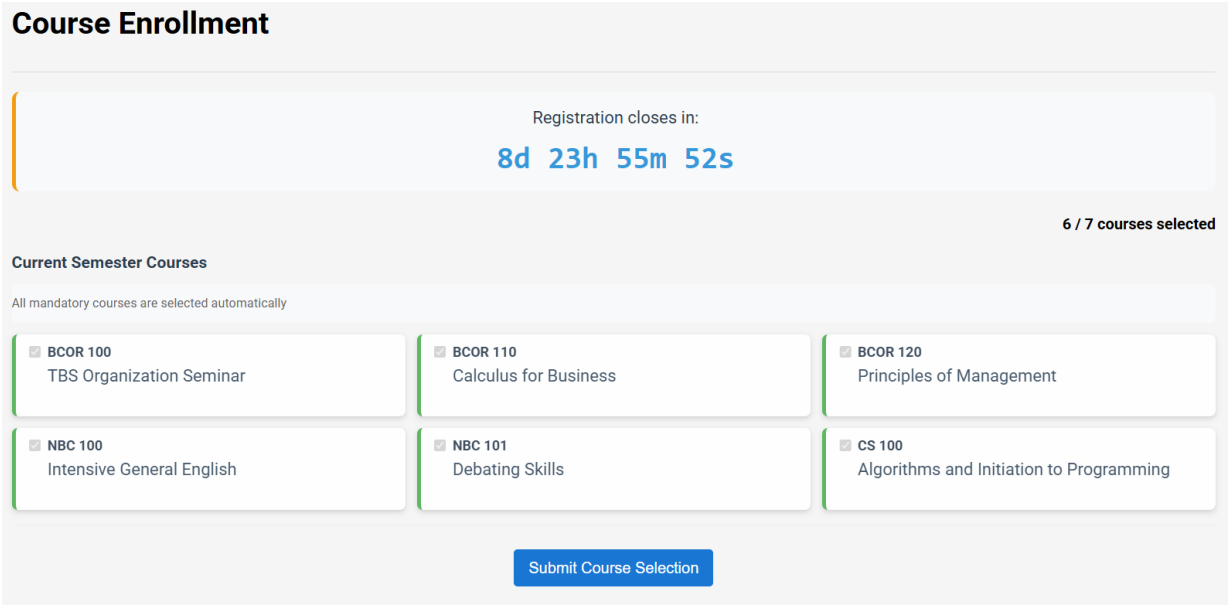


Figure 3.13: Open Course Enrollment Section - Example 1

Additionally, the student finds useful indicators such as a countdown for the registration period end as well as a counter for the currently selected courses and the total number of allowed courses in that semester. The student can also hover over any course and a popup will show up on the side to provide additional information that will help him make a better informed decision just as visualized in Figure 3.14. The course details include a short description, the course components (lecture, tutorial, or both), the prerequisites required, and the courses for which it serves as a prerequisite

Figure 3.14: Course Details Popup

It is important to not that some mandatory courses will be automatically enrolled in once the registration period ends even if the student does not confirm his choices himself as explained in Figure 2.4 of chapter 2.

Example 2:

Figure 3.15 shows a case of a student during his third year with multiple categories of courses to choose from. This is in accordance with the general course flow depicted in Figure 2.3 of chapter 2.

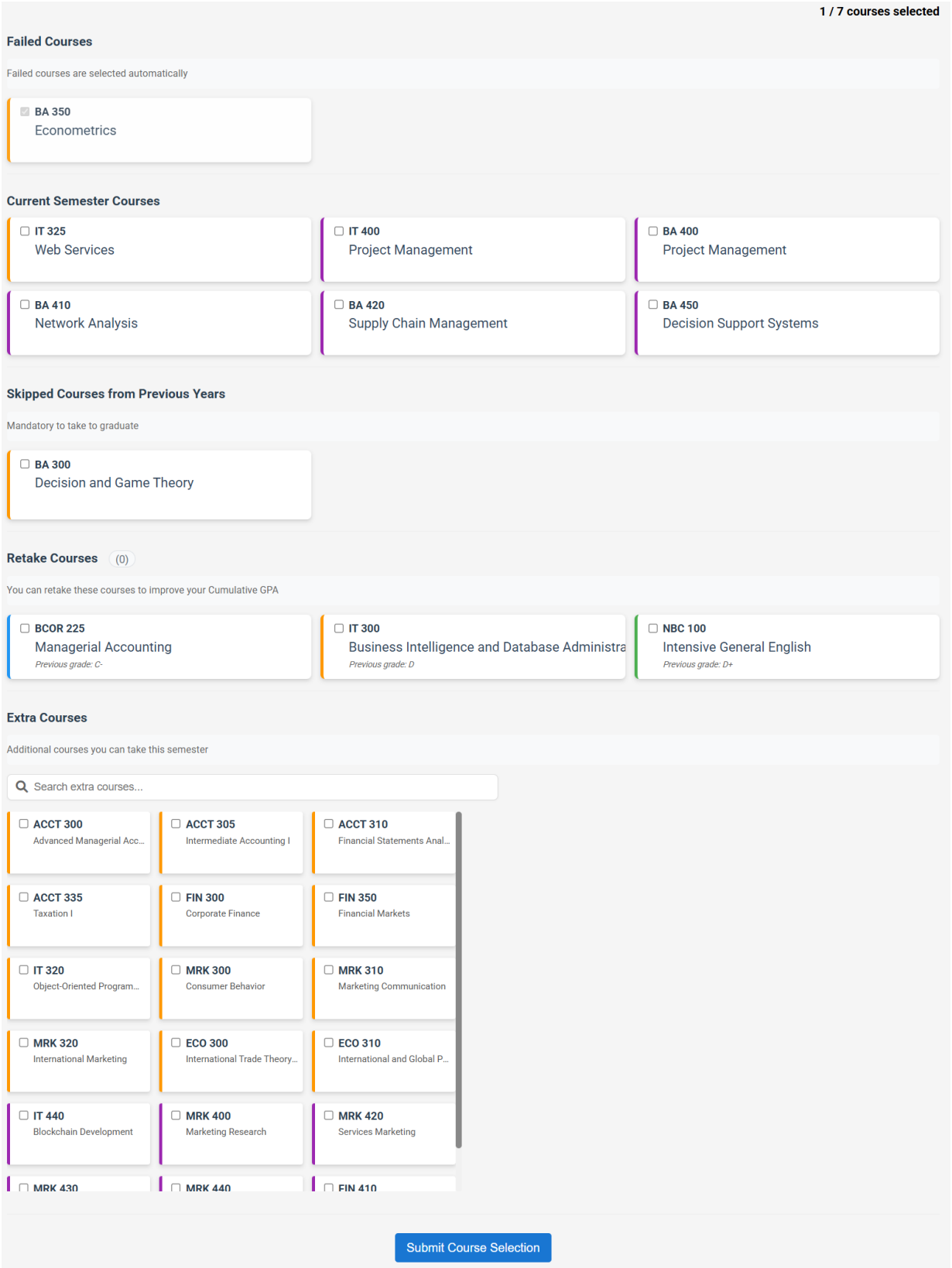


Figure 3.15: Course Enrollment Section - Example 2

Once the student is satisfied with his course selection, he can click *Run check* button that does 2 things in a specific order.

-**Run the schedule optimization model** in the background to validate the current course

selection, as illustrated in Figure 2.6 in Chapter 2. If the validation fails, meaning no feasible schedule is possible just as shown in Figure 3.16 then the student must adjust their choices and try again.

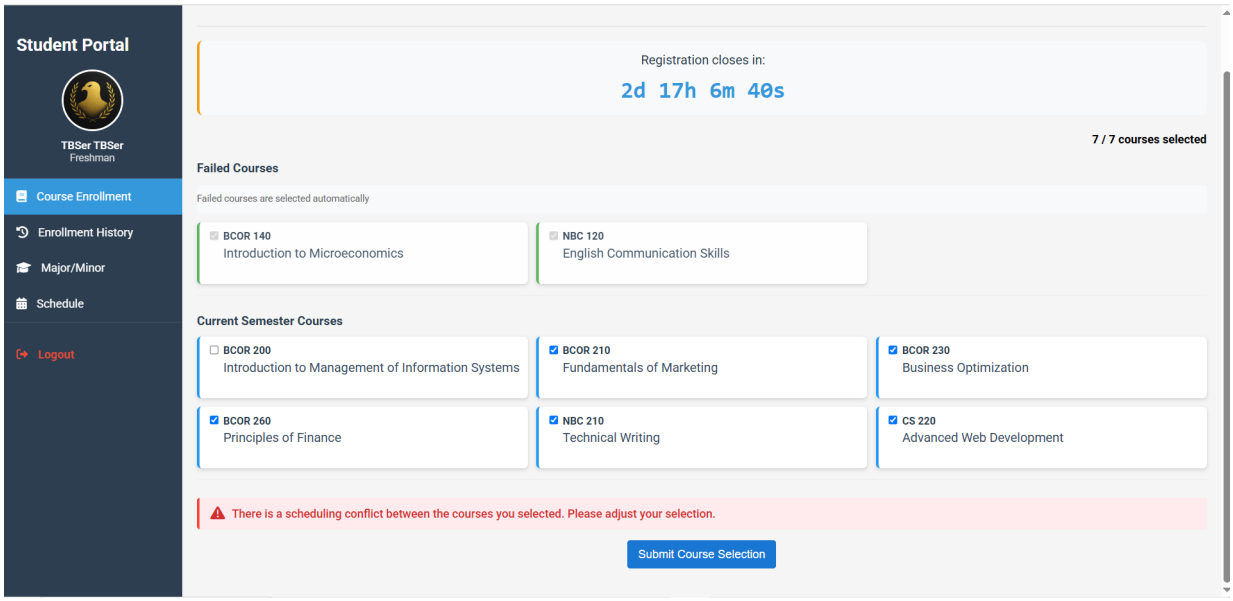


Figure 3.16: Example of Schedule Conflict for Course Selection

However if the first test passes, then the second check will start.
- **Run the assessment unit** to display proper feedback if the conditions for it are met. There are three functions that compose the assessment unit, each covering a specific case:

- **Probation:** A function is triggered when the student is under academic probation. It checks whether the current course selection is sufficient to potentially lift the probation status in the next semester. It calculates the needed GPA for this semester and suggests helpful actions if necessary. Both Figure 3.17 and Figure 3.18 depict feedback given to the same student with different course selections.

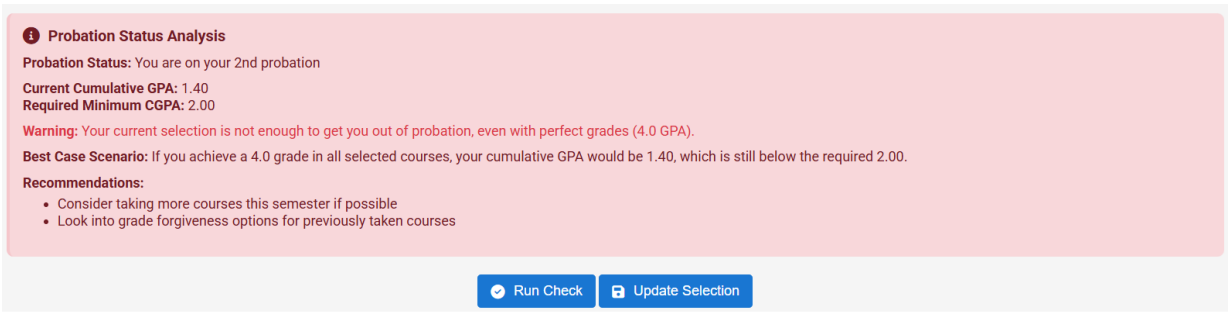


Figure 3.17: Probation Check With First Course Selection

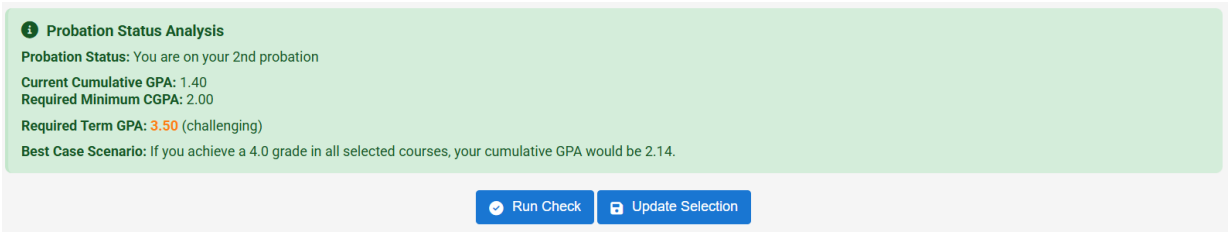


Figure 3.18: Probation Check With a Second Course Selection

- **Specialization:** A function that is generally triggered during the first two years of study to help the student understand the ramifications of their choices and set realistic goals toward the specialization they are aiming for. It calculates the specialized GPA for each major based on selected and unselected courses (when applicable), and provides feedback accordingly. The example of Figure 3.19 is that of a student during the first semester of his second year.

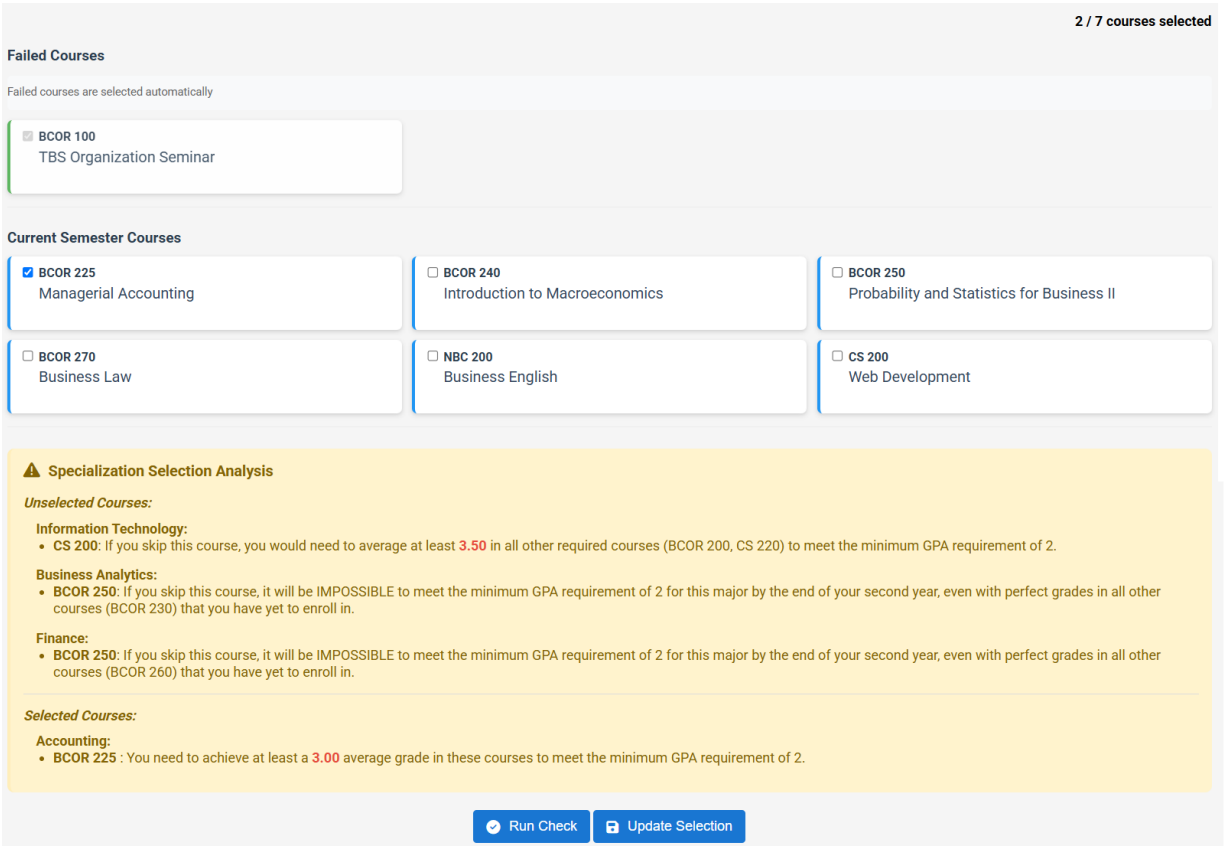


Figure 3.19: Specialization Analysis Based on Course Choices

- **Prerequisites:** A function that automatically checks the set of unselected courses and determines how skipping any of them could block access to other courses in the future. This is particularly important in multi-level prerequisite chains (e.g., A $\rightarrow$ B $\rightarrow$ C), where failing or skipping an early course like A can prevent the student from eventually enrolling in C. By highlighting these dependencies, the system helps students make more informed decisions about what to prioritize now to avoid academic roadblocks later.

Once a student is satisfied with his course selection, he can click *Submit Course Selection* button to confirm his choices and get a summary of the chosen courses indicating a successful enrollment:

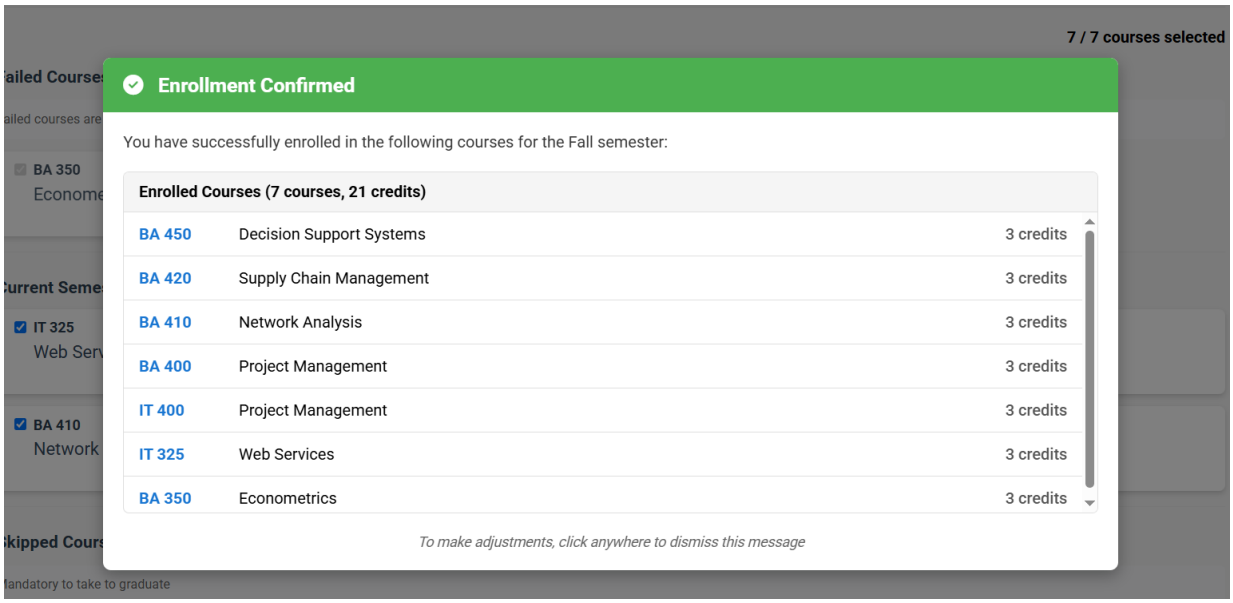


Figure 3.20: Example of a Successful Enrollment Process

Note that adjustments to the initial selection could still be freely made, even if accepted, as long as the registration period is still open.

3. Closed Registration Period

Once registration is closed, the student can no longer add courses. However, they may still submit drop course requests to the administration by clicking the red "x" on the right side as displayed in Figure 3.21.

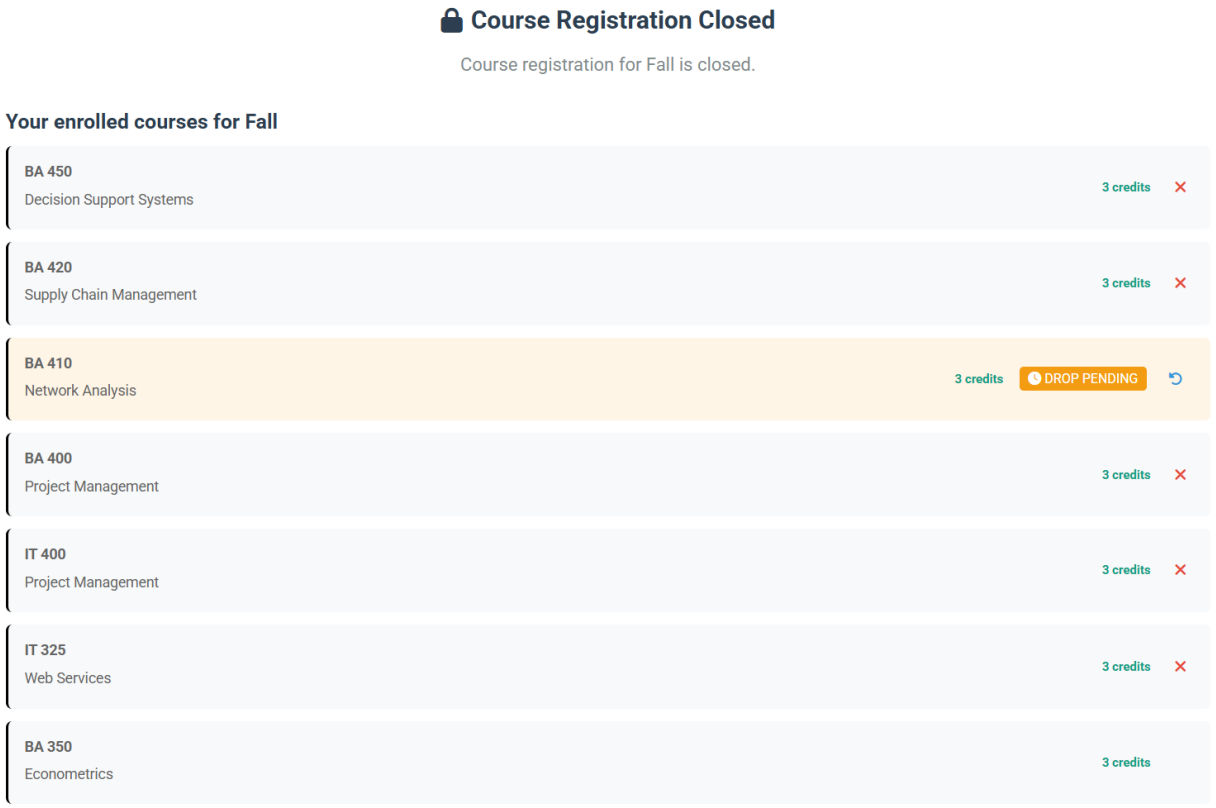


Figure 3.21: Registration Closed Notification Interface

Once the student does that, a confirmation popup will appear, requiring the student to input the word "YES" in order to submit the request, as shown in Figure 3.22. If the drop conditions are satisfied, the request will be approved by the administrators.

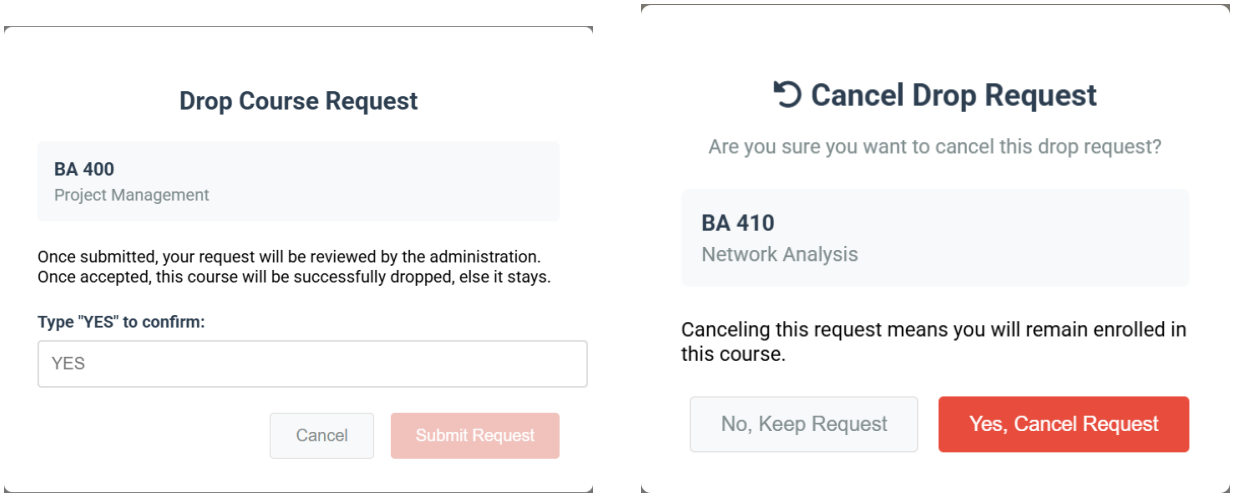


Figure 3.22: UI Screens for Sending/Canceling a Drop Course Request

4. Alerts and Restrictions

There are different kinds of alerts that gets displayed once certain conditions are met. They serve to inform and help the student understand his current standing and what he needs to do. Those alerts can be accompanied by restrictive actions.

