



People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research -Mohamed Kheider University of Biskra-
Faculty of the exact sciences, natural and life sciences.
Department of Computer Science

Master thesis

Prepared and presented by :

Soumia Athmane , Nada Saker & Soumia Mebarki
GLSD / GLSD / SIOD

Intelligent University Pedagogical Management and Scheduling Platform

Jury :

Dr.Bilal Mokhtari	MCA	University of Biskra	Supervisor
Dr.Souraya Hamida	MCA	University of Biskra	Chairperson
Dr.Hadia Hoadjli	MAA	University of Biskra	Examiner

Academic year : 2023/2024

Abstract

Scheduling university activities such as courses, exams, planning for replacements, and graduation projects is a difficult and time-consuming task. The main challenge arises from the need to schedule various types of sessions (lectures, tutorials, lab work, and exams) across different study levels, taking into account constraints such as the availability of classrooms and teachers, as well as the specific requirements of each course. In this project, we introduce a new multiplatform application (web and mobile) that is highly beneficial for administrative staff, students, and teachers, as it facilitates communication and enhances the planning of various tasks. The scheduling of pedagogical activities such as courses, exams, and assignment of graduation projects is completely automated using a Multi-objective genetic algorithm. Therefore, the application significantly reduces the manual effort required and improves overall satisfaction with scheduling requirements, making it a valuable tool for educational institutions.

ملخص

جدولة الأنشطة الجامعية مثل الحصص الدراسية، الامتحانات والحصص الإضافية هي مهمة صعبة وتستغرق وقتاً طويلاً. التحدي الرئيسي ينبع من الحاجة إلى جدولة أنواع مختلفة من الحصص (المحاضرات، والأعمال الموجهة، و التطبيقات، والامتحانات) عبر مستويات دراسية مختلفة، مع مراعاة القيود مثل توفر الأقسام الدراسية والاساتذة، بالإضافة إلى المتطلبات الخاصة لكل حصة. في هذا المشروع، نقدم تطبيقاً جديداً متعدد المنصات (ويب وموبايل) يعتبر مفيداً للغاية لكل من الطاقم الإداري، الطلاب والاساتذة، حيث يسهل التواصل ويعزز التخطيط لمختلف المهام. جدولة الأنشطة التعليمية مثل الحصص الدراسية والامتحانات، وتعيين مشاريع التخرج تتم بشكل كامل تلقائياً باستخدام خوارزمية جينية متعددة الأهداف. لذلك، يقلل التطبيق بشكل كبير من الجهد اليدوي المطلوب ويحسن الرضا العام بمتطلبات الجدولة، مما يجعله أداة قيمة للمؤسسات التعليمية.

Résumé

La planification des activités universitaires telles que les cours, les examens, la planification des remplacements et les projets de fin d'études est une tâche difficile et très chronophage. Le principal défi provient de la nécessité de programmer différents types de sessions (cours, travaux dirigés, travaux pratiques et examens) à travers différents niveaux d'études, en tenant compte des contraintes telles que la disponibilité des salles de classe et des enseignants, ainsi que des exigences spécifiques de chaque cours. Dans ce projet, nous introduisons une nouvelle application multiplateforme (web et mobile) qui est très bénéfique pour le personnel administratif, les étudiants et les enseignants, car elle facilite la communication et améliore la planification de diverses tâches. La programmation des activités pédagogiques telles que les cours, les examens et l'attribution de projets de fin d'études est entièrement automatisée à l'aide d'un algorithme génétique multi-objectifs. Ainsi, l'application réduit considérablement l'effort manuel requis et améliore la satisfaction globale vis-à-vis des exigences de planification, ce qui en fait un outil précieux pour les institutions éducatives.

Acknowledgment

*As we submit this thesis, it is only fitting to express our heartfelt gratitude first and foremost to **ALLAH**, whose guidance and blessings have been indispensable throughout this journey.*

Our deepest thanks also go to our thesis advisor, Mokhtari Bilal, whose expert guidance and unwavering support have been pivotal. His dedication to nurturing academic excellence has inspired us throughout this process.

Additionally, we extend our appreciation to the jury members for their valuable time, expertise, and constructive criticism during the defense of this thesis. Their insightful comments and suggestions have significantly contributed to enhancing the quality of our work and have provided us with a deeper understanding of our research area. Thank you for your pivotal role in this academic endeavor.

Dedication

This thesis is wholeheartedly dedicated to Allah, whose divine guidance and blessings have been our unwavering source of strength and light on this journey.

With deep love and gratitude, we extend this dedication to the Athmane, Mebarki, and Saker families, consisting of our cherished parents, sisters, and brothers, who have enveloped us in endless support and love.

Each family has uniquely shaped one of us, and we are eternally grateful for your encouragement and steadfast belief in our abilities.

Additionally, we offer our heartfelt appreciation to our friends, the extended family we chose, who have stood by us as pillars of support and companionship.

Each of you has touched our hearts deeply, fueling our journey with joy and resilience, contributing immensely to our personal and academic growth.

Soumia, Soumia and Nada

Contents

General Introduction	xv
Chapter 1: Scheduling problem	
1.1 Introduction	3
1.2 Schedule	3
1.3 Scheduling as a problem	4
1.4 Scheduling techniques	5
1.4.1 Manual techniques	5
1.4.1.1 None visual methods	5
1.4.1.2 Visual techniques	6
1.4.2 Automated techniques	6
1.4.2.1 Mathematical optimization techniques	7
1.4.2.2 Heuristics techniques	10
1.4.2.3 Machine Learning based techniques	16
1.4.2.4 Deep Learning Techniques	19
1.4.2.5 Rule-Based techniques (Expert Systems)	23
1.5 Conclusion	24
Chapter 2: Genetic Algorithm : Academic Scheduling Optimization	
2.1 Introduction	27
2.2 Evolutionary algorithms (EAs)	27
2.3 Genetic Algorithms (GA)	28
2.4 Components and processes of Genetic Algorithms	28
2.4.1 Basic concepts of GA	28

2.4.1.1	Genes	28
2.4.1.2	Chromosomes	29
2.4.1.3	Population	29
2.4.2	Phases of Genetic Algorithms	30
2.4.2.1	Initialization	30
2.4.2.2	Selection	32
2.4.2.3	Crossover	33
2.4.2.4	Mutation	35
2.4.2.5	Evaluation	37
2.4.2.6	Generation	38
2.4.3	Convergence in genetic algorithms	38
2.5	Genetic algorithms classes	39
2.5.1	Single-Objective Genetic Algorithms (SGA)	40
2.5.1.1	Elitist Algorithms:	40
2.5.1.2	Non-Elitist Algorithms	40
2.5.2	Multi-Objective Genetic Algorithms (MOGA)	42
2.5.2.1	Elitist Algorithms	42
2.5.2.2	Non-Elitist Algorithms	44
2.6	Application of Genetic Algorithms in scheduling	45
2.6.1	Problem representation	45
2.6.1.1	Representation of courses	45
2.6.1.2	Genetic representation for courses	46
2.6.1.3	Representation of exams	49
2.6.1.4	Genetic representation for exams	50
2.6.2	Scheduling with NSGA-II: From theory to practice	52
2.6.2.1	Rationale for Choosing NSGA-II	52
2.6.2.2	Application of NSGA-II for the scheduling system	52
2.7	Conclusion	62
 Chapter 3: Analysis and Conception		
3.1	Introduction	65
3.2	Unified Modeling Language definition	65
3.2.1	Use case diagram	65

3.2.2	Class Diagram	66
3.2.3	Sequence Diagram	66
3.3	System design	67
3.3.1	Professors part	67
3.3.1.1	Professors functionalities	67
3.3.1.2	Professors use case diagram	67
3.3.2	Administrator role	69
3.3.2.1	Administrator functionalities	69
3.3.2.2	Administrator use case diagram	70
3.3.3	Student role	72
3.3.3.1	Student functionalities	72
3.3.3.2	Student use case diagram	72
3.3.4	Detailed specification	73
3.3.4.1	Class diagrams	73
3.3.4.2	Sequence diagram	79
3.4	Conclusion	81

Chapter 4: Implementation and deployment

4.1	Introduction	83
4.2	Development tools and technologies	83
4.2.1	Back-end tools	83
4.2.2	Front-end tools	85
4.3	System Architecture	87
4.4	Implementation: Features and functionalities	91
4.4.1	Web Application	91
4.4.2	Mobile application	130
4.5	Results of the genetic algorithm scheduling	140
4.5.1	Genetic algorithm configuration	141
4.5.2	Obtained results	142
4.5.2.1	Courses scheduling algorithm	142
4.5.2.2	Exams planning algorithm	146
4.6	Conclusion	149
4.7	Future enhancements	149

List of Figures

1.1	Booking calendar for scheduling [1]	6
1.2	Constraint Programming Paradigm.	8
1.3	linear programming optimization techniques [1].	9
1.4	Flowchart diagram of Standard Tabu Search	11
1.5	Ant Colony Optimization Algorithm	12
1.6	Flowchart diagram of simulated annealing algorithm	13
1.7	Flow diagram of PSO algorithm	14
1.8	The evolutionary cycle of GA.	16
1.9	Illustration of Machine Learning Classification[2].	17
1.10	DNN Architecture [3]	20
1.11	Basic CNN architecture [4]	21
1.12	Standard Transformer model [5].	22
1.13	Basic architecture of a rule based system [6].	24
2.1	Evolutionary Algorithms [7].	27
2.2	Gene.	28
2.3	Chromosome.	29
2.4	A generation of chromosomes.	29
2.5	Population initialization techniques in GAs [8].	30
2.6	Parent selection methods	32
2.7	Survivor selection methods.	33
2.8	single point crossover illustration [9].	34
2.9	Two-point crossover illustration [9].	34
2.10	Uniform crossover illustration [9].	35
2.11	Bit Flip Mutation.	35
2.12	Swap Mutation.	36

2.13	Scramble Mutation.	36
2.14	Inversion mutation.	37
2.15	Evaluation process.	37
2.16	Generation in genetic algorithms.	38
2.17	Convergence in genetic algorithms [10].	39
2.18	Classification of GAs Based on objective types and elitism.	39
2.19	Non-dominated Sorting Genetic Algorithm II	43
2.20	Strength Pareto Evolutionary Algorithm 2	44
2.21	Normal session gene representation	47
2.22	Section session gene representation	47
2.23	Normal session example	48
2.24	Section session example	48
2.25	Example of chromosome representation	49
2.26	Gene representation	50
2.27	Example of gene representation	51
2.28	Example of chromosome representation	51
3.1	Professor use case diagram	69
3.2	Administrator use case diagram	71
3.3	Student use case diagram	73
3.4	Class Diagram of schedule generator	75
3.5	Class Diagram for exam scheduling	76
3.6	Class Diagram for graduation project assignment.	78
3.7	Sequence diagram for scheduling generation	79
3.8	Graduation project assignment process sequence diagram	80
4.1	Global architecture of the proposed system	87
4.2	Database tables	88
4.3	Users accounts table	89
4.4	Schedule table.	90
4.5	Bachelors assignments table	91
4.6	Web login page	92
4.7	Web login page error	92
4.8	Web admin home page	93
4.9	Web admin menu home page	93

4.10	Web professor home page	94
4.11	Web student home page	94
4.12	Web professor calendar	95
4.13	Professor proctoring schedule waiting for generation	95
4.14	Professor proctoring schedule	96
4.15	Search groups schedule	96
4.16	Display groups schedule	97
4.17	Apply request	97
4.18	Schedule a test	98
4.19	Test scheduling	99
4.20	Replacement scheduling	99
4.21	Consultation scheduling	100
4.22	Notifications	100
4.23	Schedule generator menu	101
4.24	Selecting semester schedule	101
4.25	Progress bar for waiting the scheduling process	102
4.26	Professor schedule before the generation	102
4.27	Groups schedule after the generation	103
4.28	Manipulation of the generated schedule	103
4.29	Confirming the insertion for web	104
4.30	Insert the schedule in database for web	104
4.31	Professor schedule after the generation for web	105
4.32	Publish the schedule for web	105
4.33	Email sending process	106
4.34	Professor schedule excel format	106
4.35	Exam schedule generator	107
4.36	Choosing the exam interval dates	107
4.37	The generated exam schedule for groups	108
4.38	The generated proctoring schedule for professors	108
4.39	The generated proctoring schedule excel format	109
4.40	Email reception	109
4.41	Manipulate schedule	110
4.42	Manipulation options	110

4.43	Bachelors project assignment	111
4.44	Bachelors excel file	111
4.45	Pick file code	112
4.46	Bachelors project assignment file chooser	112
4.47	Selection of the file	113
4.48	Upload code	113
4.49	Displaying the assignment result	114
4.50	Email reception after the affection	114
4.51	Masters project assignment	115
4.52	Selection of the master file	115
4.53	Displaying the masters assignment result	116
4.54	Email reception after the assignment	116
4.55	Manipulation department data main menu	117
4.56	Level manipulation	117
4.57	Add level	118
4.58	Option manipulation	118
4.59	Add option	119
4.60	Section manipulation	119
4.61	Edit, delete or add section	120
4.62	Groups manipulation	120
4.63	Add group	121
4.64	Students manipulation	121
4.65	Edit, delete or add student	122
4.66	Professors manipulation	122
4.67	Edit, delete or add professor	123
4.68	Subjects manipulation	123
4.69	Edit, delete or add professor	124
4.70	Rooms manipulation	124
4.71	Edit, delete or add room	125
4.72	Users manipulation	125
4.73	Edit, delete or add user	126
4.74	Bachelors final year project manipulation	126
4.75	Edit, truncate or add project	127

4.76	Bachelors final year project assignment manipulation	127
4.77	Masters final year project manipulation	128
4.78	Masters final year project assignment manipulation	128
4.79	Student calendar	129
4.80	Student exams calendar	129
4.81	student notifications	130
4.82	Mobile login	131
4.83	Professor profile	132
4.84	Professor schedule	133
4.85	Professor proctoring calendar	134
4.86	Mobile Apply request option	135
4.87	Professor notification	136
4.88	Student profile	137
4.89	Student schedule	138
4.90	Student exam calendar	139
4.91	Student notification	140
4.92	Courses conflict reduction across generations	143
4.93	professor session consolidation Score maximization across generations .	144
4.94	Subject sessions order score reduction across generations.	145
4.95	professors required number of sessions score across generations	146
4.96	Conflicts reduction across generations	147
4.97	One exam per day score across generations	148
4.98	Proctoring distribution variance across generations	149

List of Tables

1.1	Challenges and limitations of university scheduling	4
4.2	Parameter settings for courses scheduling algorithm	141
4.3	Parameter settings for exams planning algorithm	141
4.4	Results for Minimizing courses scheduling conflicts.	142
4.5	Results for Professors sessions consolidations	143
4.6	Results for Subject Sessions Order	144
4.7	Results for maximizing required professor sessions number score. . . .	145
4.8	Results for minimizing conflicts.	146
4.9	Results of maximizing single exam per day	147
4.10	Results for Minimizing the Proctoring Distribution Variance	148

Glossary

ACO Ant Colony Optimization.

AI Artificial Intelligence.

CNN Convolutional Neural Network.

CORS Cross-Origin Resource Sharing .

CP Constraint programming.

CSV Comma-separated values.

DEAP Distributed Evolutionary Algorithms in Python.

DNN Deep Neural Network.

EA Evolutionary Algorithm.

FCM Firebase Cloud Messaging.

GA Genetic Algorithm.

GAN Generative Adversarial Network.

HTTP Hypertext Transfer Protocol .

ILP Integer Linear Programming.

LP Linear Programming.

MOGA Multi-Objective Genetic Algorithms.

NSGA-II Non-dominated Sorting Genetic Algorithm II.

NumPy Numerical Python.

ORM Object Relational Manager.

PHP Hypertext Preprocessor.

PSO Particle Swarm Optimization.

RL Reinforcement learning.

RNN Recurrent Neural Network.

SGA Standard Genetic Algorithm.

SP Scheduling Problems.

UCSP University Class Schedule Problem.

UI User Interface .

UML Unified Modeling Language.

General introduction

General introduction

Managers across various professional fields frequently encounter the challenge of scheduling and planning work time for their employees. In hospitals, for instance, it is essential to coordinate the shifts of doctors and nurses to ensure adherence to hospital regulations. In businesses, planning involves balancing resources such as vehicles and employees, along with timing constraints to complete tasks at designated times. Effective planning is crucial for minimizing production costs, enhancing service quality for customers, and maintaining a high quality of life for employees. Similarly, in educational institutions, a major planning challenge is the allocation of time and scheduling various activities involving students and teachers. This task demands significant human and financial resources, and managing it manually is complex, requiring substantial time and effort. The process becomes even more complicated when constraints such as classroom and teacher availability must be considered. To address these challenges, we propose a new solution that primarily automates the scheduling of various activities, including planning courses, replacements, tests, exams, and graduation project assignments. Genetic algorithms have emerged as a promising solution, starting with an initial population and iteratively refining it to achieve optimal scheduling outcomes. The proposed solution is accessible through a multi-platform application, which is beneficial for students, teachers, and administrative staff, improving communication and enhancing various tasks. This manuscript is organized into four chapters.

- Chapter 1 provides an overview of the scheduling problem, emphasizing its importance in resource optimization and process efficiency. It introduces and categorizes different existing scheduling techniques, offering a comparative analysis to demonstrate their applicability and effectiveness in addressing diverse scheduling challenges.
- Chapter 2 covers the main concepts related to genetic algorithms and discusses their potential utilization for optimizing academic scheduling problems. The chapter showcases the effectiveness of genetic algorithms in solving complex scheduling issues, particularly in optimizing course and exam schedules.
- Chapter 3 focuses on the analysis and design of the proposed scheduling system

using Unified Modeling Language (UML). It includes use case, class, and sequence diagrams to illustrate the system's structure and interactions. The chapter details the design for professors, administrators, and students, highlighting their respective roles and functionalities. It concludes with a comprehensive specification summarizing the design and analysis process.

- Chapter 4 details the implementation and deployment of the scheduling system. It begins with an introduction to the development tools and technologies used, covering both back-end and front-end tools. The chapter then outlines the system architecture before describing the features and functionalities of the web and mobile applications. It also presents the results of the genetic algorithm scheduling, including the configuration and outcomes for course and exam scheduling. The chapter concludes with a discussion of future enhancements and improvements.

The thesis concludes with a general summary and conclusions.

Chapter 1

Scheduling problem

1.1 Introduction

In many different industries, scheduling plays an important and essential role for productivity and efficiency in a variety of settings, including production lines in manufacturing, computer operations, and shift management in hospitals. This chapter explores scheduling as a necessity that is also a challenging area of problem-solving in operations research. First, we outline the meaning and history of the scheduling. After that, we go over the reasons why scheduling is frequently regarded as a difficult process as well as the different situations in which it arises. Then, we look at a variety of ways and techniques used to solve scheduling problems, differentiating between automated and manual, and give a simple and straightforward explanation on how each has been applied to solve the problem. Apart from reviewing current techniques, we examine pertinent studies and advancements in the field, offering a comprehensive overview of both historical and contemporary perspectives.

1.2 Schedule

Based on [11], a schedule is a detailed plan or timetable that contain a planned activities or events that are scheduled with the times and dates when they are supposed to happen. However, the nature of these activities varies depending on the institutional context. For a clear understanding of scheduling in our modern context, lets turn back the pages of history to explore its origins and development over time. The concept of scheduling has been a fundamental aspect of organization for ages, as previously mentioned in [12], we find that the Chinese general and strategist " Sun Tzu " wrote about scheduling and strategy from military perspective 2500 years ago. Along with numerous more events, we discovered that the transcontinental railways have been built for some 200 years, none of these activities or events would have been accomplished without some form of scheduling or without the understanding of activities and sequencing.

Since the schedules are essential for effectively organizing our activities and tasks, let's explore the reasons why scheduling is a problem and explore the obstacles encountered in the process.

1.3 Scheduling as a problem

Scheduling is a globally commune challenge that affect a lot of domains, specially academic institutions such as universities where it becomes a more complex process. The administrators required to allocate limited resources such as suitable classrooms and faculty staff with the necessities of a large students body while respecting and following a broad of academic regulatory guidelines. This task is additionally complicated due to the unexpected variable enrollments and particular faculty availabilities.

The table 1.1 outlines some of the challenges and limitations that appear during the university scheduling process :

Table 1.1. Challenges and limitations of university scheduling

Challenges	Limitation
Restricted accessibility of labs and classrooms, each with specific requirement	Limited adaptability of physical resources such as predetermined number of classrooms spaces
Unpredictable changes	Operating within financial limitation that impact the growth of facilities and technology investment
Coordinating overlapping sessions and optimize the sessions to ensure the balancing of time	Managing resistance to changes in scheduling practices from various entities
Dealing with the fluctuation in course enrollment and insure that there is enough section are attainable	Handling the enormous and complicated details in scheduling for important, complex facilities
Relaying on traditional scheduling ways that may cause issues since it is not completely meet complex need that come with it	Errors in scheduling can lead to double-booked rooms and conflicting class times, causing significant disruptions

Since the concept of scheduling is not new, many attempted to solve this issues using different methods and techniques to manually adjusting or automating the process aiming to minimize scheduling conflicts and overall improve the academic outcomes.

1.4 Scheduling techniques

In the process of creating schedules, we find different preferences some favor the automated techniques while the others prefer to create their schedules manually using the traditional methods. Due of these variations, it is helpful to understand how they differ from each another more precising about their strengths and weaknesses.

1.4.1 Manual techniques

In the scheduling process,we consider manual any method that relies on the physical intervention or direct manipulation of the human user, since these manual methods fully depends on the user's active involvement in managing and organizing the required tasks. Almost all the known manual methods required the direct user manipulation and involvement which means that they have the same characteristics regarding the time and effort required,we can categorise them based on the level of the visual representation that they provide.

1.4.1.1 None visual methods

These type of methods rely mainly on text based formats to organize schedules offering a straightforward and accessible approach to the scheduling process, such as writing down tasks,appointments and deadlines on paper, using simple to do lists, text documents, etc.

The lack of the visual representation of these methods make the process challenging, specially as the complexity of the schedule increases, without the ability to visualise the schedule it will be hard to allocate the resources effectively, prioritize some activities and identify dependencies between tasks. Additionally, these methods require a lot of effort when updating and maintaining schedules since its hard to keep track of changes that have been made. Overall, while the non visual methods has there benefits in terms of simplicity and accessibility, they are not an effective choice when it comes to managing complex schedules .

1.4.1.2 Visual techniques

In contrast to the non visual methods, these methods provide a graphical representation of the schedule, making it easier in terms of interpretation and management. For example, the Figure 1.1 shows how the calendar provide a graphical overview of the scheduled tasks, appointments and deadlines in a clear and organized way.

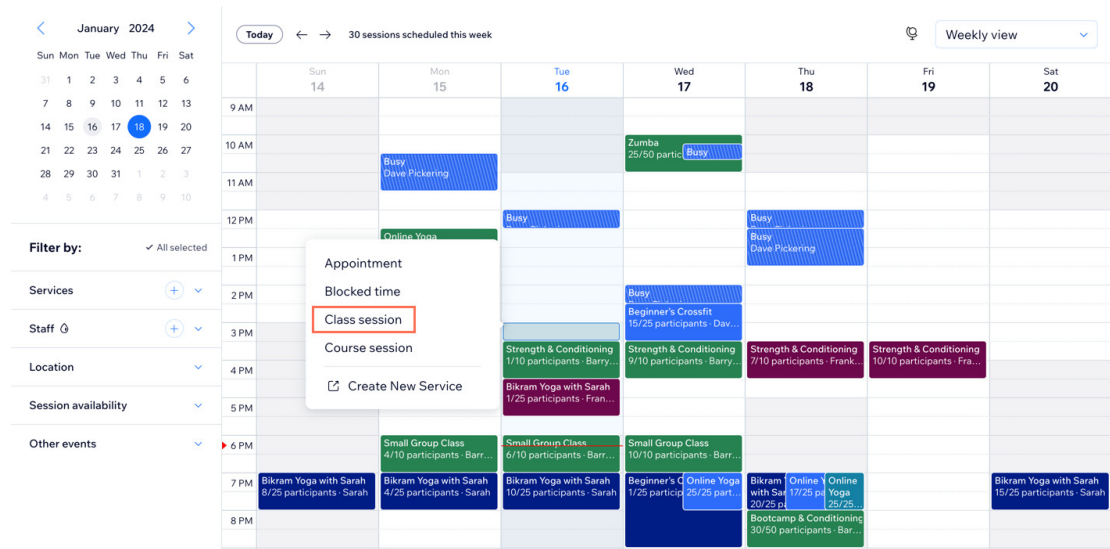


Figure 1.1. Booking calendar for scheduling [1]

Similarly, Gantt charts also are a visual method that represent the events and tasks in a timeline format showing all the important information like the duration and the dependencies. Overall, the visual methods offer in addition to the basic scheduling process clarity, communication, overlaps identification, but they still may not be efficient when the scheduling process become more big and complex.

1.4.2 Automated techniques

The shift toward developing and using the automated scheduling methods was mainly as a response to the limitations of the manual scheduling methods. These methods powered by the advancements in technology introduced data analytics, artificial intelligence and sophisticated algorithms into the scheduling problem enabling significant advancement over the manual methods, such as the ability of optimization of schedules with precision and speed , considering preferences, etc.

1.4.2.1 Mathematical optimization techniques

In analytics and decision science, mathematical optimization offers effective methods for identifying ideal solutions in a wide variety of applications in the realm of operational research and system design. Involves an objective function that is limited by a set of acceptable solutions and is susceptible of minimizing or maximization. For example, R.I kharbipov has published his article [13] in 2020, which was about how to automate the creation of the schedule at the university using the mathematical model, this research initially focuses on using the workload criteria in the selection of educational tasks. There are many various approaches in the field of mathematical optimization, each is well suited to a particular set of constraints and problem types.

Constraint Programming

As defined in [14], Constraint Programming (CP) is a powerful paradigm for solving complex problems. By using constraints between variables, it allows a direct and concise description of the problem. It offers rapid development, cost-efficient maintenance, and optimal performance as key advantages. Constraints are essential for identifying requirements and restrictions in planning and scheduling problems. These include resource availability, job dependencies, capacity limitations, time constraints, and quality standards. Incorporating these restrictions into mathematical models allows planners to ensure that plans and schedules meet all requirements and are feasible. The following figure 1.2 represent the paradigm constraint programming.

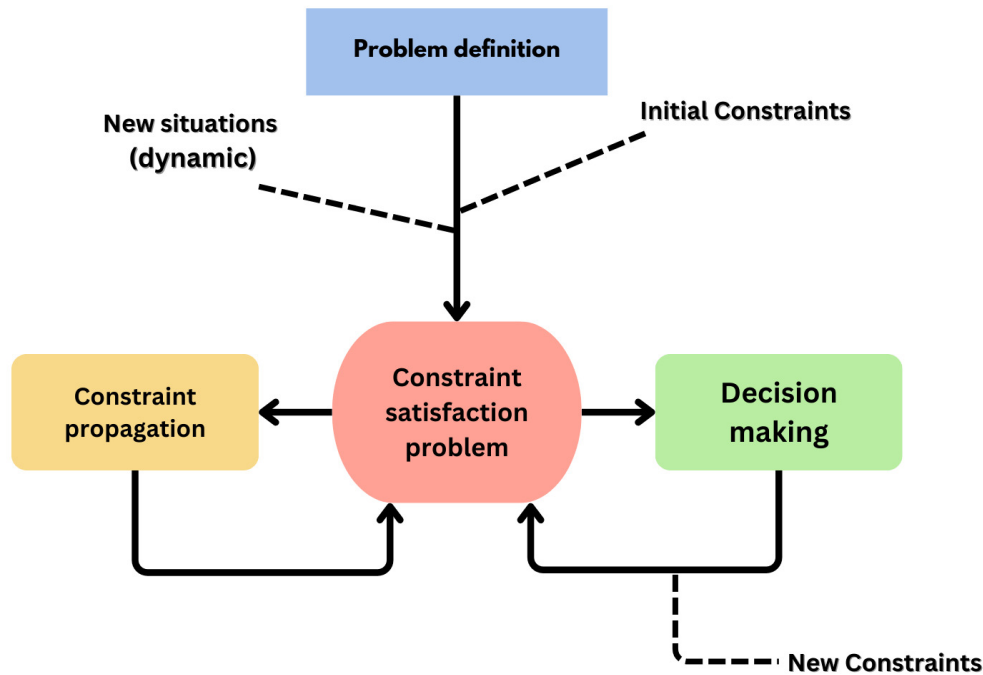


Figure 1.2. Constraint Programming Paradigm.