Probations

A probation period is when the student's CGPA falls short of the required minimum CGPA set by the administration. Consecutive probation can lead to dismissal. The Figure 3.23 above presents the alerts for a first and a second probation, urging the student to improve his CGPA. The question mark provides information about the probation system.

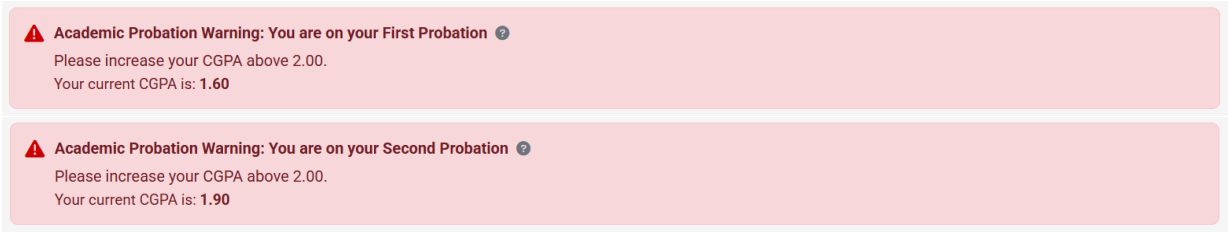


Figure 3.23: Probation Alerts: First and Second Warnings

Once the student reaches a certain number of consecutive probations as illustrated Figure 3.24 depicts a restricted student account awaiting the scientific board's decision regarding the granting of a semester extension.

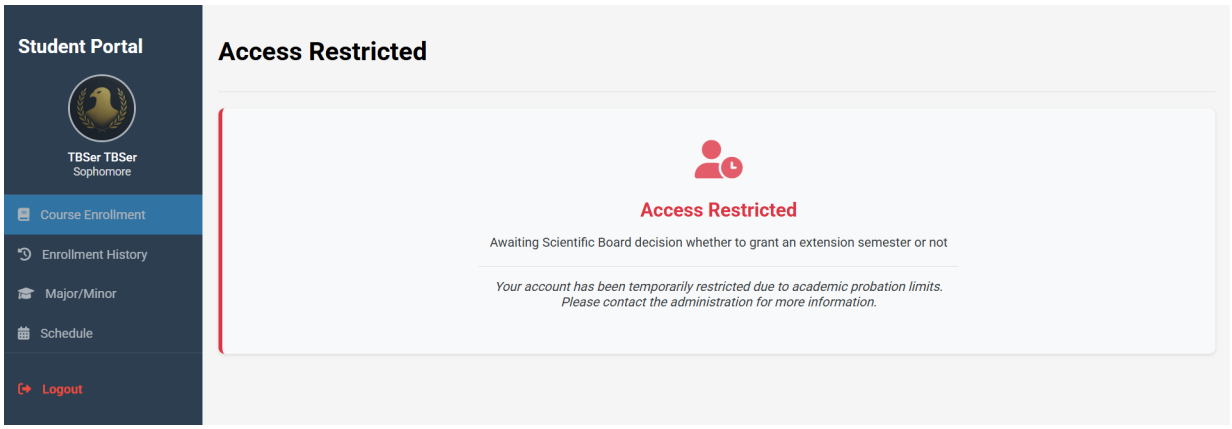


Figure 3.24: Access Restricted Message

The student loses access to their account entirely, as shown in Figure 3.25, once they reach the maximum number of allowed probations.

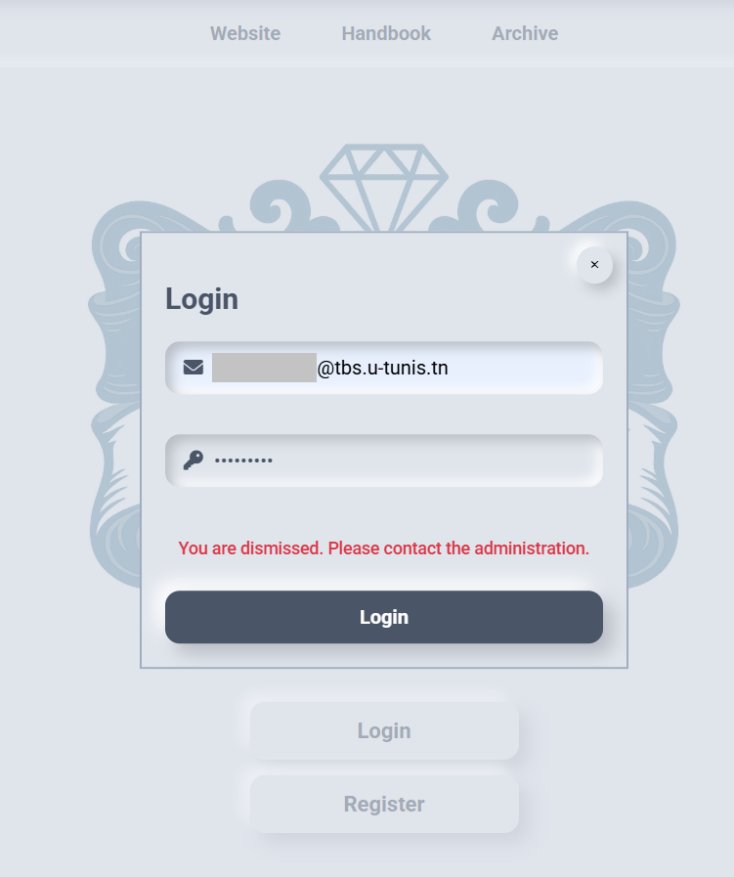


Figure 3.25: Dismissal After Final Probation

Unfulfilled Earned Credits Requirements

The alert in Figure 3.26typically appears when a student enters their third year of study but has not reached the minimum required number of earned credits. Limitations are then applied to the set of eligible courses.

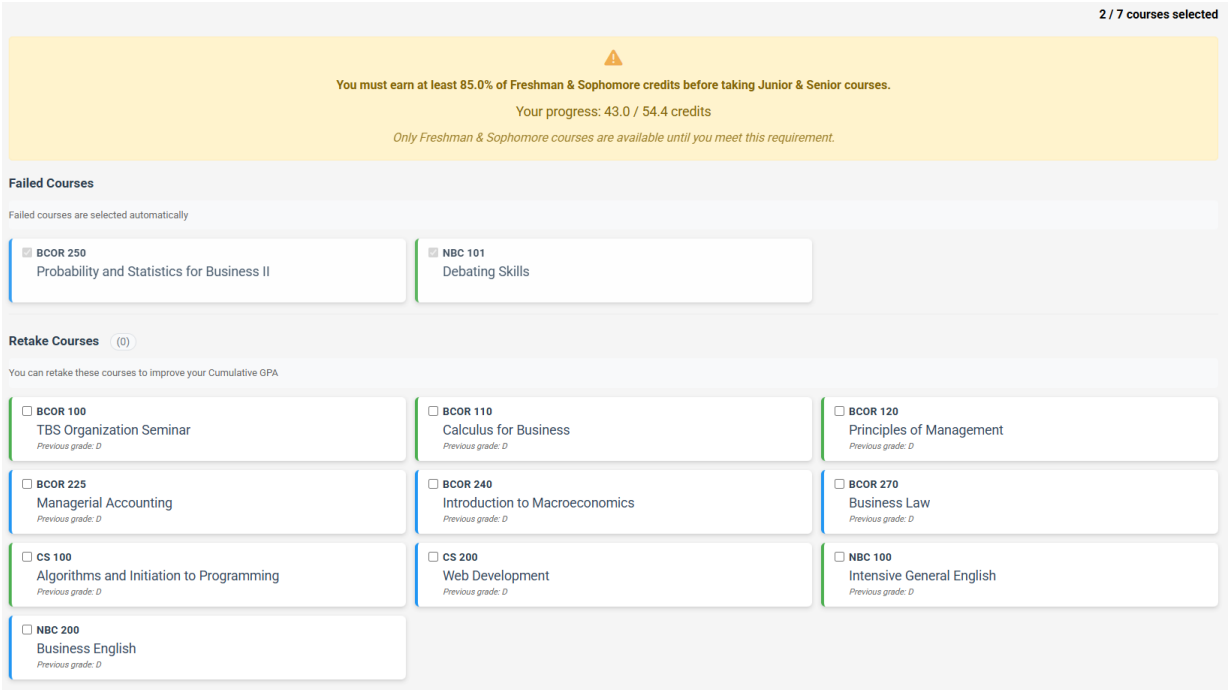


Figure 3.26: Unmet Earned Credit Notification

Major Messages

The example show in Figure 3.28 is that of a student who has enough earned credits to enroll in Junior courses but has not yet fulfilled all the requirements for selecting a major.

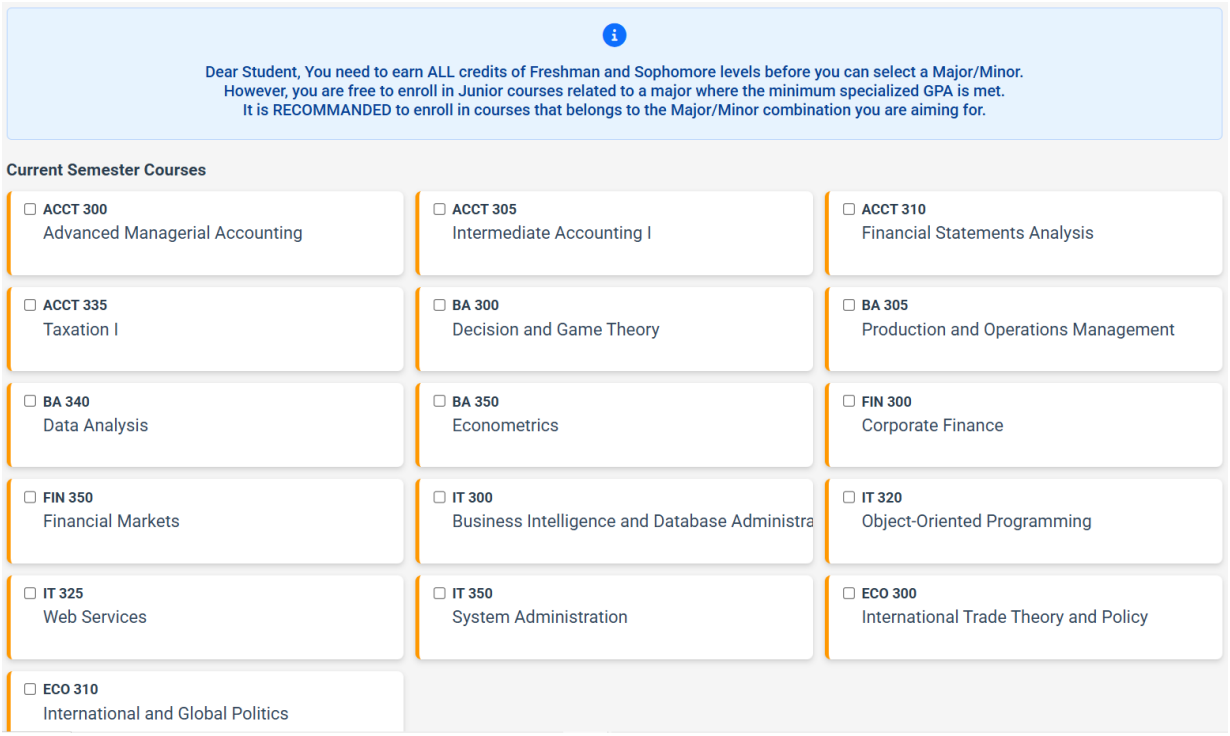


Figure 3.27: Unmet Major Requirements View

Once the student fulfills all conditions for specialization selection, then the message in Figure 3.28 will show up.

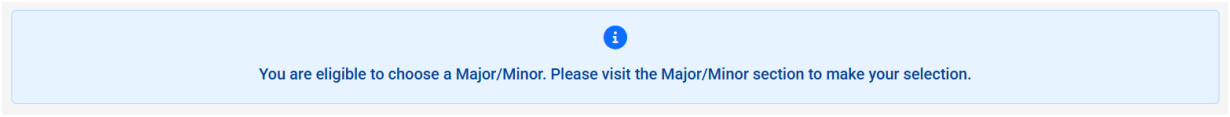


Figure 3.28: Eligible Major Selection Interface

Graduation

Figure 3.29 illustrates a celebratory notification popup displayed once the student is eligible for graduation. The student can still browse around to check his enrollment history.

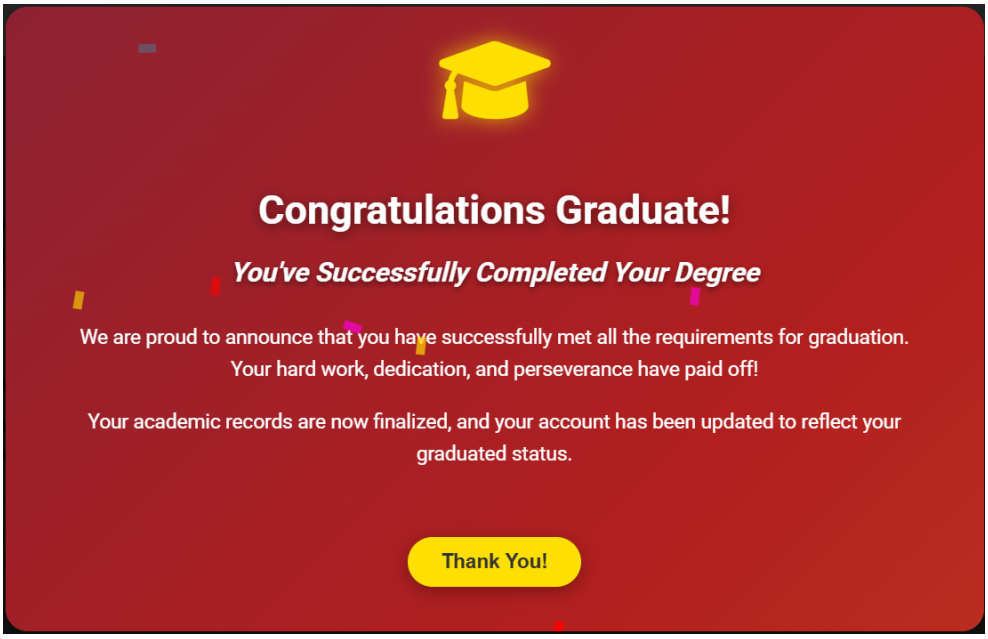


Figure 3.29: Graduation Message

3.3.3 Enrollment History

This section is designed with two main purposes in mind:

- To keep track of past enrollment records and various academic metrics:
  As presented in Figure 3.30, academic metrics such as registered credits, earned credits, and GPA are calculated for each semester. This section functions similarly to the official academic transcript, but is superior as it updates automatically whenever course enrollments or grades change, and it remains easily accessible.

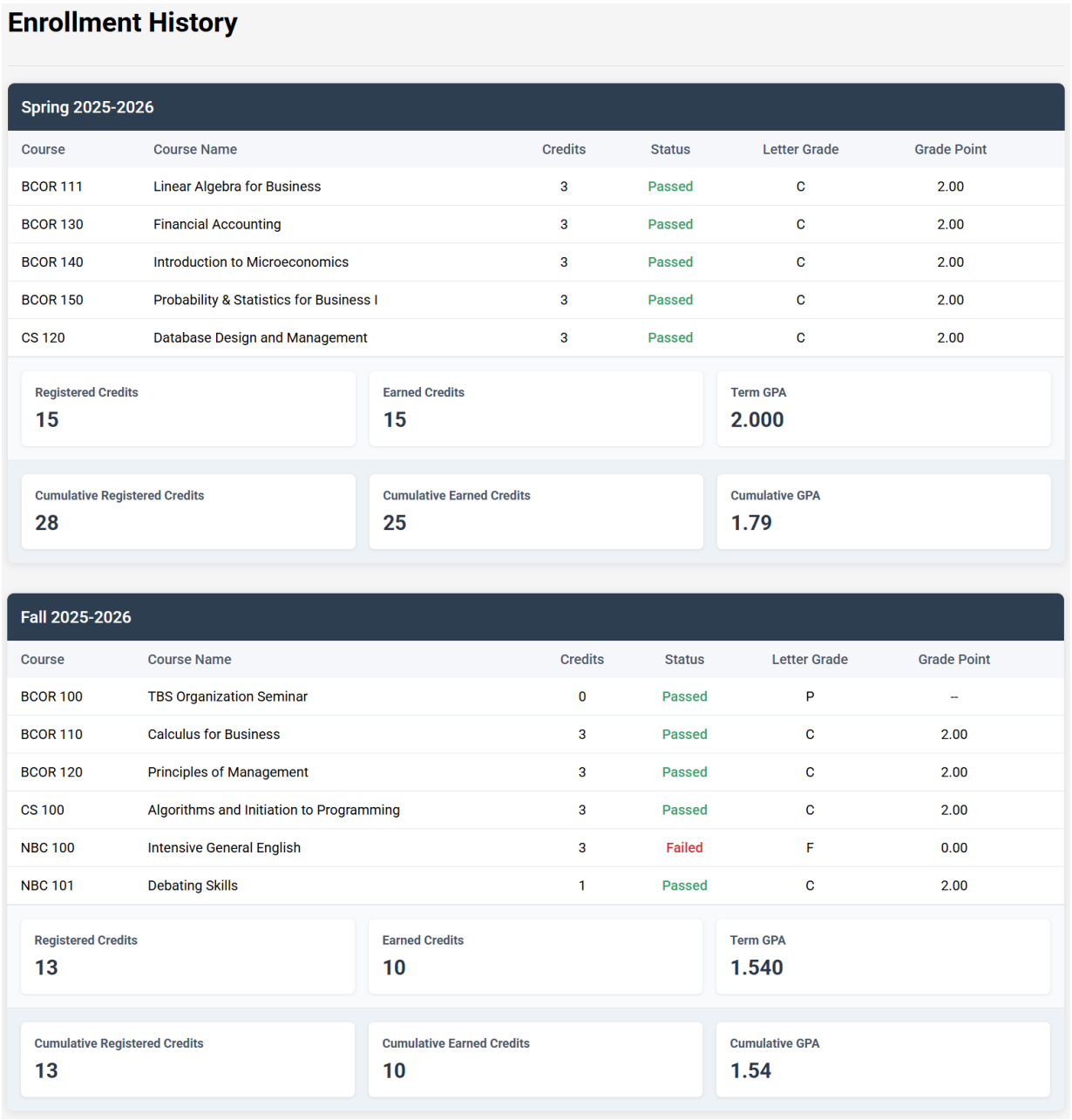


Figure 3.30: Enrollment History Section

- To allow students to apply for the forgiveness policy when qualified:
  The student can request for a course to be forgiven in accordance with the rules outlined in the official handbook. A course is eligible for forgiveness when the heart symbol appears next to it.

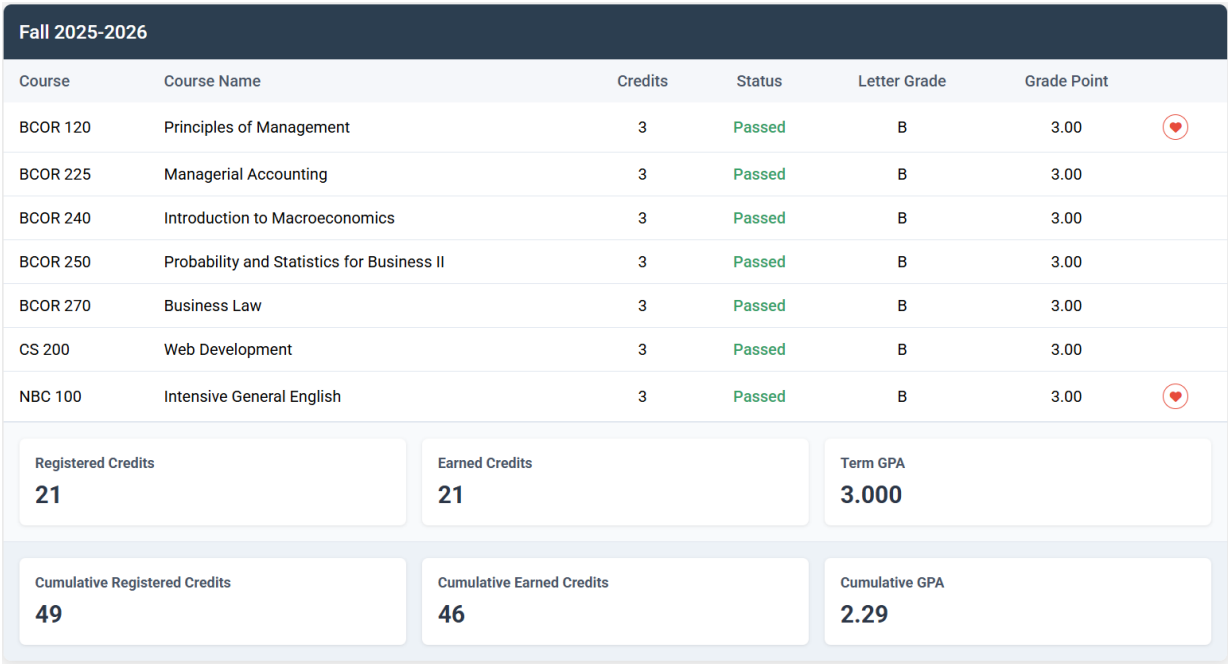


Figure 3.31: Forgivable Courses

An example of a forgiveness request is depicted in Figure 3.32. There, the student can see the changes that would occur if the forgiveness is accepted, allowing them to clearly understand the extent of the improvement.

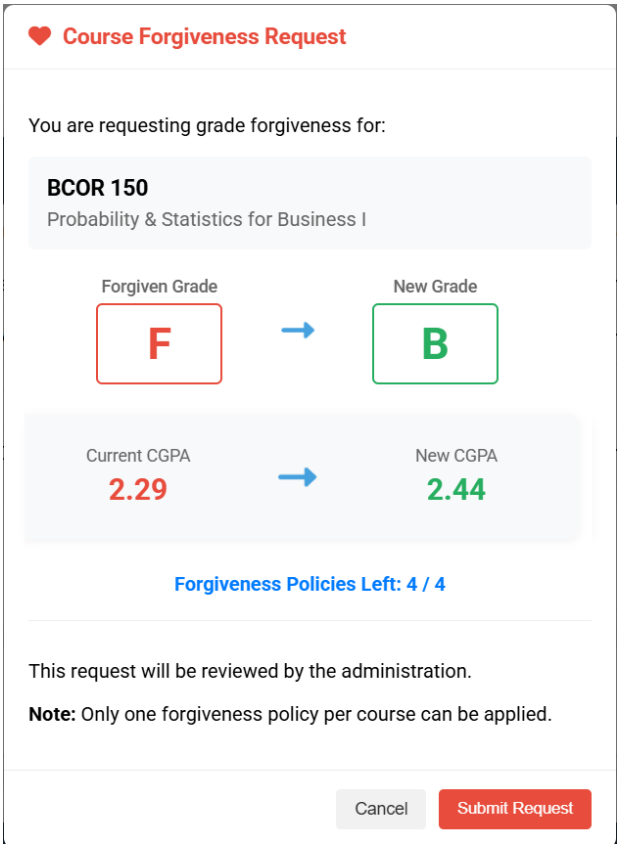


Figure 3.32: Forgiveness Request Form

3.3.4 Major/Minor Section

In this section, the student can keep track of the requirements for picking his major in the future. The numbers are updated every time a new grade is updated or inserted in the database. This provides clear visibility into how close they are to meeting their target major requirements. Students can make more informed decisions, set realistic goals, and plan their course selections strategically. It encourages proactive academic planning and minimizes surprises during the major selection period.

Illustrated in Figure 3.33, the three requirements : CGPA, earned credits from Freshman and Sophomore years, and specialized GPAs are displayed. Once these are fulfilled, the student can choose from the eligible options.

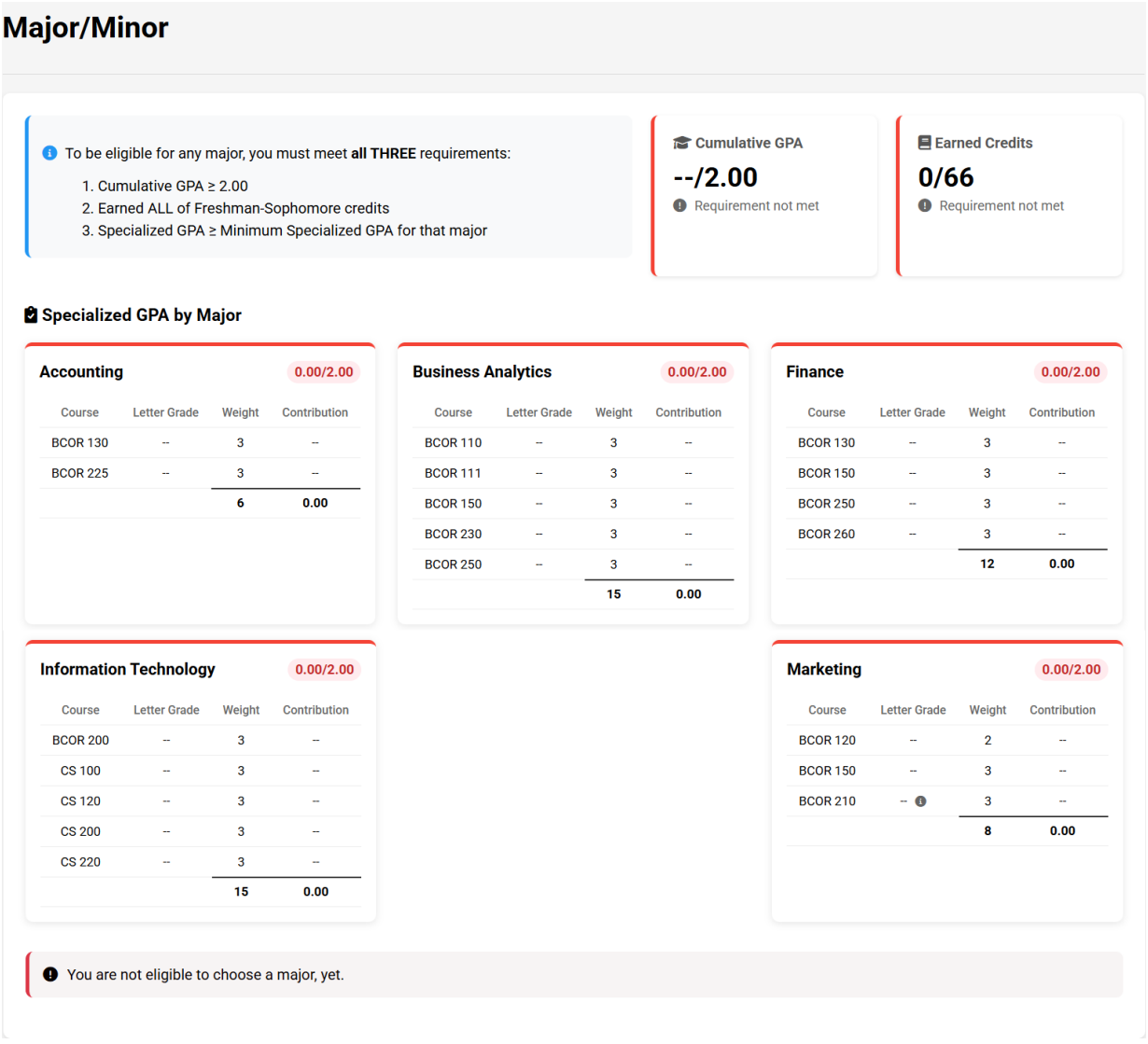


Figure 3.33: Major/Minor Section

Once the student becomes eligible to select a major and the selection window is open, as shown in Figure 3.34, they can choose from the list of valid options. The student cannot select combinations that have been rejected by the administration.

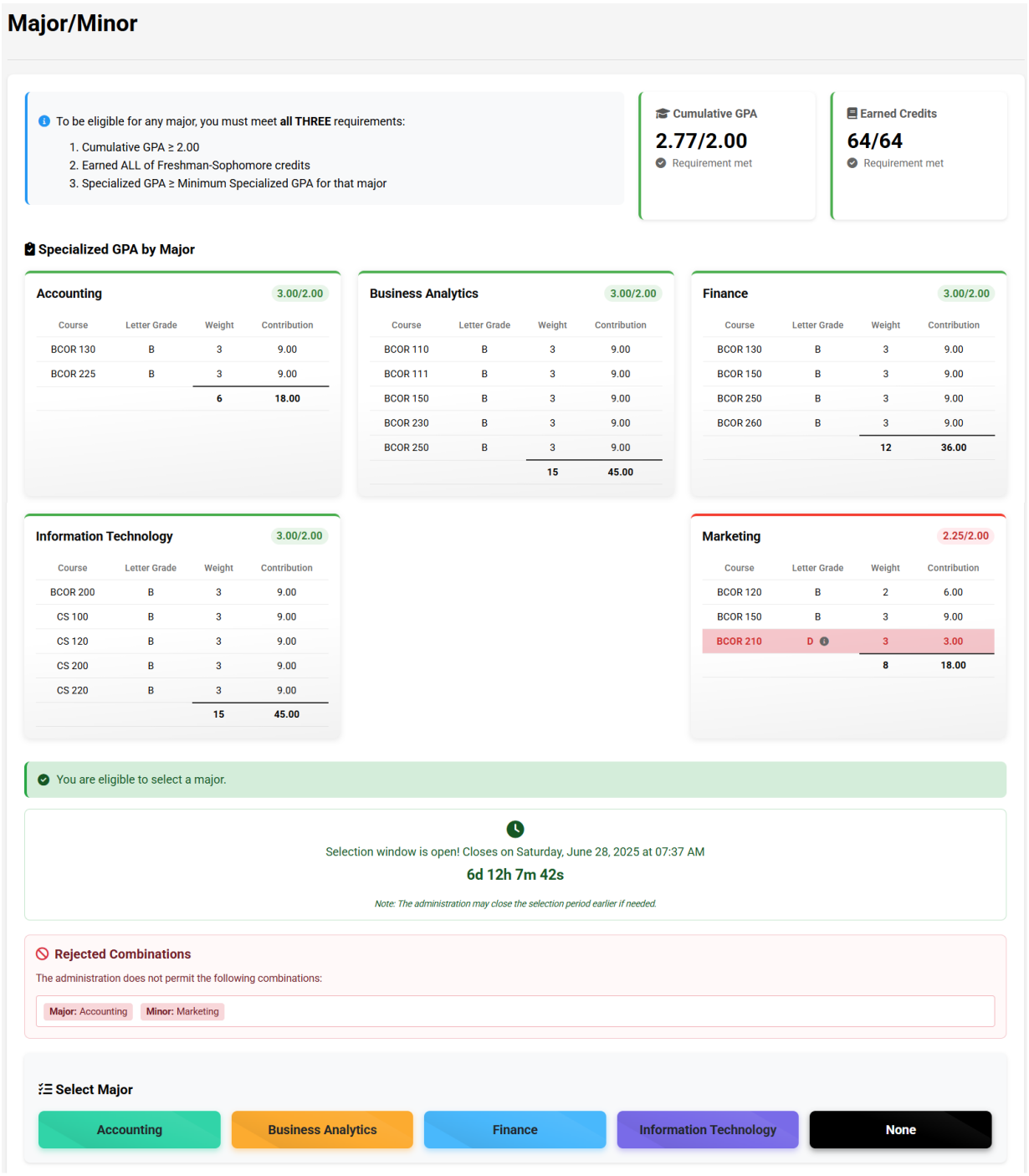


Figure 3.34: Eligible Major Pick Interface

The student is presented with the option to pick a major/minor combination , a double major combination, and "None" as presented in Figure 3.35. The "None" option is meant for those who have yet to unlock their desired major, giving them the chance to fulfill more requirements before picking again. Once the student confirms he choices, a request will be sent to the administration for validation.

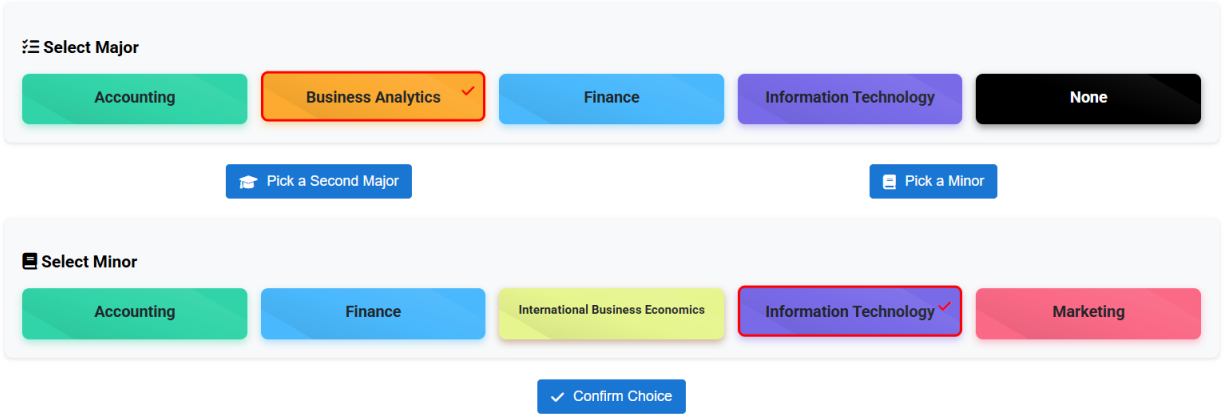


Figure 3.35: Major/Minor Combination Selection Interface

Figure 3.36 illustrates the user interface variations corresponding to each major request status: Pending, Approved, and Rejected. The student is free to alter his selection as long as the specialization selection window is open.

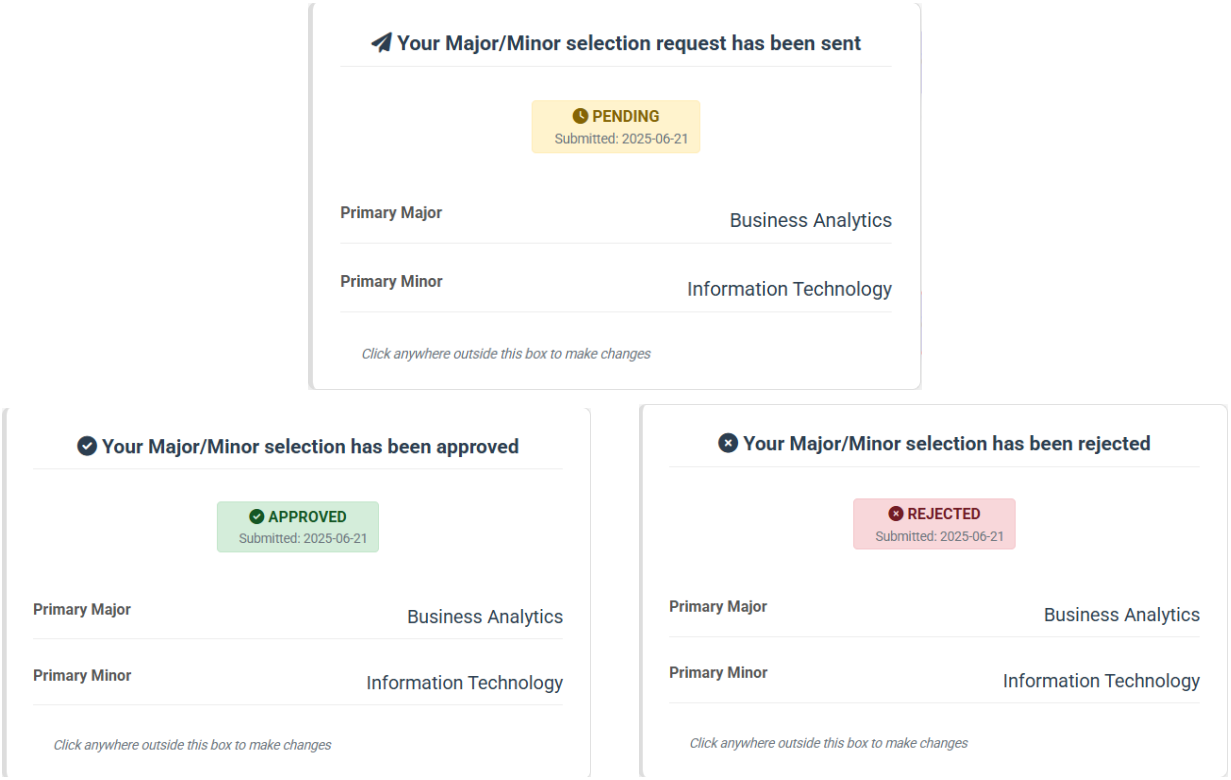


Figure 3.36: Specialty Request Status

3.3.5 Schedule

Similarly to the Course Enrollment section, the content and available actions here depend on whether the course registration period is open. If it is closed, the student can only review their saved schedule and enrolled courses for the current semester, as shown in Figure 3.37.

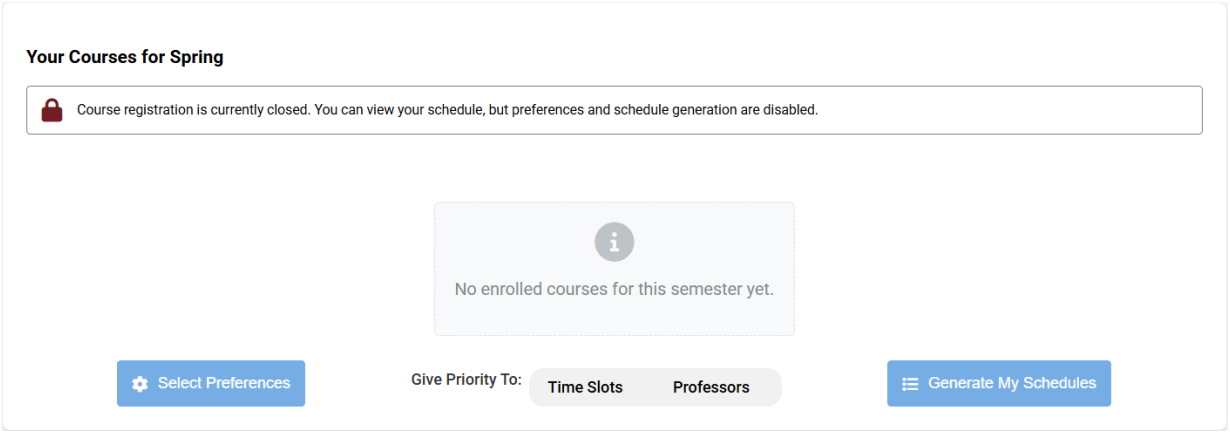


Figure 3.37: Schedule Section When Registration Is Closed

But once the registration period is open, the student begins by setting their scheduling preferences, including preferred professors and time slots, as well as the relative priority between them. Based on these inputs, the system generates one or more optimized schedules that strictly respects the predefined constraints, while aiming to maximize the overall satisfaction score derived from the student's preferences. At this stage, the student also validates the specific study groups they will be enrolled in for each course, thereby finalizing the course registration process for the semester.

Requirements

For the model to function correctly, several preparations are required. The official schedules must first be inserted into the database, thoroughly cleaned, and properly indexed in a way that makes filling to the sets previously defined in our model easier. These preparations ensure that the optimization process runs smoothly and produces valid, conflict-free schedules. Indexing was done either as **absolute**, where each unique identifier receives a unique index (for example, the time slots field), or as **dependent indexing**, where indexes reset when the dependency changes. For example, professor indexes reset per course and session type. This is not a problem since the logic does not require comparing professors across courses, and it provides a cleaner display when testing.

course_code	week_day	start_time	end_time	professor	classroom	group	session_type	course_index	session_number_index	time_slot_index	professor_index	Tutprof_index	lect_prof_index
BCOR 140	Friday	08:30:00	10:00:00	N. Khraief	A5	F.1	lecture	3	1	21	2	NULL	2
BCOR 140	Friday	10:00:00	11:30:00	N. Khraief	A5	F.1	lecture	3	2	22	2	NULL	2
BCOR 150	Friday	11:30:00	13:00:00	A. Dridi	A6	F.1	lecture	4	2	23	1	NULL	1
BCOR 130	Friday	13:30:00	15:00:00	Nejia Moumen	A6	F.1	lecture	2	1	24	4	NULL	4
BCOR 130	Friday	15:00:00	16:30:00	Nejia Moumen	A6	F.1	lecture	2	2	25	4	NULL	4
BCOR 111	Friday	08:30:00	10:00:00	I. Rassas	A8	F.2	lecture	1	1	21	3	NULL	3
BCOR 111	Friday	10:00:00	11:30:00	I. Rassas	A8	F.2	lecture	1	2	22	3	NULL	3
BCOR 130	Friday	13:30:00	15:00:00	Nejia Moumen	A9	F.2	lecture	2	1	24	4	NULL	4
BCOR 130	Friday	15:00:00	16:30:00	Nejia Moumen	A9	F.2	lecture	2	2	25	4	NULL	4
NBC 130	Friday	08:30:00	10:00:00	Ben alaya K.	S10	F.3	lecture	12	1	21	1	NULL	1
BCOR 111	Friday	10:00:00	11:30:00	I. Khemir	A4	F.3	lecture	1	1	22	2	NULL	2

Figure 3.38: Schedule table sample

Preferences

The student is provided with an interface to set their preferences. By clicking the *Select Preferences* button a popup with two options will open:

- **Professors:** The student can rank professors for each session type of each course as presented in Figure 3.39. Alternatively, they may check the *No Preferences* box if they are indifferent to the options. Higher-ranked professors will be assigned greater weights and if there is no specific preference, a weight of 1 will be assigned. The selection is finalized by clicking the *Save Professor Preferences* button.

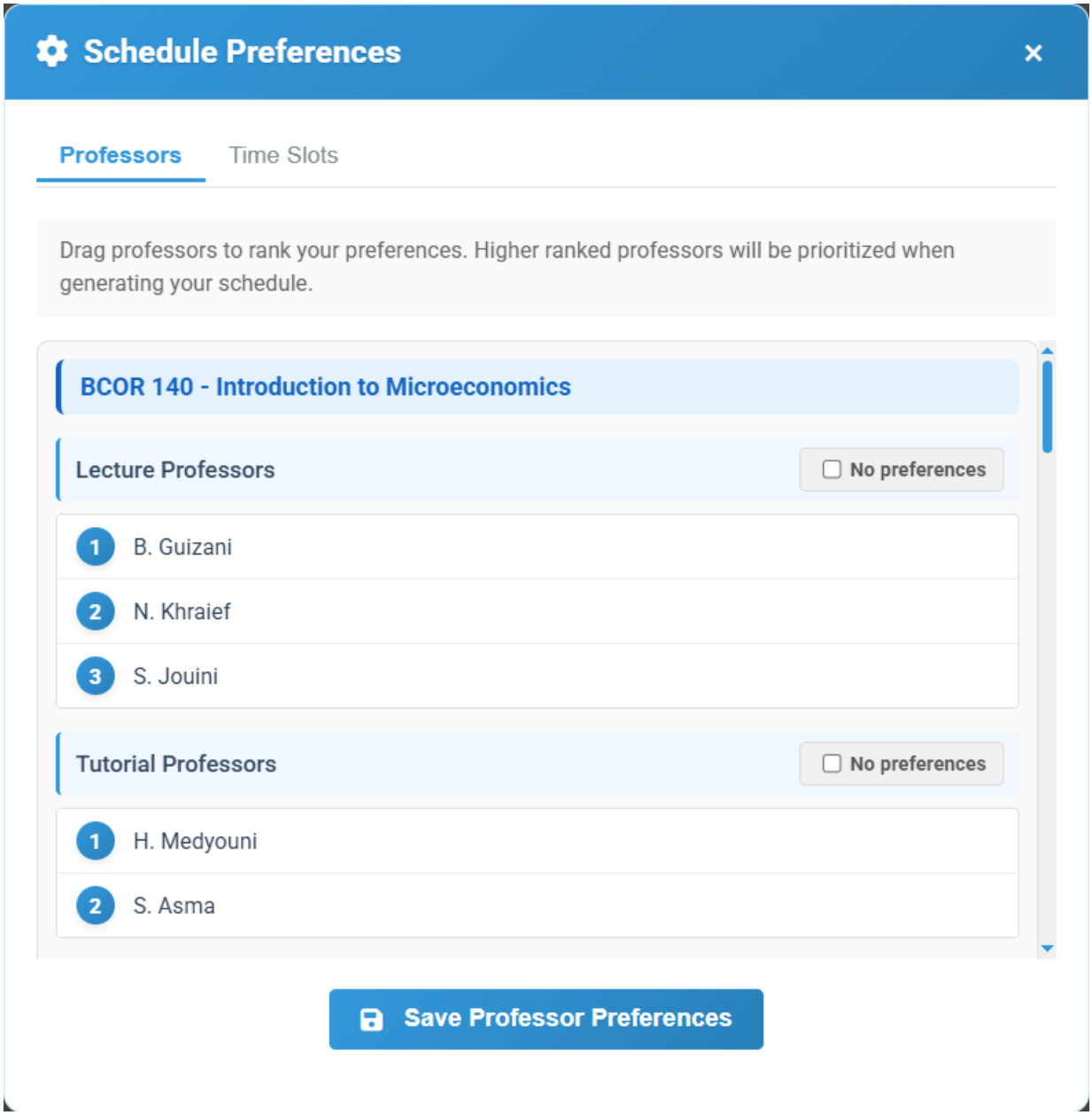


Figure 3.39: Professor Preferences Input

- **Time Slots:** The student can select preferred time slots by clicking individual cells, or entire rows or columns to select all slots in that row or column just as portrayed in Figure 3.40. Favorable time slots are assigned a varying weight β that depends on which type of preference will be given priority, while non-favorable ones receive a weight of 0. Preferences are confirmed by clicking the *Save Time Slots Preferences* button.

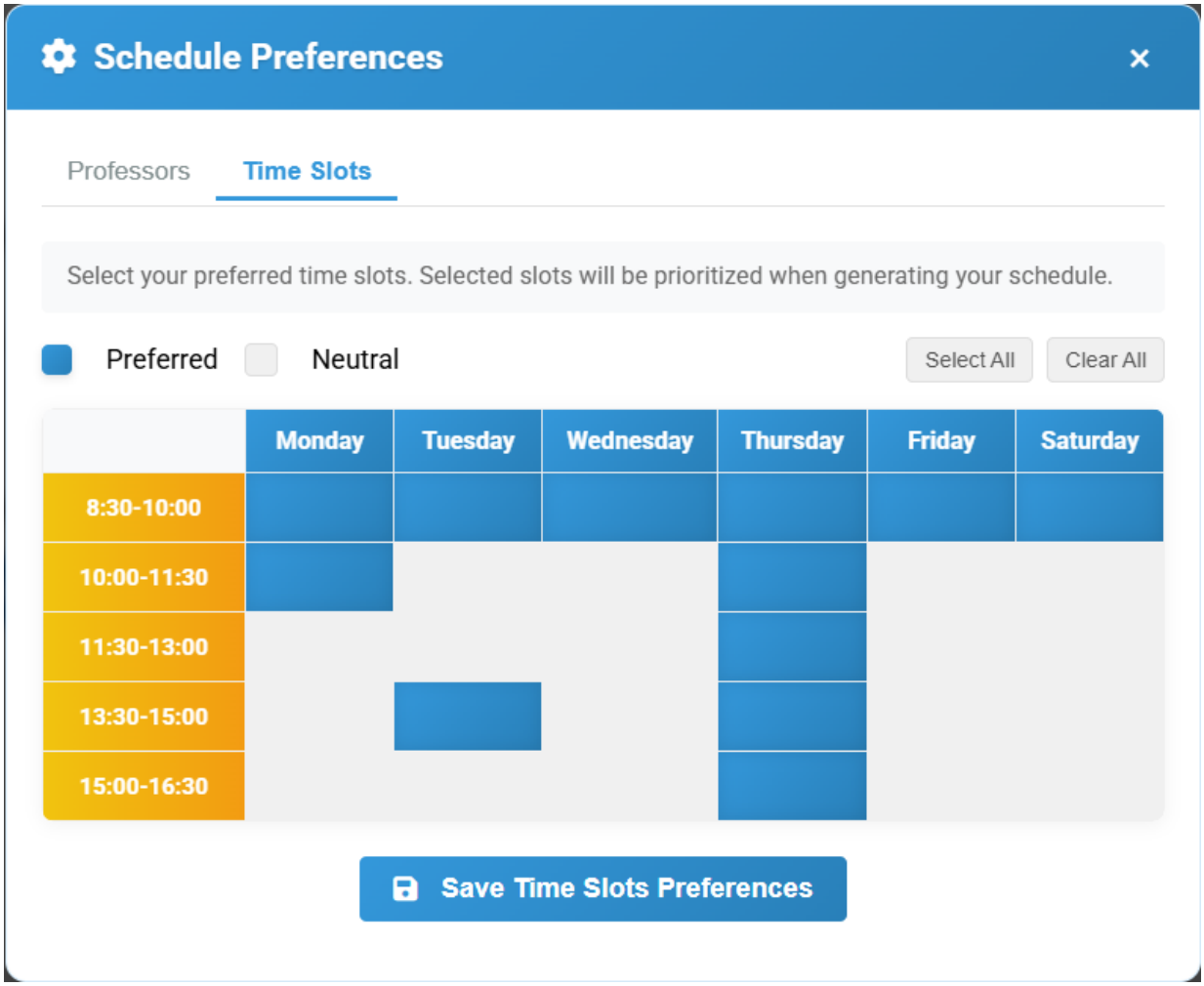


Figure 3.40: Student Time Slots P references

After setting their preferences, the student must specify which preference category takes priority: **professor preferences** or **time slot preferences**. This selection determines the value of β assigned in the optimization model.