In scheduling, CP is sensitive to how constraints and decision variables are formulated. This sensitivity originates from the need to formulate constraints and decision variables carefully, handle large-scale and complicated problem instances quickly, and optimize solutions within realistic computation times.

Linear Programming

Another sub-field of mathematical programming is known as linear programming (LP). Noted in [15] as a quantitative analytical approach to decide whether or not to achieve the intended objective. In many fields, LP is considered the most important optimization technique, because it holds significant value and relevance across a wide range of applications, industries, and problem domains. Used generally to find the most ideal solution to the problem within a set of limitations. the technique consists of an objective function and linear inequalities, which depending on certain restrictions can take the form of equations or inequalities. Depending on the restrictions reflected in the linear relationship, this approach is used to maximize or reduce the objective function of

the provided mathematical model, the Figure 1.3 provides a clearer visualization of the main phases.

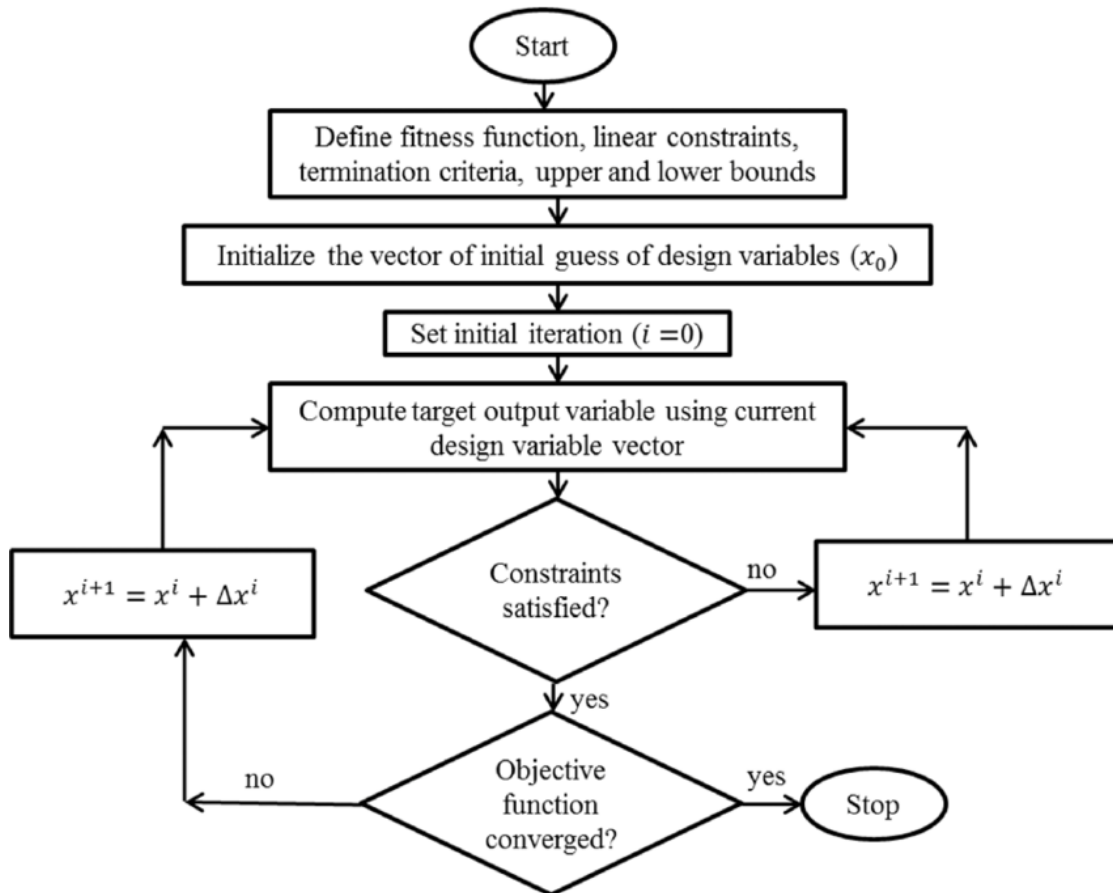


Figure 1.3. linear programming optimization techniques [1].

Linear Programming (LP) can be used to solve scheduling and planning problems by setting the objective, setting constraints, developing the LP model, generating solutions to the using LP solvers, evaluating the results and refining the model iteratively. All of these aspects are expressed as mathematical functions. This method enhances the efficiency of planning, task scheduling and resource allocation, making it effective in different areas, particularly in project management, production planning and logistics management. As an applications of this method, professors Ahmed Wasfy and Fadi A. Aloul as detailed in [16] presented how to solve the university class scheduling problem using advanced Integer Linear Programming techniques starting from how to formulate

the university schedule as an ILP problem. The ILP models are solved using advanced generic based and boolean satisfiability based ILP solvers.

Although the method works well for planning, it also has downsides, including assumptions about linearity and issues with discrete variables. For more complex problems, other methods may be better.

1.4.2.2 Heuristics techniques

Heuristics techniques especially in real-world applications, provide practical solutions to optimization problem that are too large or complex for precise algorithms. These techniques are distinguished by their adaptability and flexibility, and they are particularly valuable in finding solutions to NP-hard problems. Follows, The commonly used heuristic techniques.

Tabu search

Tabu search, as described in [17] is a strategy for solving combinatorial optimization problems whose applications range from graph theory and matroid settings to general pure and mixed integer programming problems.

This algorithm altered the scheduling problem by introducing a more efficient approach that elevated and improved the quality of solutions, using a memory based strategy that keep track of all the visited solutions (Tabu list), so they will be avoided when looking for the near optimal one, improving the quality of the found solution. Figure 1.4 shows the flowchart of a Standard Tabu Search.

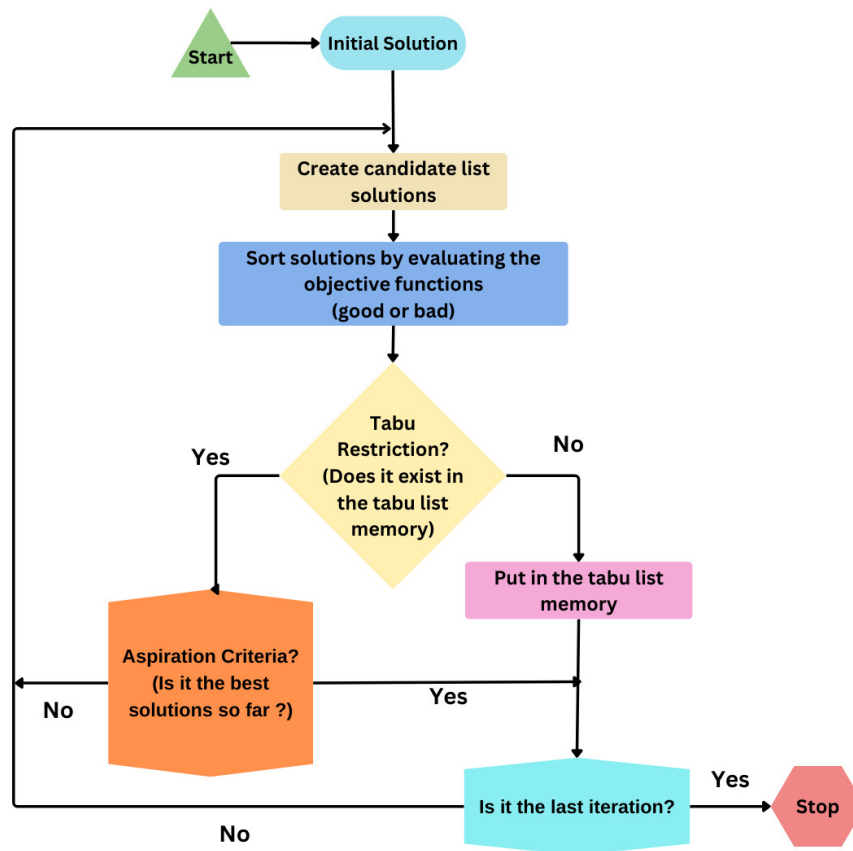


Figure 1.4. Flowchart diagram of Standard Tabu Search

However, the reliance of the algorithm on memory can lead to high memory requirements, specially with large scheduling problems, making it harder to maintain and update the Tabu list, increasing the processing time which is considered unpractical for dynamic and real-time scheduling environments.

Ant colony Optimization (ACO)

Ant colony Optimization is a probabilistic search technique, that simulate the foraging behavior of an ants colony in the nature to discover the shortest path to the food by leaving behind pheromones that motivate the other ants to follow their path. Which considered as indirect form of communication as referenced in [18, 19].

Some researchers used this technique as a solution to the scheduling problems by employing ACO as a graph-based representation where nodes represent the tasks of the SP

and the possible scheduling decisions are illustrated as edges. This algorithm used the population of the artificial ants to navigate through the graph enabling the colony to dynamically explore and evaluate different scheduling options within the solution space. For more clarity the next figure 1.5 describe the steps of the ACO algorithm.

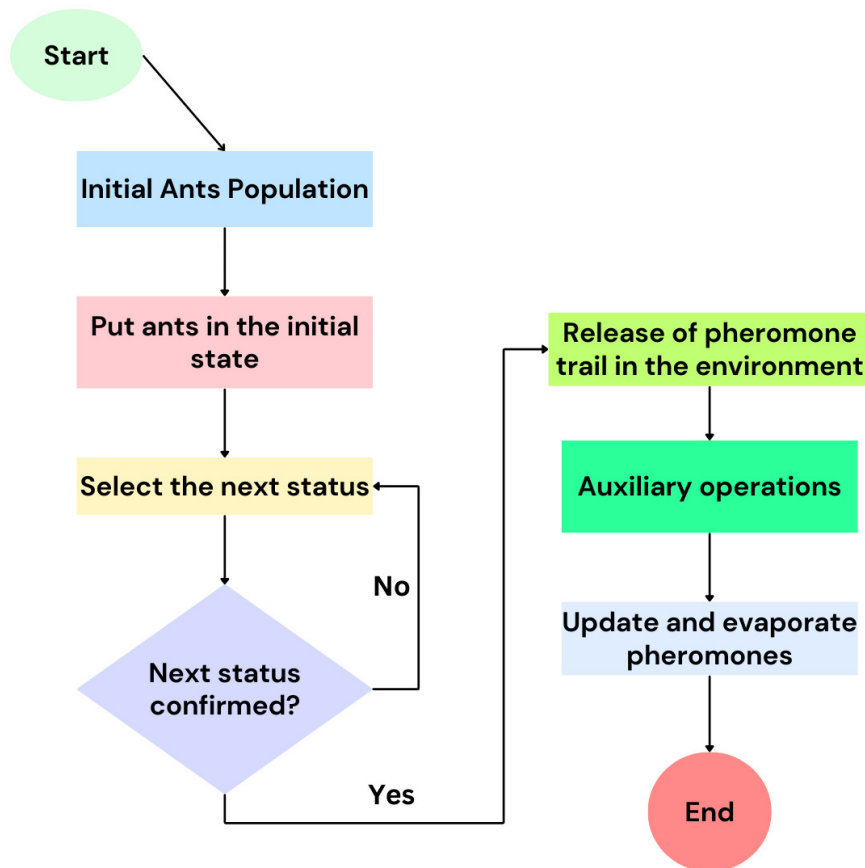


Figure 1.5. Ant Colony Optimization Algorithm

In this case, the authors of the article [20] used the ACO-based algorithm for solving the airline crew scheduling problem. In addition, Edson Flórez¹, Wilfredo Gómez² and MSc. Lola Bautista described the implementation of an ACO algorithm model to find a solution to Job Shop Scheduling Problem, same as [21] which used this approach for the resource-constrained project scheduling problem.

ACO algorithm in SP is aiming to focus on effectively distribute workload across resources in order to assign sessions and tasks to specific time and space, leading to reduce conflicts and enhance efficiency.

Although ACO is an effective technique for addressing SP, there are also some limitations such as the sensitivity in parameter settings and possibly may not scale well with complex problems, making the solving process expensive.

Simulated Annealing

The simulated annealing algorithms as mentioned in [22] is one of the preferred heuristic methods for solving the optimization problems, used to find approximate solutions. It mimics the physical process annealing in metallurgy, where a low-energy crystalline state is attained by heating and then slowly cooling the material to minimize energy and reduce problems. When we are seeking to attempt to minimize (or maximize) an objective function f with respect to a set of variable, this technique is commonly used, the issues can range from scheduling task to circuits. This method's initial solution start with a preliminary resolution of the problem, this can be generated randomly or by applying some heuristic. For more clarity of how the simulated annealing algorithm work, the figure 1.6 represents a flowchart diagram of the process.

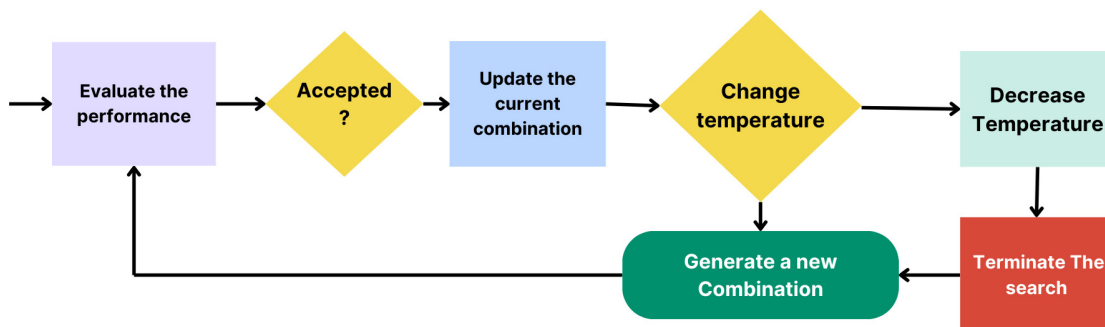


Figure 1.6. Flowchart diagram of simulated annealing algorithm

In the scheduling process, the algorithm begin with an initial schedule and continuously make small changes like swapping the order of tasks or adjusting resource allocations in order to explore the neighboring schedules. The algorithm accept a new schedule in every iteration made, based on multiple factors such as the the temperature parameter and a determined probability. The sensitivity of simulated annealing towards parameter selection, such as cooling schedule and initial temperature, is one of its main drawbacks. It can be difficult to find the ideal combination because these elements have

a big impact on performance, especially when handling complicated scheduling problem.

Particle Swarm Optimization (PSO)

Particle Swarm Optimization as defined in [23], is an algorithm inspired from the nature, especially from the movement and the behavior of animals and insects that are living in swarms such as birds and fish by keeping each individual's memory of the experience along with the data that his group provides, all in the process of determining the optimal food regions.

In this algorithm, an optimization problem's possible solution is shown as a "particle". In the solution space, every particle has a position and velocity, these particles come together to form a "swarm", each swarm is initialized by random positions within the solution space. The figure 1.7 provides a clear overview of the main phases that algorithm go through.

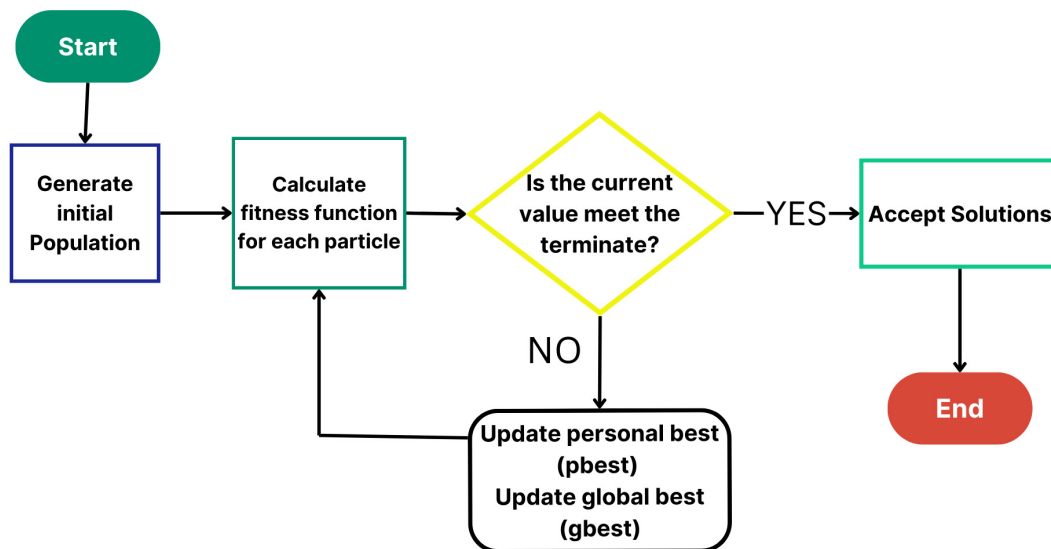


Figure 1.7. Flow diagram of PSO algorithm

PSO can effectively applied to scheduling problems across various domains such as project and task scheduling, along with their scheduling parameters like start time,

duration, resource requirements and dependencies are represented in SP. Within the framework of PSO, any potential schedule can be considered as a solution or a particle. In this regard, the authors of the article, Hossain, S. I., Akhand, M. A. H., Shuvo, M. I. R., Siddique, N. and Adeli, H [24] explain how PSO is significant when it applied to scheduling, who used this method with the help of Selective Search to optimize the university course scheduling problem. this proposed technique is different from many of the existing methods that are currently in use, such as PSO-based approaches, where UCSP is converted into an equivalent floating point numeric domain. PSO Selection Search approach uses a swap sequence based discrete PSO with a number of modifications.

Although This algorithm is effective in various optimization tasks. But, this does not imply that it immune to limitations such as having difficulty to deal with the complex scheduling problem's constraints, which could lead to insufficient solutions. Also, in large-scale scheduling scenarios with many tasks, resources and constraints, The PSO may not be able to scale well.

Genetic Algorithms (GA)

A genetic algorithm is a search algorithm that traverses a space of solutions in the quest for the optimal one, to solve a given problem. The way of handling this exploration is by producing a "population" of possible solutions, that are encoded as a chromosomes or string of genes, and allowing them to evolve through successive generations. They generate and evaluate potential solutions to scheduling problems by mimicking natural selection. The figure 1.8 illustrates the cycle of the genetics algorithms.

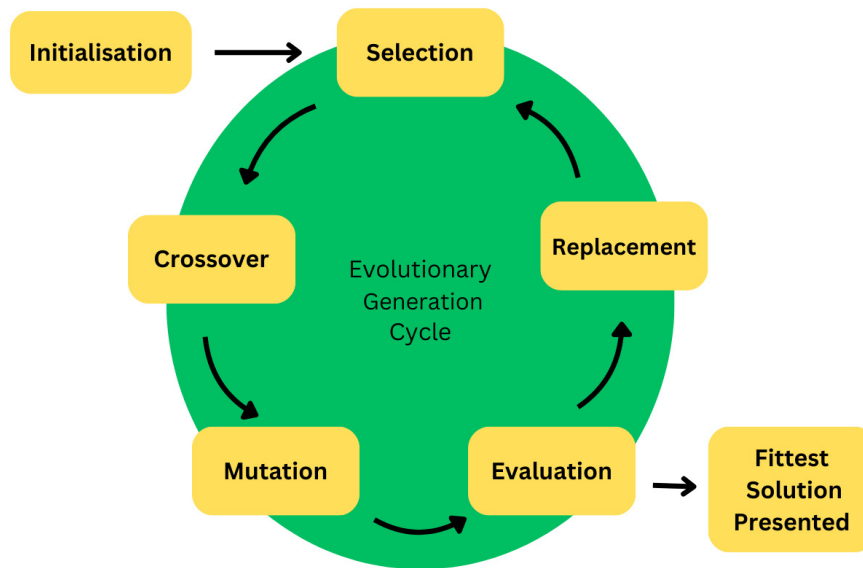


Figure 1.8. The evolutionary cycle of GA.

Genetic algorithms identify and enhance schedules that fulfill particular requirements, making them valuable tools for optimizing scheduling processes across various domains for their ability to handle complicated constraints by representing each potential schedule as a chromosome which can consist of genes that represent tasks or activities. Many researchers have used GA to solve scheduling problems; each one has developed their own technique for implementing this algorithm. Such as, Herth A.K. [25] has proposed in 2017 that a standard GA based on constraints tournament can be applied to solve the university course timetabling problem using a standard genetic algorithm based on constraints tournament, and Abdullah ELEN [26] found that depending on the constraints determined, the application of GA accurately creates exam scheduling without overlapping. GA(s) face challenges in terms of scalability, premature convergence, and adaptability to dynamic environments, but remain powerful optimization tools when adjusted and used appropriately for specific problems.

1.4.2.3 Machine Learning based techniques

Artificial intelligence (AI) has advanced significantly with machine learning, which provides powerful tools that let computers analyze and interpret large and complex amounts

of data to learn and make decisions. By implementing techniques that adaptively enhance their performance as the amount of data available for learning increases, which are critical in making predictions or recommendations.

Typically, there are three primary classification for this field like the figure 1.9 shows, which are individually tailored for certain data sets and objectives:

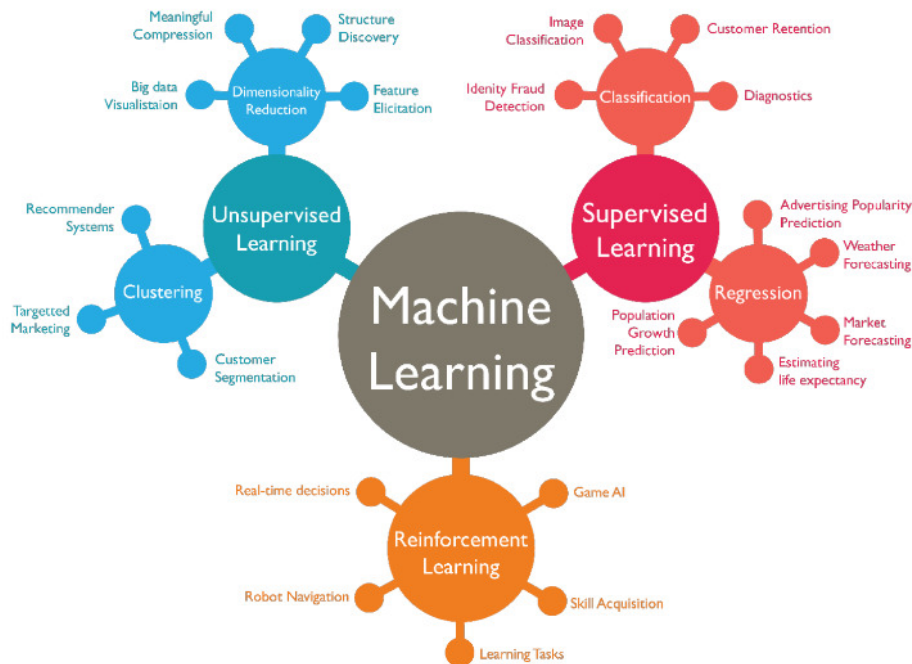


Figure 1.9. Illustration of Machine Learning Classification[2].

Supervised Learning

As defined in [27], Supervised learning is a machine learning method that relies on training data that includes both inputs and their corresponding outputs. This data is often called labeled or supervised data. The goal is to build a system that can predict output, based on new inputs by learning the relationship between inputs and outputs. If the outputs are discrete values, it leads to classification, while if the outputs are continuous, it leads to regression.

As supervised learning is used to address planning challenges, historical data is leveraged to anticipate future planning needs or to categorize tasks and assignments based

on different criteria such as priority or Resource requirements Before predictions or classifications can be generated from labeled data, models must be trained beforehand. This leads to improved decisions regarding scheduling and resource allocation. Applying supervised learning to scheduling problems can face various challenges. Constraints related to the availability of labeled data, Challenges of handling dynamic scheduling environments, Limited ability to generalize to unknown scenarios, Some difficulties include capturing complex planning links and possible model interpret-ability issues.

Unsupervised Learning

Unsupervised learning is a sort of machine learning that requires as little human supervision as possible to uncover data groupings or hidden patterns without utilizing labels. It's often known as self-organization, allows for the modeling of probability densities across inputs instead of supervised learning, which typically uses human-labeled data. In unsupervised learning, cluster analysis is used to group or segment data sets with similar features to identify algorithmic linkages. In scheduling and planning, unsupervised learning techniques are used to reveal hidden patterns, group similar tasks or resources, detect anomalies, reduce dimensionality, discover recurring patterns, and maximize resource allocation. This technique enable firms to make more informed decisions, increase productivity, and improve overall scheduling and planning operations through the examination of historical data.

This technique, for planning have several limitations, such as difficulty in finding clusters, uneven quality of clusters, scalability issues, insufficient optimization guidance, reliance on evaluation of data quality and tedious efforts to adapt to configuration changes. Nonetheless, they continue to be useful resources for uncovering hidden insights, especially when combined with additional techniques and in-depth domain knowledge.

Reinforcement Learning (RL)

As mentioned in [27], RL is a unique branch of machine learning from the previously cited ones. In reinforcement learning, an agent engages with the environment to optimize its gains by acting and gaining knowledge from its outcomes. Two techniques are used to compel the agent to act: (i) exploitation, in which the agent chooses to act based on its past experiences—including setbacks and mistakes—and (ii) exploration,

in which the agent decides to act in a way that is unrelated to its prior experiences. the technique is used for planning and scheduling, by encoding states that represent planning configurations, selecting actions such as task assignment or resource distribution, and creating rewards to promote desired results such as meeting deadlines or optimizing the use of resources. Through repeated policy learning, the algorithm improves decision strategies, harmonizing the search for new methods with the use of proven approaches to adjust to changing environments and maximize performance in different domains. The important methods in RL are Deep Q-Networks (DQN) for deep learning applications, Reinforce for policy gradients, and Q-Learning and SARSA (State-Action-Reward-State-Action) for value updates. Value and policy approaches are combined in Actor-Critic methods (A2C, PPO), etc.

Within this approach, limitations in scheduling problems manifest in managing complexity, computational demands, and reward formulation, requiring customized strategies and domain-specific adjustments for effective optimization.

1.4.2.4 Deep Learning Techniques

With a suite of powerful algorithms that emulate human brain activity, Deep Learning represent the next evolution in computational technology. Deep learning, is a sub field of AI that uses in several methods through neural networks structured layers of algorithm, each one of this methods has certain advantages and limitation. Here follows some essential deep learning techniques, which can be used in scheduling.

Deep Neural Networks (DNN)

DNNs are a class of deep learning that is used widely in data-driven modeling. And as clarified in [28] , they have layers with nodes and edges that reflect mathematical relationships. Training updates these associations iteratively via back propagation. These updated relationships function as the formulas used to predict output variables from input variables after training as shown in Figure 1.10. Consequently, the capacity of DNNs to capture intricate and nonlinear interactions found in a system is one of its main advantages. It has been used as an automation technique to the scheduling process specially for optimization and demand forecasting. DNNs reliably forecast future demand for resources or services by analyzing historical data and patterns and improve schedul-

ing efficiency by continuously fine-tuning resource allocation based on real-time data, guaranteeing optimal resource use. One major limitation of this technique is the large amount of labeled data required, also it can be computationally demanding, involving substantial computational resources for both training and inference.