Following the current settings, if time slot preferences are prioritized, each preferred slot receives a large weight of 10, a sufficiently large value to outweigh any professor related weight value. On the other hand, if professor preferences are prioritized, each preferred time slot is given a β value of 1, making it equivalent to the least desired professor in any ranking.

Schedule Generation

Once the student clicks *Generate My Schedules* button, a backend function is triggered to compute all optimal solutions. This function runs the model to find an initial solution, stores it in a dictionary, and then adds a constraint to the model to exclude that solution from being generated again. This is achieved by summing the active decision variables from the previous solution and constraining their total to be less than before. The process iterates until one of the following conditions is met: no further feasible solutions exist, the maximum number of allowed solutions is reached, or the time limit for model execution is exceeded.

During this computation, a loading screen is displayed for up to 10 seconds, as illustrated in Figure 3.41. In practice, the waiting time may be significantly shorter (as little as one second), depending on the number and complexity of the imposed constraints.

Concurrently, an additional task is performed to format the raw solution into a readable and intuitive structure. Instead of returning indexed decision variables from the model, the system retrieves relevant information from the database, such as classrooms, groups, and professor names to construct complete and user-friendly schedule options.

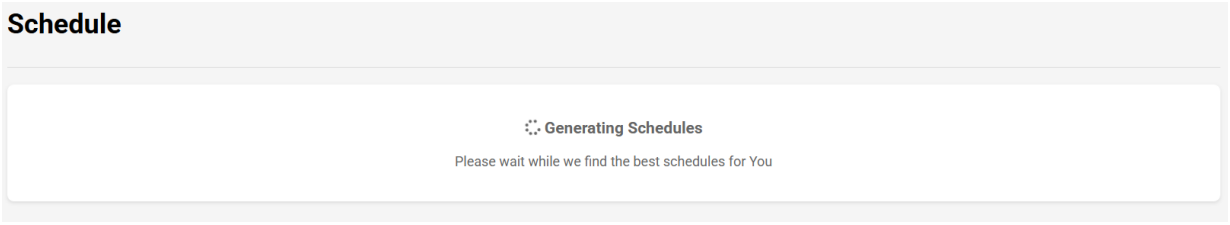


Figure 3.41: Loading Screen During Schedules Generation

Once generation is complete, all solutions with the same satisfaction score are numbered and presented to the student for review as visualized in Figure 3.42. To conserve resources and avoid overwhelming the user, a maximum of 10 solutions is displayed. If too many solutions are returned, the student is encouraged to refine their preferences to narrow down the results.

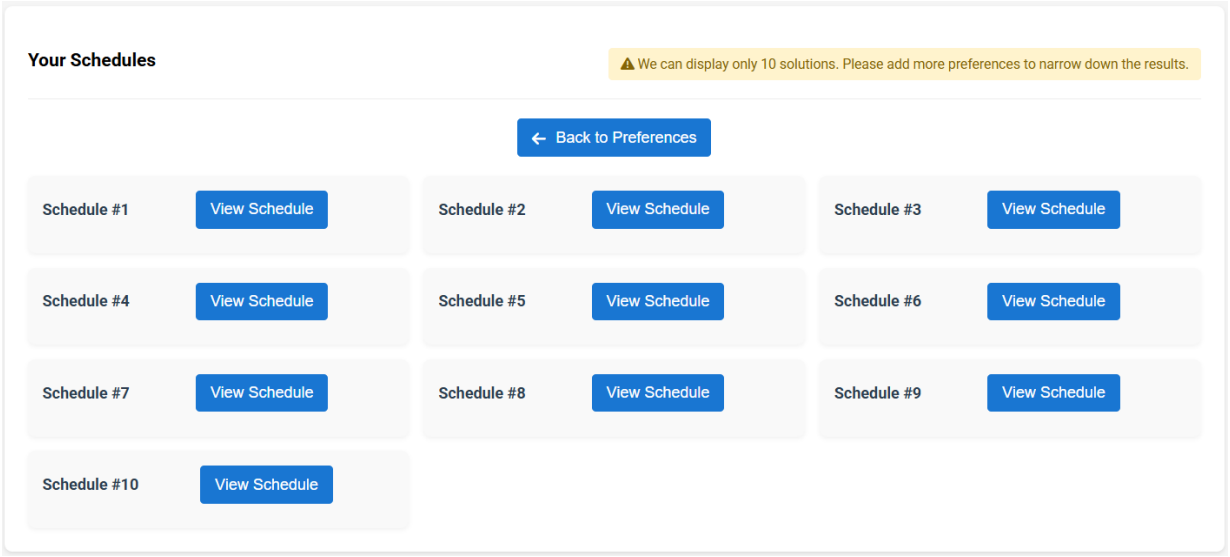


Figure 3.42: Example of Broader Results With Fewer Preferences

After adjusting the preferences to be more precise, the results are narrowed down in Figure 3.43:

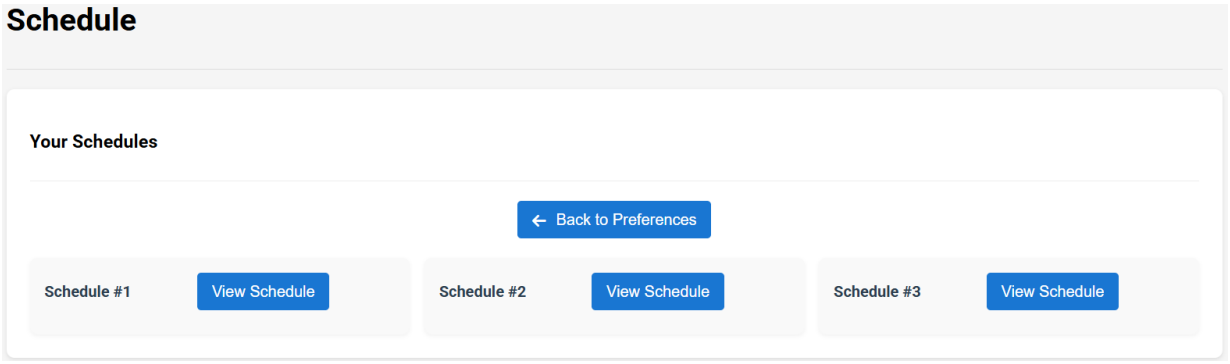


Figure 3.43: Same Example but Using More Preferences

Schedule Selection

The student can browse through the proposed schedules one at a time using a clean and intuitive interface. The main components of the interface of Figure 3.44:

- *Back to All Solutions* button: Returns the student to the previous interface displaying all solutions.
- *Previous* and *Next* buttons: Allow navigation between the generated schedules.
- *Download Schedule* button: Enables downloading the schedule as a PDF, with options such as landscape orientation or black-and-white format.
- *List of Enrolled Courses*
- *Color-Coded Timetable*: Provides a visual representation of the schedule, showing all relevant session information.
- *Select Schedule* button: Finalizes the selection and enrolls the student in the specified groups for each course.

Although all generated schedules are theoretically equally satisfactory from the model's perspective, the student may still evaluate them based on additional informal preferences not captured in the model, such as the desire to enroll in certain groups to study with friends or the preference for specific classrooms related to personal reasons. The student is also free to go back and modify their preferences to regenerate new schedules, or even change their course selection entirely,if permitted, in the **Course Enrollment** section.

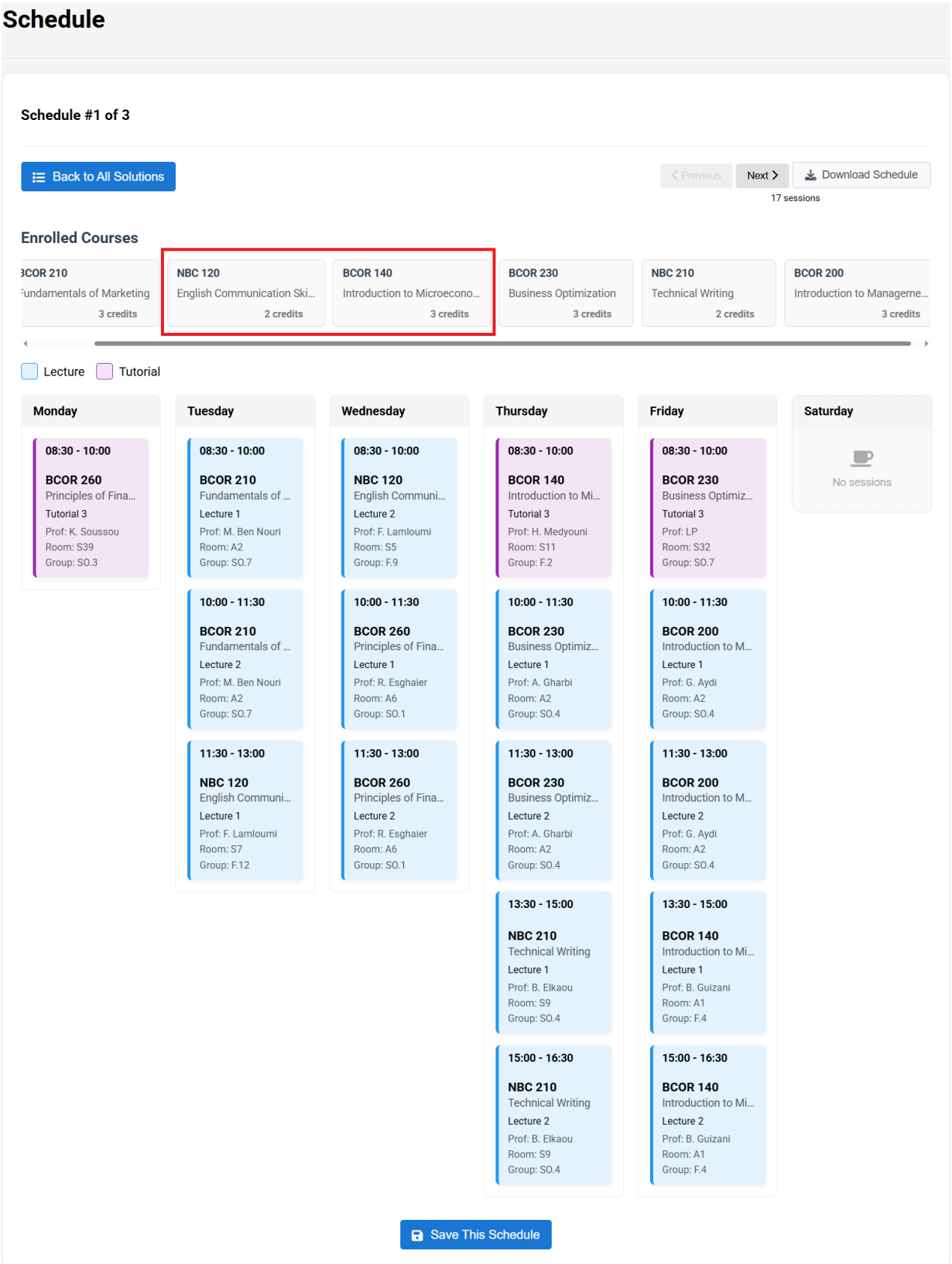


Figure 3.44: A Generated Schedule for a Freshman Student

Note that even though there are enrolled courses from the Sophomore level, this is still a Freshman student, hence the presence of year one courses in the red rectangle.

Testing Other Possibilities

We begin with the case depicted in Figure 3.44 , and rerun the model for a Sophomore student with no failed nor skipped courses. This implies that all current semester courses will be automatically selected, leaving no real decision-making to the student.

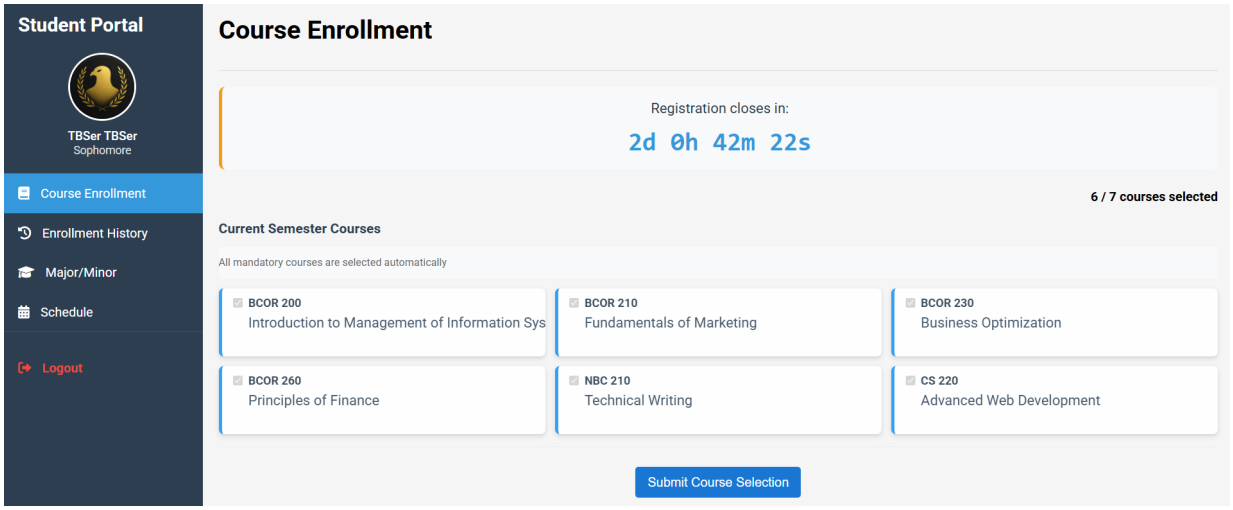


Figure 3.45: Course Enrollment for a Sophomore Student

The student confirms the selection of courses, indicating that at least one valid schedule can be generated.

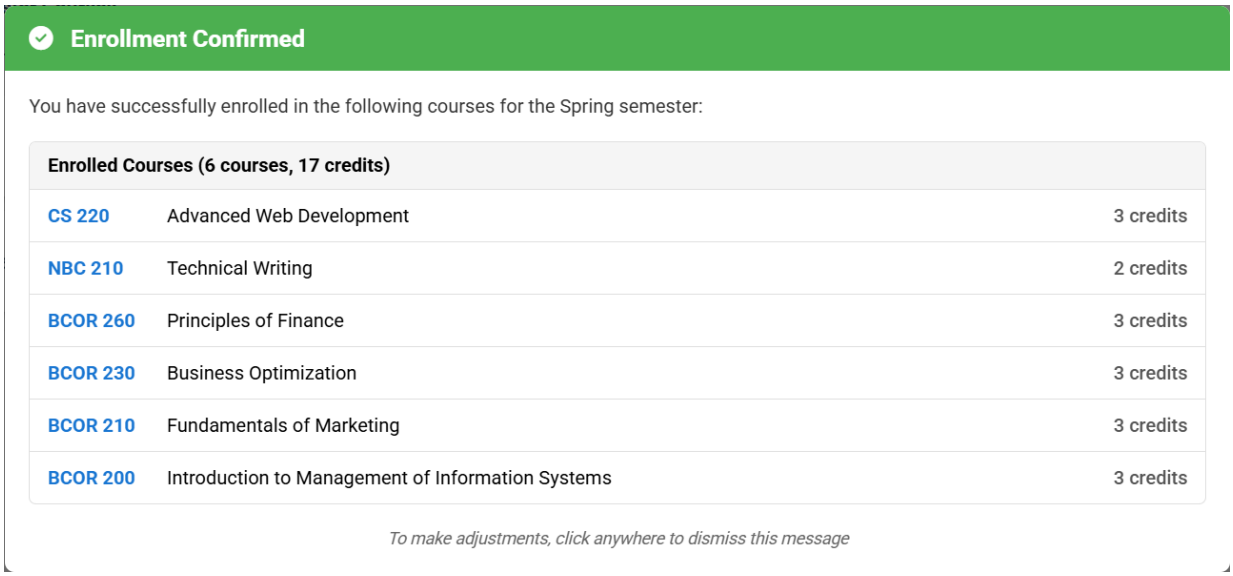


Figure 3.46: Confirmation of Enrollment

No professor preferences are specified in this case.

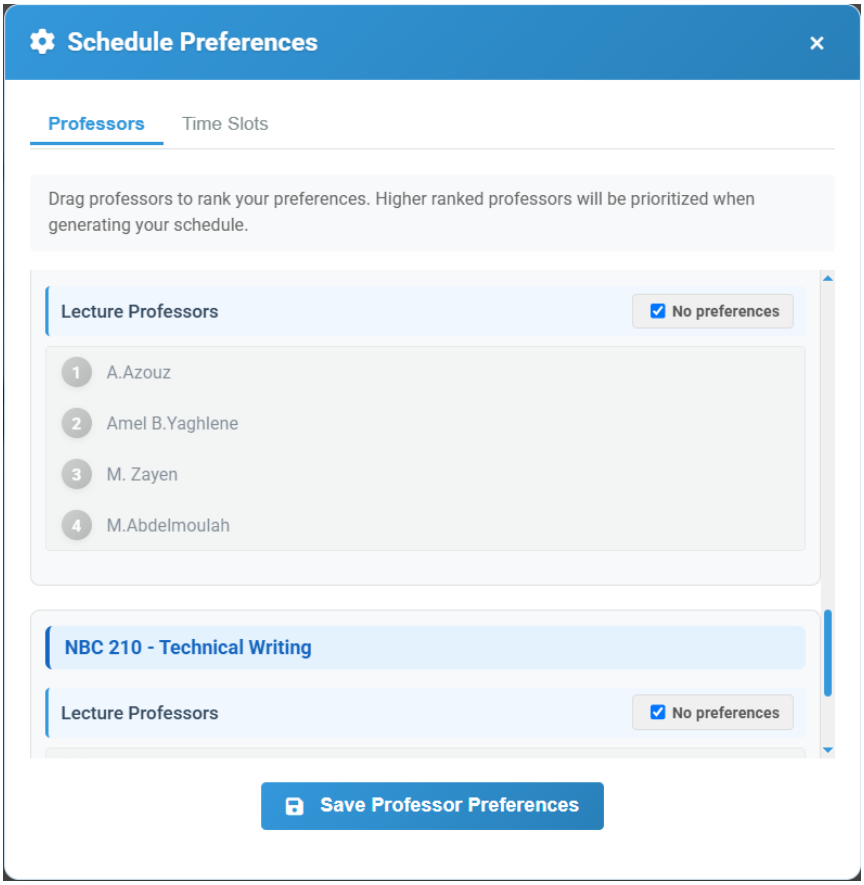


Figure 3.47: Schedule Generation with no professor preferences

Time slot preferences are also kept limited, with only the 8:30 A.M. sessions marked as favorable.

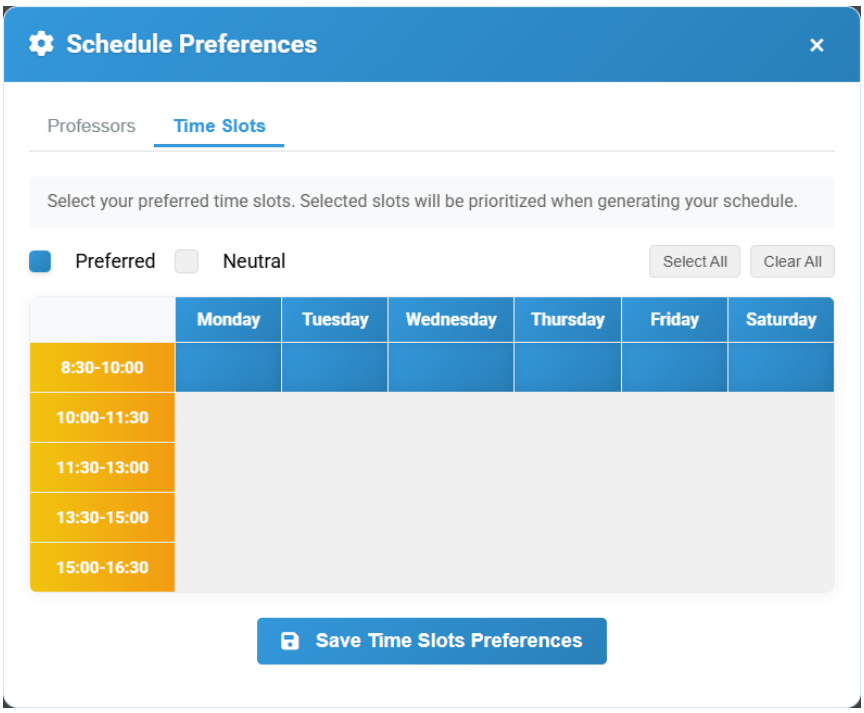


Figure 3.48: Limited Time Slot Preferences

Finally, the student sets priority to time slots and generate the solutions. Shown in Figure 3.49 and Figure 3.50, only one solution was found, and it did not satisfy any of the student's time slot preferences and even contained 2 gap sessions. This outcome is the result of the restrictions of Equations (2.13) and (2.14) in our scheduling model, which becomes enforced once the sets `FixSlot` and `FixProf` are populated. In fact, these sets are filled after verifying that the student has no currently failed nor skipped courses. At that point, a function is triggered to match the student's group with all sessions in the database that correspond to that same group. The relevant indexes are then retrieved and inserted into the fixed sets.

This mechanism effectively forces the student to enroll exclusively in their officially assigned group, limiting the flexibility of the scheduling model and potentially overriding time slot preferences but it is in accordance with the academic regulations and rules.

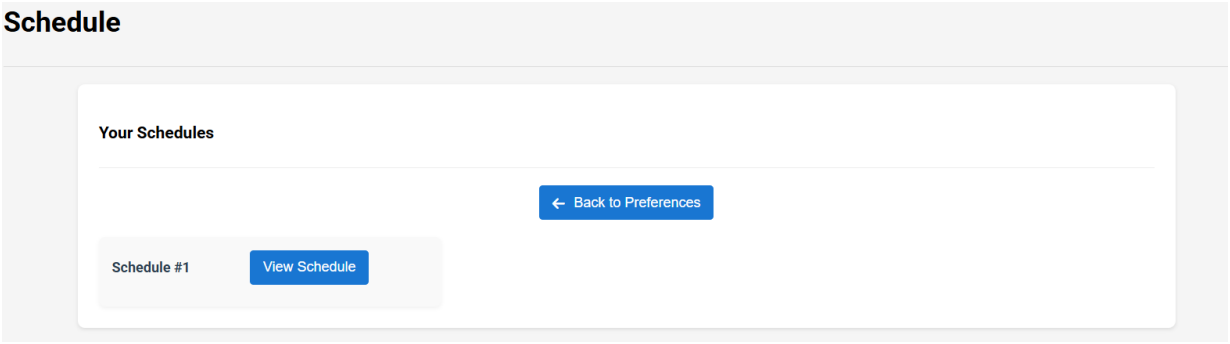


Figure 3.49: Only One Valid Schedule Found

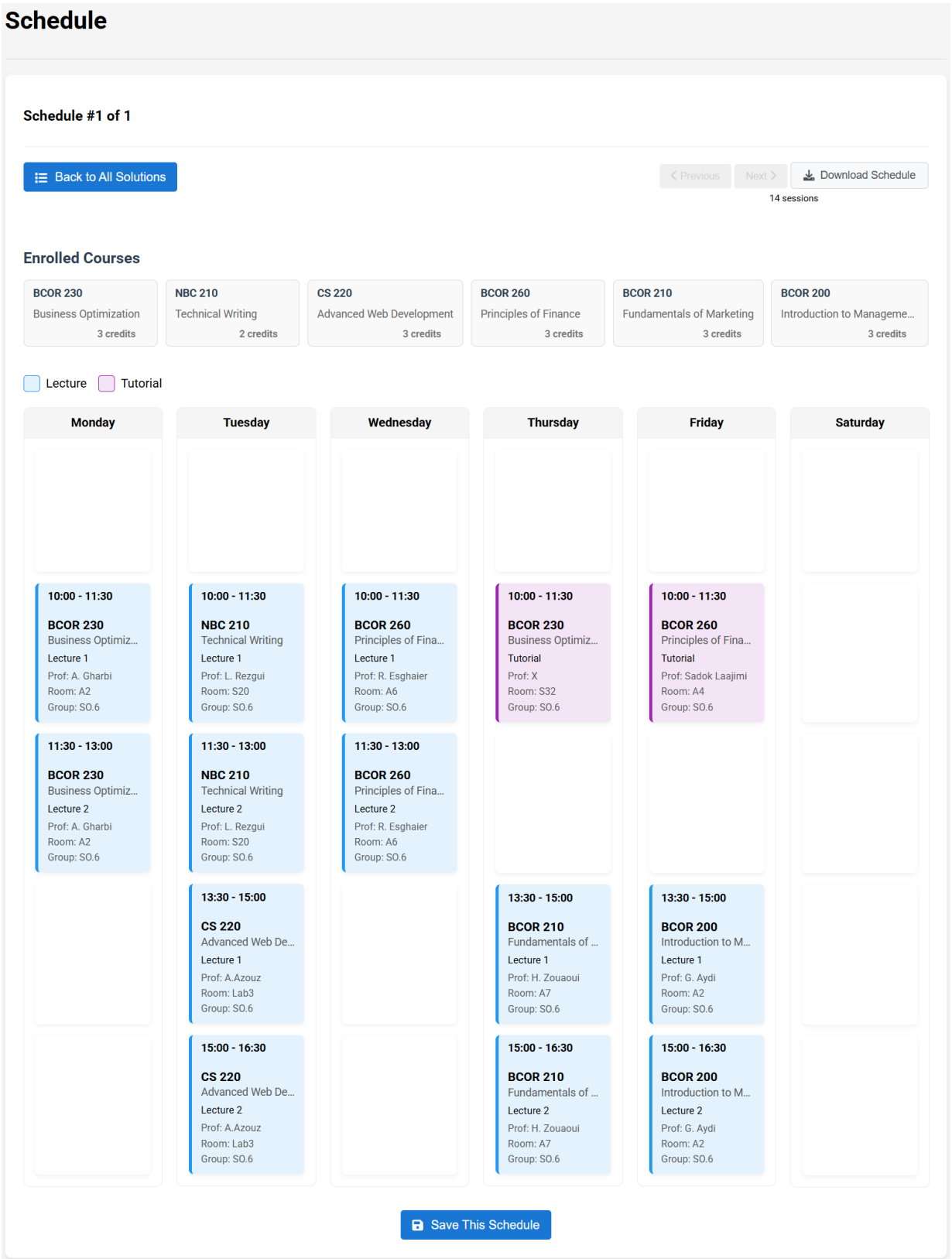


Figure 3.50: Schedule of a Sophomore student

Another case where this constraint is applied is when a Junior or Senior student is repeating a failed course. In such scenarios, the course must be repeated with a group that matches the student's major/minor selection. In Figure 3.51 we consider the case of a Junior student with a

major In Business Analytics and a minor in Information Technology.

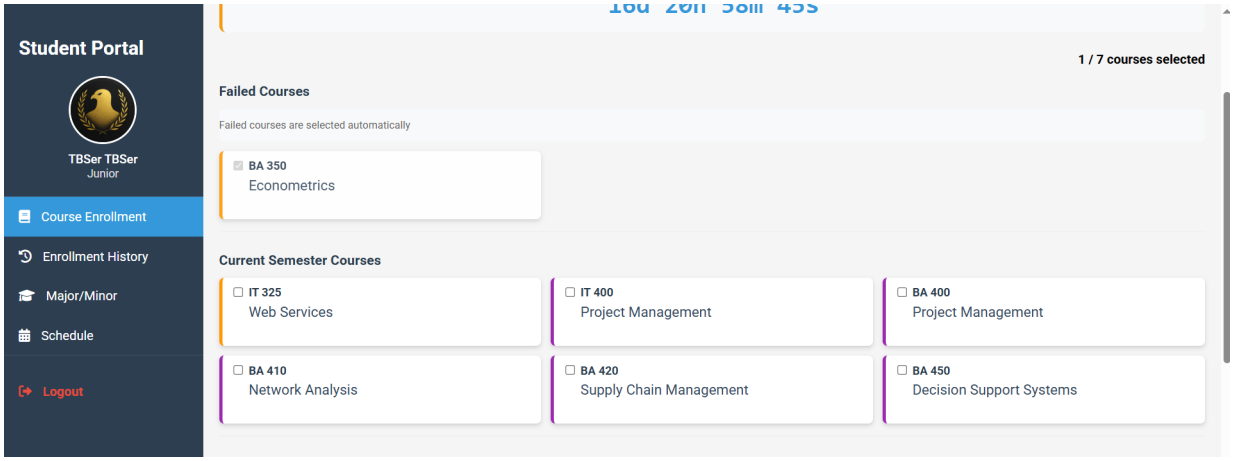


Figure 3.51: Case: a Junior Student With a Failed Course

We will do a simple test by enrolling in the only failed course "Econometrics" as show in Figure 3.52.

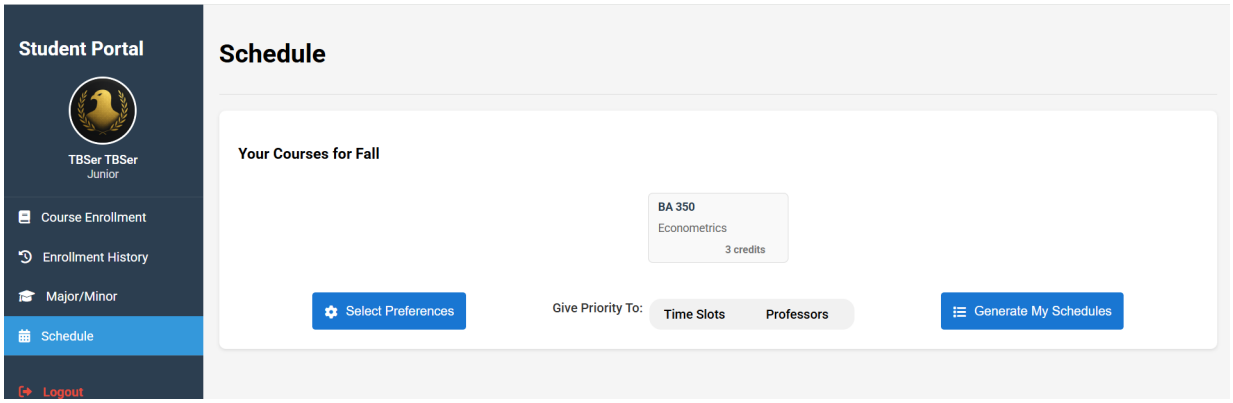
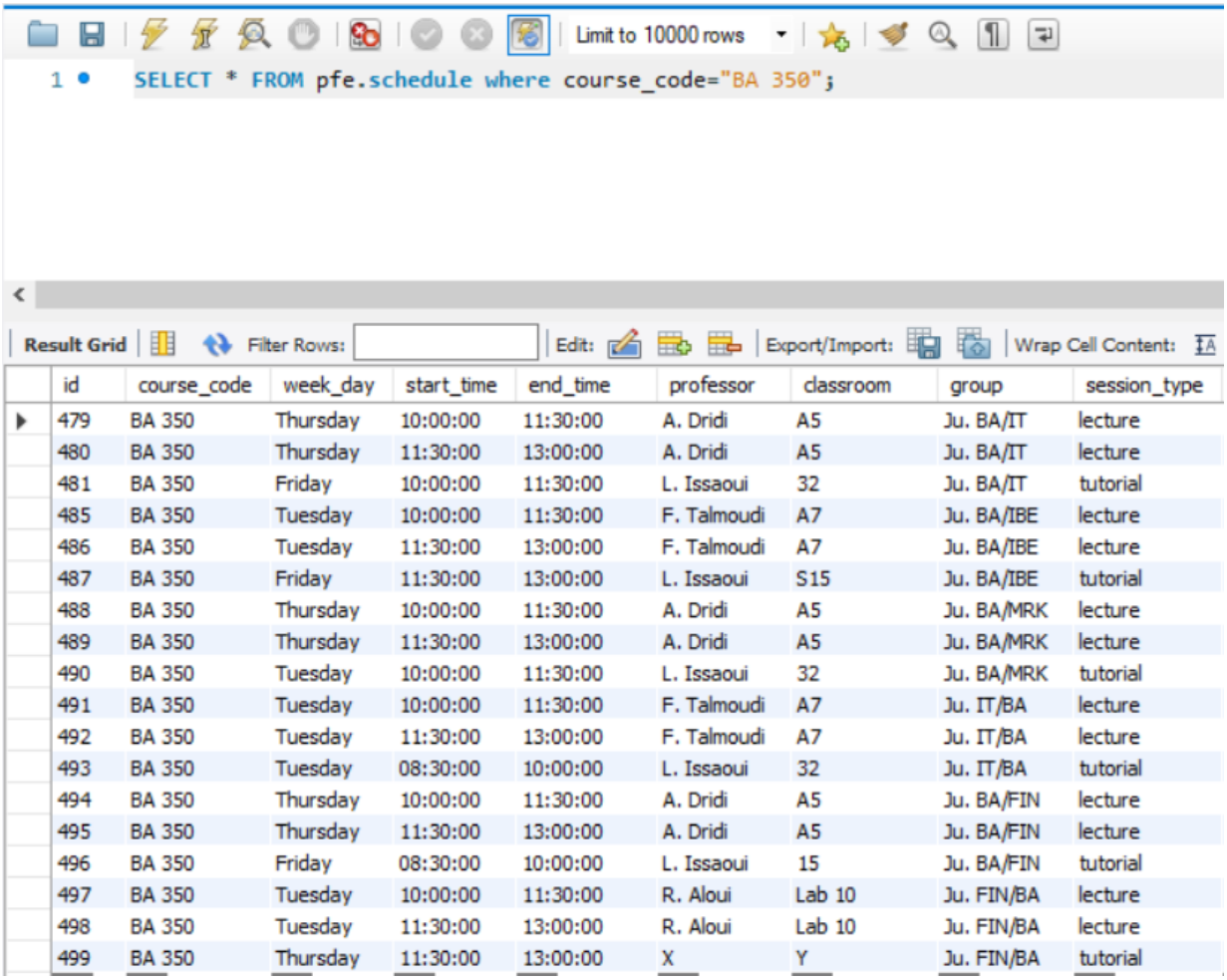