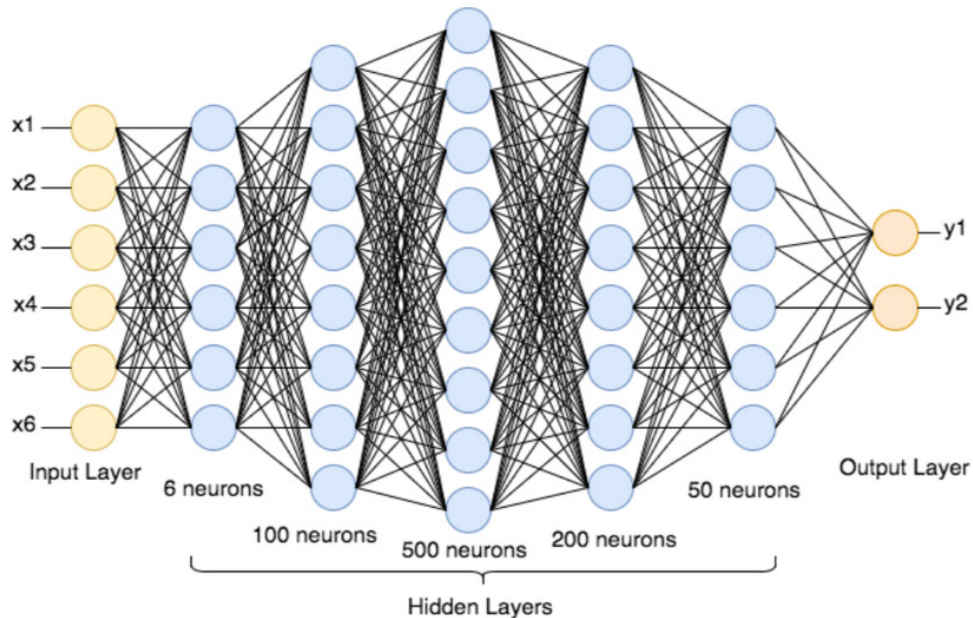


Figure 1.10. DNN Architecture [3].

Convolutional Neural Networks (CNN)

Convolutional Neural Networks, a class of deep learning algorithms created and used for processing structured grid data like IMG1. According to [29], and like its shown in the Figure 1.11, CNNs process input data by applying a number of convolutional layers. Every layer is made up of several filters, or kernels, that move across the input data to identify different characteristics like edges, textures, or patterns. A fully connected layers are employed after multiple convolutional and pooling layers to create the final predictions. In the scheduling process, CNNs convert the data into a format that is structured and easy to interpret, like grids or sequences. In order to identify patterns and correlations in the data, such as time task sequences or spatial resource distributions. As mentioned in [30], a CNN consists of one output layer, many hidden layers, and input layers. The data is sent to the input layer to begin training. Subsequently, the

information is routed through many concealed levels. Multiple filters and pooling layers are included in the hidden layer. Better features are produced when these layers are added between concealed layers. Through the process of identifying patterns, and then forecast the best times to assign tasks or resources, reducing the likelihood of delays and improving system performance as a whole. Numerous fields, including manufacturing, energy management, and job shop scheduling, can benefit from this broad strategy. But due to their computationally demanding nature and need for vast quantities of labeled data for training, CNNs may not be as appropriate for real-time or extremely dynamic scheduling scenarios.

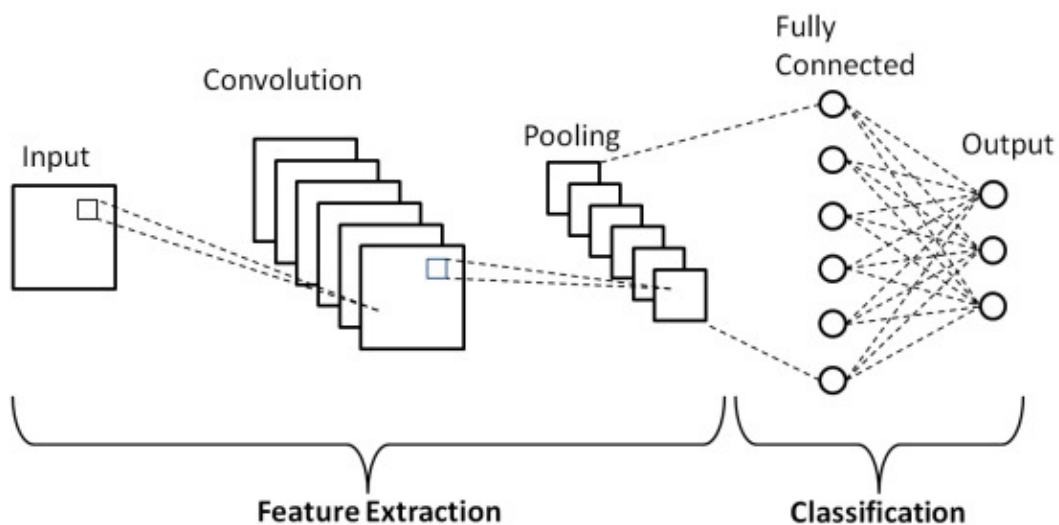


Figure 1.11. Basic CNN architecture [4]

Recurrent Neural Networks (RNN)

Recurrent Neural Networks are defined in [29] as a type of artificial neural networks, designed to identify patterns in data sequences like speech, text, or time series. RNNs, in contrast to conventional neural networks, feature connections that create directed cycles, which preserve information, due to their ability to retain a "memory" of prior inputs in a sequence. Utilizing past scheduling data, RNNs learn to identify patterns and dependencies among tasks in scheduling processes. They are trained to comprehend the order of tasks and their interdependences, retaining a "memory" of prior tasks during scheduling, enabling it to automatically determine the optimal next task while taking

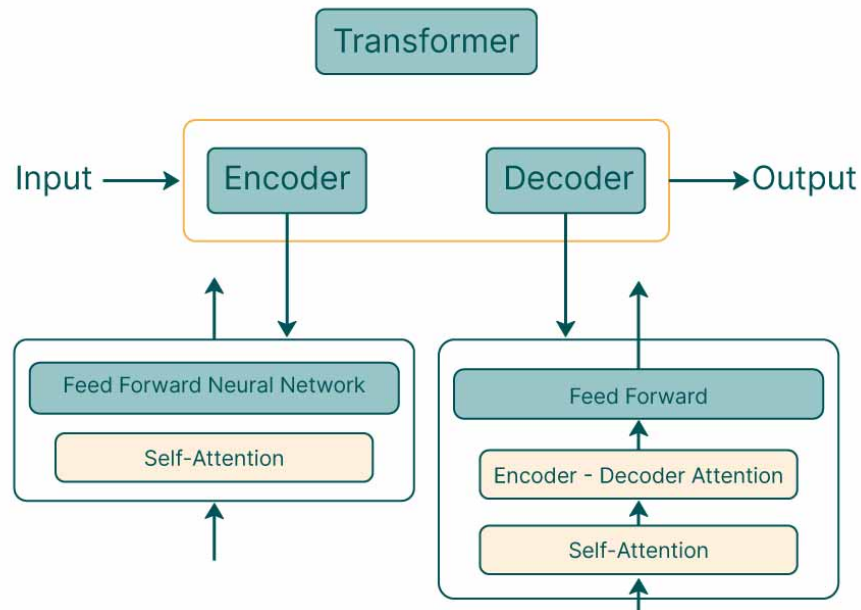


Figure 1.12. Standard Transformer model [5].

into account the limits and circumstances of the moment. By minimizing delays and optimizing resource usage, this aids in the creation of timetables that are more flexible and efficient. Unlike CNNs and DNNs, RNNs use their design to handle temporal data, which makes them especially useful for tasks involving sequences.

Transformer Models

Transformer models according to [31], are also a type of the neural networks, intended for processing sequential input. Compared to earlier models like RNNs, they are better at capturing long-range dependencies because they analyze incoming data in parallel using a method called "self-attention".

The encoder and decoder, the two components of the transformer, are shown in the figure 1.12. For the model to process and produce responses based on the complete input sequence at once, as opposed to one element at a time, each block in the encoder and decoder is essential. Transformer models have been adapted to the scheduling process by applying their self-attention mechanism, which enables them to evaluate every aspect of a schedule simultaneously rather than one component at a time. This is especially help-

ful when scheduling complex tasks and resources, as different activities and resources interact in complex ways. By using Transformers to assess the entire schedule collectively, the model may determine the optimal times and sequences for tasks depending on overarching objectives like avoiding downtime or balancing workloads. This all-encompassing strategy aids in the creation of more adaptive and effective schedules, particularly in settings where conditions change quickly.

Generative Adversarial Networks (GAN)

According to [32], Generative Adversarial Networks are a class of artificial intelligence algorithms that are carried out by a system of two neural networks in a zero-sum game framework, competing with each other. In 2014, Ian Goodfellow and colleagues first introduced this technique. Simply, the generator network mimics real data instances to learn how to generate data, while the discriminator network tries to separate real data from fake. Because of the rivalry, both networks progressively improve their methods, making it possible to produce realistic-looking synthetic data. GANs have been successfully applied to optimize scheduling. For example, They drive deep reinforcement learning in large computing environments to improve task scheduling in giant computer systems, increasing resource allocation efficiency significantly. In the energy sector, GANs are used in a similar fashion to offer accurate wind power scenarios that facilitate day-ahead scheduling and enhance energy distribution planning. These illustrations demonstrate the great adaptability of GANs as a modeling and optimization tool for complex scheduling tasks across multiple industries, making them a significant asset for planning and forecasting. However, they are limited by the large amounts of data and the processing power they require, and problems like mode collapse can make optimization difficult.

1.4.2.5 Rule-Based techniques (Expert Systems)

The rule-based systems, also known as the expert systems, were among the earliest automated scheduling methods. Mentioned in [6] as approaches that solve issues or make judgments based on predetermined, understandable rules. These systems as it is shown in figure 1.13 composed of three main components. A knowledge base, which stores the rules and facts used for decision-making, an inference engine, which applies the

rules from the knowledge base to specific situations to derive conclusions or decisions, and a knowledge acquisition module, which is responsible for gathering and updating the knowledge base with new rules and information. They are frequently used to automate and expedite decision-making in a variety of procedures. Among the most popular strategies of rule-based systems are those that prioritize tasks by specific criteria, such as priority rules, which rank tasks based on the earliest deadline or shortest processing time. Also, there is dispatching rules which determine the sequence in which tasks are processed on machines, while batch scheduling rules enhance efficiency by grouping related tasks together for processing.

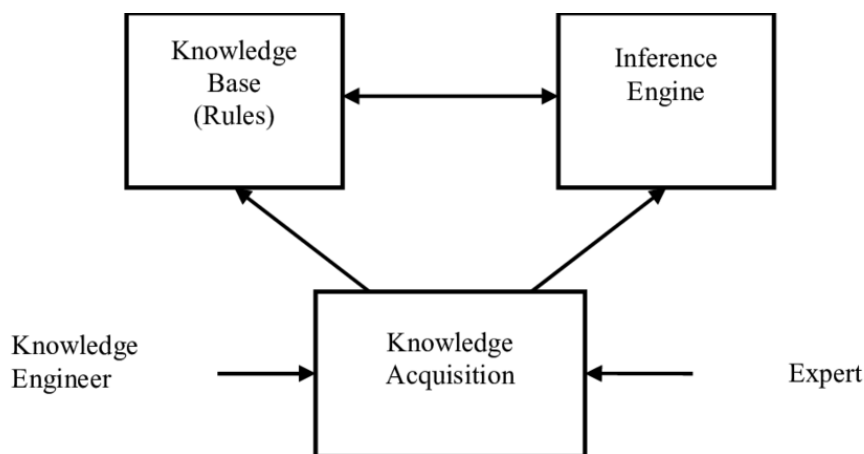


Figure 1.13. Basic architecture of a rule based system [6].

The simplicity of expert systems in implementation make them unique, simple to comprehend, and adaptable to certain operational requirements. However, they are limited as the set of the rules cant cover all the scheduling scenarios, or readjust well to the changing circumstances and conditions.

1.5 Conclusion

In this chapter to cover the scheduling problem and the numerous techniques used to address this complex challenge, from sophisticated machine learning to traditional manual approaches. Giving a general idea on how each technique has been specifically applied to the problem and mention the main limitations of each one. The goal is to provide

a basic understanding for selecting the most effective strategy based on the specific requirements and constraints of different scheduling scenarios. For our specific scenario, which is scheduling university courses and exam sessions, we decided to go with GAs as our preferred approach. This decision is based on their ability in handling multiple constraints and objectives simultaneously. They excel in adapting to the complex and dynamic nature of the academic scheduling, where a lot of factors must be balanced efficiently such as room and professor availability, and the needs of both professors and students.

Because GAs are more flexible, efficient, and adaptable than other options, they are a better option for university scheduling tasks. In the next chapter, more explanation on how we used the GAs as a solution to the academic scheduling specifically university courses/exams scheduling, exploring the detailed setup and implementation of them demonstrating how they deal with the challenges, constraints and limitations of academic scheduling.

Chapter 2
Genetic Algorithm : Academic
Scheduling Optimization

2.1 Introduction

In this chapter, we will explore the fundamental components of Genetic Algorithms (GAs), such as genes, chromosomes, and fitness functions. We will explain the roles of various phases, including initialization, selection, crossover, and mutation, and discuss how each contributes to the algorithm's convergence. Additionally, we will classify Genetic Algorithms into different types, highlighting their specific applications. Finally, we will focus on the application of GAs to academic scheduling, discussing effective problem representations and the practical implementation of these algorithms to solve our scheduling challenges.

2.2 Evolutionary algorithms (EAs)

According to [33], evolutionary algorithms (EAs) are stochastic search based techniques which use a population of solution to. They mimic natural evolution by employing mechanisms such as selection, crossover, and mutation to evolve solutions over generations. Points in the search space are considered as individuals, or potential solutions, which collectively constitute a population. Their fitness rating is a numerical representation of their suitability for the given objective. EAs can include three key elements in addition to initialization and termination, which are the core elements of all algorithms: a set of search operators (typically implemented as "recombination" and "mutation"), an enforced control flow, and a representation that links suitable variables to feasible solution candidates. The figure 2.1 illustrates the different families of Evolutionary Algorithms, including Evolutionary Programming, Genetic Algorithms (which encompass Genetic Programming), and Evolution Strategies.

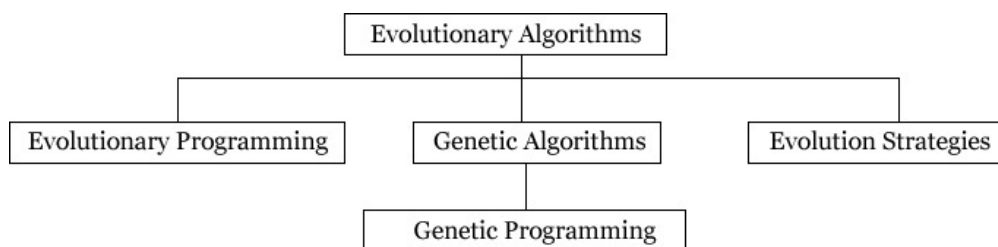


Figure 2.1. Evolutionary Algorithms [7].

2.3 Genetic Algorithms (GA)

Genetic Algorithms are considered as a sub-class of Evolutionary Algorithms as shown in the previous section, which means that EA is an umbrella term includes all the Algorithms inspired by the idea of Darwinian evolution. In addition, according to [34, 35], the term Genetic algorithm, universally abbreviated now into GA, is a search strategy that is inspired by Neo-Darwinism, Darwin's classical theory of evolution, aside with Weismann's natural selection theory and Mendel's concept of genetics. All are theories of natural evolution that is based on processes of reproduction, mutation, competition and selection which are an essential property of life. It was first introduced to the world by John Holland in 1975, which considered as a pivotal year in the development of GA, explaining for the first time its basic principles.

2.4 Components and processes of Genetic Algorithms

Before starting the discussion on how we applied the GAs in our proposed scheduling system, it is essential to clarify and explain some key concepts and standard structure that defines the GA.

2.4.1 Basic concepts of GA

2.4.1.1 Genes

A gene in a GA is the smallest unit of information, representing a single parameter or value of a potential solution to the problem. Binary strings are the most common way to represent genes. However, there are alternative ways to encode them as well, depending on the situation. In Figure 2.2, the gene is depicted within a binary string. Each position

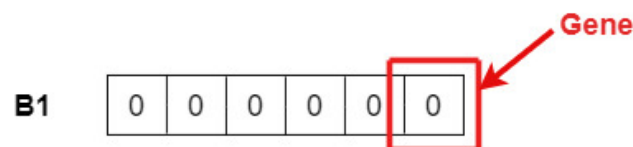


Figure 2.2. Gene.

in the string represents a bit, which can take the value of 0 or 1. The highlighted section indicates a single gene within the chromosome.

2.4.1.2 Chromosomes

A chromosome is formed by an array of genes, representing a possible solution to the problem. GA makes them evolve through generations using genetic operators, namely crossover and mutation, until finding the best solution (fittest chromosome). In Figure

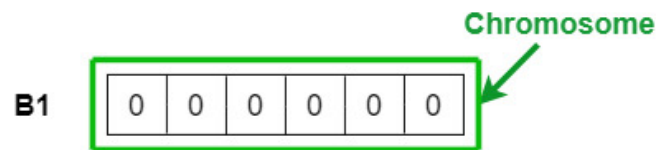


Figure 2.3. Chromosome.

2.3, the chromosome is depicted as a binary string composed of multiple genes. Each position in the string represents a bit, which can take the value of 0 or 1. The entire string represents a potential solution to the problem, with genetic operators acting on it to evolve and improve the solution over generations.

2.4.1.3 Population

A population is a set of chromosomes. As the GA progress, the population changes and evolves over the generations creating new populations.

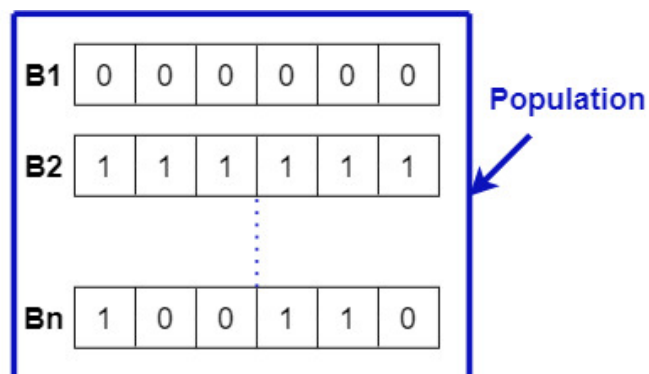


Figure 2.4. A generation of chromosomes.

In Figure 2.4, the population is depicted as a collection of chromosomes. Each chromosome represents a potential solution to the problem. Over successive generations, the genetic algorithm modifies the population through operations such as selection, crossover, and mutation, leading to the evolution of the population and the discovery of better solutions.

2.4.2 Phases of Genetic Algorithms

2.4.2.1 Initialization

The first step in the GA process is population initialization [36]. This involves generating a group of solutions, also known as a set of chromosomes, for the current generation. The initial population, P_0 , is typically created randomly. In subsequent generations, new populations, P_t , are formed for each generation. t (where $t = 1, 2, \dots, n$). There are various types of population initialization techniques, each with different characteristics. As mentioned in [8], a new categorization is introduced that encompasses all existing population initialization techniques. This proposed classification groups the techniques to three distinct and easy to understand perspectives. These aspects are randomness, compositionality, and generality. Within each of these three categories, techniques are further classified into sub-categories, and their properties are described in detail. Figure 2.5 illustrates this hierarchy.

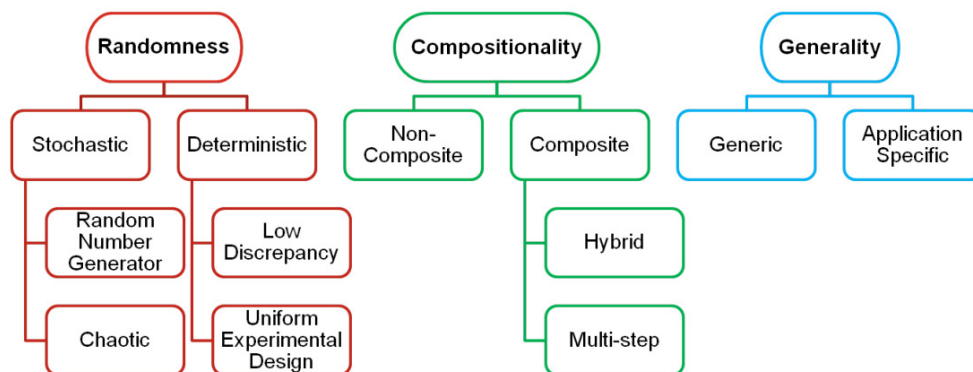


Figure 2.5. Population initialization techniques in GAs [8].

- **Randomness:** Refers to whether the initialization method involves random processes. It determines if the method uses random elements or follows a fixed, predictable pattern. It includes two main sub-classes of techniques, stochastic and deterministic techniques.

Stochastic techniques: They incorporate random elements, including Random Number Generators which use random numbers to create populations, and chaotic systems that introduce randomness through chaotic behavior.

Deterministic techniques: Techniques that do not depend on randomness, including Low Discrepancy sequences that ensure evenly distributed points, and Uniform Experimental Design that provides a structured sampling of the solution space.

- **Compositionality:** Refers to the number of independent procedures used in a technique. Based on this, population initialization techniques are classified as either composite or non-composite.

Non-Composite: Basic techniques that generate populations in a single step are called non-compositional. This includes methods that are stochastic, deterministic, generic, or application-specific but cannot be broken down into separate population initialization methods.

Composite: Involves using multiple methods to improve the initialization, such as Hybrid approaches that integrate different techniques for combined strengths, and Multi-step methods that apply a sequence of techniques in stages.

- **Generality:** Refers to the versatility of a population initializer across various domains. It indicates how broadly the initializer can be applied to different types of problems and situations.

Generic: Techniques that can be applied to all types of optimization problems are known as generic techniques. These methods generally assume that the given optimization problem is a black-box puzzle.

Application-Specific: Tailored for particular real-world problems, using domain knowledge to improve results and speed up the convergence of EA. They are effective for targeted problems but may not work well elsewhere.

2.4.2.2 Selection

This phase involves selecting individuals according to their fitness from the current population. Fittest individuals have more chance to be selected. This phase includes two stages.

- **Parent Selection:** Choosing individuals from the current generation to reproduce and generate offspring.
- **Survivor Selection:** Selecting individuals from both the parents and their offspring to form the next generation.

The various parents selection techniques used in genetic algorithm are illustrated in the figure 2.6.

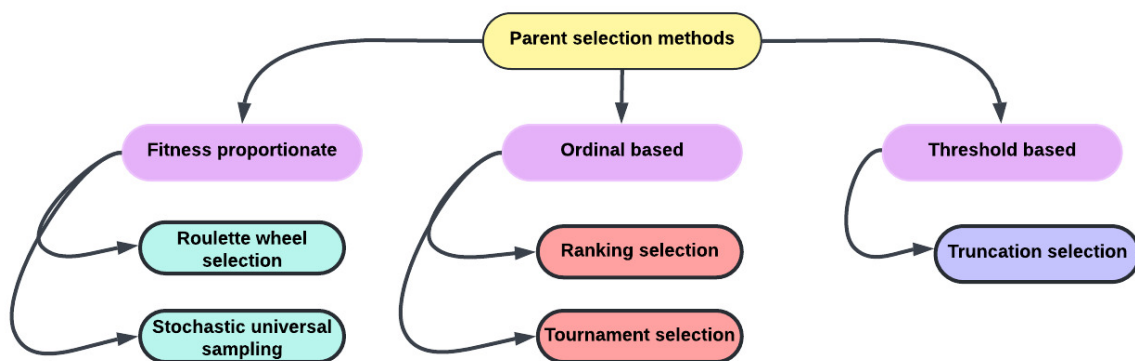


Figure 2.6. Parent selection methods .

This figure 2.6 categorizes parent selection methods in genetic algorithms into three main groups: fitness proportionate, ordinal based, and threshold based. Fitness proportionate methods, such as Roulette Wheel Selection and Stochastic Universal Sampling, select individuals based on their fitness levels. Ordinal based methods, including Ranking Selection and Tournament Selection, rank individuals or have them compete in tournaments to determine selection. Threshold based methods, like Truncation Selection, only allow individuals above a certain fitness level to be selected. This classification highlights the various techniques used to choose parents for reproduction in genetic algorithms. Survivor Selection decides which individuals will be removed and which

will remain in the next generation. This process is also referred to as replacement. As illustrated in the figure below .

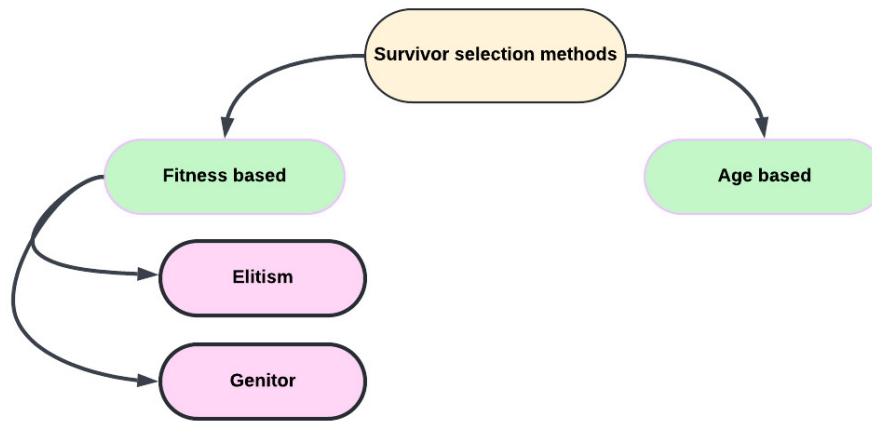


Figure 2.7. Survivor selection methods.

The figure 2.7 illustrates survivor selection methods used in genetic algorithms, including age-based and fitness-based approaches. In age-based selection, individuals are removed after a set number of generations, while fitness-based selection replaces the least fit individuals with new offspring using techniques like tournament selection. Elite preservation (elitism) ensures the best chromosomes are copied to the next generation, enhancing performance, and the GENITOR method (steady-state selection) continuously replaces the least fit individuals, promoting gradual population improvement, thus ensuring a balance of new and high-quality individuals in each generation.

2.4.2.3 Crossover

As defined in [37], the crossover operator is similar to the biological process of crossing over and recombination of chromosomes during cell meiosis. This operator exchanges a subsequence between two selected chromosomes to produce two offspring. Based on the unique requirements and characteristics of the problem at hand, various crossover techniques can be utilized. Each technique has its own approach to combine parental genes, affecting the algorithm's convergence and efficiency. Below, we explore the different types of crossover techniques.

Single Point Crossover

A crossover point is chosen on the parent organism string, and all data beyond this point is exchanged between the two parent organisms. This method is characterized by positional bias, as illustrated in the figure 2.8.

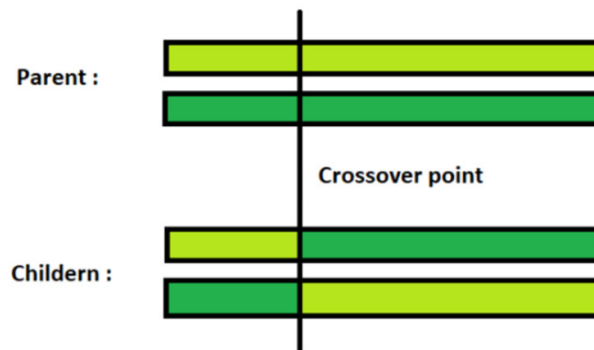


Figure 2.8. single point crossover illustration [9].

Two-Point Crossover

This is a particular example of the N-point crossover technique. As shown in the figure 2.9, two random points are selected on the individual chromosomes (strings), and genetic material is swapped at these points.

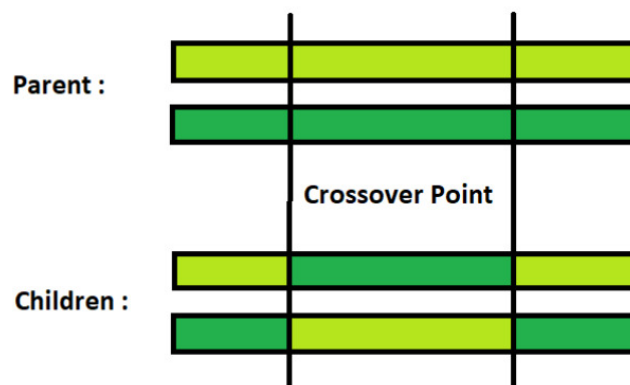


Figure 2.9. Two-point crossover illustration [9].