Figure 3.52: Schedule Interface For Preferences Selection

Before generating the solution, we will check the existing available sessions for the Econometrics course with a simple SQL query.



```
SELECT * FROM pfe.schedule where course_code="BA 350";
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	course_code	week_day	start_time	end_time	professor	classroom	group	session_type
479	BA 350	Thursday	10:00:00	11:30:00	A. Dridi	A5	Ju. BA/IT	lecture
480	BA 350	Thursday	11:30:00	13:00:00	A. Dridi	A5	Ju. BA/IT	lecture
481	BA 350	Friday	10:00:00	11:30:00	L. Issaoui	32	Ju. BA/IT	tutorial
485	BA 350	Tuesday	10:00:00	11:30:00	F. Talmoudi	A7	Ju. BA/IBE	lecture
486	BA 350	Tuesday	11:30:00	13:00:00	F. Talmoudi	A7	Ju. BA/IBE	lecture
487	BA 350	Friday	11:30:00	13:00:00	L. Issaoui	S15	Ju. BA/IBE	tutorial
488	BA 350	Thursday	10:00:00	11:30:00	A. Dridi	A5	Ju. BA/MRK	lecture
489	BA 350	Thursday	11:30:00	13:00:00	A. Dridi	A5	Ju. BA/MRK	lecture
490	BA 350	Tuesday	10:00:00	11:30:00	L. Issaoui	32	Ju. BA/MRK	tutorial
491	BA 350	Tuesday	10:00:00	11:30:00	F. Talmoudi	A7	Ju. IT/BA	lecture
492	BA 350	Tuesday	11:30:00	13:00:00	F. Talmoudi	A7	Ju. IT/BA	lecture
493	BA 350	Tuesday	08:30:00	10:00:00	L. Issaoui	32	Ju. IT/BA	tutorial
494	BA 350	Thursday	10:00:00	11:30:00	A. Dridi	A5	Ju. BA/FIN	lecture
495	BA 350	Thursday	11:30:00	13:00:00	A. Dridi	A5	Ju. BA/FIN	lecture
496	BA 350	Friday	08:30:00	10:00:00	L. Issaoui	15	Ju. BA/FIN	tutorial
497	BA 350	Tuesday	10:00:00	11:30:00	R. Aloui	Lab 10	Ju. FIN/BA	lecture
498	BA 350	Tuesday	11:30:00	13:00:00	R. Aloui	Lab 10	Ju. FIN/BA	lecture
499	BA 350	Thursday	11:30:00	13:00:00	X	Y	Ju. FIN/BA	tutorial

Figure 3.53: BA 350 Course Sessions

Once the generation process is complete, the outcome is a single optimized schedule.

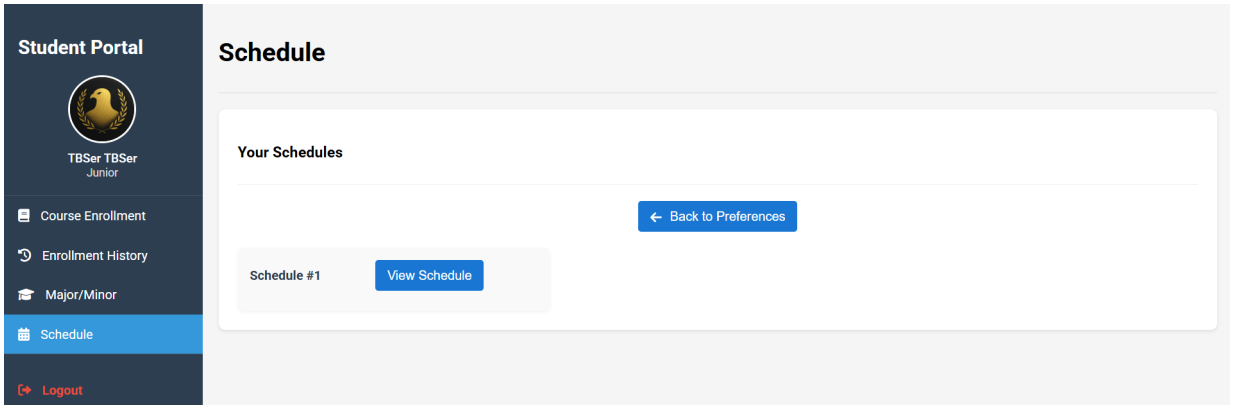


Figure 3.54: Successful Schedule Generation

Upon viewing the schedule, we can confirm that the sessions were indeed scheduled with the student's assigned group.

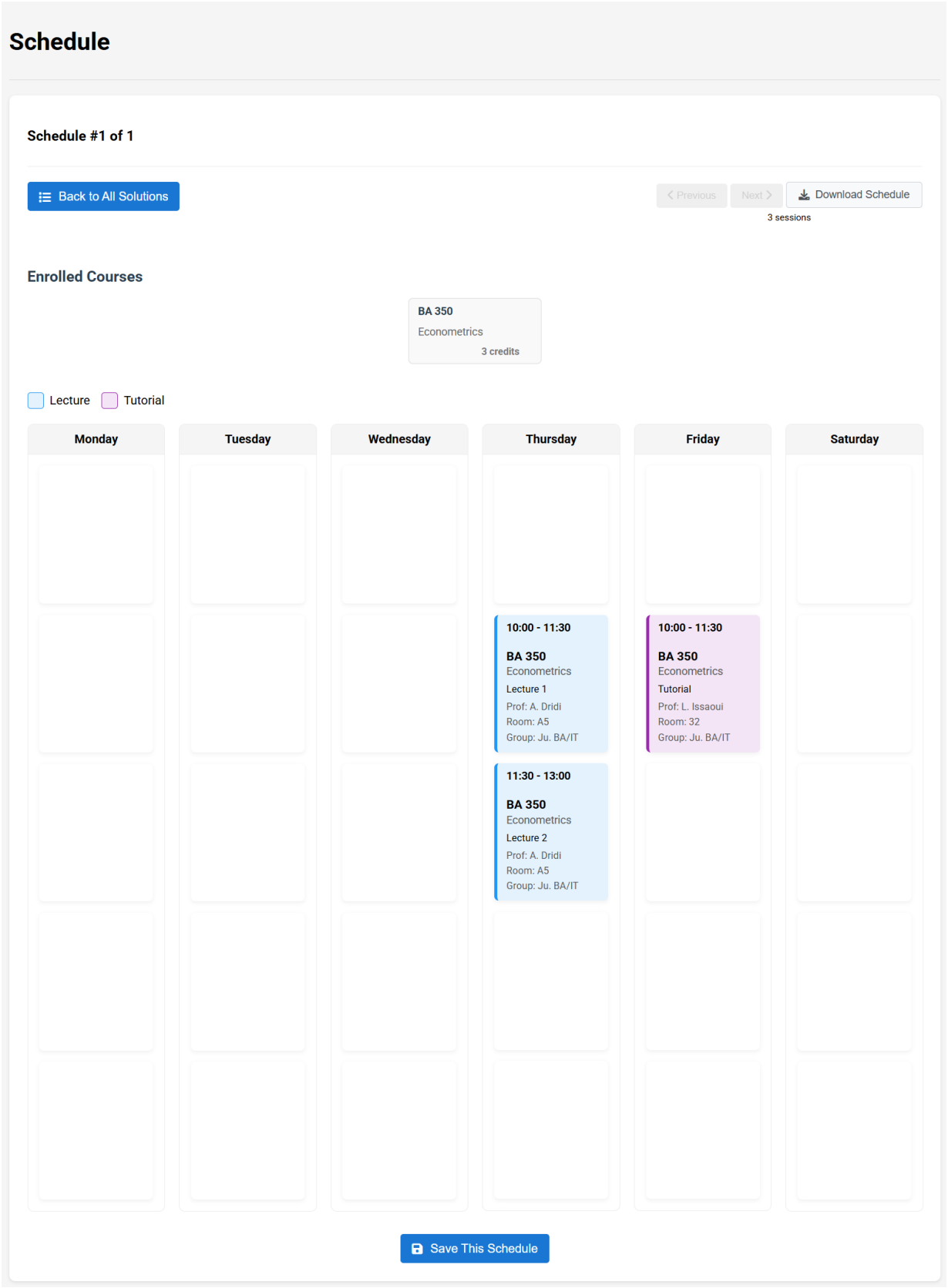


Figure 3.55: Optimized Schedule for a Junior Student With No Failed Courses

However, upon a closer examination of Figure 3.53, it becomes evident that three different groups (BA/IT, BA/MRK, and BA/FIN) that attend the same lecture sessions. This raises the

question: why was the group BA/IT specifically assigned?

The reason lies in a complementary function designed to prioritize the assignment of the group that matches the student's own group. This prioritization is applied only to sessions defined by the `FixSlot` and `FixProf` sets. Applying it selectively in this way prevents inconsistencies that could otherwise arise during schedule construction.

3.3.6 Makeup Session

This hidden section in the side menu depicted in Figure 3.68 becomes visible only when a makeup session period is initiated by the admin, specifically after a Spring semester ends and before a new Fall semester begins. It is available exclusively to students who are nearing graduation and have no more than two failed or skipped courses. In such cases, they become eligible to enroll in those courses during the makeup period.

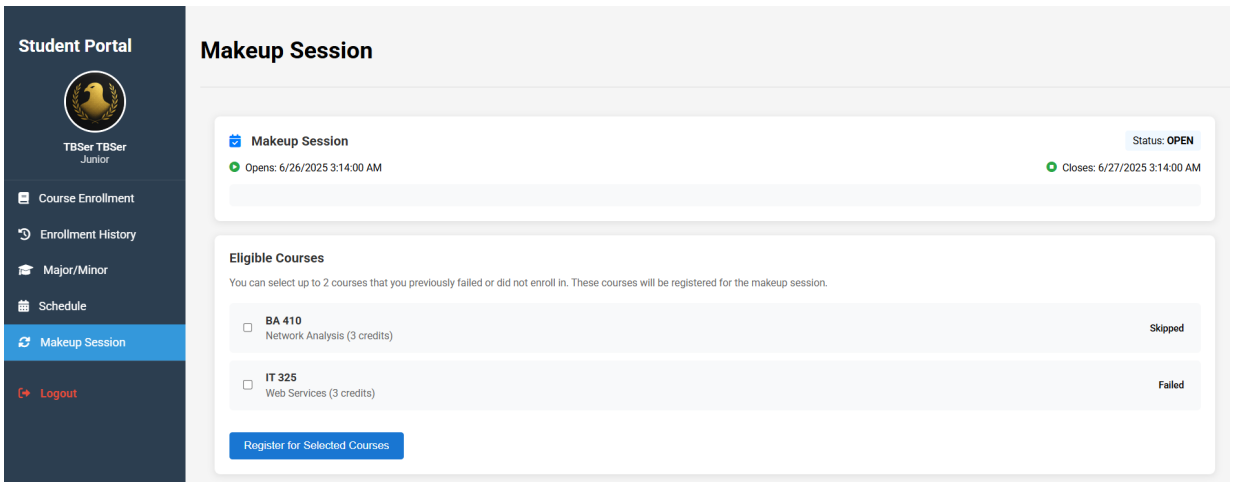


Figure 3.56: Makeup Session Interface

The student is free to enroll in these courses, even if they are failed courses or courses from different semesters.

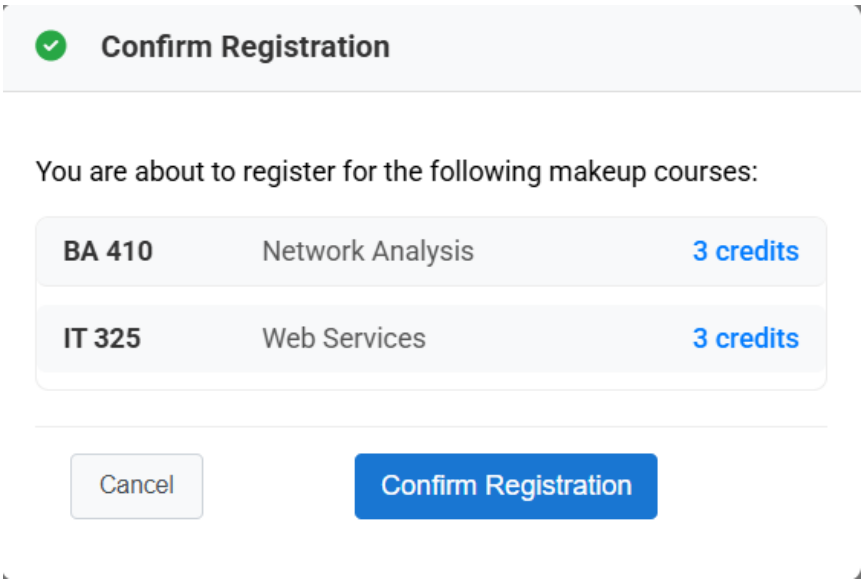


Figure 3.57: Confirm Registration Button

For the makeup session, course registration is straightforward which means there is no course selection validation by the schedule optimization model as shown in Figure 3.58

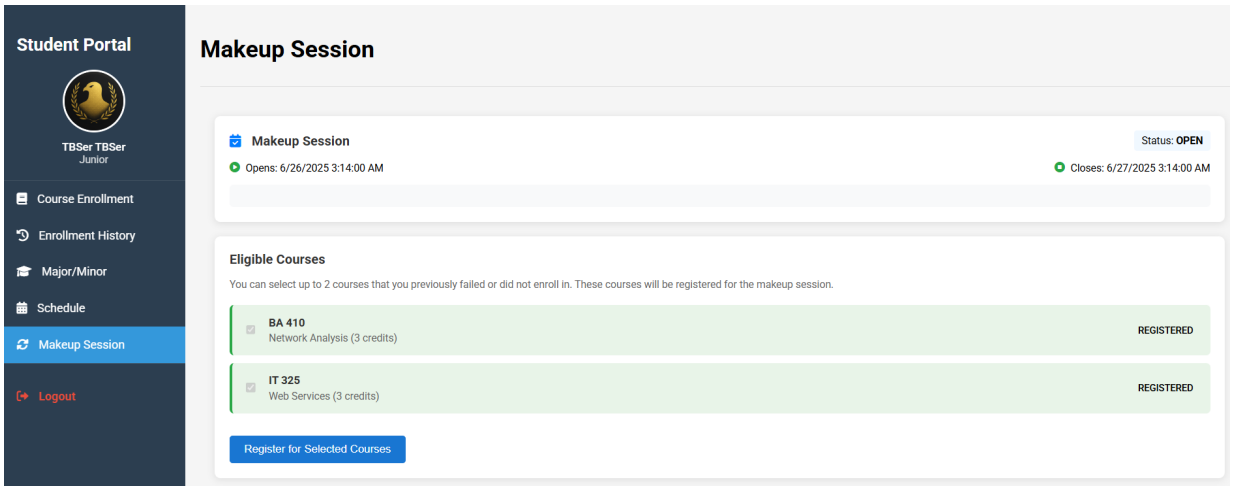


Figure 3.58: Registered Makeup Session Overview

The enrolled courses will be displayed under "Makeup Session" in Enrollment History section.

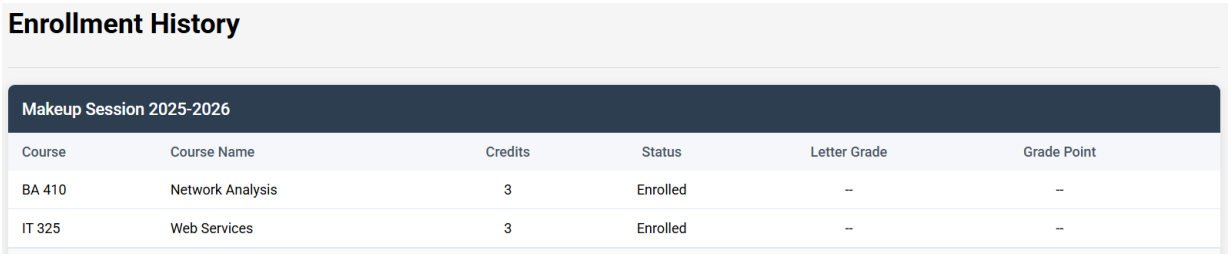


Figure 3.59: History of Enrolled Makeup Courses

3.4 Admin Portal

Once logged in, the admin gets access to a side menu presented in Figure 3.60 with the following elements from top to bottom:

- **A clickable profile picture:** that gives access to personal information and the ability to edit them.
- **Management and Request Handling section:** Handles user requests and manages academic periods, including course selection, semester transitions, and makeup sessions.
- **Major/Minor Validation section:** Takes care of most matters related to specialty selection, including managing the selection period and accepting or rejecting combination choices.
- **Settings and Adjustments section:** Encompasses managing all types of settings, both system-wide and individual, as well as handling special cases and manual changes.
- **Dashboard section:** Includes various charts that provide useful academic and system-related insights.
- **Logout Option:** Allows the admin to log out of the system at any time.
- **Theme Toggle:** Enables switching between light and dark modes for a personalized interface experience.

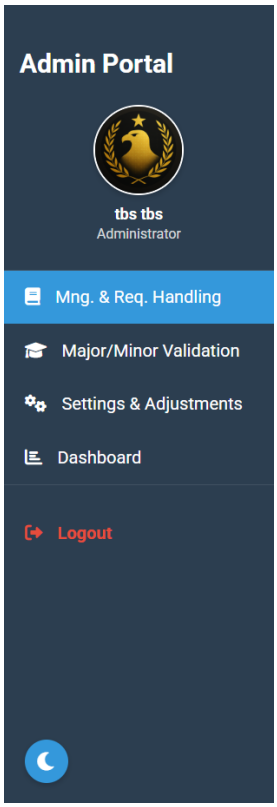


Figure 3.60: Admin's Side Menu

3.4.1 Personal Information

Once the admin clicks his profile picture, he can edit his personal information through this user interface. Pressing *Save Changes* will commit to the changes, and pressing *Cancel* will dismiss them.

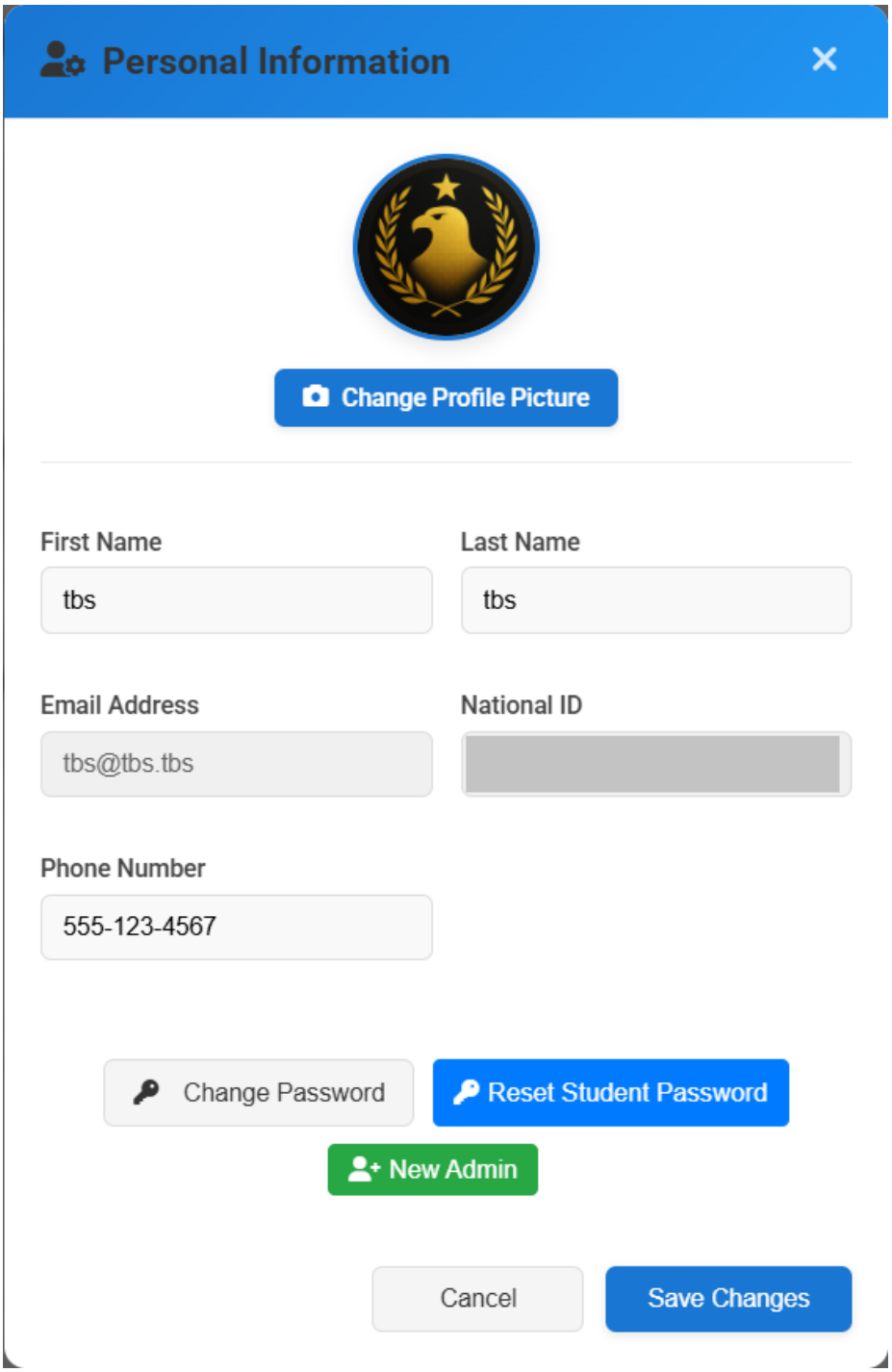


Figure 3.61: Admin Personal Information Interface

He is also presented with options to change his own password, reset a student's password manually in cases where the student has forgotten it and cannot access their email to perform a standard reset, and finally, to create a new admin account (as shown in Figure 3.62.

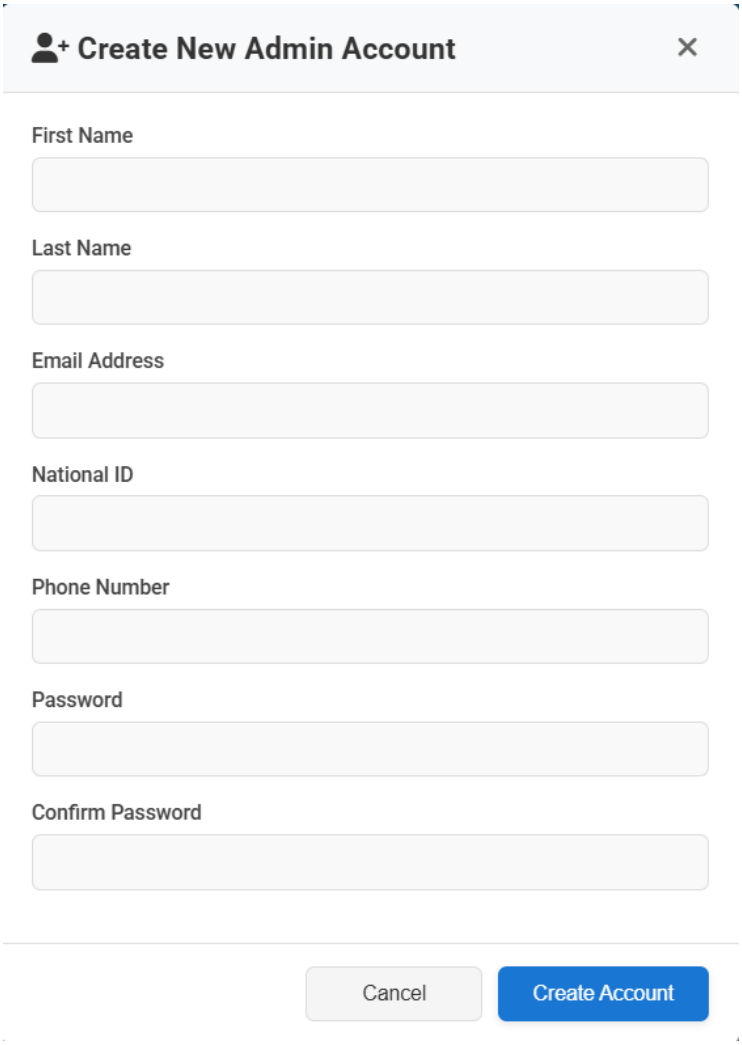


Figure 3.62: Creating a New Admin Account

3.4.2 Management and Requests Handling

In this section illustrated in Figure 3.63, the admin can control the start and end of semesters, course enrollment periods, and makeup sessions. Additionally, the admin can approve or deny student requests related to dropping courses, forgiveness policy, and manage semester extensions for the academic probation based on the official decision of the scientific board.

Figure 3.63: Management and Request Handling Section

Semester Management

The admin can start and end semesters, a critical action that triggers several back-end processes. For example, ending a semester will close active registration periods, update student levels, identify graduates, truncate certain database tables, and more.

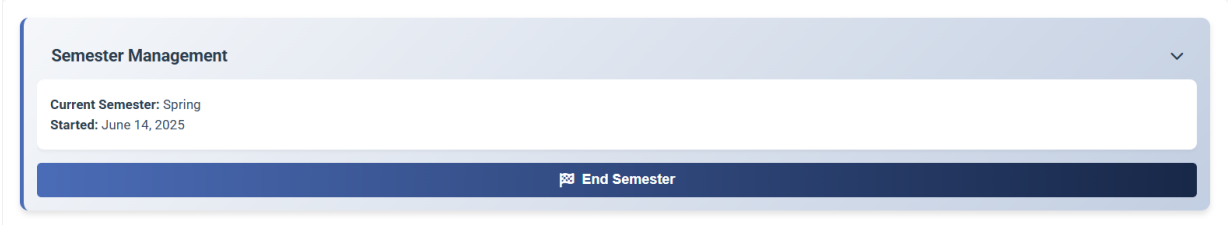


Figure 3.64: Semester Management Interface

Before this action can be performed, precautions were taken, by verifying that there are no missing grades or ongoing enrollments in the database (with a few exceptions).

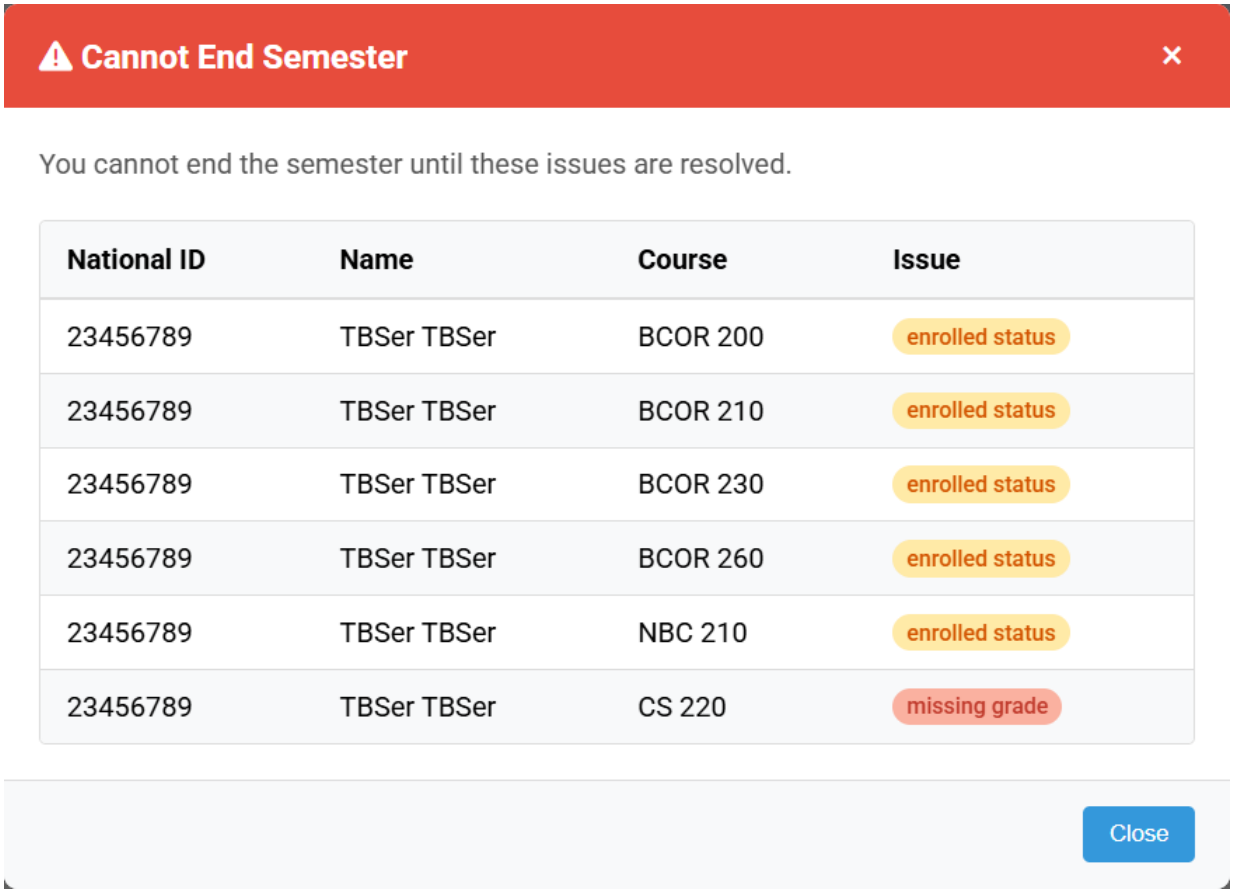


Figure 3.65: Semester End Blocked Due to Missing Data