Uniform Crossover

As demonstrated in the figure 2.10, each gene is chosen at random from the corresponding genes of the parent chromosomes, using a method similar to tossing a coin.

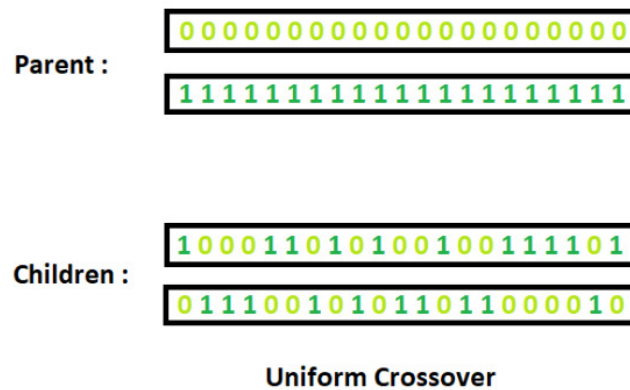


Figure 2.10. Uniform crossover illustration [9].

2.4.2.4 Mutation

As a key element in genetic algorithms, mutation handles the exploration of the search space and it is crucial for convergence, unlike crossover. It involves making small random changes to a chromosome to generate new solutions, thereby maintaining and introducing diversity within the genetic population. Typically applied with a low probability to avoid turning the GA into a random search. Here, we discuss some of the most widely used mutation operators.

Bit Flip Mutation

As represented in the figure 2.11, in bit flip mutation, one or more random genes are selected and flipped. This technique is used for binary encoded genetic algorithms.

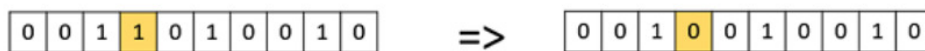


Figure 2.11. Bit Flip Mutation.

Random Resetting

An extension of bit flip mutation for integer representation is random resetting. In this method, a randomly selected gene is assigned a random value from the set of permissible values.

Swap Mutation

As illustrated in the figure 2.12, in this technique, two positions on the chromosome are selected at random, and their values are interchanged. This method is commonly used in permutation-based encoding.

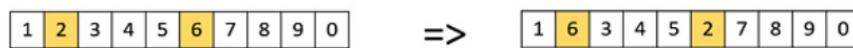


Figure 2.12. Swap Mutation.

Scramble Mutation

As visualized in the figure 2.13, scramble mutation is also popular with permutation representations. In this method, a subset of genes from the entire chromosome is selected, and their values are randomly scrambled or shuffled.

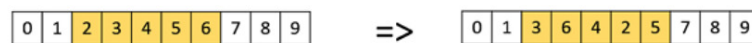


Figure 2.13. Scramble Mutation.

Inversion Mutation

As detailed in the figure 2.14, this technique involves selecting a subset of genes, like in scramble mutation, but instead of shuffling, the entire string within the subset is reversed.

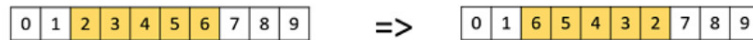


Figure 2.14. Inversion mutation.

2.4.2.5 Evaluation

As defined in [38], the fitness function acts as the evaluation metric that steers the genetic algorithm towards finding the optimal solution. It assesses and quantifies the quality and suitability of each individual solution within the population. By assigning a fitness score to each candidate, the function determines how well each solution meets the objectives and adheres to the constraints of the optimization problem. Higher fitness scores represent more effective solutions, enabling the algorithm to recognize and prioritize the most promising candidates for further evolution and reproduction. This ongoing process allows the algorithm to progressively improve the population, steadily moving closer to the optimal solution with each generation.

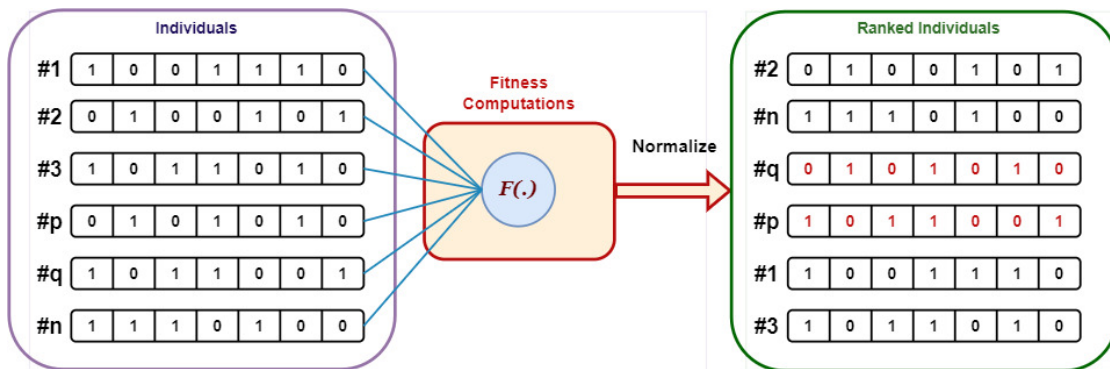


Figure 2.15. Evaluation process.

The figure 2.15 illustrates the evaluation process within a genetic algorithm. On the left side, we have a population of individuals represented by binary strings (chromosomes). In the middle, the fitness function F evaluates each individual. This function assigns a fitness score based on how well each solution meets the problem's objectives and constraints. On the right side, individuals are ranked based on their fitness scores. The individuals with higher fitness scores are considered better solutions and are prioritized for further evolution and reproduction.

2.4.2.6 Generation

As shown in the figure 2.16, a generation in genetic algorithms (GAs) is defined as one complete iteration that includes all phases from initialization to the selection of offspring for the new population.

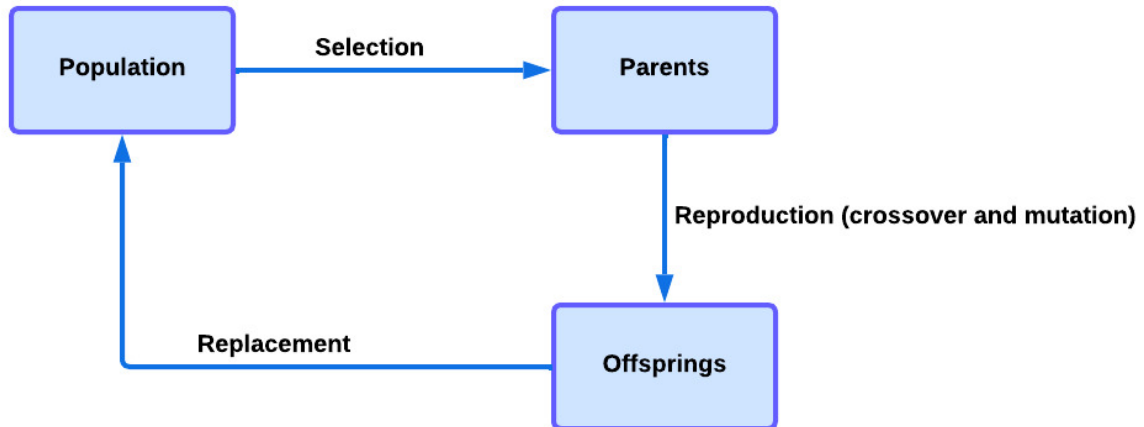


Figure 2.16. Generation in genetic algorithms.

2.4.3 Convergence in genetic algorithms

As [?] defined, convergence happens when every individual of a population become identical. Complete convergence is observed in genetic algorithms that apply just crossover. The population may settle on a sub-optimal solution, which is known as premature convergence. However, populations don't necessarily stabilize throughout time, with the best individuals sharing genetic and behavioral traits.

In order to optimize the efficiency of evolutionary computing, it is necessary to determine the critical variables that affect convergence. Important components including population size, recombination operators, reproduction strategy, and parameter representation are illustrated in the figure 2.17.

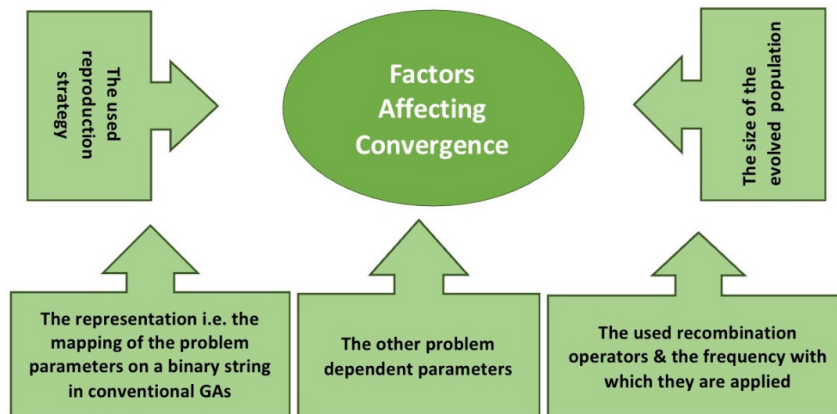


Figure 2.17. Convergence in genetic algorithms [10].

2.5 Genetic algorithms classes

Although there are several ways to classify GAs, such as genetic representation or selection techniques. This discussion will specifically focus on categorizing them based on objective types and elitism.

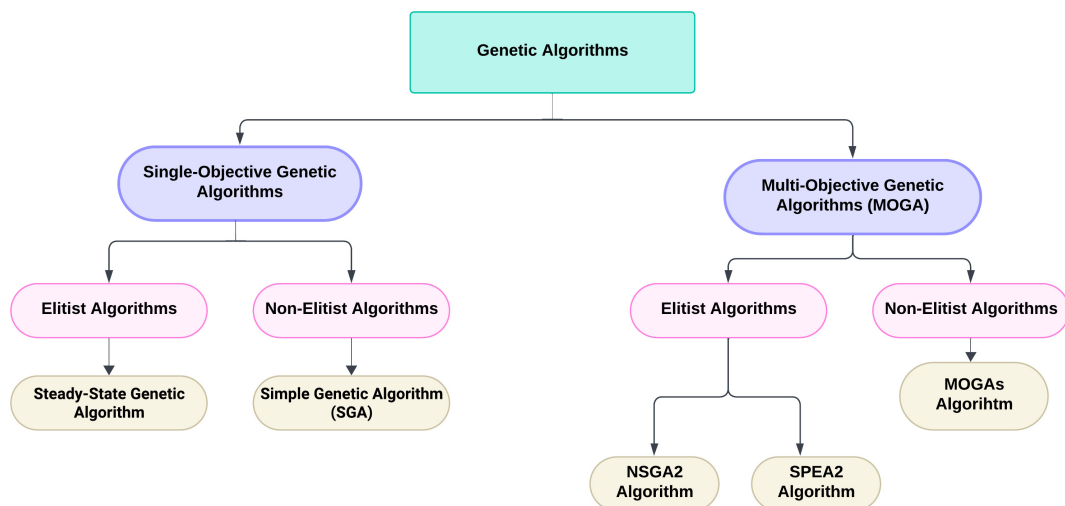


Figure 2.18. Classification of GAs Based on objective types and elitism.

As illustrated in the figure 2.18, we can see a systematic classification of genetic al-

gorithms (GAs) based on their optimization objectives and solution retention strategies, divided into Single-Objective Genetic Algorithms (SGA) and Multi-Objective Genetic Algorithms (MOGA). Each category is further categorized into elitist and non-elitist algorithms, with each type tailored to specific operational mechanics and strategic focuses.

2.5.1 Single-Objective Genetic Algorithms (SGA)

These algorithms optimize a single criterion, focusing on finding the best solution to either maximize or minimize that specific goal.

2.5.1.1 Elitist Algorithms:

Several elitist algorithms designed to optimize solutions are classified as single-objective genetic algorithms. These algorithms take different approaches, but they all aim to preserve high-quality solutions across generations. While there are various algorithms in this category, the Steady-State Genetic Algorithm is particularly well-known. These algorithms use different strategies to retain the best individuals in the population, increasing the efficiency of the genetic search process.

Steady-State Genetic Algorithm

Maintains a constant population size by selectively replacing only a few individuals at a time, thereby ensuring the retention of the best individuals. This approach effectively prevents the loss of optimal genetic solutions, keeping high-quality genes within the population across generations.

2.5.1.2 Non-Elitist Algorithms

Non-elitist algorithms, on the other hand, do not preserve the best individuals across generations. Instead, they concentrate on the fundamentals of genetic processes. These algorithms can be more exploratory, providing a wider range of solutions while risking the loss of high-quality solutions. The Simple Genetic Algorithm (SGA) is a prominent example in this category.

Simple Genetic Algorithm (SGA)

Operates on the basic principles of selection, crossover, and mutation, cycling through generations without explicitly preserving the best solutions. This method focuses on exploring the genetic landscape broadly, which may result in the loss of good solutions but simultaneously enhances the diversity of the gene pool. The Standard Genetic Algorithm (SGA) can be described by the pseudo-code given by the algorithm 1.7

Algorithm 1 Pseudo-code for Simple Genetic Algorithm (SGA)

- 1: Choose an initial population P_0 of individuals;
 - 2: Evaluate the fitness of all individuals;
 - 3: Choose a maximum number of generations: t_{\max} ;
 - 4: **while** not satisfied and $t < t_{\max}$ **do**
 - 5: $t \leftarrow t + 1$;
 - 6: Select parents for offspring production;
 - 7: Apply reproduction and mutation operators;
 - 8: Create a new population of survivors: P_t ;
 - 9: Evaluate P_t ;
 - 10: **end while**
 - 11: Return the best individual of P_t ;
-

The algorithm apply the basic steps of GAs. Starting by generating the initial population of individuals P_0 , evaluating and assigning a fitness value to each individual. Then it starts looping until a satisfaction criterion is met or a maximum number of generations, t_{\max} , is reached. In each iteration, parents are selected from the existing population to become the next one's parents, and genetic processes like reproduction and mutation are used to create new offspring. The fitness of this newly created population P_t , is then assessed. When the procedure has completed running the loop, it returns the best individual from the final population, which represents the optimal solution found during the process, ensuring that the solutions will continue to be improved.. The (SGA) is commonly utilized in function optimization, and to help with feature selection and parameter tuning for a variety of models, including as classifiers and regressors, in elementary machine learning tasks. SGA is a favorite in educational and experimental contexts to demonstrate the fundamentals of evolutionary algorithms since it also functions as a key tool in basic evolutionary computing assignments.

2.5.2 Multi-Objective Genetic Algorithms (MOGA)

These algorithms handle multiple, often conflicting objectives, aiming to find a Pareto front of solutions where each is optimal without dominating the others in all criteria.

2.5.2.1 Elitist Algorithms

Elitist algorithms in multi-objective genetic algorithms, similarly to their single-objective counterparts, aim to maintain high-quality solutions across generations. Ensuring that the best solutions are retained, they preserve a diverse and high-quality set of options. Two well-known elitist algorithms in this category are the Non-dominated Sorting Genetic Algorithm II (NSGA-II) and the Strength Pareto Evolutionary Algorithm 2 (SPEA2).

Non-dominated Sorting Genetic Algorithm II (NSGA-II)

As mentioned in [39] The population is initialized and sorted into fronts based on non-domination. The first front is fully non-dominant, with subsequent fronts dominated by previous ones. Individuals are assigned fitness values based on their front, starting with 1 for the first front. A crowding distance parameter, measuring proximity to neighbors, is also calculated. Larger crowding distances enhance population diversity. Parents are chosen from the population through binary tournament selection, considering both rank and crowding distance. An individual is selected if it has a lower rank or a higher crowding distance compared to the other [40].

The selected population generates offspring through crossover and mutation. The combined population is then re-sorted by non-domination, and only the top N individuals, based on rank and crowding distance, are retained. The figure 2.19 depicts the sequential phases of the NSGA-II algorithm, illustrating how it classifies and evolves solutions to maintain a diverse and optimal Pareto front.

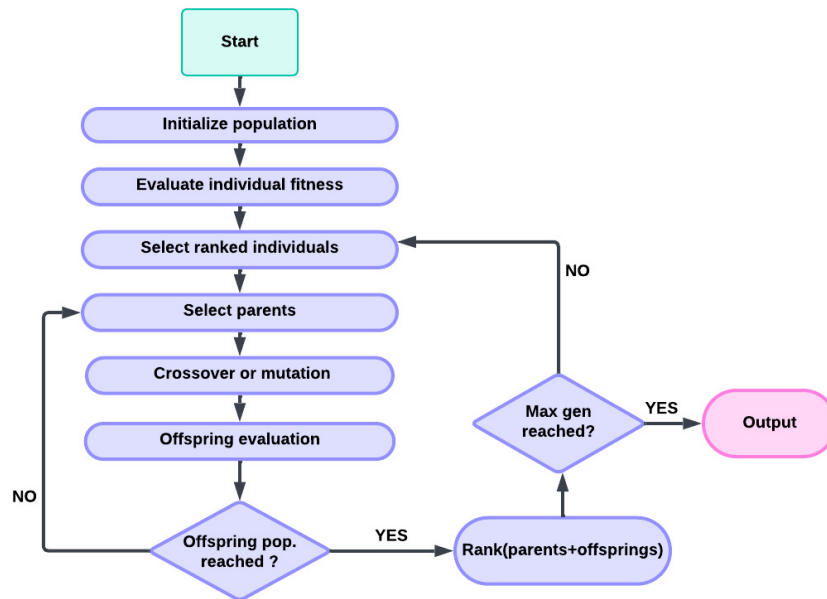


Figure 2.19. Non-dominated Sorting Genetic Algorithm II

Strength Pareto Evolutionary Algorithm 2 (SPEA2)

An external archive is used to preserve a selection of the top solutions, which are then combined with the main population for future generations. This method evaluates fitness based on both the dominance and density of solutions, helping to maintain a diverse set of optimal solutions along the Pareto front. The figure 2.20 illustrates the step-by-step process of SPEA2, highlighting its method for maintaining an archive of non-dominated solutions and evolving the population.

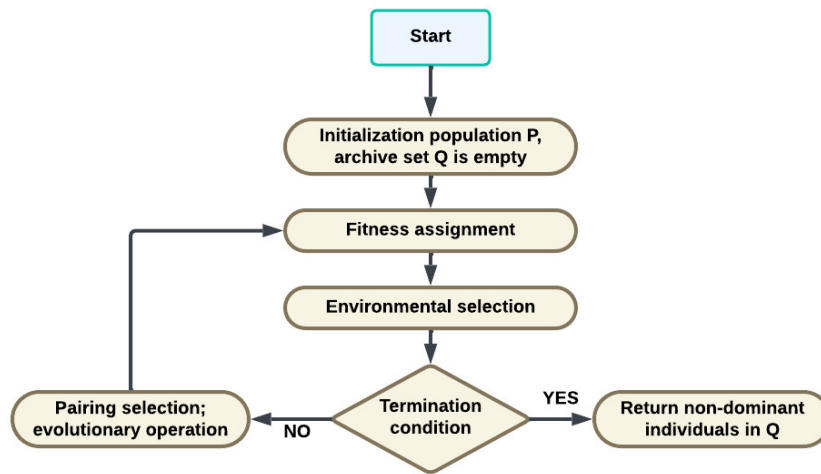


Figure 2.20. Strength Pareto Evolutionary Algorithm 2

2.5.2.2 Non-Elitist Algorithms

Non-elitist algorithms in multi-objective genetic algorithms behave similarly to their single objective counterparts, but in a multi-objective context. They do not pass on the most effective solutions to future generations. Instead, they concentrate on the fundamentals of genetic processes, with the goal of exploring a wide range of potential solutions across multiple objectives. These algorithms can be more exploratory, providing a broader range of solutions while risking losing some high-quality ones. The Generic Multi-Objective Genetic Algorithm serves as an example of this approach.

Generic MOGA Algorithm

Multi-Objective Genetic Algorithms (MOGAs) address optimization problems with multiple conflicting objectives by finding a set of non-dominated solutions known as a Pareto front. These earlier methods prioritize basic optimization strategies over advanced features like elitism. MOGAs are useful in scenarios like engineering design, resource management, and economic planning, helping decision-makers understand trade-offs and make informed choices.

2.6 Application of Genetic Algorithms in scheduling

Scheduling courses and exams is not an easy task, and this is due to the different requirements and constraints that should be considered, so both professors and students can work in a favorite environments. This section provides an explanation about the use of genetic algorithm in the context of scheduling courses and exams. Moreover we offer the findings and analysis, showing how well our strategy worked to solve the university's scheduling problem.

2.6.1 Problem representation

Since in our work we are solving two problems of scheduling, scheduling department courses and exams, we decided to divide the representation of the problem into two separate sides so we can explain each side individually.

2.6.1.1 Representation of courses

In the course scheduling of the computer science department, we have five levels: three bachelor's levels (L1, L2, L3) and two master's levels (M1, M2). Each level is divided into sections, and each section contains groups. The data we used is selected from the second semester of the last year.

The bachelor's Levels (L1, L2, L3) is organized as follow:

- L1: Two sections, each one contains six groups.
- L2: One section contains eight groups.
- L3: One section contains seven groups.
- Since all the bachelor levels has the same option (SI), all the groups and sections in the same level have the same subjects.

The masters Levels (M1, M2) is organized as follow:

- Both levels M1 and M2 has five different options (AI, SIOD, GLSD, RTIC, IVA).

- Each option has different subjects to schedule.
- Each option contains one section and one group in the section for each level.

Requirements for course scheduling

In order to ensure a functional and efficient schedule, we have identified several non negotiable requirements that are essential for the scheduling process.

- Each course must be assigned a specific time slot and room.
- Professors must be available at the assigned times.
- No two courses can be scheduled in the same room at the same time.
- Each session must be assigned to a suitable room: practical sessions in a lab, lectures in an amphitheater, and tutorial sessions in a normal classroom.
- Each professor must be scheduled for the subjects they are assigned to.
- Lectures must be assigned only to the professor responsible for the subject.

Besides the essential requirements, there are also some favorable requirements that elevates the scheduling process.

- Professors should be scheduled for a maximum of three days per week to consolidate their sessions.
- Each professor should have a maximum of 9 hours of teaching per week, equivalent to 6 sessions.
- The order of sessions for each subject should ideally be Lecture, Tutorial, and then Lab work.

2.6.1.2 Genetic representation for courses

In this section, we delve into the fundamental components of our GA designed for optimizing course scheduling. The effectiveness of a GA largely hinges on how well the

problem domain is encoded through its genetic representation. Here, we detail the structure and functionality of genes, chromosomes, and populations, tailored specifically for the complexities of course scheduling.

1. Gene

In our GA we initially represented the gene as one session for a group. However, because lectures are shared across sections, this strategy created problems for lecture sessions. Consequently, lectures frequently violated the main requirements, since the algorithm interpreting them as multiple sessions scheduled in the same room and by the same professor. To address this problem, we revised our approach to represent the gene in two different ways. Both representations still reflect a single session from the schedule, but we introduced two types of sessions: normal sessions and section session. Normal sessions and section sessions genes are represented respectively in figures 2.21 and 2.22.

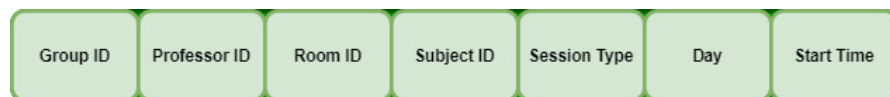


Figure 2.21. Normal session gene representation

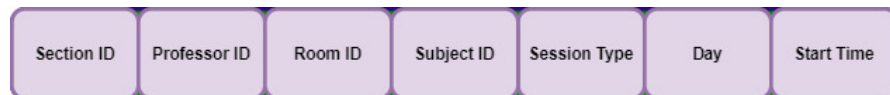


Figure 2.22. Section session gene representation

This way, we schedule lecture sessions for the entire section instead of individual groups, which prevents violations of the requirements. By doing so, we ensure that lectures are not mistakenly interpreted as multiple sessions in the same room or with the same professor, thereby maintaining compliance with the main constraints. Examples of normal sessions and section sessions are given respectively by figures 2.23 and 2.24.

Group ID	Professor ID	Room ID	Subject ID	Session Type	Day	Start Time
03	01	06	01	TD	Monday	09:40

Figure 2.23. Normal session example

Section ID	Professor ID	Room ID	Subject ID	Session Type	Day	Start Time
01	16	27	02	Lecture	Sunday	08:00

Figure 2.24. Section session example

2. Chromosome

In our genetic algorithm, the chromosome represents the entire schedule for all the groups and sections in the department. Each chromosome is a comprehensive sequence of genes, where:

- Each gene corresponds to a scheduled session, either for an individual group (normal session) or for an entire section (section session).
- The chromosome contains the entire set of sessions, ensuring that the scheduling constraints and requirements are met for all courses within the department.

By using this method, the genetic algorithm is able to evaluate and optimize the schedule entirely, ensuring that all sessions are allocated to the proper time slots and rooms, ensuring no conflicts. An example of a Chromosome is given by the figure 2.25

	Group/Section ID	Professor ID	Room ID	Subject ID	Session Type	Day	Start Time
Gene 1	01	01	30	01	Lecture	Tuesday	08:00
Gene 2	05	20	10	03	TD	Monday	13:10
				⋮			
Gene N	47	72	16	27	TP	Wednesday	11:20

Figure 2.25. Example of chromosome representation

Notice that N represents the total number of sessions in the complete schedule, encapsulating all the scheduled sessions for all groups and sections in the department.

2.6.1.3 Representation of exams

Exam schedules at the computer science department cover four academic levels: three bachelor's levels (L1,L2 and L3) and one master's level (M1 and M2). Each level is split into sections, and those sections are further divided into groups, with the levels having the same representation as in the courses. The second semester of the preceding academic year is covered by the considered data. Notably, no exam is given at the M2 level within the specified time frame. As a result, the level has been eliminated from the exam schedule.

Requirements for exams planning

Several fundamental, non-negotiable elements for the scheduling process have been established in order to guarantee an efficient and successful exam schedule.

- A specific time slot and room must be assigned to each exam session.
- Proctors must be available during the assigned times.
- Exam sessions cannot be in the same room at the same time across all levels.

- Each session must be assigned to a suitable room for its level. For bachelor levels exam sessions are scheduled in an amphitheater, while master's level sessions take place in "TD" rooms, which are normal classrooms.
- It is essential to schedule each proctor for the subjects to whom they have been assigned.
- the professor in charge is always listed as a proctor for every exam session.
- Each room must accommodate the number of groups and proctors that correspond to its capacity.
- Proctors for an exam session should be qualified professors of the subject. If only one qualified professor is available for a subject, additional proctors will be added to fulfill the room's required proctors number.
- The number of groups assigned to an exam session in a specific time and room must match the room's capacity.

In addition to the necessary requirements, there are a few optional requirements that improve the scheduling processes.

- All proctors should have the same number of proctored sessions, except for those associated with subjects having a coefficient of 1, who should only be assigned a single exam session to proctor.
- Each group, across all sections and levels, have only one exam session per day.

2.6.1.4 Genetic representation for exams

Gene

The gene was depicted in our exam planning algorithm as one exam session for a group, as shown in the figure 2.26.

Section id	Room id	Subject id	Day	Start time	Proctors id	Groups id
------------	---------	------------	-----	------------	-------------	-----------

Figure 2.26. Gene representation

An example is given in the figure 2.27

3	26	16	Sunday	08:30	[10,66,67,4]	[21,22,23,24]
---	----	----	--------	-------	--------------	---------------

Figure 2.27. Example of gene representation

Chromosome

A chromosome represents the department's exam planning for all groups and sections across all levels, it is consisting of a sequence of genes where each gene corresponds to a specific exam session (see figure 2.28).

	Section id	Room id	Subject id	Day	Start time	Proctors id	Groups id
Gene 1	3	26	16	Sunday	08:30	[10,66,67,4]	[21,22,23,24]
Gene 2	1	28	4	Tuesday	12:30	[1,2]	[1,2]
				⋮			
Gene N	4	4	26	Saturday	10:30	[71]	[38]

Figure 2.28. Example of chromosome representation

N is the total number of exam sessions in the whole planning, which includes all the sessions that are scheduled for each group and section within the department.

2.6.2 Scheduling with NSGA-II: From theory to practice

In scheduling, particularly when dealing with complex multi-objective problems, finding an optimal solution that balances various conflicting objectives is a significant challenge. To solve this, we chose and used the Non-dominated Sorting Genetic approach NSGA-II, a popular multi-objective optimization approach. This section explains why we chose NSGA-II for our scheduling challenge and how we used it in our work, its implementation in our specific context and the results and analyses derived from its application.

2.6.2.1 Rationale for Choosing NSGA-II

The selection of NSGA-II for our scheduling problem was based on several factors. While there are many advantages to use NSGA-II, the main factors that led us to choose it are listed below:

- **Multi-Objective Optimization:** Our scheduling problem has multiple conflicting objectives, such as minimizing conflicts and managing the different preferences of the involved entities. NSGA-II excels at handling multi-objective optimization problems, offering a wide range of solutions that balance these objectives.
- **Pareto Front Diversity:** NSGA-II maintains diversity in the Pareto front through a crowding distance mechanism. This is critical for our scheduling problem because it ensures that we have a diverse set of solutions to accommodate varying scheduling constraints and preferences.
- **Non-Dominated Sorting:** The NSGA-II's non-dominated sorting approach ranks solutions effectively based on their dominance levels. This allows us to clearly identify and select the best schedules that balance competing objectives, resulting in a high-quality solution set.

2.6.2.2 Application of NSGA-II for the scheduling system

In the process of implementing NSGA-II for our scheduling needs, we applied a consistent methodology across both exam scheduling and course scheduling in several key phases of the algorithm. The initialization, mutation, selection, and crossover processes

are identical in their execution for both applications, reflecting our commitment to a unified genetic algorithm framework that leverages robustness and efficiency. While these phases are uniformly implemented, the distinctive nature of each scheduling problem necessitates a tailored approach during the fitness evaluation phase.

Initialization:

The initial population in both algorithms exams scheduling and courses scheduling is generated semi-randomly. It takes into account a number of constraints, including suitable classrooms, qualified professors, and session requirements. This careful initialization ensures that the resulting schedules are somehow viable from the start, which reduces the need for large modifications later in the algorithm. Given the problem's complexity, we determined that this approach was needed for our algorithm's performance. By starting with a population that already meets some fundamental constraints, we significantly minimize the search space and convergence time, resulting in faster and more efficient optimization. During the initialization process, both algorithms the exams algorithm and the courses algorithm perform the following steps tailored to their respective scheduling needs.

1. Courses scheduling initialization

The initialization in course scheduling is tailored to satisfy the needs of every section and group. First, the algorithm determines which subjects should be scheduled according to the option and level of the scheduled sections and groups. Subsequently, it assigns qualified professors to guarantee that every topic is taught by a fitting professor for the scheduled session. It then selects time slots and rooms that are suitable for each session type, scheduling the required number of lectures, tutorials, and lab sessions.

2. Exams scheduling initialization

In this process, the algorithm assigns proctors, time slots to the exam sessions, and determines subjects for each group in each section based on associated options to ensure appropriate coverage. Rooms are allocated based on the number of groups to reduce capacity issues—for example, amphitheater H accommodates 6 groups. It also ensures that there is an exam session for every given subject for every

group in a section across all levels, efficiently controlling space and resources across sections and levels.

Evaluation

In our algorithm, the evaluation of scheduling quality addresses multiple objectives independently, recognizing the distinct challenges and criteria specific to course and exam scheduling. This necessitates a separate consideration for each scheduling type to ensure precision and effectiveness. The method employed to manage and compare these objectives is based on the principle of Pareto dominance. According to this approach, a solution 'P' is said to dominate another solution 'Q' if it is no worse than 'Q' in all objectives and superior in at least one. This comparative analysis is guided by weights that indicate the desired direction higher or lower for each objective's values when assessing dominance. This method ensures that our optimization process comprehensively evaluates each potential solution's overall performance across the varied and competing objectives.

• Courses scheduling evaluation

In course scheduling, we emphasize the following objectives, each treated independently to reflect the multi-dimensional nature of the optimization:

- Minimize conflicts: To ensure minimal conflicts between courses, enhancing the feasibility of the schedule for students and professors.
- Maximize professor session consolidation Score: To consolidate teaching sessions for professors for maximum of three days, thereby reducing their idle time and improving operational efficiency.
- Maximize professor sessions number score: To distribute teaching loads evenly across faculty members, promoting fairness and optimal resource utilization.
- Minimize subject order penalty: To sequence courses logically (e.g., lectures before labs), facilitating a coherent learning experience.

Each objective is assigned a weight indicating its importance, which guides the optimization process. The weights are defined as follows:

- w_{conflict} : Weight for minimizing conflicts (-2).

- $w_{\text{prof.consolidation}}$: Weight for maximizing professor session consolidation (1).
- $w_{\text{prof.sessions}}$: Weight for maximizing professor sessions number (-1).
- w_{order} : Weight for minimizing subject order penalty (-1).

Objective functions

In this section, we delve into the specific objective calculations for course scheduling.

1. Conflicts calculation (P_{conflict})

The equation 2.1 represents the objective function of conflicts, and the goal of the scheduling algorithm would be to minimize P_{conflict} .

$$P_{\text{conflict}} = \sum_{i=1}^N C_i \quad (2.1)$$

Where:

- C_i represents each individual conflict penalty component.
- N is the total number of conflict types (professor availability, room availability, group schedules, section session overlaps, professor-subject unsuitability, section-group overlaps).

The equations (2.2) to (2.7) represent the different conflicts that the function aims to minimize.

Professor Availability Conflicts (C_{prof}):

$$C_{\text{prof}} = \sum_{s \in S} \sum_{d \in D} \sum_{t \in T} \delta_{\text{prof}}(s, d, t) \quad (2.2)$$

Room Availability Conflicts (C_{room}):

$$C_{\text{room}} = \sum_{s \in S} \sum_{d \in D} \sum_{t \in T} \delta_{\text{room}}(s, d, t) \quad (2.3)$$

Group Schedule Conflicts (C_{group}):

$$C_{\text{group}} = \sum_{s \in S} \sum_{d \in D} \sum_{t \in T} \delta_{\text{group}}(s, d, t) \quad (2.4)$$

Section Session Overlaps (C_{section}):

$$C_{\text{section}} = \sum_{s \in S} \sum_{d \in D} \sum_{t \in T} \delta_{\text{section}}(s, d, t) \quad (2.5)$$

Professor-Subject Unsuitability ($P_{\text{prof.suit}}$):

$$C_{\text{prof.suit}} = \sum_{s \in S} \delta_{\text{prof.suit}}(s) \quad (2.6)$$

Section-Group Overlaps ($C_{\text{section.group}}$):

$$C_{\text{section.group}} = \sum_{s \in S} \sum_{d \in D} \sum_{t \in T} \delta_{\text{section.group}}(s, d, t) \quad (2.7)$$

Where:

- s represents a session, which includes details about the scheduled activity.
- d represents a day, indicating when the session is scheduled.
- t represents a time slot, specifying the exact time when the session occurs.
- Each δ function is a condition-checking function that returns the count of specific conflicts or unsuitabilities detected for given sessions, days, and time slots.

2. Consolidation score (S_{prof})

The equation 2.8 represents the objective function of consolidation (the penalties each time a professor is scheduled to teach on more than three days in a week to enhance consolidation efficiency).

$$S_{\text{prof}} = \sum_{p \in P} \text{consolidation_score}(p) \quad (2.8)$$

Where:

- p represents a professor.

3. Session number score ($C_{\text{prof_sessions}}$)

The equation 2.9 represents the objective function of required sessions number for professors. $C_{\text{prof_sessions}}$.

$$C_{\text{prof_sessions}} = \sum_{p \in P} \text{session_count_penalty}(p) \quad (2.9)$$

Where:

- p represents a professor
- the score is a penalty applied inversely to the number of sessions conducted by the professor.

4. Subject order penalty (P_{order})

The equation 2.10 represents the objective function of the subject sessions order (the penalties each time a subject session does not respect the preferred order of sessions for a group. We aim to minimize P_{order} to ensure that the curriculum is delivered in a logical sequence.

$$P_{\text{order}} = \sum_{g \in G} \sum_{s \in S} \text{subject_order_penalty}(g, s) \quad (2.10)$$

Where:

- g represents a group.
- d represents a day.
- s represents a session.
- aiming to enforce correct sequential order of sessions of the same subject within a week (Lecture, Tutorial then Lab work).

• **Exams scheduling evaluation**

In exams scheduling, our objectives include:

- Minimize conflicts To ensure minimal conflicts between courses, enhancing the feasibility of the schedule for students and professors.
- Maximizing that each group in each section and level has only one exam session per day
- Minimizing the proctoring distribution variance
- $w_{\text{conflicts}}$: Weight for minimizing conflicts (-2).
- $w_{\text{single_exam}}$: Weight for maximizing the possibility that each group has only one exam per day across all sections and levels (1).
- $w_{\text{proctored_sessions}}$: Weight for minimization of proctoring distribution variance (-1).

Objective functions Here, we delve into the specific objective calculations for exams scheduling.

1. Conflicts calculations: The equation 2.11 represents the objective function of conflicts, aiming to minimize P_{conflict} .

$$P_{\text{conflicts}} = \sum_{i=1}^N C_i \quad (2.11)$$

Where:

- Minimize P_{conflict} is the total conflict penalty.
- C_i represents each individual's conflict .
- N is the total number of conflict types (professor availability, room availability,time availability, sessions overlaps).

2. Scheduling one exam session per day The equation 2.12 represents the objective function of scheduling one exam session per day (the penalties each time more than

one exam is scheduled in a day for a section, aiming to minimize $P_{\text{single_exam}}$.

$$P_{\text{single_exam}} = \sum_{g \in G} \text{one_exam_per_day}(g) \quad (2.12)$$

Where:

- G is the set of groups.
- $\text{group_sessions_per_day}(g)$ is the score for group g having limited exam sessions per day .

3. Proctoring distribution variance : The equation 2.13 represents the objective function of proctoring distribution (variance).

$$J_{\text{proctored_sessions}} = \sum_{i=1}^P (x_i - \mu)^2 \quad (2.13)$$

where:

- J represents the objective function to be minimized.
- x_i denotes the actual number of proctoring sessions assigned to the i -th professor.
- μ is the ideal number of sessions per professor, calculated as $\frac{\text{total number of subjects}}{\text{total number of professors}}$.
- P is the total number of professors.

Selection

In our algorithm, we went through two stages in the selection process: parent selection and survival selection. These stages helps maintain a balance between preserving high-quality solutions and introducing new genetic material for exploration.

1. Parents selection :

- Parents are selected from the population using the NSGA-II selection method, which is predefined in the DEAP library. Each individual in the population is assigned a crowding distance value since selection is based on rank and

crowding distance. Crowding distance is calculated within each front, making comparisons between individuals in different fronts less meaningful. The method for calculating crowding distance is as follows:

For each front F_i with n individuals:

- Initialize the crowding distance to zero for all individuals:

$$F_i(d_j) = 0 \quad \text{for all } j$$

- For each objective function m :

- * Sort the individuals in front F_i based on objective m :

$$I = \text{sort}(F_i, m)$$

- * Assign infinite distance to the boundary individuals:

$$I(d_1) = \infty \quad \text{and} \quad I(d_n) = \infty$$

- * For $k = 2$ to $n - 1$:

- Update the crowding distance:

$$I(d_k) = I(d_k) + \frac{I(k+1).m - I(k-1).m}{f_{max}^m - f_{min}^m}$$

where $I(k).m$ is the value of the m -th objective function of the k -th individual in I .

The crowding distance measures the Euclidean distance between individuals within a front based on their objectives in the multi-dimensional space. Boundary individuals always receive an infinite distance, ensuring their selection.

2. Survivor selection :

- The current population (parents) and the offspring are combined, and the best individuals are selected to form the next generation's population. This ensures that the best traits are carried over while introducing new genetic

material.

- Elitism: The Hall of Fame (HoF) mechanism ensures that the best individual from the current population is preserved and carried over to the next generation to ensure that the highest quality solution is not lost.

Crossover

The crossover process is implemented using a one-point crossover mechanism. This mechanism selects a random crossover point for each pair of parent individuals and exchanges their genetic material at this point to produce two new offspring. This approach ensures that new offspring inherit characteristics from both parents, promoting genetic diversity and exploring new parts of the solution space.

This function works as follows:

- It takes two parent individuals, 'ind1' and 'ind2', as inputs.
- It checks if the length of either individual is less than or equal to 1, in which case no crossover is performed.
- The maximum crossover index is determined based on the shorter of the two parents to ensure valid crossover points.
- A random crossover index is selected within the valid range.
- The genetic material of the parents is swapped up to the crossover point to create two new offspring.

Mutation

The mutation process is implemented using bit flip mutation. This helps in exploring new solutions and prevents the algorithm from getting stuck in local optima. This function works as follows:

- It takes an individual as input.
- It iterates through each session in the individual.

- A mutation type is selected from "time slot", "room", or "professor" for courses and "time slot", "room" for exams .
- Depending on the mutation type, the session's day and time, room, or professor is altered.

Solution extraction

The final step in our genetic algorithm is the extraction of the best solution after the algorithm has completed its run. This involves selecting the best individual from the population, which represents the optimal schedule based on the defined objectives and constraints. After the completion of the specified number of generations, the best solution is extracted from the Pareto front. The solutions are ordered based on the conflict penalty and crowding distance:

- **Conflict Penalty:** The solutions are first ordered based on the conflict penalty. The solution with the least conflict penalty is considered the best.
- **Crowding Distance:** Among the solutions with the same conflict penalty, the crowding distance is used to further distinguish and select the best solution. The solution with the higher crowding distance is preferred as it promotes diversity.

This function works as follows:

- It finds the minimum conflict penalty among the solutions in the front.
- It selects all solutions that have this minimum conflict penalty.
- It calculates the crowding distance for these selected solutions.
- It selects the solution with the maximum crowding distance as the best solution.

2.7 Conclusion

In this chapter, we provided a comprehensive overview of GAs, explaining their basic concepts, types, and primary phases. Explored key elements like populations, chromosomes, and genes and discussed how they are coordinated through phases like ini-

tialization, selection, crossover, mutation, and evaluation. We showcased the practical application of these algorithms in our academic scheduling problem using NSGA-II.

Chapter 3

Analysis and Conception

3.1 Introduction

In this chapter, we explore the main features of our multi-platform scheduling system created specifically for the computer science department. The primary objective is to make the scheduling of courses, exams, and the assignments of graduation project fully automatic. Both web and mobile platforms are supported by our platform. Thus, we go over the major goals, functions, and features of the system. Additionally, we will provide UML diagrams that demonstrates user-system interactions and highlights the ways in which the application satisfies the demands of administrators, professors and students.

3.2 Unified Modeling Language definition

Unified Modeling Language (UML) is a standardized modeling language, which offers a collection of graphical notation tools for building visual representation of object-oriented software system. UML helps in the specification, construction, documentation and visualisation of a software system's artifacts which enable developers and stakeholders understand and design complex systems. Due to these characteristics, we chose UML to help in modeling, explaining and providing a better visualization of our system. UML contains over then 14 types which divide into two categories: 7 types of diagrams represent structural information, and seven more types represent general UML diagrams for behavioral modeling. In the next context, we will explain the UML diagrams that we used in our project modeling.

3.2.1 Use case diagram

The functional needs of a system can be represented by a use case diagram, which is a type of diagram described by the Unified Modeling Language (UML). According to the definition provided by [41], use case diagram gives a high-level overview of the system. It is primarily used to model a system's behavior as observed by actors or external individuals and to capture its dynamic characteristics. In our scenario, the actors are students, professors and administrators, each one of them interact with different features offered by the system and represent the roles that users play inside it.

Use cases illustrate the features of the system by offering particular services to the actors. On the other hand, the system boundary defines its extent and sets it apart from the surrounding environment.

Use case diagrams include a variety of relationships between the actors and use cases, such as generalization which indicate inheritance between actors or use cases from specific roles or functions, as well as associations that are represented by lines that connect actors and use cases for illustrate the interaction between the two of them. In addition, to include and extend relationships, in the include side, a use case's behavior is integrated into another, and in the extend a use case is expanded for optional or exceptional cases.

3.2.2 Class Diagram

As defined in [42], a class diagram in software engineering is a form of static structure diagrams, which illustrates the classes, attributes, operations, and interactions between objects in a system. Class diagrams rely on relationships between classes since they demonstrate how different classes relate to one another. These relationships include composition , aggregation, association, inheritance and dependencies. This diagram helps with comprehension and communications by providing a clear picture of the architecture of the system, which is essential for scalability, maintenance and documentation.

3.2.3 Sequence Diagram

UML Sequence Diagrams are interaction diagrams that demonstrate the steps involved in implementing an software, as defined by [43]. They represent how items interact with one another as they function together. Sequence diagrams are time-focused and visually depict the sequence of the interaction by displaying the time, which messages are sent, and when to display the interaction's sequence visually on the vertical axis of the diagram. There are several advantages for using sequence diagrams, one of them is making the flow control in the system easier to understand by providing a clear and thorough perspective of the sequence in which objects interact with each other. They also improve communication among stakeholders, including developers, designers and non-technical users, by offering an illustration of the system's behavior over time. For

the purpose of maintenance and upcoming development, this type of diagrams provides full documentation of how things work on the entire process.

3.3 System design

In this section, we provide an explanation of our multiplatform's conceptual framework while highlighting the key actors in the system, which are administrators, professors and students. Each individual plays an essential part in the efficiency of the system. Along with providing diagrams that demonstrate the features and interactions that support each actor accomplish the overall objectives of the system.

3.3.1 Professors part

Within the computer science department, professor is an academic staff member who is responsible of teaching courses and doing his own research. In our case, he is able to more effectively manage academic tasks. For more details, we will explore the features and interaction that may happened between our actor which is the professor and the multi-platform.

3.3.1.1 Professors functionalities

Professors can use the same features on both web and mobile platforms. The main features are:

- The ability to check his own schedule.
- Consult schedules of all the groups.
- Schedule a replacement, test or consultation session, by filling all the needed information to apply the request. The concerned group will be notified accordingly.
- Receives notifications or emails whenever an update happened to his schedule.

3.3.1.2 Professors use case diagram

The figure 3.1 represents the use case diagram for the professors role.

1. The professor should access to his personal account using the login page, which provide him the features and tasks that he need.
2. Professors can check out their schedules through this feature, which makes it simpler for them to efficiently manage their time and duties.
3. During exams season, the professors can check their proctoring schedule to know when and where they have session to supervise.
4. The professor can see the schedule of a specific groups, which gives him an overview at any time.
5. This feature will assist the professor in submitting requests of any type that correspond to the cases that will be discussed below.
 - (a) This feature can be used whenever the professor want to add a session (Lecture, Lab work, Tutorial) to a specific group(s) for any reason such as making up for a missed class or in the case of a course delay.
 - (b) The professor can use this feature to schedule a test session (Lecture, Tutorial).
 - (c) To schedule a session to consult the applied works or project of the practical sessions, This is the ideal use case for this situation.
6. Professors use this feature to successfully exit and log out from their accounts.

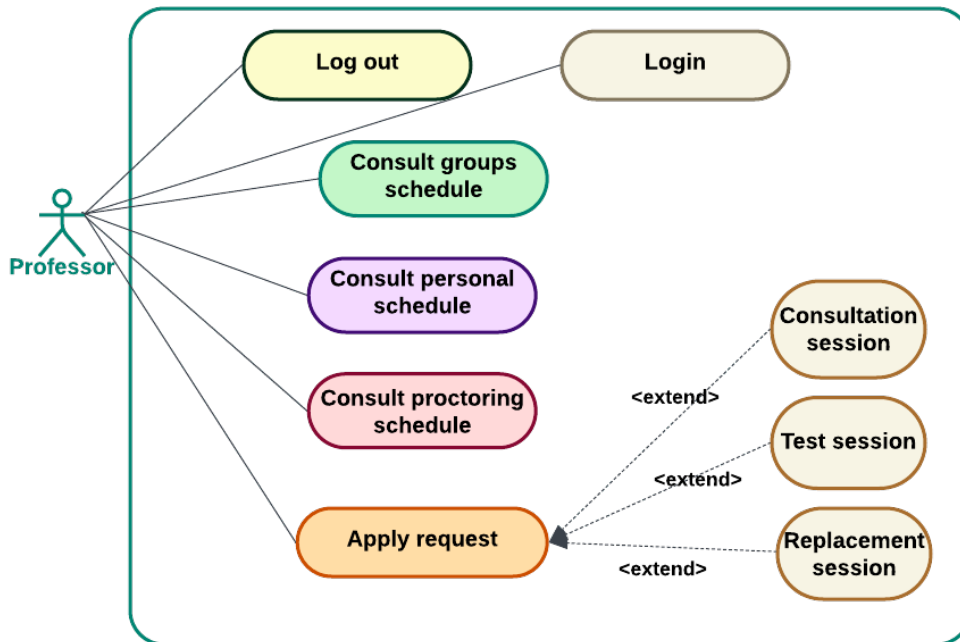


Figure 3.1. Professor use case diagram

3.3.2 Administrator role

An administrator, who we refer to on the platform as "admin", is a professor that is in charge to handle administrative tasks and operations such as the head of department and his assistant. In each interface, whether web or mobile, gives the admin access to different features or tasks. Since our system is a multi platform, in the following context we will describe the features that this actor can use on each user-screen.

3.3.2.1 Administrator functionalities

Administrators maintain all of the features of academics since they are professors with extra administrative responsibilities. However, their web interface has special capabilities designed and implemented for their duties. Among these functionalities, we can mention:

- Admin can manage professors and students accounts by adding, deleting or editing.

- Manipulate the database of the department by examine, update, edit or delete information from the databases (Levels, students, subject, etc).
- Generates the groups, professors schedule of the current academic semester.
- Has the ability to generate the groups or professors exam schedule.
- Able to launch the automatic processing of the allocation of the subjects to students based on their individual choices and rank, and then make them available to those concerned.
- The administrator has the ability to consult each group schedule and then he can make updates on it by adding , deleting and editing sessions.

3.3.2.2 Administrator use case diagram

As we knew before that the administrator is in fact a professor but with more special tasks and since we detailed the professors features in the previews use case, we are going to discuss the special features of the admin. The figure 3.2 illustrates the administrator use case diagram. In follows, the main components of the diagram.

1. The admin can generate multiple type of schedule automatically, which include semester schedule and exam schedule for both groups and professors. However, he can manually modify the generated schedule if he desired.
 - (a) The administrator has the ability to generate the current semester schedule for the groups and all the professors. Thus, the end of this process all the concerned parts will be notified.
 - (b) In each evaluation season, this feature can be used by the admin to generate the schedules of the exams. A notification will be send to inform all the groups and professors.
2. Admin can modify the schedules by adding, editing or deleting sessions.
3. The administrator has the hand to make changes or updates to the department information such as updates the users list, adding the new students or professors, deleting the canceled subjects, etc.

4. It is about assigning the graduation project for the bachelors and masters levels. This features can only be launched by the administrator during the graduation period. All the involved students will be notified of the results after the submission.
5. This feature used to inform all the concerned staff whenever any changes happened to the schedule.



Figure 3.2. Administrator use case diagram

3.3.3 Student role

One of the primary users of this system are the students, who take benefits of it to enhance their academic journey. Subsequent, we will highlight their special features.

3.3.3.1 Student functionalities

Same as the professors, the student features can be used on both platform web and mobile which can be presented as:

- Consult their schedules.
- Receive notification if the schedule get updated.
- Get notified if any session is added to the schedule (replacement, test, consultation session).
- Consult exam planning.
- If the student is in the bachelors or masters level, he got notified when the desired project is assigned.

3.3.3.2 Student use case diagram

The figure 3.3 represents the use case diagram of the student role. Here follow the main components of the diagram.

1. Like the professors, the students can use this feature to consult their schedules, whether it's for the semester or for exams.
2. if any change accrued in the schedule such as adding replacement session, test session, etc, the student is notified through this feature.

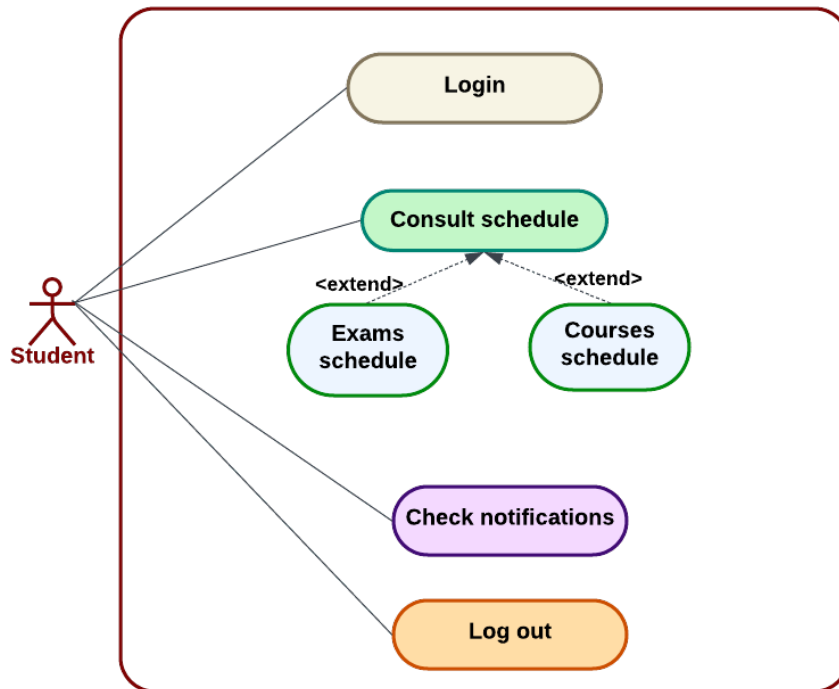


Figure 3.3. Student use case diagram

3.3.4 Detailed specification

In-depth descriptions of some significant administrative features are presented in this section. Class diagram will be used to show the system's static structure, and sequence diagram will be implemented to represent the sequential interactions between different components during various operations. A solid understanding of the architecture and operation of the system will be attainable due to these diagrams.

3.3.4.1 Class diagrams

A class diagram is essential since it connects use cases, illustrations and software design diagrams (interaction diagrams and class diagrams). Some of the main features will be explained in this section.

Class diagram for semester schedule generator

Before starting the generation process, we should have a better insight and understand-

ing of the structure and logic behind this generation process. The figure 3.4 represent the class diagram of the courses scheduling generator for the semester. This process is built by eight key classes, each class has an important role in managing the generation process of several schedule.

- The "*ScheduleGenerator*" class is considered as the primary widget in charge to serve the schedules generator interface. The inbuilt state for this class relies on its management state class which is "*Schedule-GeneratorState*".
- Using the TabController, the "*Schedule-GeneratorState*" class controls the tab navigation and keep managing the content view list for the exam and general scheduling. The "*iniState*" method is used to initialize the interface state. This class also contains functions like "*groupschedule-generation*", "*examschedule-generation*" and "*menuGenerator*". For more clarity, the "*menuGenerator*" is used to display an interface-user list of the generation options such as general scheduling (managed by "*group-schedule-generation*" method) and exam scheduling (managed by "*exam-schedule-generation*" method).
- Two of the main tab classes are "*GroupSchedule-Tab*" and "*Profschedule-Tab*", which are controlled by the "*groupschedule-generation*" method from the "*ScheduleGeneratorState*" class. Each one of these tab classes is responsible to create a user interface for displaying the resulted schedules for the groups and professors. Both "*ProfscheduleTabState*" and "*GroupscheduleTabState*" are inherited respectively to these tabs classes. These state classes are responsible to execute the scheduling algorithm and get the returned schedules, all of that is handled by the methods "*runAlgorithm*" and "*getTimetable*".
- Supporting to the scheduling process, The "*FlaskSchedule*" class, is originally a web application framework. This class represents the backend Flask application that runs the scheduling algorithm executed by the "*ScheduleGeneration*", returning a mapped list to the "*getTimetable*" method. The "*ScheduleGeneration*" is a class that provides the methods responsible for the scheduling algorithm and other methods to handle the mapping of the schedule to be returned when the program is executed.