Registration Period Management

Shown in Figure 3.66, the admin can initiate course enrollment periods for students. If the specified dates are in the future, the period will be scheduled accordingly.

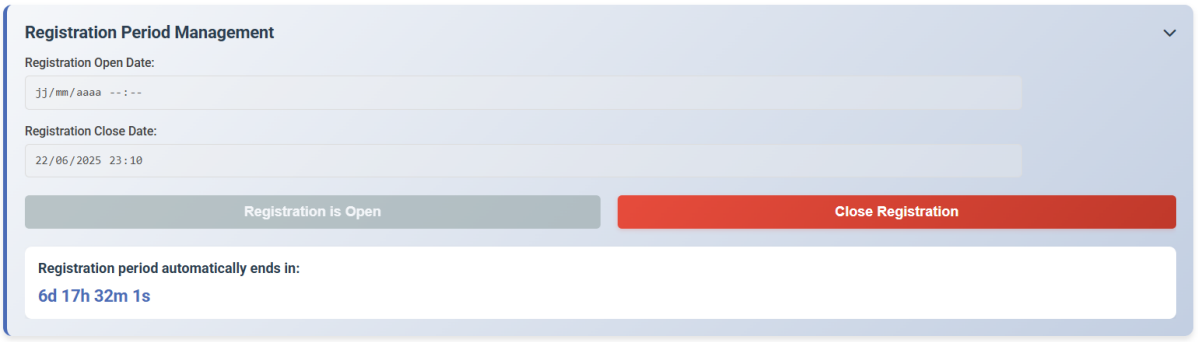


Figure 3.66: Enrollment Period Management Interface

Makeup Session Management

Opening the makeup session unlocks a hidden section on the student interface, where only eligible students that satisfies certain conditions are allowed to see and use it to enroll in courses.

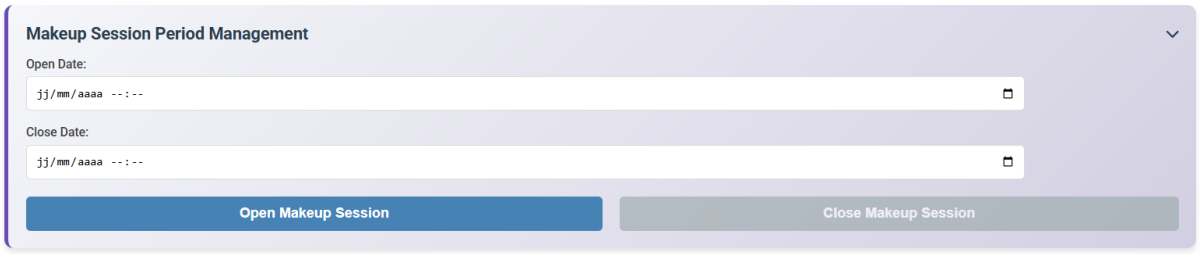


Figure 3.67: Makeup Session Management Interface

Dropping Course Handling

The admin can approve or reject student requests to drop specific courses using the interface presented in Figure 3.68. This functionality becomes relevant once the course enrollment period has ended. And the final decision depends on whether the student has already sat for any form of evaluation in the course (e.g. midterm, quiz, or oral test). If such participation is detected, the request will be denied.

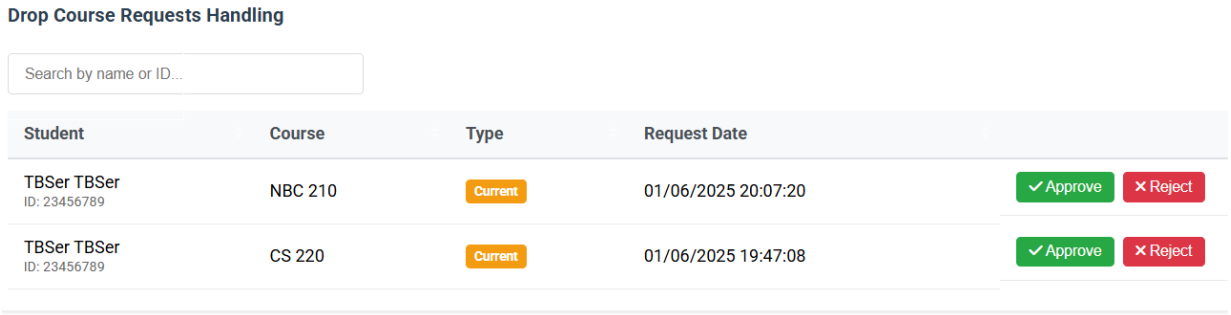


Figure 3.68: Drop Course Requests Handling Interface

Forgiveness Requests Handling

The admin can approve or reject the student's request as well as check already handled requests.

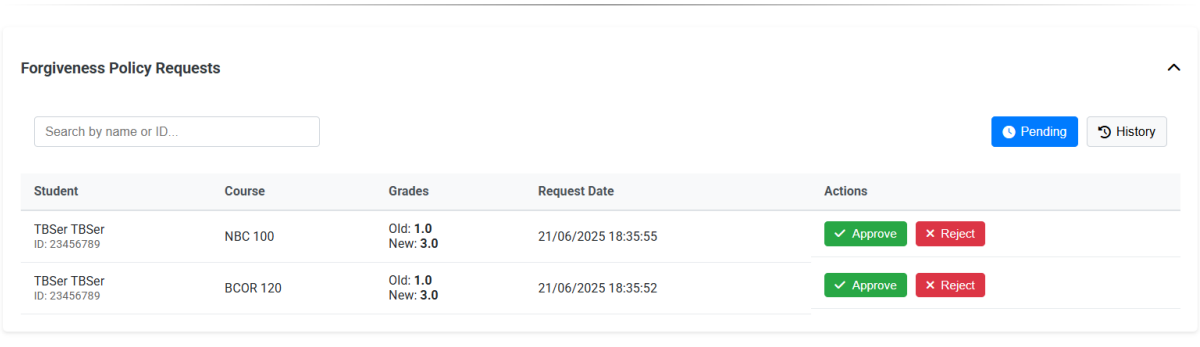


Figure 3.69: Forgiveness Policy Requests Handling Interface

Probation Extension Handling

In Figure 3.70, the requests that gets sent there are automatically triggered by the system once a student reaches the maximum number of consecutive probation semesters allowed. While the admin does not make the final decision, they are responsible for processing and recording the scientific board's verdict.

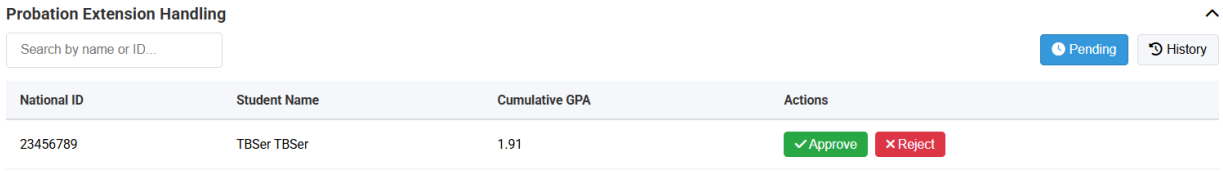


Figure 3.70: Probation Handling Interface

After the scientific board renders its decision, the admin may record any accompanying comments for the student to review in the section displayed in Figure 3.71. If the extension is approved, the student will be granted an additional semester at TBS to improve their cumulative GPA above the required threshold. If the request is denied, the student will be dismissed from the institution immediately.

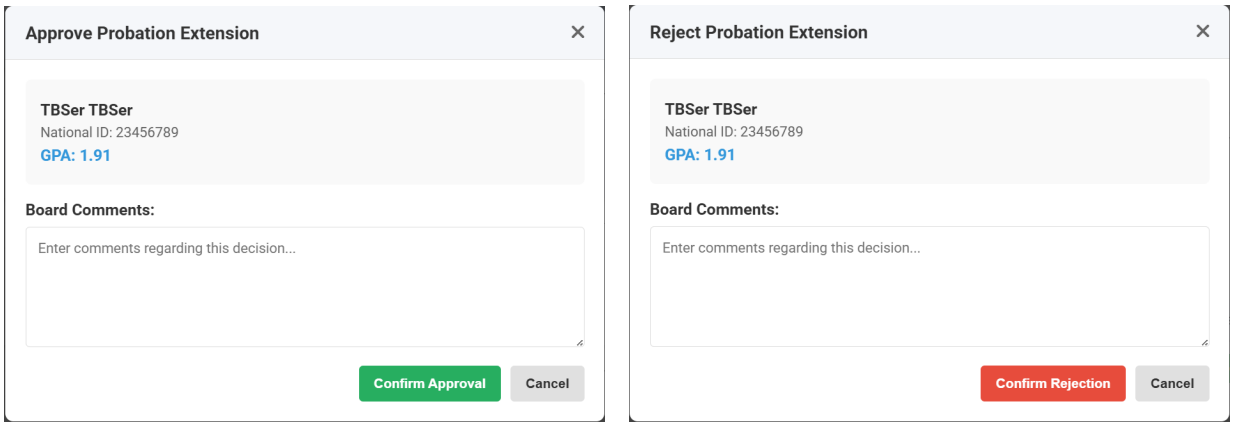


Figure 3.71: Examples of Approving and Rejecting Probation Extension Requests

3.4.3 Major/Minor Validation

In this menu option shown in Figure 3.72, the administrator performs actions related to student specializations. The general layout of the interface consists of three main components: (1) a **Selection Period Management** panel, which controls the opening and closing of the specialization selection period; (2) a **Statistics Section**, which displays relevant data regarding major/minor combinations submitted by students; and (3) a **Specialization Requests Section**, where submitted requests are listed and can be either approved or rejected by the admin based on the institution's capacity to accommodate the selected combination within its available resources.

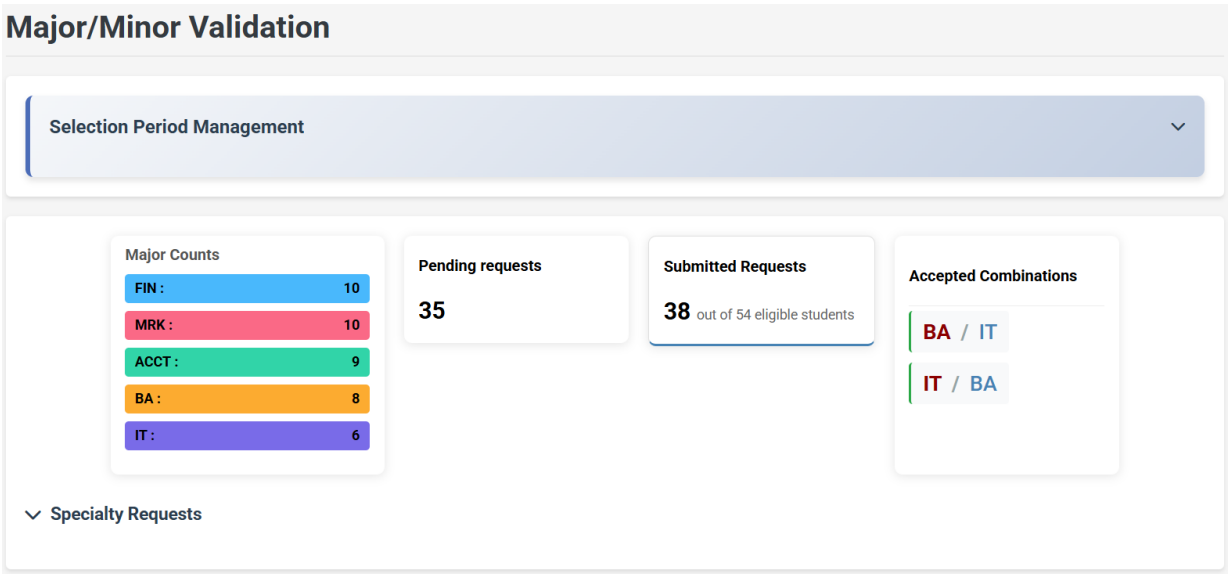


Figure 3.72: Major/Minor Combination Validation Interface

Requests are grouped by combination type (Major/Minor or Major/Major) and by the specific pairing chosen (e.g., BA/IT, ACCT/FIN) as portrayed in Figure 3.73. Once a combination is approved by the administrator, it gets added to the list of "accepted combinations" and all subsequent requests for that same combination are automatically approved. Conversely, if a combination is rejected, students are no longer allowed to submit requests for it.

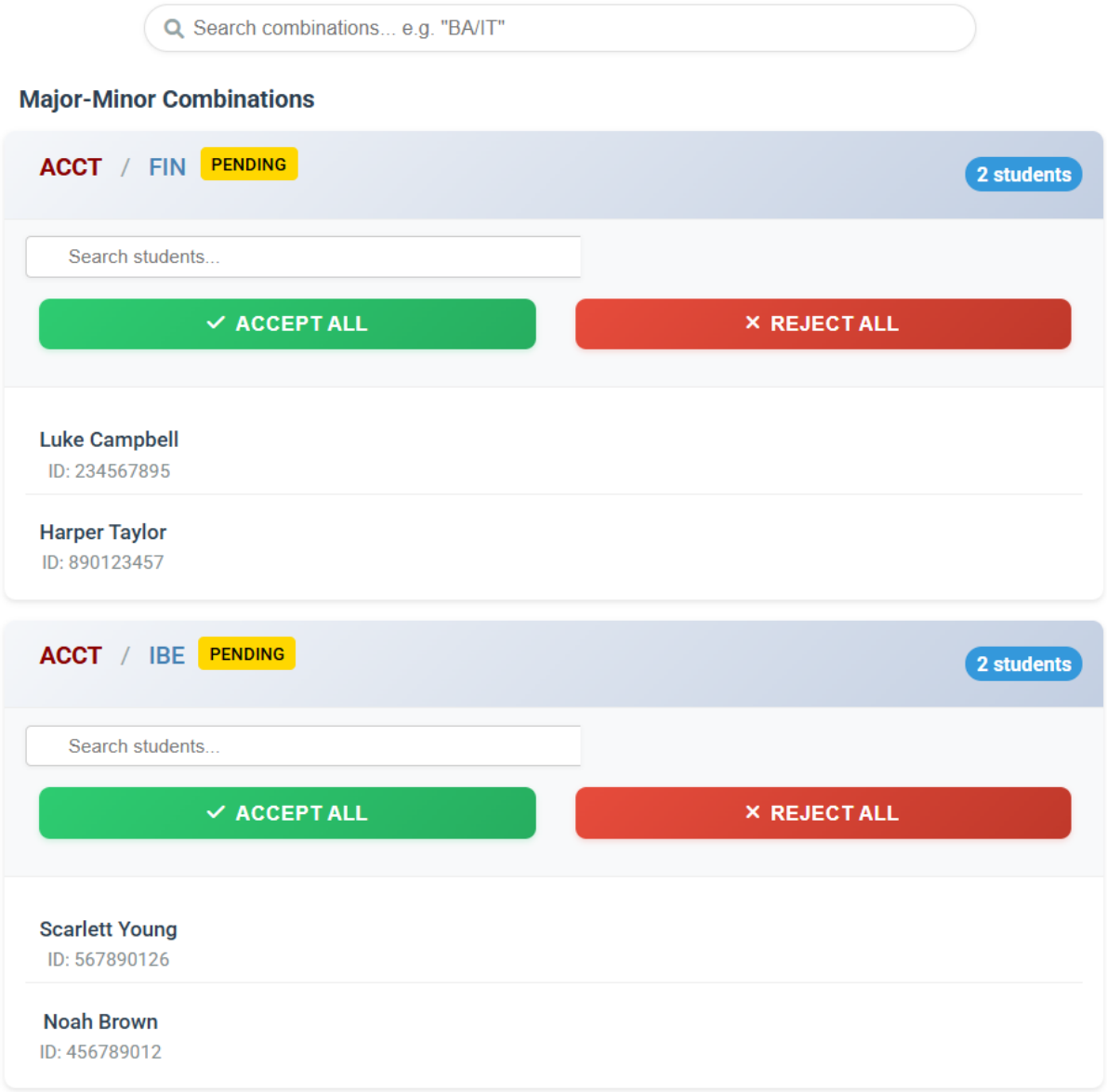


Figure 3.73: Specialization Requests Handling Interface

3.4.4 Settings and Adjustments

This part of the menu presented in Figure 3.74, there is a total of 5 sections : "Course Settings","Major Settings", "General Settings", "Schedule Settings" and "Special Cases" that grants the admin full control over both system-wide and individual parameters. Such flexibility is essential to ensure the system remains adaptable and effective in the face of evolving academic scenarios throughout any semester.

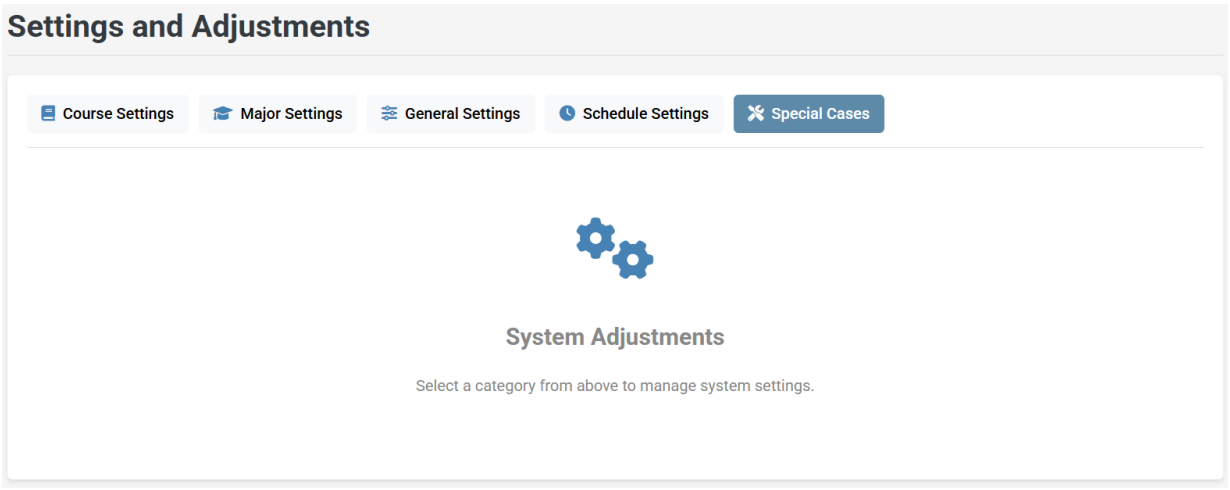


Figure 3.74: Settings and Adjustments Interface

Course Settings

In this drop-down menu, the admin can select to add or edit any existing course.

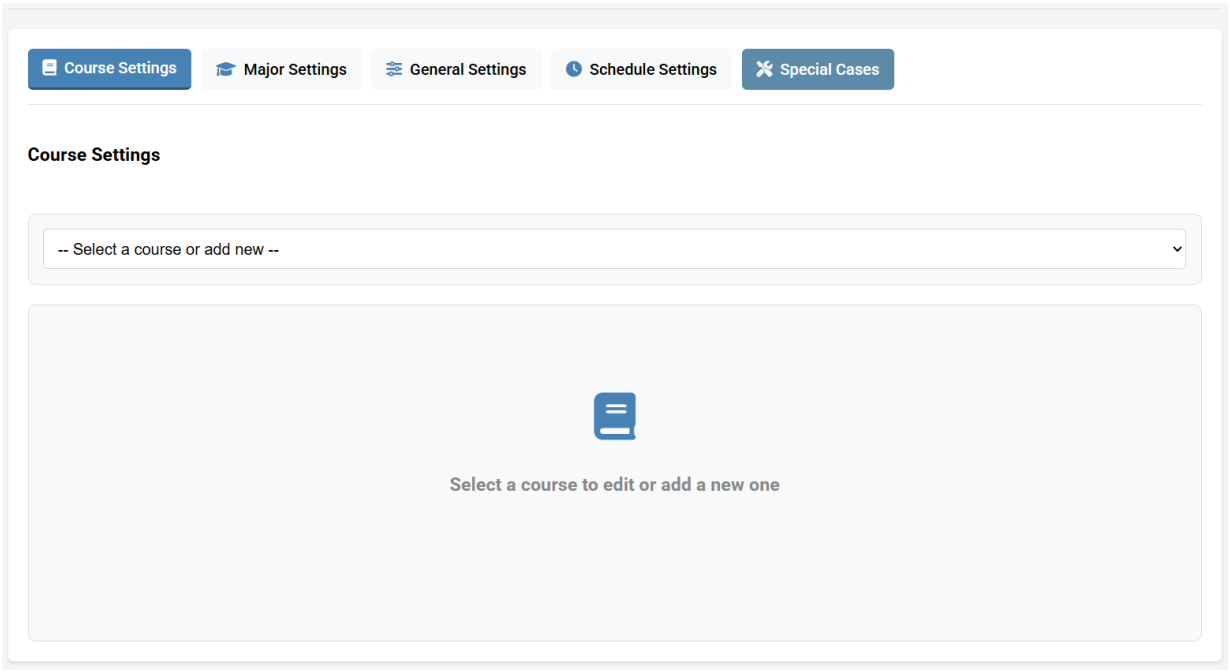


Figure 3.75: General Course Settings Interface

Once the course is selected, the admin can edit its configuration as portrayed in Figure 3.76. The “Save Changes” button commits to those changes.

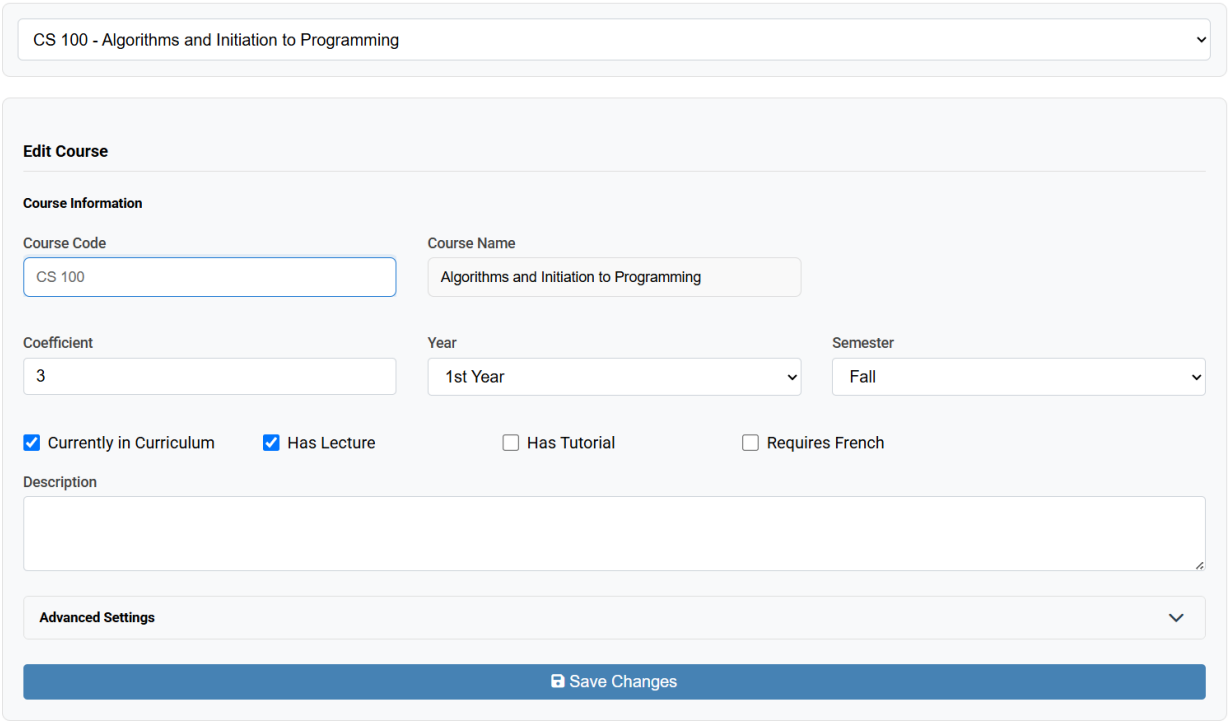


Figure 3.76: Basic Course Settings

If the Advanced Settings option is clicked, other configurations related to which major or minor the course is offered to will be displayed. Fields like *"For Minor If Major Is"* and *"Minor Study Year"* serve the purpose of handling specific cases, such as when courses are offered as minors only if a certain major is selected, or when the course is offered at a different year than the one defined by its course code.
Example: `IT 325` is mandatory for BA/IT Senior students.

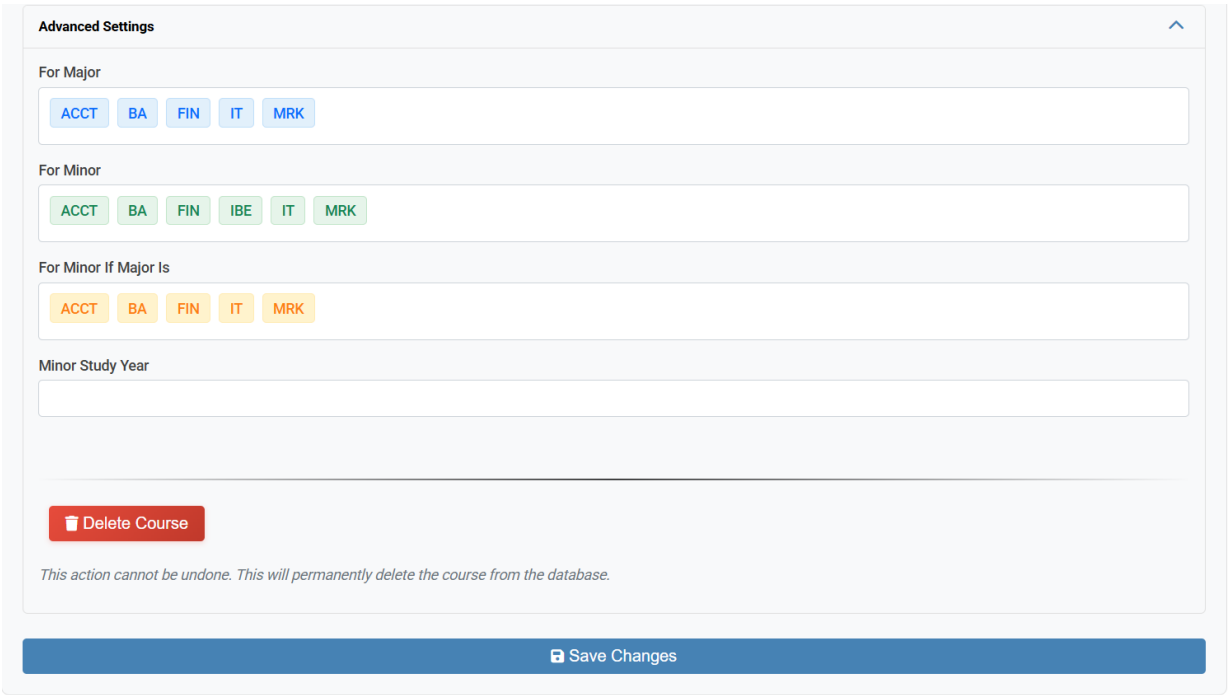


Figure 3.77: Advanced Course Settings

Major Settings

Major settings can be either on a system level or an individual level.

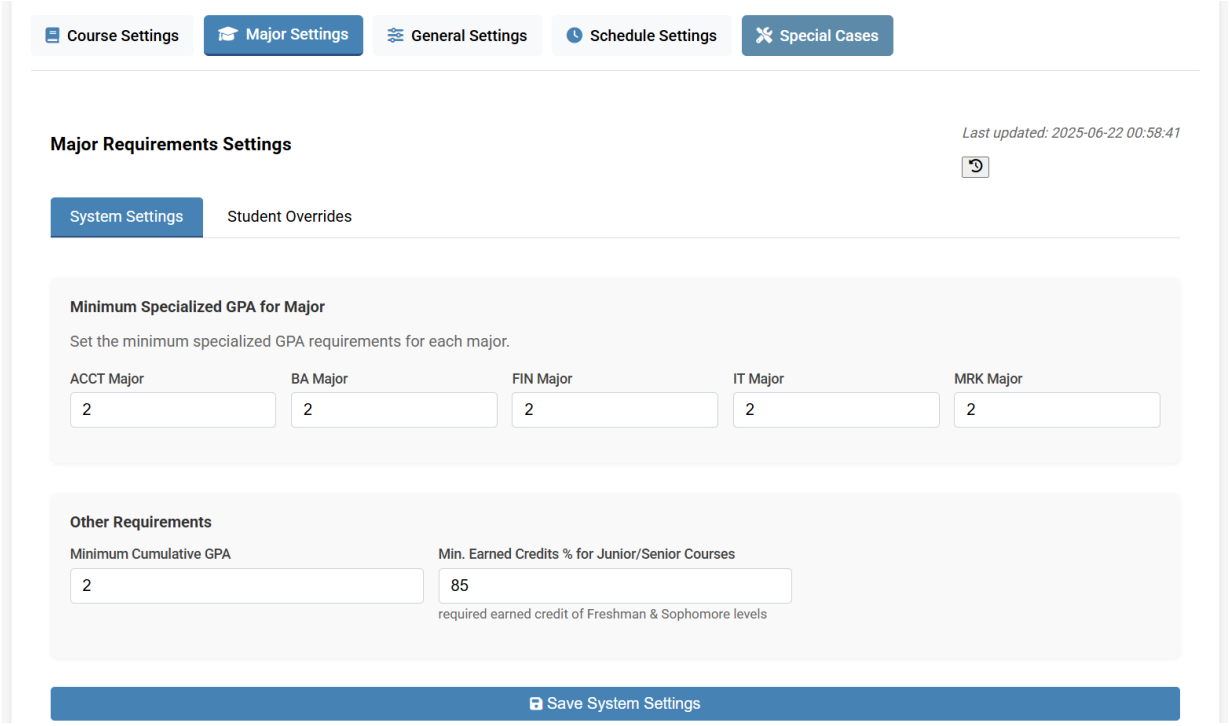


Figure 3.78: System-Level Basic Settings for Majors

The admin can adjust the specialized GPA for all majors, as well as other configurations related to the selection of specialty and selection of Junior/Senior related courses.

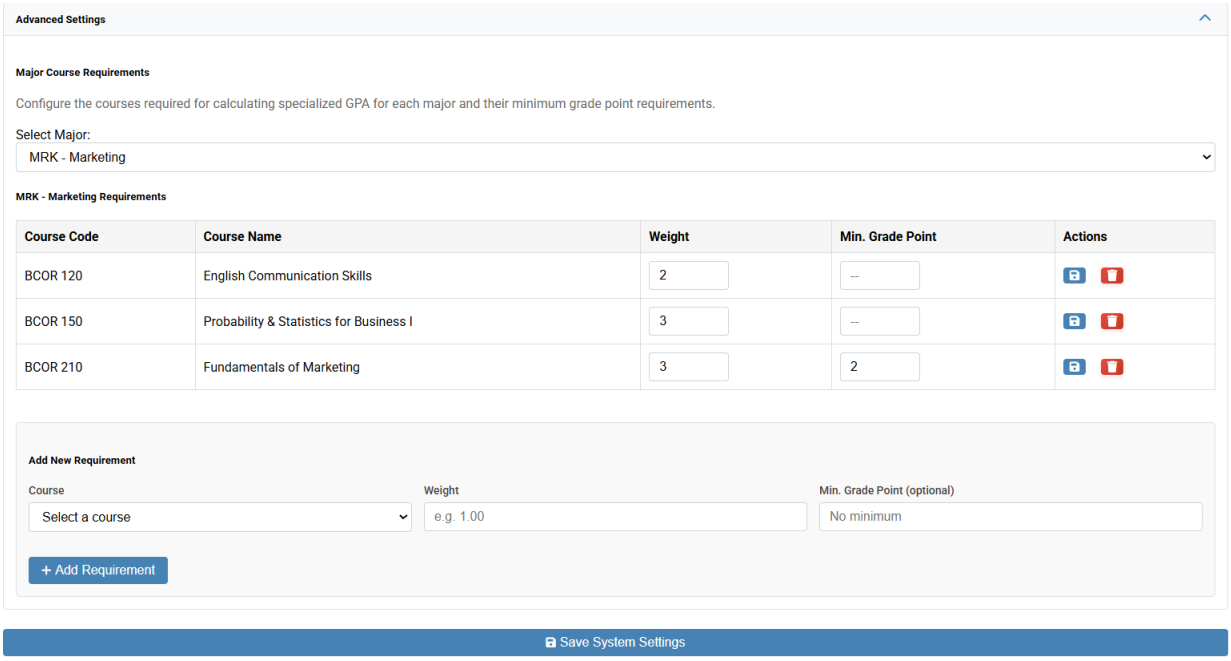


Figure 3.79: System-level Advanced Settings for Majors

In the Advanced Settings section, the admin can configure the requirements that define the

specialized GPA for any major. This includes adjusting the list of contributing courses, their respective weights, and the minimum required grade points as needed.

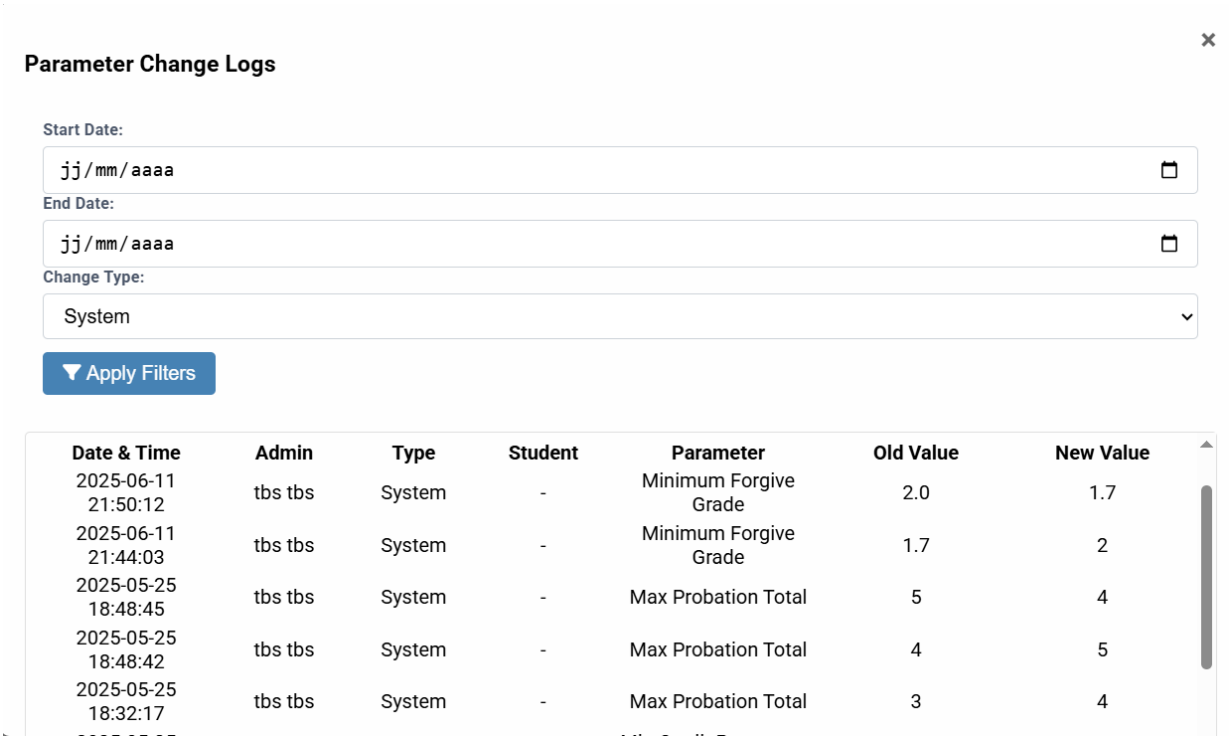


Figure 3.80: Major Settings Log History

The admin can conveniently view logs for both system-wide and individual changes, with the ability to apply filters for easier navigation and targeted searches.

General Settings

In this settings option, the administrator can freely adjust general parameters related to probation rules, the forgiveness policy, and graduation criteria. These modifications can be applied either globally across the entire system:

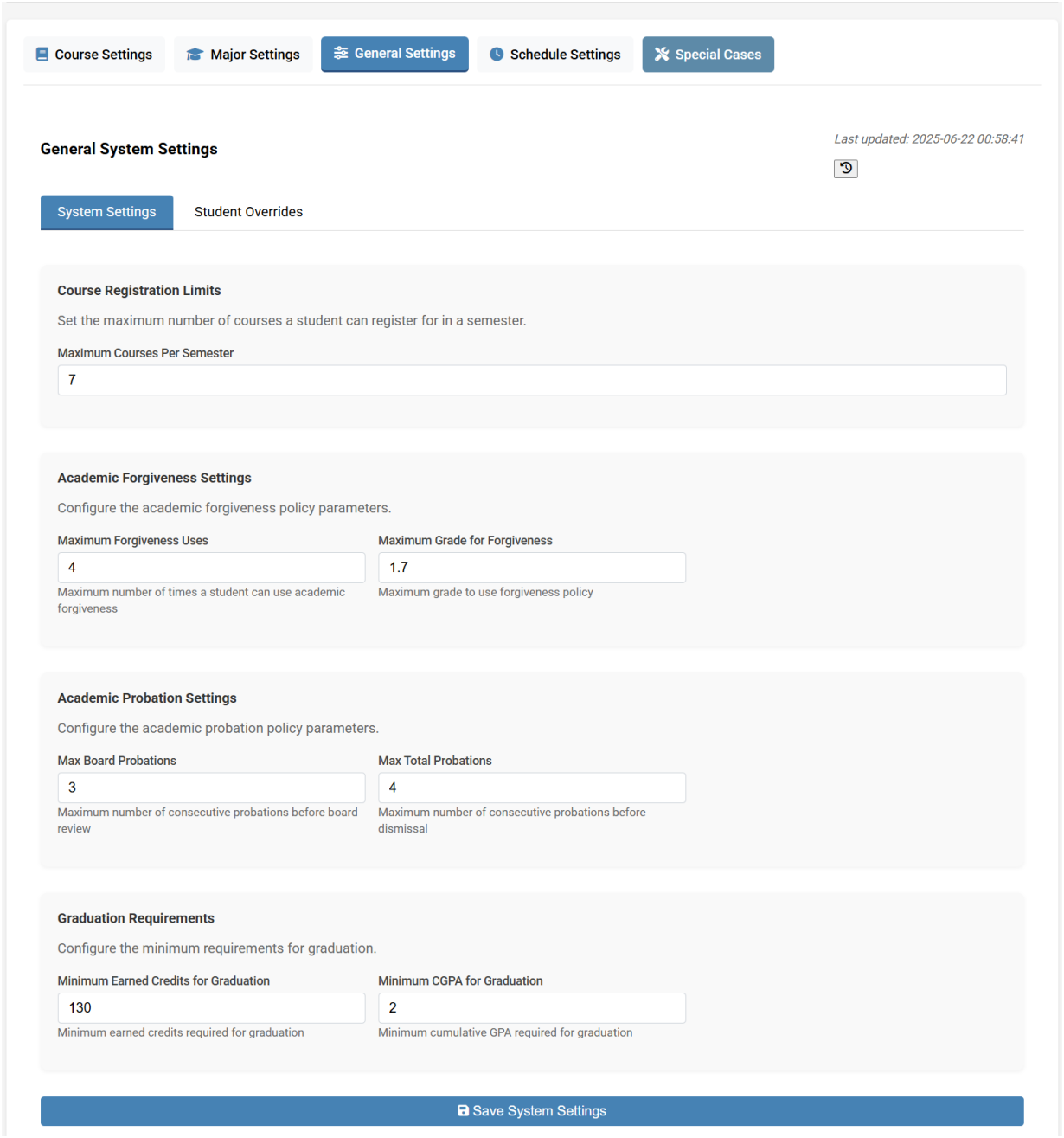


Figure 3.81: System-wide General Settings

or targeted to specific individuals only, with some limitations, as illustrated in the figure below:

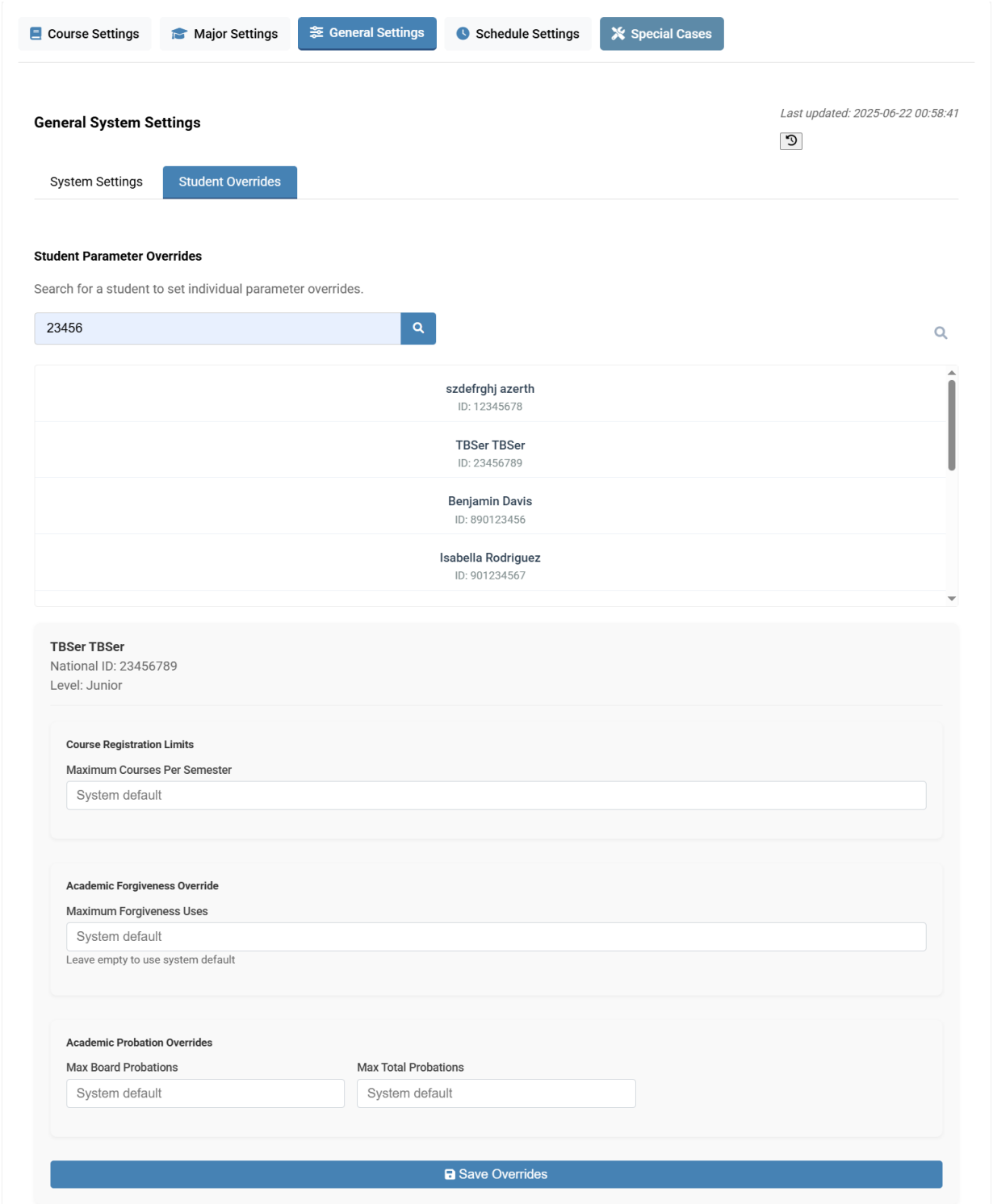


Figure 3.82: Individual-level General Settings

Schedule Settings