- Once the resulted data is obtained, the method *getTimeTable* handle the data by decoding the response from the flask server to a map, then convert each map in the JSON map to a *TimeTableEntry* model.
- The *parseAndSchedule* method intervenes by receiving this map as an input parameter, this map contain a detailed list of type *TimeTableEntry* that consist of groupid, professorid, roomid, subjectid, day and starttime, each item in the *TimeTableEntry* list represents one session. After the necessary detailed data are extracted, a session detailed map is created and then this map is used by the schedules map to organize the sessions.

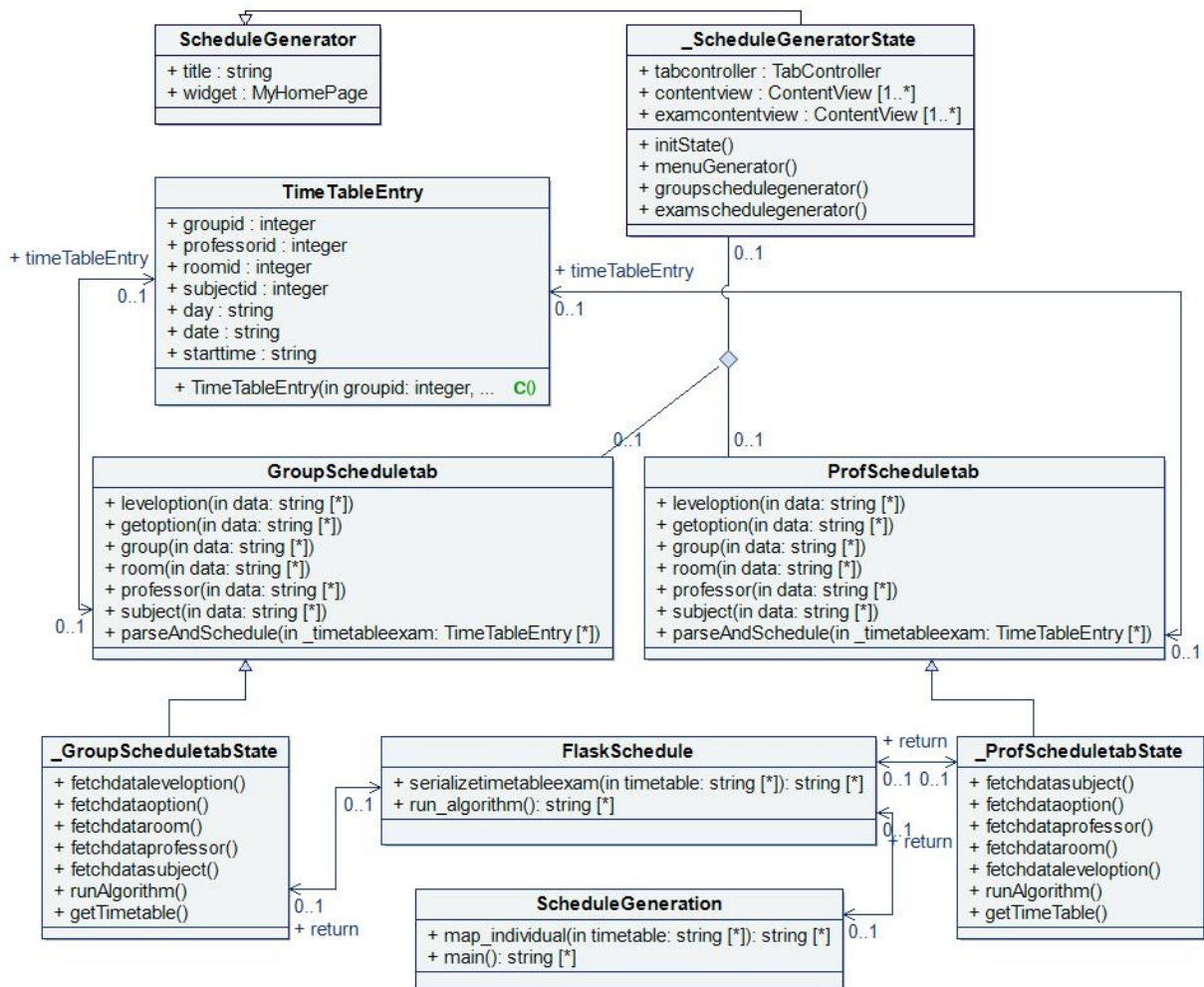


Figure 3.4. Class Diagram of schedule generator

Class diagram for exams scheduling

As we have provided a detailed explanation of the general schedule class diagram, we will do the same to the exam scheduling diagram, which is represented in the figure 3.5. This generator relies on the same key classes as the general scheduling process, with some differences in certain classes.

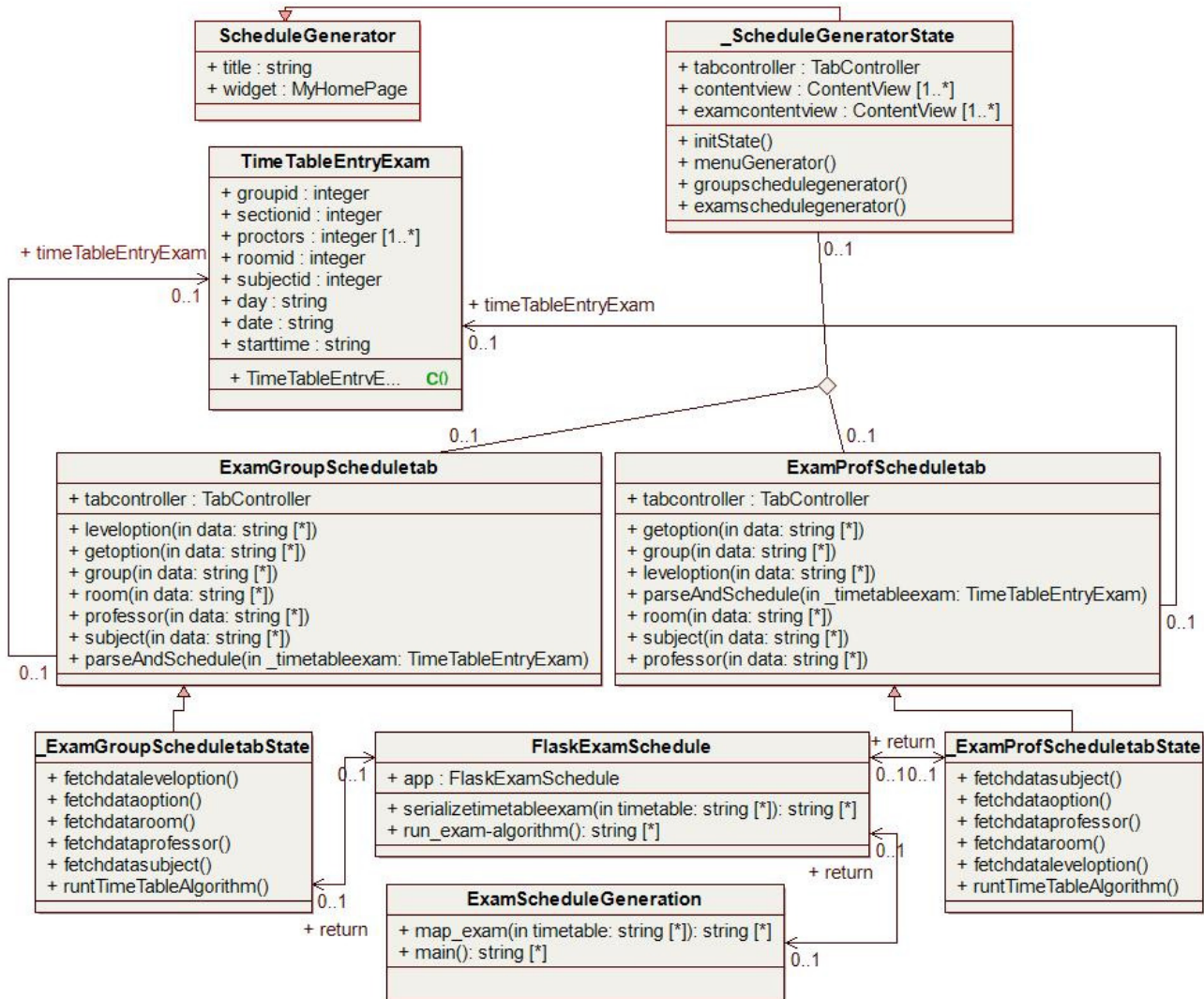


Figure 3.5. Class Diagram for exam scheduling

- With a spotlight on generating the exams schedule automatically, the diagram presents the "ExamGroupScheduletab" and "ExamProfScheduletab" classes.

These classes are responsible for creating a user interface to display the resulting schedules for professors and students, associated from the *"ScheduleGeneratorState"*. *"ExamGroupScheduletabState"* and *"ExamProfScheduletabState"* are in charge of managing the state for these tabs.

- The *"ExamScheduleGeneration"* class is used to run the scheduling algorithm, and this process will be executed only by the *"getTimetable"* method from the *"ExamGroupScheduletabState"* as part of the *"FlaskExam"* class's backend processing.
- After getting the data from the flask server, the *"getTimeTable"* handle the returned data by decoding it to a map then converted it to a *"TimeTableEntryExam"*, then the method *"parseAndSchedule"* step in to handle the creation of the schedule map by extracting the detailed session data that represents by the groupid, sectionid, proctorsid, roomid, subjectid, day, starttime and finally the date of each exam.

Class diagram for graduation project assignment

The figure 3.6 illustrates the class diagram of the graduation project assignment.

- The main widget responsible for providing functionality of the assignment result interface has been identified as *"GPAssignment"* class.
- The main classes for creating a user interface for displaying the resulted assignment project are *"Bachelor"* and *"Master"* classes. Both *"BachelorState"* and *"MasterState"* are responsible to manage the state of these tabs and these two states are inherited from them.
- The method *"filechooserwidget"* is responsible of importing the inputted excel file from the interface by calling the function *"pickfile"* to display the file picker. When the file is imported, the method *"runAlgorith"* invoked to start the algorithm by sending the imported file name. The flask server receive the name of the file leading to transfer it to the class of the assignment process. The *"processcsyfile"* function uses the name of the transferred file as a parameter to calculate the general mark of each student then to the each group by the functions *"calculmoyG"*

and *calculmoyGgroup*". After affected the project to each group the *insertResultedData*" step in to insert the detailed data such as groupid, students name, their choices and the affected choice.

- After executing the algorithm from the classes *GABachelorsAssignment*" and *GAMastersAssignment*" for both masters and bachelors, the method *fetchdata*" is responsible for fetching the result from the database to display it in the interfaces.

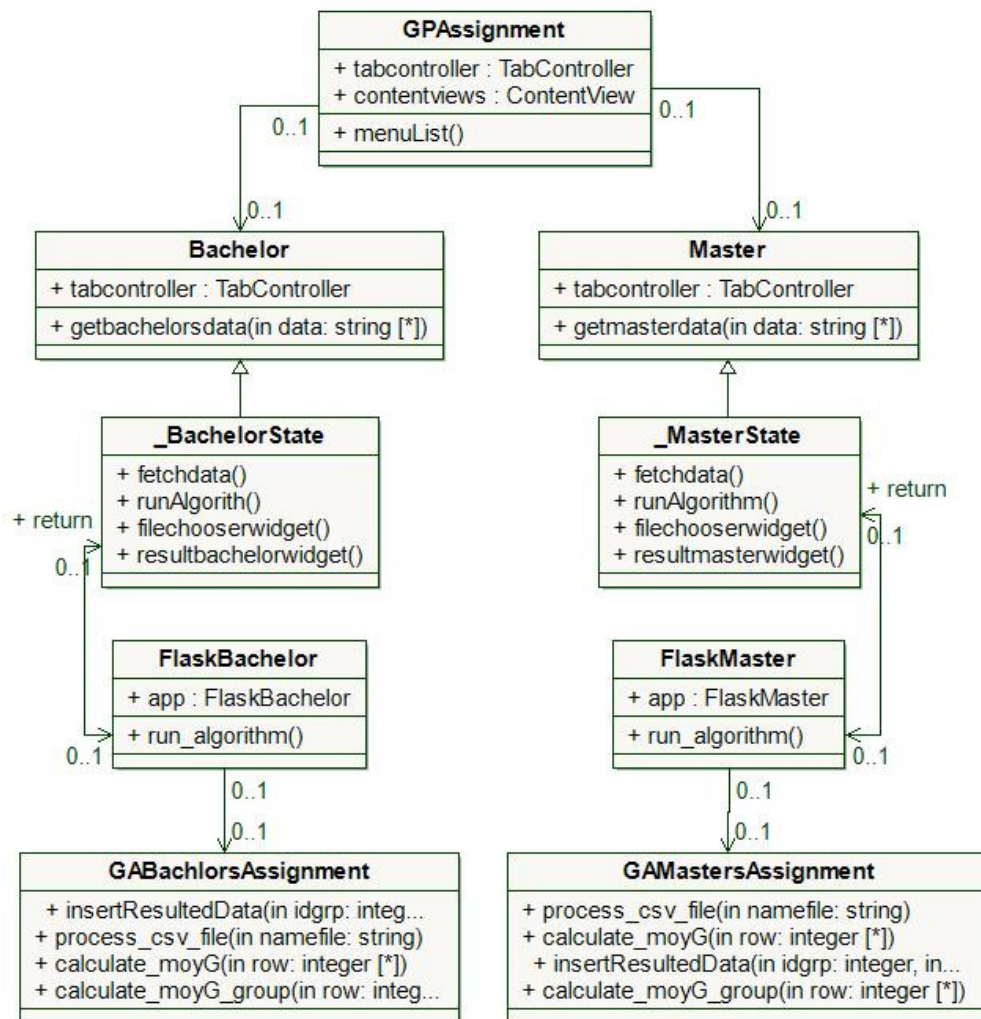


Figure 3.6. Class Diagram for graduation project assignment.

3.3.4.2 Sequence diagram

For a greater understanding of the system, the sequence diagram will be the appropriated tool to use. We shall delve deeply in the process explanation in the following context.

Sequence diagram for schedule generation process

The sequence diagram presented in the figure 3.7 illustrates the generation process for both exams and semester schedule.

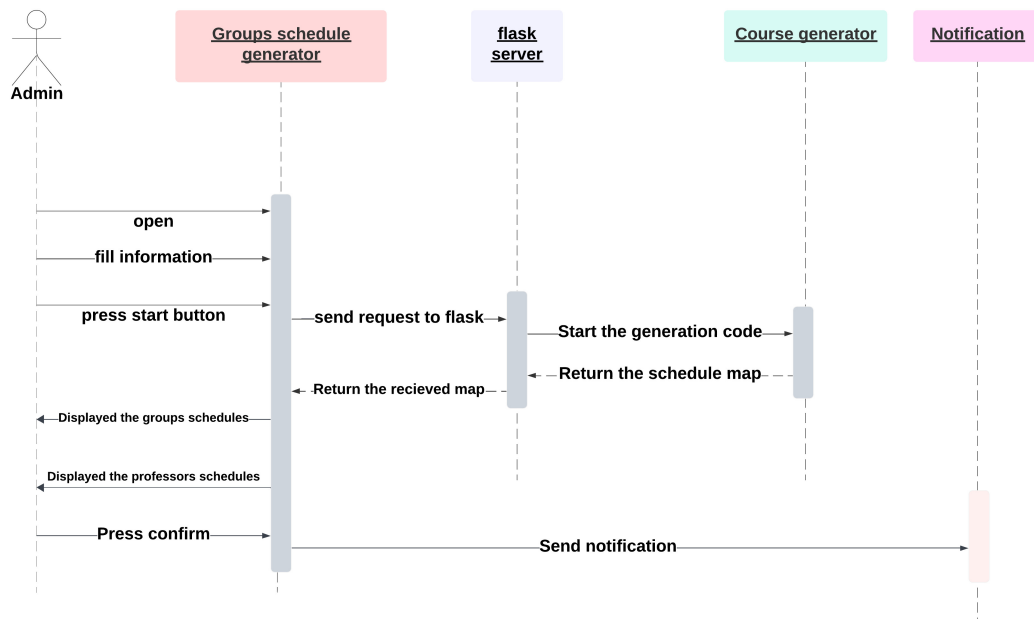


Figure 3.7. Sequence diagram for scheduling generation

Initially, the admin start this process by opening the groups schedule generator interface and fill the necessary information such as selecting the current semester and entering the current year. The groups schedule generator send a request to backend flask server when the start button is pressed. In order to handle this request, this server runs the course generator algorithm. After the scheduling process is finished, the algorithm class return the created schedule map to the flask server, which transmits it to the groups schedule generator interface leading to display the new schedules of both groups and professors to admin at the areas that have been set up for that. Once the schedule has been approved and verified by the administrator, he can press the confirm button

which allow the notification service to notify the involved users. All of these steps are the same for the exam schedule generator.

Sequence diagram for the graduation project assignment process

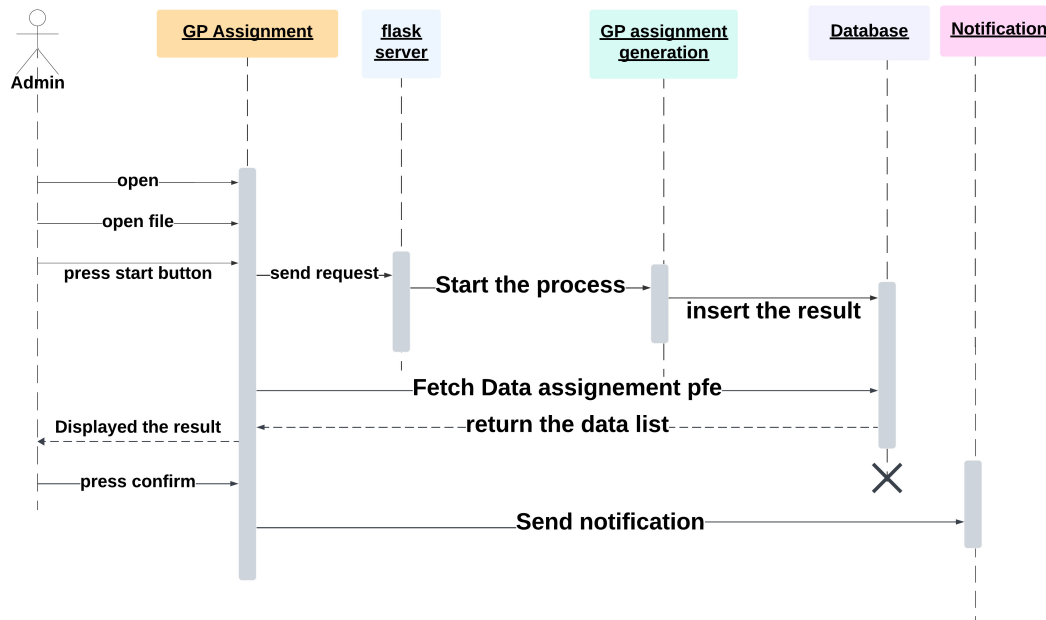


Figure 3.8. Graduation project assignment process sequence diagram

As illustrated in the figure 3.8, the administrator open the "GA assignment" interface which reveal to him a input field to import the excel file that contain the necessary data such as groupid, name students, students yearly mark and their desired project choices. After the user imported the excel file, he can press the start button, which leads to send a request to the flask server that runs in the backend application. The graduation project assignment initialize by the "GA assignment generation", when the process is accomplished the resulted data will be inserted in the database. In order to display the assignment result, the "GA assignment" class will fetch the data from the database. The admin press the confirm button whenever he completed the verification to the resulted assignment leading to send notification to the involved students.

3.4 Conclusion

In this chapter, we have given an in-depth analysis and conception of the scheduling system, applying a variety of UML diagrams and detailed specifications to highlight its behavioral and structural elements. As a way to make sure the system satisfies the demands of the administrators, students and professors, these diagrams captured the user interactions, system architecture, and dynamic processes in detail. In the upcoming chapter, we will go over the requirements and give a summary of the platform.

Chapter 4

Implementation and deployment

4.1 Introduction

The main focus of this chapter is to provide the implementation details and results of the proposed scheduling system. We proceed by describing the system requirement, which include the used software, the key components of the application, the main features and the user-friendly interfaces of our platform. We also illustrate the technical details of the used genetic algorithm for scheduling courses and exams. Finally, we conclude the chapter by highlighting the forthcoming updates and planned additions, ensuring the system's continuous development to satisfy needs in the future.

4.2 Development tools and technologies

Both front-end and back-end applications are required for our scheduling system. The operating systems, databases, frameworks, and libraries needed for the front-end and back-end are all included in this section. Efficient and dependable system functioning is ensured by following the recommendations.

4.2.1 Back-end tools

In this section, The important back-end tools are described, along with the critical software elements required for the best possible system performance.

- **Python Language:** As defined in [44], python is a popular interpreted, object-oriented, high-level programming language that is easy to learn and understand. It is suitable for a range of applications, from web development and data analysis to machine learning and automation.
- **Numerical Python (NumPy):** Large multi-dimensional arrays and matrices are supported by NumPy, it is a core package scientific computing in Python, and offers a set of mathematical operations for large arrays. It provides the foundation for numerous other Python scientific and data analysis libraries and is commonly utilized for numerical and array-oriented computing workloads.
- **Distributed Evolutionary Algorithms in Python (DEAP):** The DEAP library [45] is a framework designed for rapid prototyping and testing of evolutionary

algorithms. It emphasizes making algorithms explicit and data structures transparent, facilitating ease of use and integration with parallelization mechanisms such as multiprocessing.

- **Matplotlib:** Referred to [46], it is a comprehensive library for creating static, animated and interactive visualizations in Python. It is an adaptable tool for data visualization in scientific and analytical applications since it offers an object-oriented API for embedding plots into applications and supports a variety of plot and chart types.
- **Math:** The Math module in Python comprises various mathematical functions and constants. Designed to facilitate a wide range of mathematical tasks. As a built-in module, it provides essential functions for basic arithmetic operations such as addition (+), subtraction (−), multiplication (*), and division (/), alongside advanced operations including trigonometric, logarithmic, and exponential functions.
- **MySQL Connector:** It is a database connector, which means that it uses a query to retrieve data from a database. By selecting Database from the toolbar at the top of the window, you may navigate to the connection page for this and other database connectors in the Data Center.
- **Random:** The Python random module [47] generates pseudo-random numbers, meaning they are not truly random. This module can be used for various random actions, such as generating random numbers and selecting random values from a list or string.
- **Pandas:** The Pandas library in Python according to [48], is a powerful, open-source tool designed for data manipulation and analysis. It is built on top of the NumPy package, providing fast and flexible data structures such as Series (1-dimensional) and Data-Frame (2-dimensional) that are designed to work with structured data.
- **Comma-separated values:** CSV files can be read and written in Python using an integrated module called this library. It is a useful tool for data modification and interchange since it makes managing csv files easier by providing classes and

methods to read data into dictionaries or lists and write data from dictionaries or lists into csv format.

- **Flask:** Flask [49], is a web framework for Python, designed to facilitate the development of web applications with ease. As a micro-framework, it features a small, easy-to-extend core and does not include an Object Relational Manager (ORM) or similar components. Despite its lightweight nature, Flask offers many useful features, such as URL routing and a template engine. It operates as a WSGI web application framework, making it a powerful tool for building web applications.
- **Cross Origin Resource Sharing (CORS):** CORS [50] is a security feature that leverages HTTP headers to enable servers to specify which external origins (domains, schemes, or ports) are permitted to access their resources. This mechanism is essential for bypassing the same origin policy restrictions, which typically prevent web applications from making requests to a different origin than the one that served the web page. CORS allows for more flexible and secure interactions between web pages and different origins, facilitating safer cross origin requests and data sharing.

4.2.2 Front-end tools

The front-end tools listed below outline the essential software elements required for a productive and responsive user experience.

- **Flutter:** It is developed by google, which is an open-source UI software development kit. It used to create desktop, web and cross-platform applications from a single codebase , including IOS, Android and Windows.
- **Dart:** by google created Dart, it is a client-optimized programming language, used to help the developers creating quick apps for any platform. With the Flutter framework, Dart is frequently used to create cross-platform mobile applications.
- **Hypertext Transfer Protocol (HTTP):** It is a library which enables an effort to communicate with web services via HTTP requests in flutter. HTTP gives developers the ability to transmit and retrieve data over the internet using an intuitive

API for GET, POST, PUT, DELETE, and others. For Flutter apps to integrate network connection, this package is necessary.

- **Ngrok:** As mentioned in [51], Ngrok lets you make a local server public on the internet. It allows external access to your local development environment by establishing a secure tunnel to your localhost. Web developers that need to test webhooks, APIs, or show off their local development work to others without deploying it to a distant server may find this to be especially helpful.
- **File Picker:** It is a package that enables the users to select files from the devices storage. As defined in [52], it is a vital resource for applications that need file input capabilities since it offers a straightforward and adaptable API for selecting one or multiple files.
- **Cloud functions:** This library allows flutter apps to interact with Google Cloud Functions. It enables to trigger server-side code written in Google's cloud environment directly from the flutter app, without needing to set up a server.
- **Cloud fire store:** This library provides access to Google Cloud Fire store, a scalable database for mobile, web, and server development. It allows to store and synchronize data across multiple clients and backend services in real-time.
- **XAMMP:** It is an open-source cross platform web server solution stack package developed by Apache Friends. It provides an easy-to-install and use environment for web development, combining various components that are essential for building and running web applications. The acronym XAMPP stands for: X: Cross-platform (works on multiple operating systems, such as Windows, macOS, and Linux), A: Apache HTTP Server, M: MariaDB (formerly MySQL), P: PHP, P: Perl
- **Hypertext Pre-processor (PHP):** as defined in [53], PHP is a server-side scripting language designed for web development but also used as a general-purpose programming language.

4.3 System Architecture

This section describes the architecture of our system. It employs a client/server model using Flutter technology for creating web and mobile interface. The architecture is detailed in the figure 4.1.

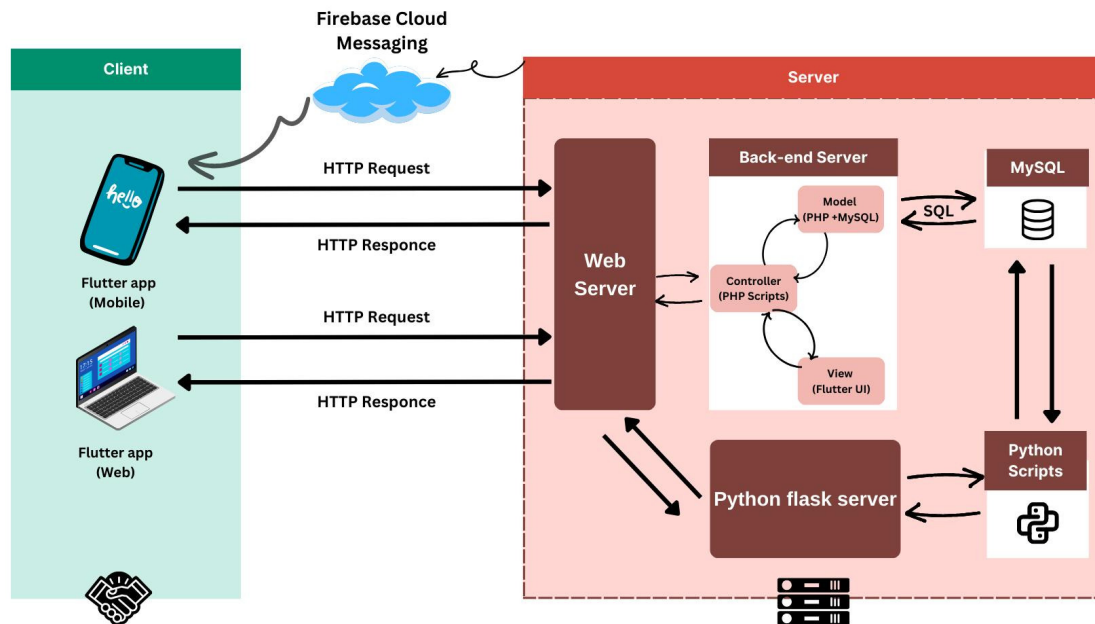


Figure 4.1. Global architecture of the proposed system

Here, we present the architecture shown in figure 4.1 within our system, highlighting the interactions among client applications, servers, and database structures.