This settings interface allows the administrator to configure various parameters that influence the behavior of the scheduling model. Specifically, it enables adjustments to the weights used in the objective function, along with parameters that define the upper bounds on solution count and execution time.

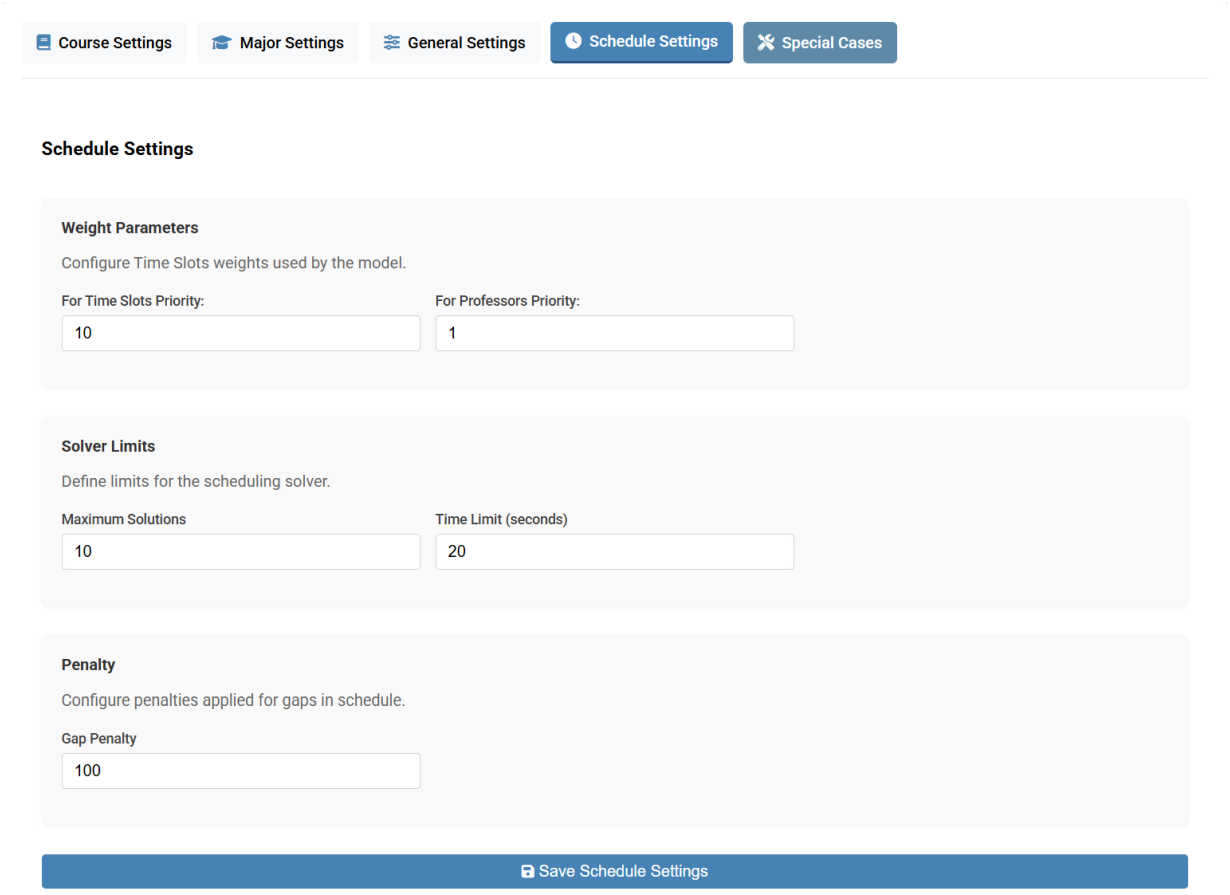


Figure 3.83: Configuration interface for schedule-related parameters

Special Cases

The interface shown in Figure 3.84 is designed to manage cases related to transfer students and those who take a break from their studies for personal or medical reasons (note that the validation of these cases is handled outside the system). The admin can search for students using their National ID numbers.

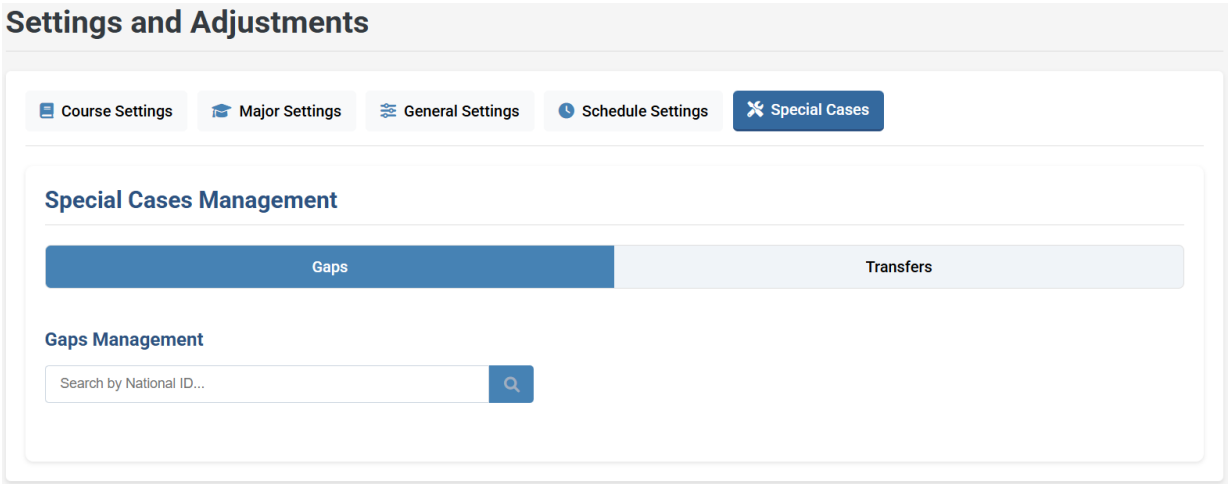


Figure 3.84: Special Cases Management Interface

For students taking one or more semesters off for valid reasons, the admin can mark their

current semester courses with the status *"not enrolled"* in the back-end. In the front-end, these courses are shown as *"skipped"*, following the course flow structure illustrated in Figure 2.3. This ensures that the system's academic logic remains consistent.

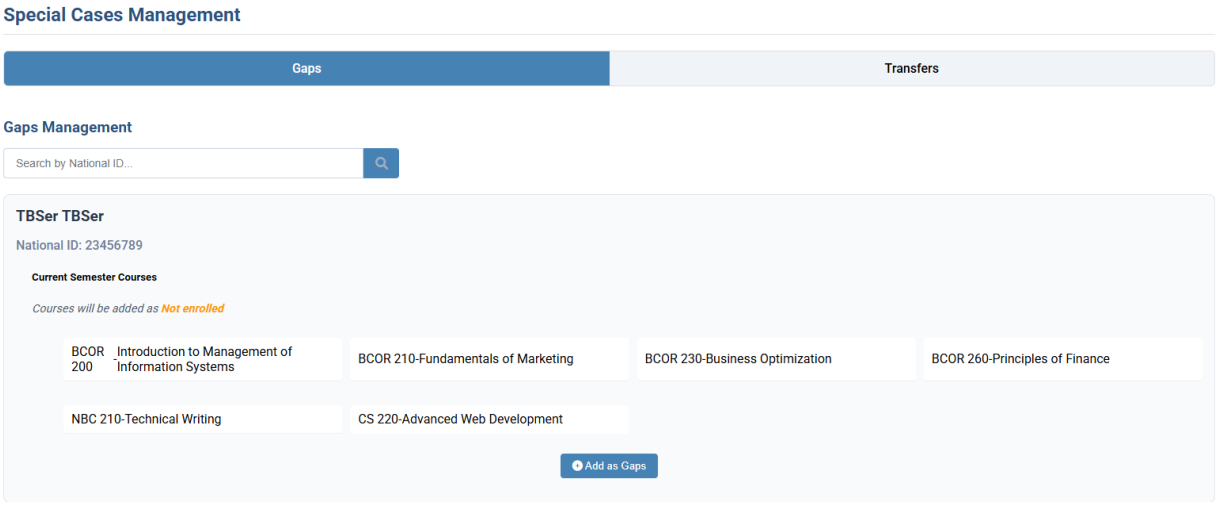


Figure 3.85: Gap Semester Interface

On the other hand, students who spend a semester at another higher education institution must have their credits transferred, if applicable, upon returning to TBS.

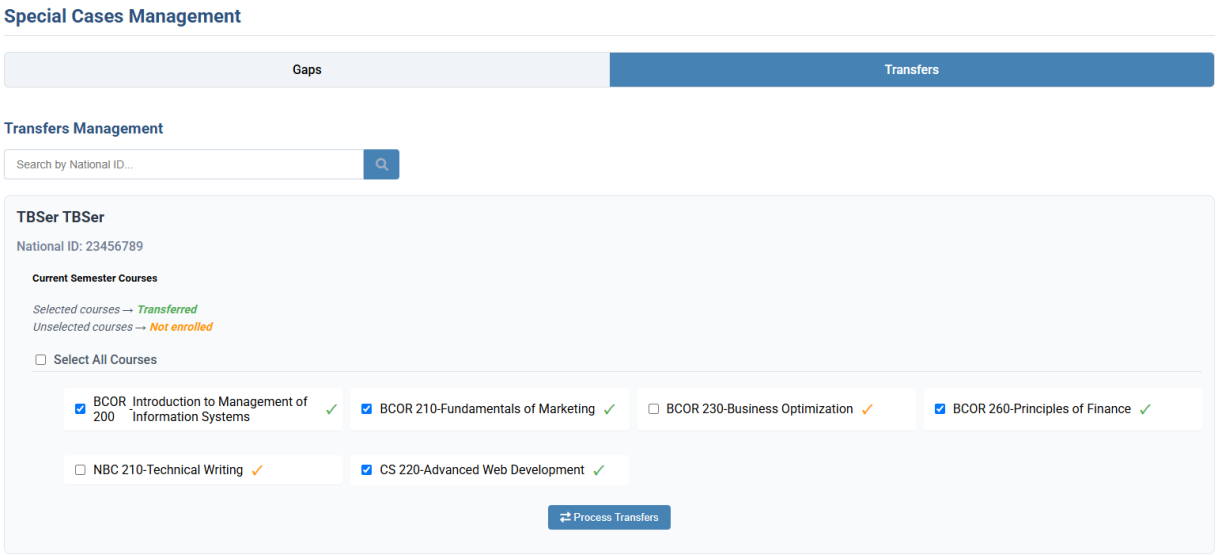


Figure 3.86: Transfer Courses Management Interface

Dashboard Interface

The admin has access to a dashboard portrayed in Figure 3.87,that provides an overview of key academic and system statistics, including general metrics, cumulative GPA distributions by major, major selection breakdown, and various performance indicators. This allows monitoring of student performance trends and early identification of academic concerns to improve curriculum design.

Note: The displays in the dashboard were done using random data.

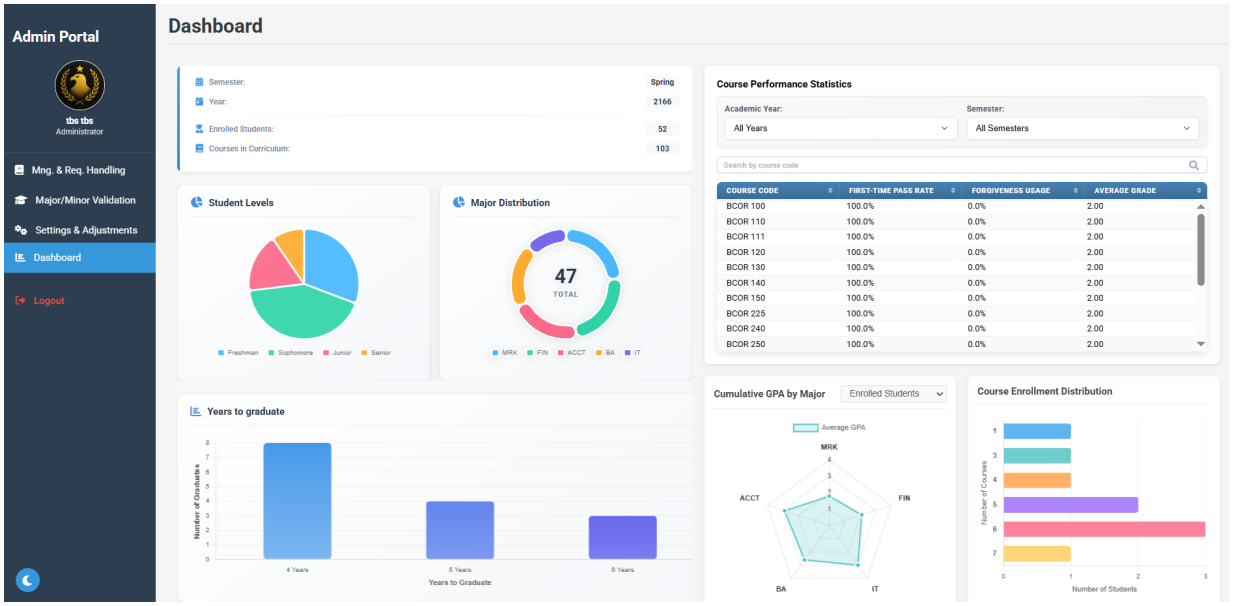


Figure 3.87: Admin Dashboard Interface

3.5 Conclusion

This chapter presented the complete implementation of the course registration system, highlighting both the technical foundation and user functionalities. It showcased how the system supports students through guided decision-making, while also equipping administrators with tools to manage academic workflows and enforce policies. By integrating optimization, automation, and interactive design, the system ensures a streamlined, scalable, and policy-compliant registration experience for all users.

Discussion & Conclusion

In this project, we designed and implemented a web-based course registration system tailored for Tunis Business School's academic environment. Over several iterative development cycles, we translated the school's handbook rules into precise system logic and built a unified platform for both students and administrators.

The resulting prototype streamlines the entire registration workflow: it enforces eligibility and prerequisite constraints, allows students to select courses and specializations, and automatically generates valid timetables using a linear optimization model. At the same time, administrators can define academic periods, handle requests, adjust parameters, and get access overall statistics through a dashboard.

These components together mark a significant modernization of TBS's registration process. In the new system, students benefit from clear guidance and immediate feedback as they select their courses, specializations, and build their schedules. Administrators benefit from reduced manual workload and fewer ad-hoc overrides.

From a development standpoint, this project was also a valuable learning experience. Using an Agile, scenario-driven methodology allowed us to implement features in the same sequence that they occur during a student's academic journey This iterative approach meant that each sprint delivered a usable subset of functionality, which we tested and refined before moving on.

Despite this progress, the system has limitations in its current form. It has not yet been stress-tested under high user concurrency. Furthermore, a system of this importance requires a higher level of security to prevent any unauthorized manipulation of grades or enrollment data. Additionally, the system currently assumes that inputs such as student grades and personal information are provided and inserted into the database for use whenever they are available, which highlights the need for proper integration with existing institutional systems.

Future Work:

- Expand the Advisory Module: This could involve training on historical student data to predict course success, or introducing peer-based recommendations such as "Students like you also took..."
- Implement Additional Features: Including open communication channels between students and administrators to facilitate support and clarification, or a course progression simulator that allows students to visualize their current academic standing, simulate different course-taking scenarios, and instantly see the impact on their academic level, major/minor progress, unlocked prerequisites, and estimated graduation timeline.

In conclusion, this project demonstrates a strong proof-of-concept for digitizing and optimizing academic course registration. By centralizing the process, enforcing rules systematically, and providing automated schedule generation, the platform lays the groundwork for a more efficient and transparent registration experience. With the recommended enhancements, the system can evolve into a comprehensive tool that not only simplifies enrollment but also aids academic planning and decision-making for the entire school. This aligns with research findings that such systems greatly improve efficiency and user satisfaction

References

[1] Prizament M. How higher ed institutions can complete their digital transformation. *EdTech Magazine*, April 2023.

[2] Kuali. Workflow automation in higher education: Driving digital transformation. *Kuali Blog*, January 2025.

[3] Ellucian. How do students feel about the registration process? *Ellucian Blog*, 2025.

[4] Ellucian. Modern one-stop student service centers transform the student experience. *Ellucian Blog*, 2024.

[5] Mahmoud et al. University timetabling challenges. *Journal of Scheduling*, 2015.

[6] A. Wren. Scheduling, timetabling and rostering: A special relationship? *International Conference on the Practice and Theory of Automated Timetabling*, 1996.

[7] Khalid A. Fakieh et al. Decision support systems in higher education. *International Journal of Applied Information Systems*, 9(2):32–40, 2015.

[8] S. Samaranayake et al. An interactive decision support system for college degree planning. *Athens Journal of Education*, 10(1):100–115, 2023.