1. **Client:** A request is sent by the client to the server. After receiving the request from the controller, the server forwards the data to the model for processing. The controller replies to the client after processing. This exchange of information takes place via the HTTP protocol. The client is represented by:

- **Flutter app (Mobile):** The mobile application is developed in Dart using VS code. Data exchange between the application and the server is in the lightweight JSON format.
- **Flutter app (Web):** The web application is developed using Dart.

2. **Servers:** The servers used in our application are XAMPP and Flask.
3. **Notifications handler:** We used Firebase Cloud Messaging (FCM) to send notifications from the server to the Android client.
4. **Database:** The database (illustrated in Figure 4.2) was created using phpMyAdmin. We propose an architecture with 20 tables which satisfies the requirements of this work.

Table	Action	Rows	Type	Collation	Size	Overhead	
affected_gp2	12	InnoDB	utf8mb4_general_ci	176.0	KiB	-	
affected_gpm2	3	InnoDB	utf8mb4_general_ci	160.0	KiB	-	
exams_schedule	59	InnoDB	utf8mb4_general_ci	64.0	KiB	-	
exam_proctors	59	InnoDB	utf8mb4_general_ci	48.0	KiB	-	
groupe	37	InnoDB	utf8mb4_general_ci	64.0	KiB	-	
groups_schedules	455	InnoDB	utf8mb4_general_ci	128.0	KiB	-	
level	5	InnoDB	utf8mb4_general_ci	16.0	KiB	-	
level_option	13	InnoDB	utf8mb4_general_ci	32.0	KiB	-	
notification	42	InnoDB	utf8mb4_general_ci	32.0	KiB	-	
option	6	InnoDB	utf8mb4_general_ci	32.0	KiB	-	
option_sub	51	InnoDB	utf8mb4_general_ci	48.0	KiB	-	
professor	77	InnoDB	utf8mb4_general_ci	16.0	KiB	-	
project Lec3	41	InnoDB	utf8mb4_general_ci	32.0	KiB	-	
project_m2	41	InnoDB	utf8mb4_general_ci	32.0	KiB	-	
room	30	InnoDB	utf8mb4_general_ci	16.0	KiB	-	
section	14	InnoDB	utf8mb4_general_ci	48.0	KiB	-	
student	192	InnoDB	utf8mb4_general_ci	128.0	KiB	-	
subject	51	InnoDB	utf8mb4_general_ci	32.0	KiB	-	
subject_profs	126	InnoDB	utf8mb4_general_ci	48.0	KiB	-	
userlist	9	InnoDB	utf8mb4_general_ci	80.0	KiB	-	
20 tables	Sum	1,323	InnoDB	utf8mb4_general_ci	1.2	MiB	0

Figure 4.2. Database tables

The figures 4.3, 4.4 and 4.5 show some tables as examples.

- **Users accounts (userList):** For controlling user access within the application, the "userList" table is essential. Fields like *iduser*, *username*, and *password* are among them. Users are categorized by *role*, with professors, students and admins being distinguished by the type field. If the user is a professor or admin, their *idprof* links to their professor record, if the user is a student, their *idstudent* links to their student record. In order to facilitate efficient user communication, the *fcmtoken* field is also utilized to send notifications to the user's device. In general, controlling user access and interactions inside the program is largely handled by the userList table.

Showing rows 0 - 8 (8 total. Query took 0.0125 seconds)

```
SELECT * FROM `userlist`
```

	iduser	username	password	type	idstudent	idprof	fcmtoken
<input type="checkbox"/>	1	74440304	37006367	student	1	NULL	e2SGV1SvQgm1VVzTdBNyEY-APA91bHU3LlSeDhXa2lyM5T7F8...
<input type="checkbox"/>	2	38033089	41353302	teacher	NULL	10	NULL
<input type="checkbox"/>	3	49173454	49034190	admin	NULL	60	NULL
<input type="checkbox"/>	4	64743513	89635464	teacher	NULL	15	NULL
<input type="checkbox"/>	15	20220222	2502516	teacher	NULL	9	NULL
<input type="checkbox"/>	16	16050203	16050203	student	60	NULL	e2SGV1SvQgm1VVzTdBNyEY-APA91bHU3LlSeDhXa2lyM5T7F8...
<input type="checkbox"/>	18	1452020	15202452	teacher	NULL	23	e2SGV1SvQgm1VVzTdBNyEY-APA91bHU3LlSeDhXa2lyM5T7F8...
<input type="checkbox"/>	19	5225302	96525420	student	101	NULL	e2SGV1SvQgm1VVzTdBNyEY-APA91bHU3LlSeDhXa2lyM5T7F8...
<input type="checkbox"/>	20	5859652	5245568	teacher	NULL	20	e2SGV1SvQgm1VVzTdBNyEY-APA91bHU3LlSeDhXa2lyM5T7F8...

Figure 4.3. Users accounts table

- Schedule (groupsschedules):** An essential part of the application for controlling professor and group schedules is the "groupsschedules" table. It has a number of important fields with comprehensive details about every scheduled session. Each schedule entry is uniquely identified by the *id* field, and the *idgrp* field provides a link to the particular group for which the schedule is meant. The room where the session will take place is indicated by the *idroom* element, while the *idsub* field relates to the topic related to the session. The professor in charge of leading the session is also linked to the *idprof* field. The starttime field shows the exact time of the session, while the day field denotes the day of the week it is planned to take place. The session type, such as lecture, tutorial, or lab work, is defined in the *sessiontype* field. Additionally, the *role* field in the session is used only in situations where temporary sessions, tests, and consultations are added, specifying the scheduled date.

	id	idgrp	idsub	idroom	idprof	day	starttime	sessiontype	role	date
<input type="checkbox"/>	1	1	1	26	1	Monday	09:40:00	LECTURE	NULL	NULL
<input type="checkbox"/>	2	2	1	26	1	Monday	09:40:00	LECTURE	NULL	NULL
<input type="checkbox"/>	3	3	1	26	1	Monday	09:40:00	LECTURE	NULL	NULL
<input type="checkbox"/>	4	4	1	26	1	Monday	09:40:00	LECTURE	NULL	NULL
<input type="checkbox"/>	5	5	1	26	1	Monday	09:40:00	LECTURE	NULL	NULL
<input type="checkbox"/>	6	6	1	26	1	Monday	09:40:00	LECTURE	NULL	NULL
<input type="checkbox"/>	7	7	1	26	1	Monday	09:40:00	LECTURE	NULL	NULL
<input type="checkbox"/>	8	1	2	28	16	Tuesday	14:50:00	LECTURE	NULL	NULL
<input type="checkbox"/>	9	2	2	28	16	Tuesday	14:50:00	LECTURE	NULL	NULL
<input type="checkbox"/>	10	3	2	28	16	Tuesday	14:50:00	LECTURE	NULL	NULL
<input type="checkbox"/>	11	4	2	28	16	Tuesday	14:50:00	LECTURE	NULL	NULL
<input type="checkbox"/>	12	5	2	28	16	Tuesday	14:50:00	LECTURE	NULL	NULL
<input type="checkbox"/>	13	6	2	28	16	Tuesday	14:50:00	LECTURE	NULL	NULL
<input type="checkbox"/>	14	7	2	28	16	Tuesday	14:50:00	LECTURE	NULL	NULL
<input type="checkbox"/>	15	1	3	30	17	Monday	11:20:00	LECTURE	NULL	NULL
<input type="checkbox"/>	16	2	3	30	17	Monday	11:20:00	LECTURE	NULL	NULL
<input type="checkbox"/>	17	3	3	30	17	Monday	11:20:00	LECTURE	NULL	NULL
<input type="checkbox"/>	18	4	3	30	17	Monday	11:20:00	LECTURE	NULL	NULL
<input type="checkbox"/>	19	5	3	30	17	Monday	11:20:00	LECTURE	NULL	NULL
<input type="checkbox"/>	20	6	3	30	17	Monday	11:20:00	LECTURE	NULL	NULL
<input type="checkbox"/>	21	7	3	30	17	Monday	11:20:00	LECTURE	NULL	NULL
<input type="checkbox"/>	22	1	4	29	24	Tuesday	13:10:00	LECTURE	NULL	NULL

Figure 4.4. Schedule table.

- Bachelors assignments:** Students group assignments are managed via the *affected_gp2* table according to their desires. It has a number of important fields that are essential to this process. Groups can be balanced according to academic performance by using the *moyG-group* field, which shows the group's average grade, and the *idgrp* field, which acts as a unique identification for each group. The names of the students assigned to each group are stored in the variables *stu-1*, *stu-2*, *stu-3*, and *stu-4*, with a maximum of four students per group. The group preferences for the assignment are represented by the fields *choice1* through *choice5*, where *choice 1* denotes their preferred choice and option 5 their least preferred one. The group affected choice, determined by a number of circumstances, is indicated in the *affected-choice* field.

	idgrp	moyG_group	stu_1	stu_2	stu_3	stu_4	choice1	choice2	choice3	choice4	choice5	affected_choice
<input type="checkbox"/>	1	11.78	ABDELOUAHAB Dounia	ASSASSI Brahim	ASSASSI Salah eddine	NULL	38	32	36	6	32	32
<input type="checkbox"/>	2	14.50	BENAOUJ Assia	BENBRAIKA Mohamed zakaria	BERBAKH Ikram	BOUABID Abdessettar	17	1	39	3	7	17
<input type="checkbox"/>	3	12.88	BOUAICHA Djihane	ELGUERRI Mohamed seddik	GASMI Mahdi taki eddine	NULL	4	33	19	9	11	33
<input type="checkbox"/>	4	14.08	GHANEMI Oussama	GHERBIA Meriem	NULL	NULL	28	36	8	9	9	28
<input type="checkbox"/>	5	13.84	GHERD Issam	HADEF Mohamed badis	HAFAYED Abdelaziz	NULL	38	4	40	30	25	38
<input type="checkbox"/>	6	13.27	HELIS Islam	HENNI Abdelkader	KAZAR Louaye	NULL	6	9	2	37	1	6
<input type="checkbox"/>	7	14.98	KHARFALLAH OUSSAMA	MEKHEL Amel	NULL	NULL	31	19	3	10	29	31
<input type="checkbox"/>	8	13.89	MILLOUJ Nahla	NULL	NULL	NULL	4	40	9	38	25	4
<input type="checkbox"/>	9	13.67	MOHAMMEDI Rabab	MOUAKI BACHIRI Imane	NOUAR Elmoustaz billah	NULL	38	30	1	40	12	30
<input type="checkbox"/>	10	14.96	SALHI Ali	SEDRATI Ahmed abderraouf	TABBI BACHIR	NULL	29	35	1	30	3	29
<input type="checkbox"/>	11	12.44	YOUCEF Abdelhalim	ZEGHADA SIHEM	NULL	NULL	30	3	25	12	34	3
<input type="checkbox"/>	12	12.71	BENAMEUR Meriem	NULL	NULL	NULL	38	17	4	28	31	NULL

Figure 4.5. Bachelors assignments table

4.4 Implementation: Features and functionalities

This section provides a detailed overview of the features and functionalities implemented in our application. It is divided into two parts: Web and Mobile. Each subsection highlights the various interfaces available to different users, including professors, students and administrators.

4.4.1 Web Application

The web application serves as the primary platform for managing and accessing the various features of our system. It includes interfaces for professors, students, and administrators, each tailored to their specific needs and roles. Here following the key interfaces available in the web application.

1. **Login:** Every user sees the interface shown in figure 4.6 after establishing a connection to the web application. He connects to his account successfully and displays the home page if the username and password are accurate. If not, it denies the connection and displays an error message, as shown in figure 4.7.

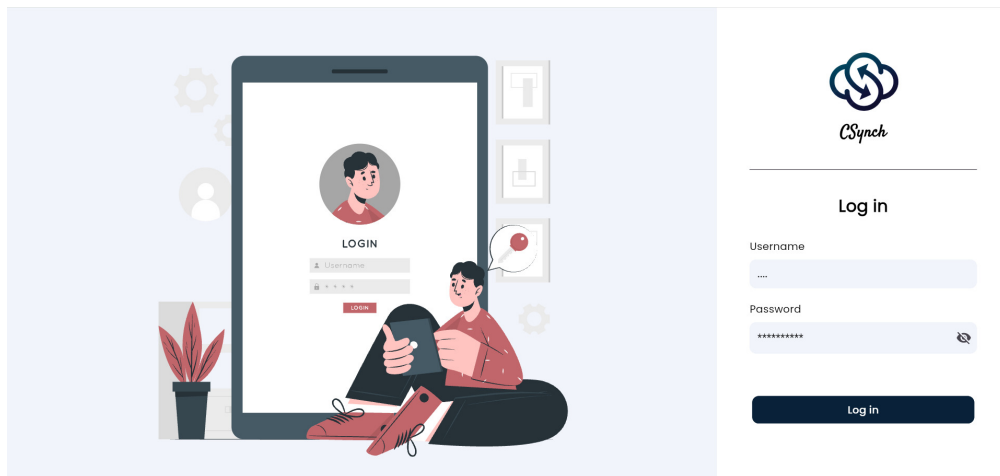


Figure 4.6. Web login page

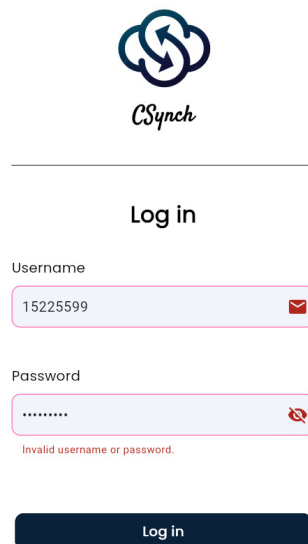


Figure 4.7. Web login page error

2. **Home page:** After a successful login, each user's home page whether they are a student, professor or administrator will show up with features specific to their roles. To improve the applications usability and functionality, each menu item is made to provide instant access to the needed areas. Students, professors and administrators

have different needs, and the navigation system is designed to meet those needs. This makes it possible for all users to quickly locate and make use of the functions that are required for their particular responsibilities. figures 4.8, 4.9, 4.10 and 4.12 present the unique user interfaces for each user group, highlighting the personalized navigation features offered by the application.

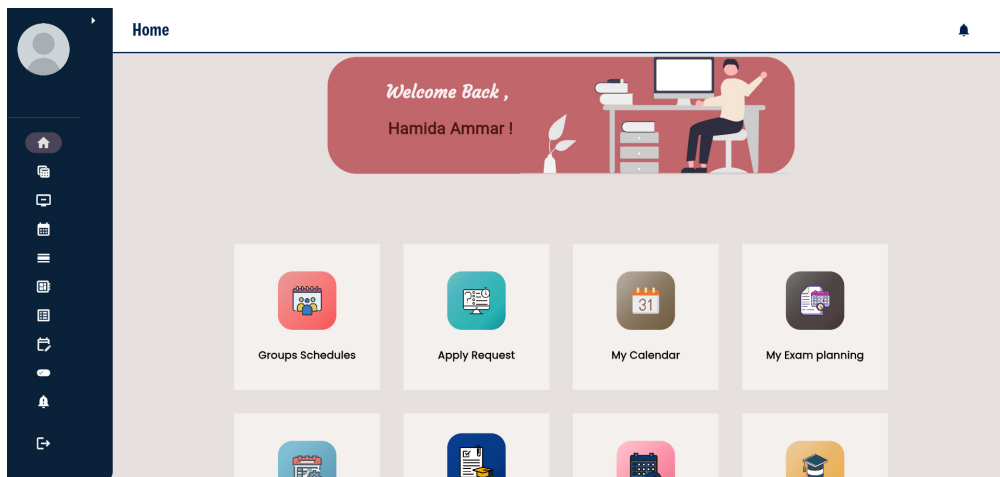


Figure 4.8. Web admin home page

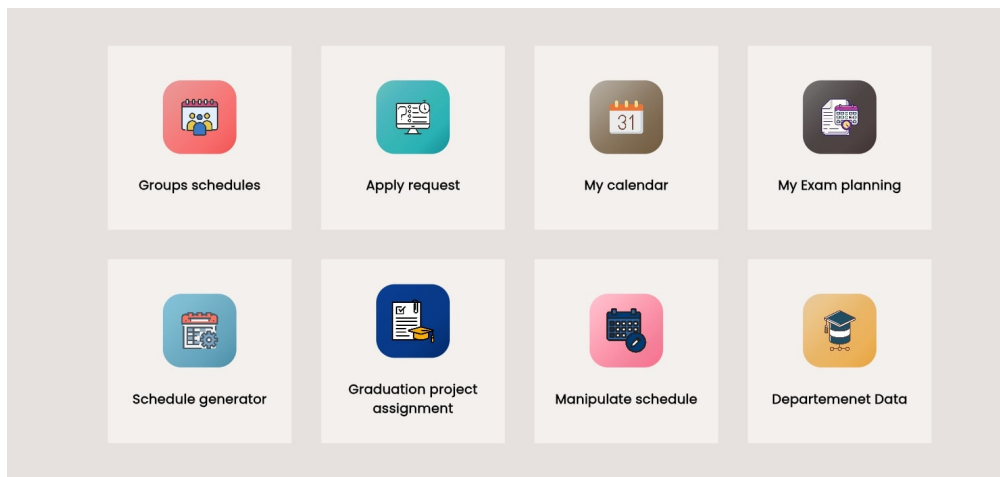


Figure 4.9. Web admin menu home page

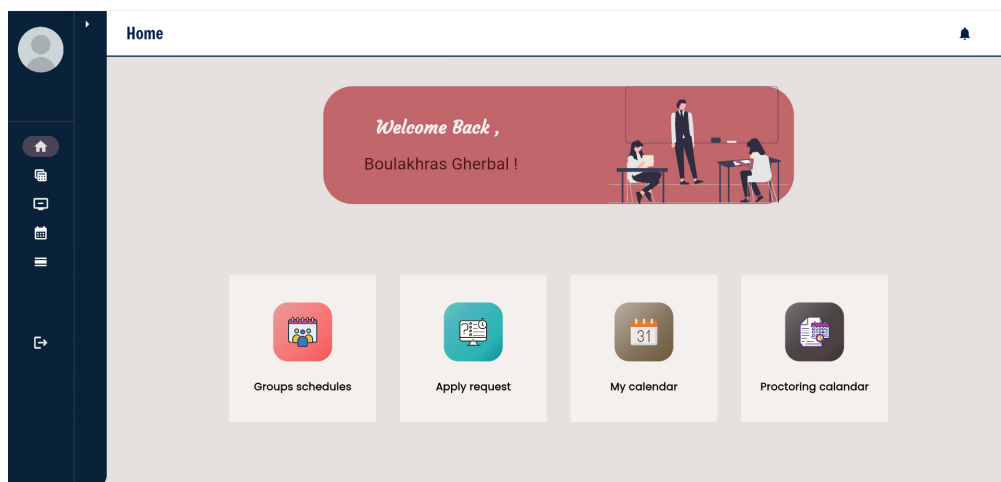


Figure 4.10. Web professor home page

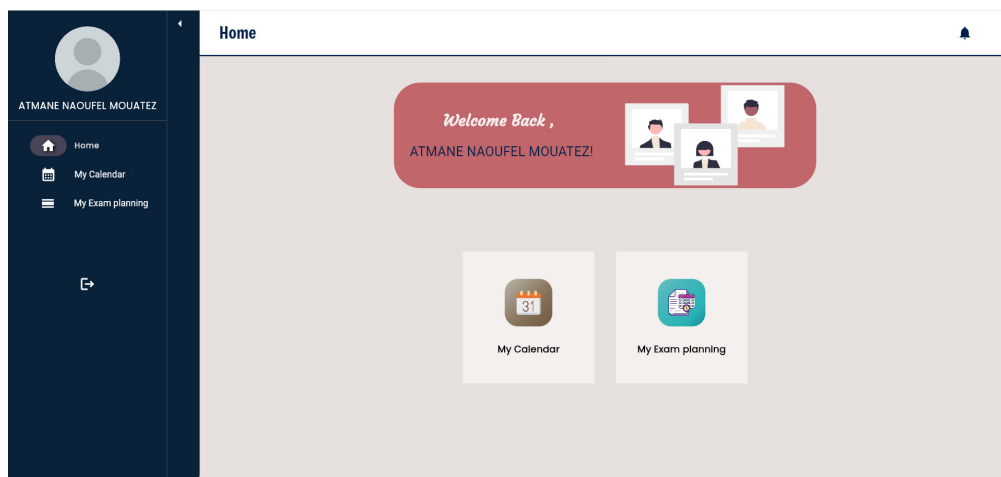
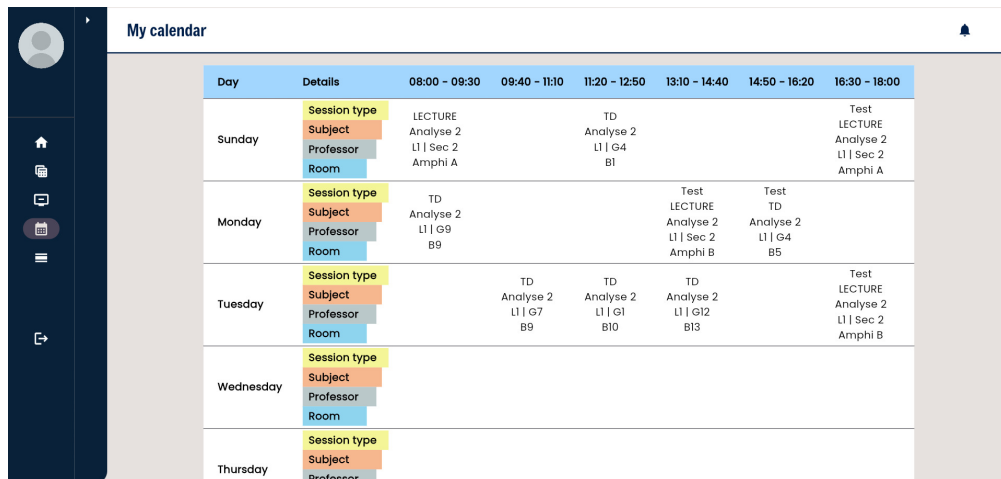


Figure 4.11. Web student home page

3. Professor Interfaces

The professors can consult their daily and proctoring schedules, manage group schedules, and apply for temporary sessions such as tests, replacements, and consultations. Additionally, professors receive notifications about any updates or changes to their schedules, ensuring they stay informed and organized.

- **Calendar:** This interface displays the daily schedule of a professor, see figure 4.12.



Day	Details	08:00 - 09:30	09:40 - 11:10	11:20 - 12:50	13:10 - 14:40	14:50 - 16:20	16:30 - 18:00
Sunday	Session type Subject Professor Room	LECTURE Analyse 2 LI Sec 2 Amphi A		TD Analyse 2 LI G4 B1			Test LECTURE Analyse 2 LI Sec 2 Amphi A
Monday	Session type Subject Professor Room		TD Analyse 2 LI G9 B9		Test LECTURE Analyse 2 LI Sec 2 Amphi B	Test TD Analyse 2 LI G4 B5	
Tuesday	Session type Subject Professor Room		TD Analyse 2 LI G7 B9	TD Analyse 2 LI G1 B10	TD Analyse 2 LI G12 B13		Test LECTURE Analyse 2 LI Sec 2 Amphi B
Wednesday	Session type Subject Professor Room						
Thursday	Session type Subject Professor						

Figure 4.12. Web professor calendar

- Proctoring Schedule:** Similar to the calendar, this interface also displays the professors proctoring schedule which is linked to specific dates. The display adjust depending on the state of the exam schedule, if the exam schedule is not generated yet, it will be shown in the figure 4.13.

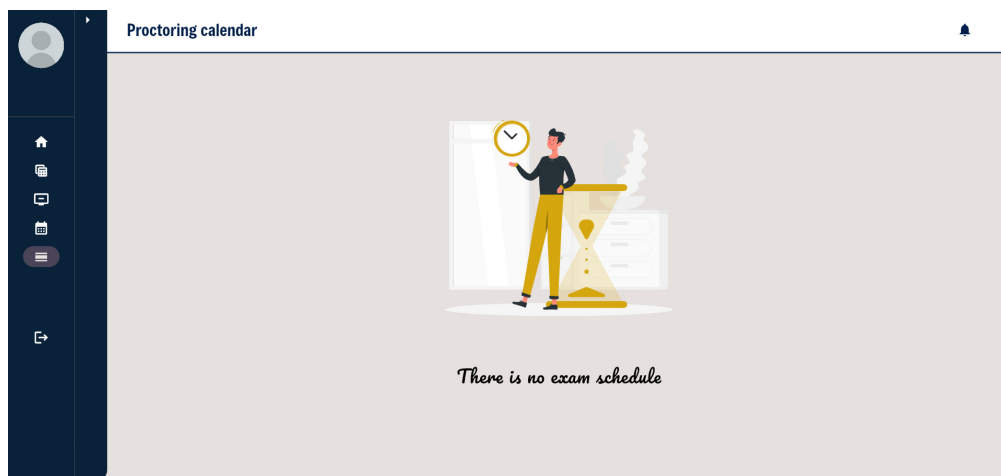


Figure 4.13. Professor proctoring schedule waiting for generation

When the schedule is generated, the figure 4.14 illustrates how the interface will appear.

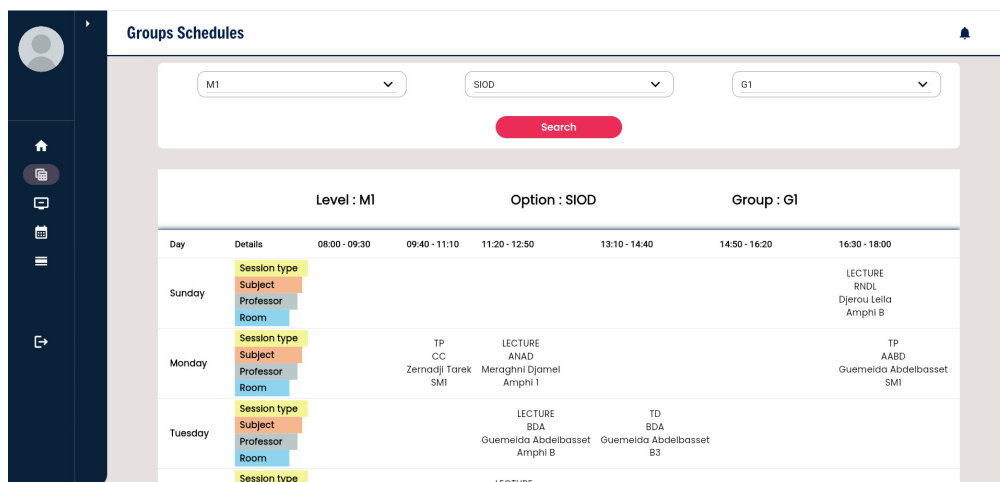
Day	Details	08:00 - 09:30	10:30 - 12:00	12:30 - 14:00	14:30 - 16:00
Saturday	Subject Professor Room				
Sunday 2024-06-16	Subject Professor Room		Analyse 2 Sec2 111 Amphi B		
Monday	Subject Professor Room				
Tuesday	Subject Professor Room				
Wednesday	Subject Professor Room				
Thursday	Subject Professor Room				

Figure 4.14. Professor proctoring schedule

- **Groups schedule:** this interface displays the schedule of all the groups in the department by specifying the level, option and group name. Initially, the interface will be displayed as the figure 4.15 where the professor can select the level, option and group.

Figure 4.15. Search groups schedule

As displayed in figure 4.16, the interface refreshes to provide a full view of the schedules for the selected group once these selections are made.



		Level : M1		Option : SIOD		Group : G1	
Day	Details	08:00 - 09:30	09:40 - 11:10	11:20 - 12:50	13:10 - 14:40	14:50 - 16:20	16:30 - 18:00
Sunday	Session type Subject Professor Room						LECTURE RNDL Djerrou Leila Amphi B
Monday	Session type Subject Professor Room		TP CC Zernadji Tarek SMI	LECTURE ANAD Meraghni Djamel Amphi I			TP AABD Guemeida Abdelbasset SMI
Tuesday	Session type Subject Professor Room			LECTURE BDA Guemeida Abdelbasset Amphi B		TD BDA Guemeida Abdelbasset B3	

Figure 4.16. Display groups schedule

- Apply request:** Here, a menu will appear as shown in figure 4.17, allowing the professor to manage scheduling temporary sessions (tests, replacements, and consultations) by navigating in the menu items. Specifying the session type (lecture, tutorial, lab work), level, option, group(s), day, date, time, room, and finally subject, With each selection, the available choices are filtered based on the previously selected options. After filling out all the necessary information, the professor confirms the request. If the request has been accepted, a confirmation message will show up.

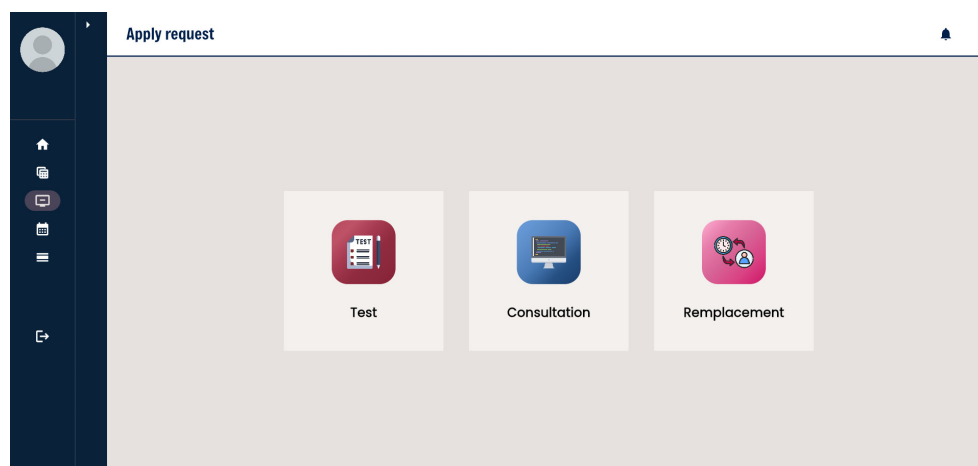
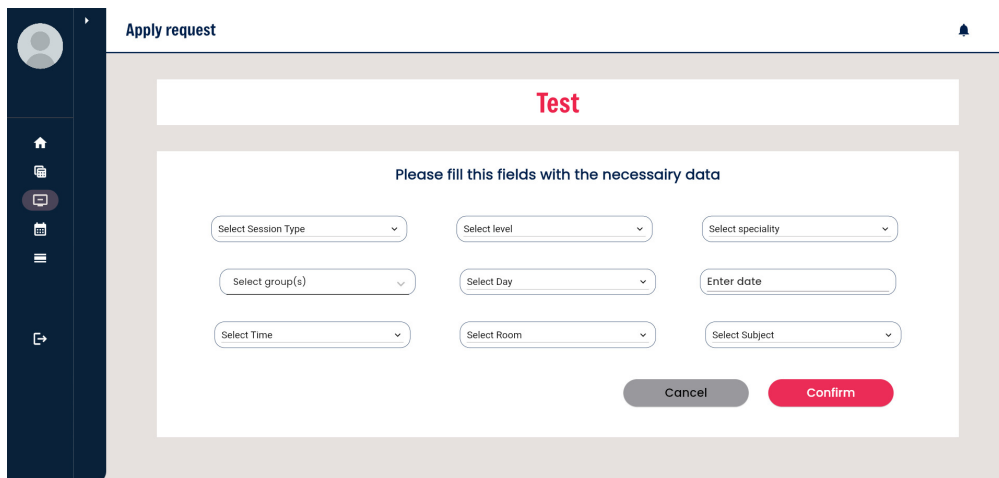


Figure 4.17. Apply request

Test: If the professor want to schedule a test session, this is the appropriate interface to use. The initial screen shown in figure 4.18 presents empty fields for selecting the session details. The level, option and subject field will be dynamically filtered based on the professor's teaching schedule. After choosing the group(s), the desired day and the date, the time field will also be dynamically filtered based on the free time that the session can be scheduled on. The same process is applicable to the room. First, it will be filtered based on the session type, then for the available rooms at the selected time.



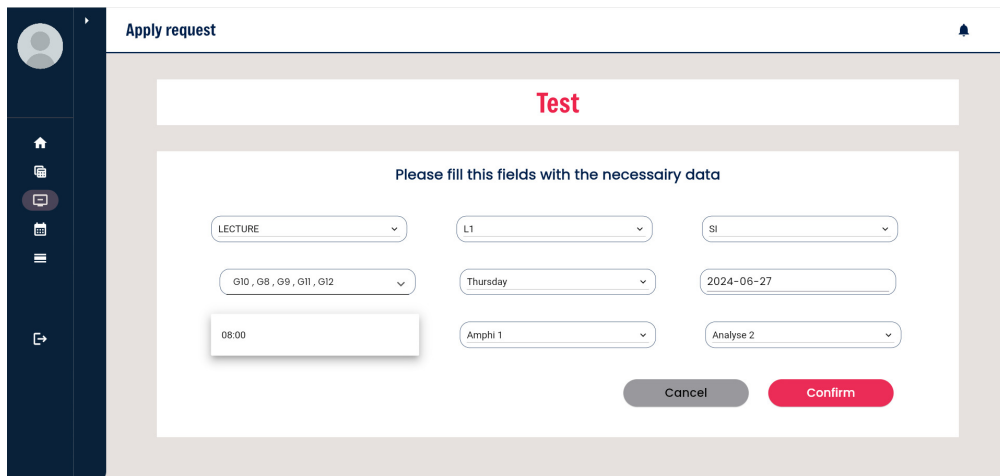
The screenshot shows a web application interface titled "Apply request" with a user profile icon and a notification bell in the top right. The main content area is titled "Test" in red. Below the title, there is a prompt: "Please fill this fields with the necessary data". The form consists of nine input fields arranged in a 3x3 grid:

- Row 1: "Select Session Type" (dropdown), "Select level" (dropdown), "Select speciality" (dropdown)
- Row 2: "Select group(s)" (dropdown), "Select Day" (dropdown), "Enter date" (text input)
- Row 3: "Select Time" (dropdown), "Select Room" (dropdown), "Select Subject" (dropdown)

At the bottom right of the form, there are two buttons: a grey "Cancel" button and a red "Confirm" button.

Figure 4.18. Schedule a test

Once the professor fill out this fields, exactly as shown in figure 4.19, then by clicking on the confirm button the session will be add to the concerned group(s) schedule, which initialize automatic notification to alert the related group(s).



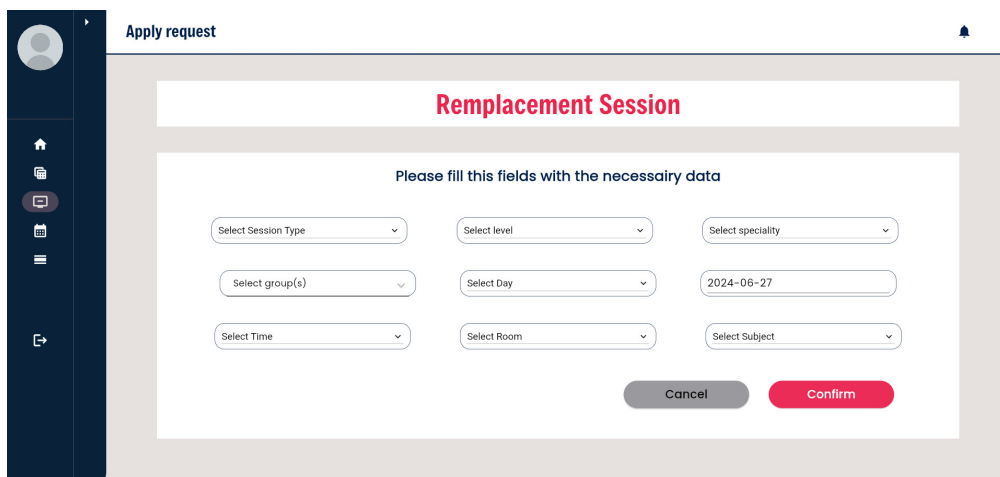
The screenshot shows a web interface titled "Apply request" with a notification bell icon in the top right. The main content area is titled "Test" in red. Below the title, there is a prompt: "Please fill this fields with the necessary data". The form contains the following fields:

- LECTURE (dropdown menu)
- L1 (dropdown menu)
- SI (dropdown menu)
- G10, G8, G9, G11, G12 (dropdown menu)
- Thursday (dropdown menu)
- 2024-06-27 (text input)
- 08:00 (text input)
- Amphi 1 (dropdown menu)
- Analyse 2 (dropdown menu)

At the bottom right of the form, there are two buttons: "Cancel" (grey) and "Confirm" (red).

Figure 4.19. Test scheduling

Replacement: Scheduling a replacement session as displayed in figure 4.20, following the same process as the test session.



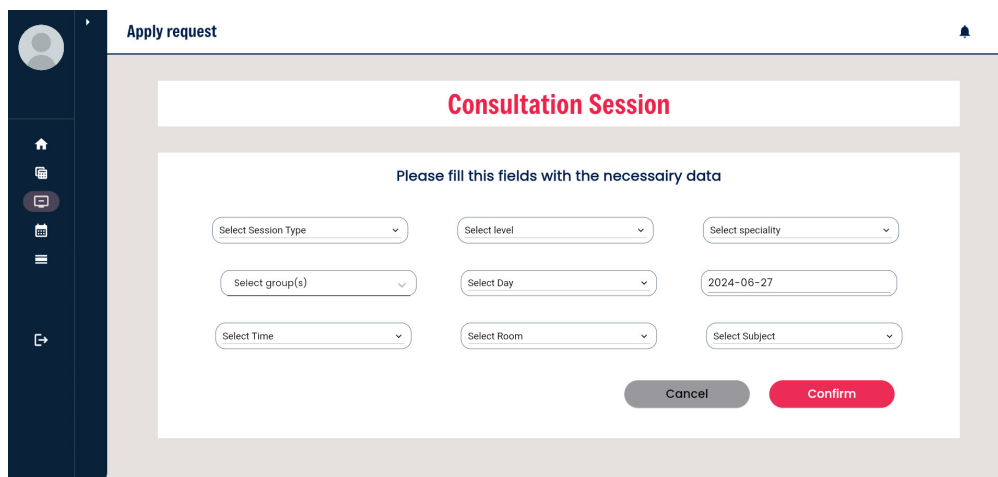
The screenshot shows a web interface titled "Apply request" with a notification bell icon in the top right. The main content area is titled "Replacement Session" in red. Below the title, there is a prompt: "Please fill this fields with the necessary data". The form contains the following fields:

- Select Session Type (dropdown menu)
- Select level (dropdown menu)
- Select speciality (dropdown menu)
- Select group(s) (dropdown menu)
- Select Day (dropdown menu)
- 2024-06-27 (text input)
- Select Time (dropdown menu)
- Select Room (dropdown menu)
- Select Subject (dropdown menu)

At the bottom right of the form, there are two buttons: "Cancel" (grey) and "Confirm" (red).

Figure 4.20. Replacement scheduling

Consultation: For scheduling a consultation session, the professor will follow the same steps as scheduling a test session, but the only difference between these two processes is that in this feature, the professor can only schedule lab work evaluation sessions.



The screenshot shows a web interface for scheduling a consultation session. The page title is "Apply request". The main heading is "Consultation Session" in red. Below the heading, there is a prompt: "Please fill this fields with the necessary data". The form contains several input fields: "Select Session Type", "Select level", "Select speciality", "Select group(s)", "Select Day", "2024-06-27", "Select Time", "Select Room", and "Select Subject". At the bottom right, there are two buttons: "Cancel" (grey) and "Confirm" (red).

Figure 4.21. Consultation scheduling

- **Notification:** After applying any updates in the schedule, a notification message will appear as it presented in figure 4.22.

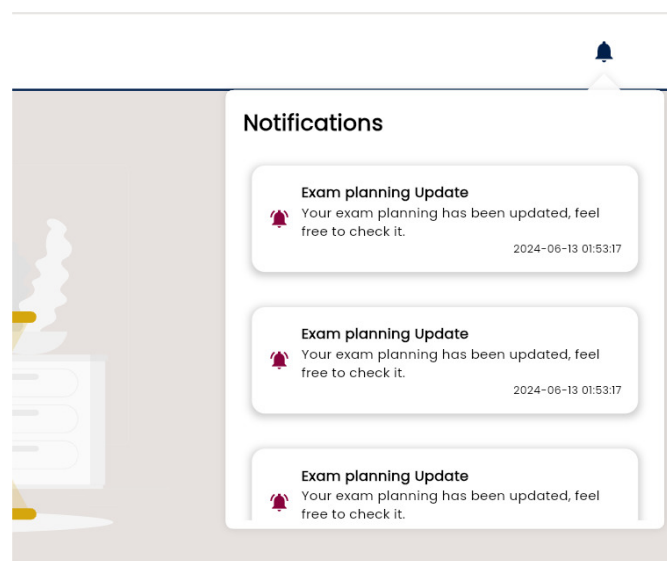


Figure 4.22. Notifications

4. Admin Interfaces

The admin, who is also a professor, have access to all the functionalities available to professors, as well as administrative features. Since the professor functionalities have been previously explained, we will focus here only on the administrative features.

- **Generate Schedules:** This interface includes the menu for selecting generation options, such as generating the semester schedule or exam schedule, like it is presented in figure 4.23.

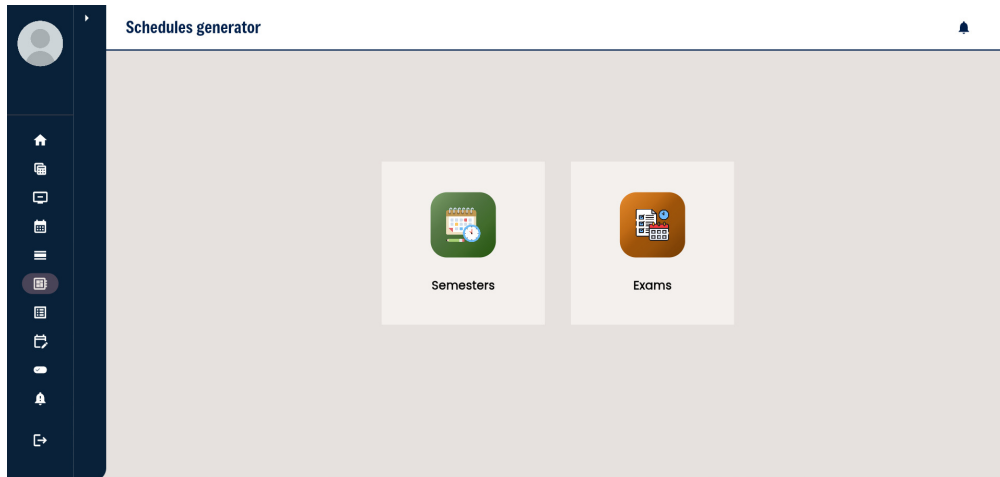


Figure 4.23. Schedule generator menu

Semester: As shown in figure 4.24, the admin start the automatic generation of schedules for both professors and groups, after specifying the semester and the current academic year.

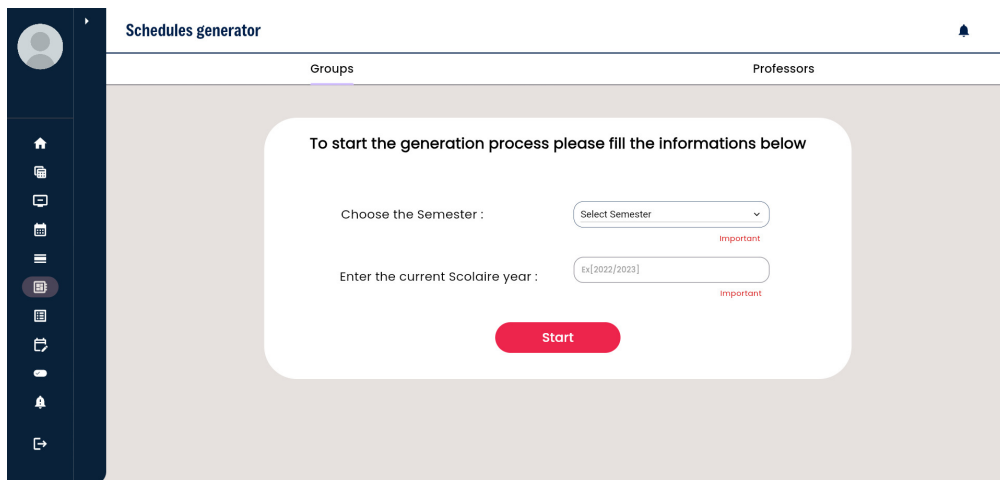


Figure 4.24. Selecting semester schedule

Afterwards, a progress bar will be appeared, as seen in figure 4.25, which indicate

that the generation is still processing.

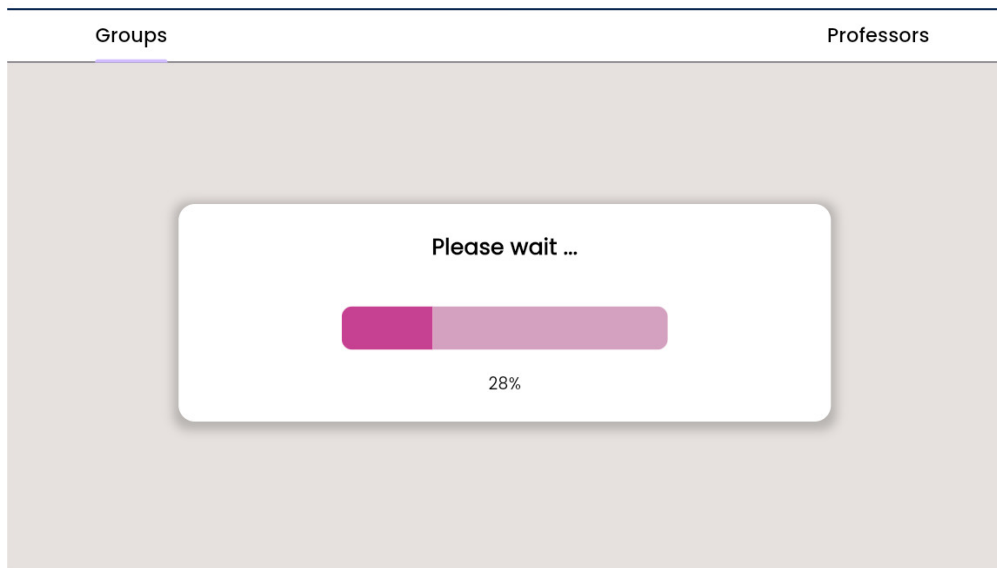


Figure 4.25. Progress bar for waiting the scheduling process

Meanwhile, the professor schedule generation can't be done till the groups schedule is generated, and will be displayed to the admin as shown in the figure 4.31.

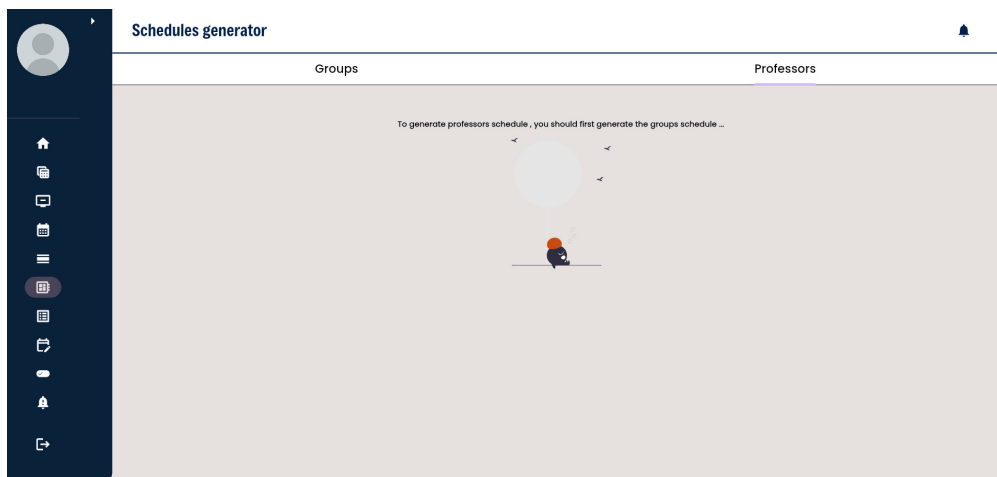


Figure 4.26. Professor schedule before the generation

After the generation process is done, the generated groups schedule will be displayed as seen in the figure 4.27.

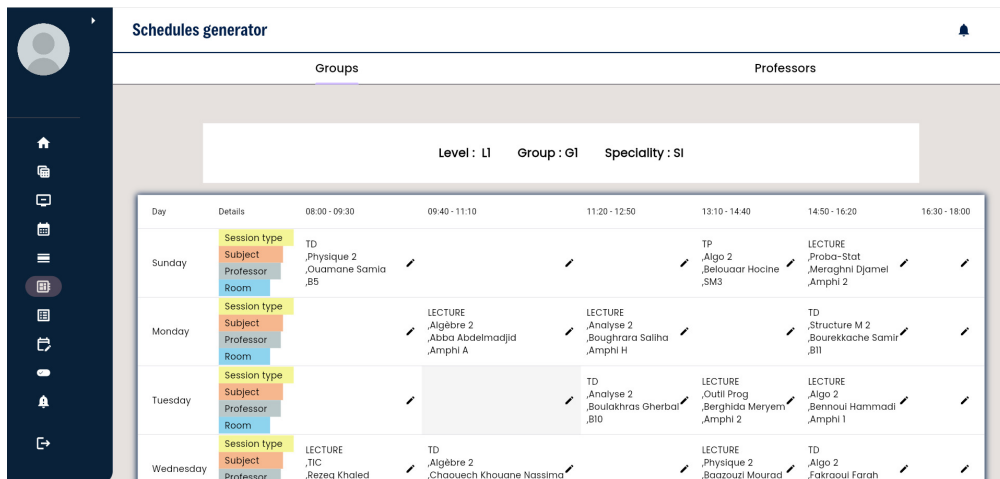


Figure 4.27. Groups schedule after the generation

If the admin want to add, edit or delete a session, he can press on the table cell and then a dialog field will be shown in the figure 4.28. After the admin verify the



Figure 4.28. Manipulation of the generated schedule

schedule, he can press the confirm button in the figure 4.29, which automatically insert the schedule into the database.

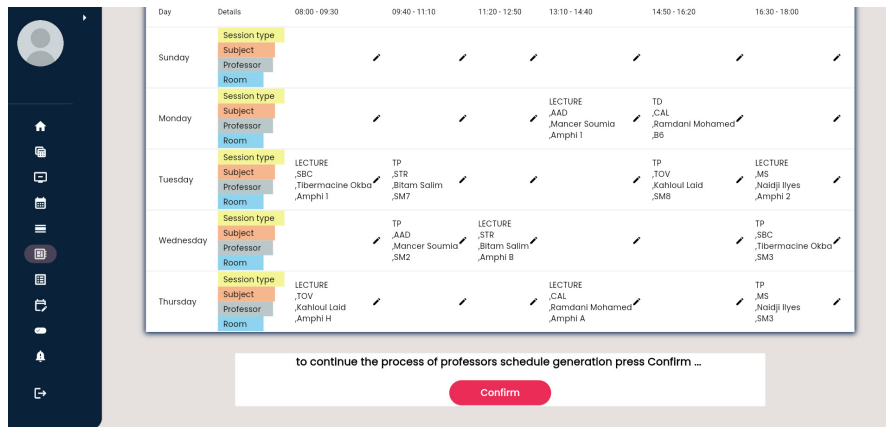


Figure 4.29. Confirming the insertion for web

For informing the admin that the inserting data has been successfully done, a notification will be appeared as the figure 4.30 represents.

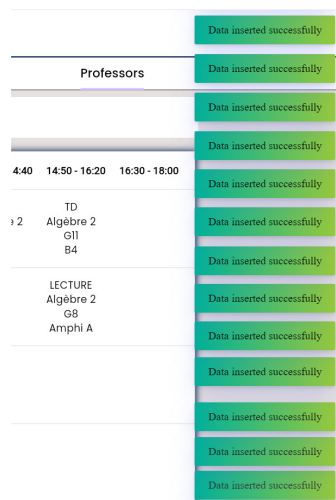


Figure 4.30. Insert the schedule in database for web

The professor schedule will be displayed same as the figure 4.31 illustrate.

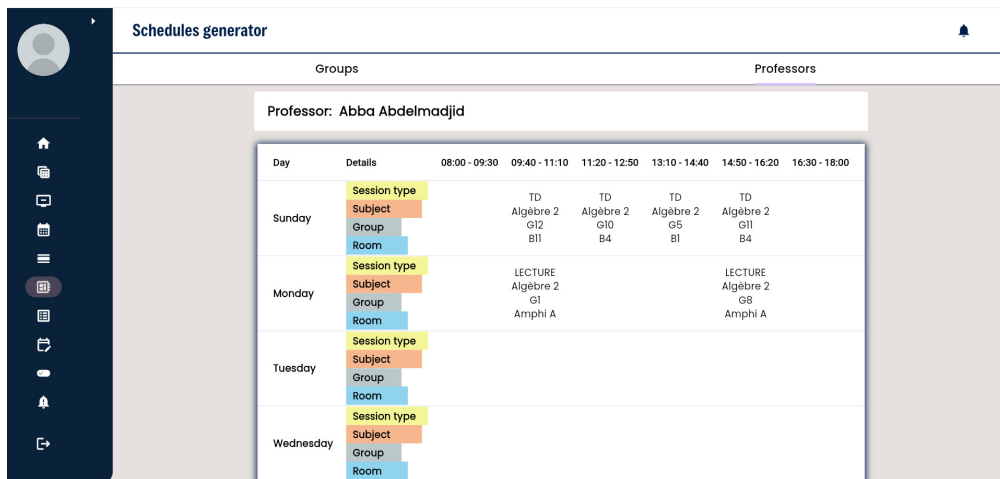


Figure 4.31. Professor schedule after the generation for web

For publishing the schedule, the admin press on the publish button that is displayed on the figure 4.32, which is in charge of notifying professors via email with their schedules in Excel format are attached.

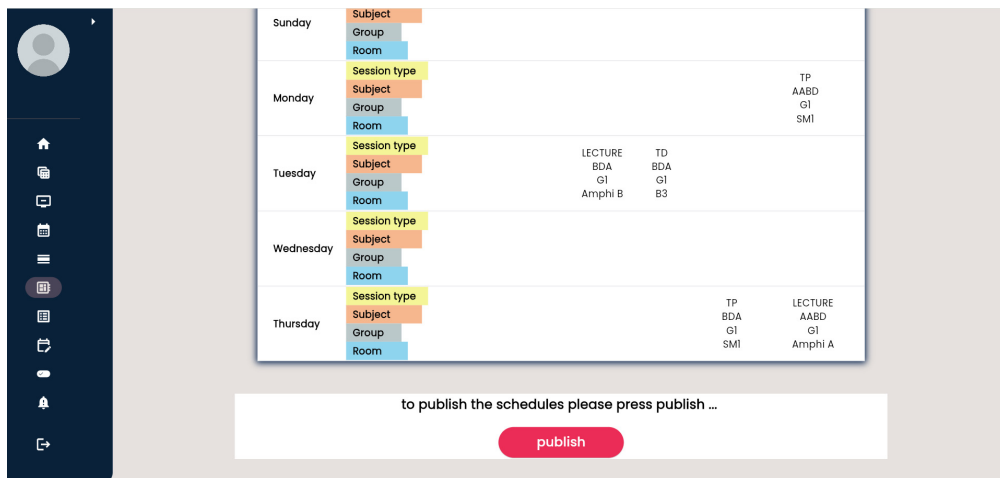


Figure 4.32. Publish the schedule for web

The figures 4.33 and 4.34 describe the publishing process results.

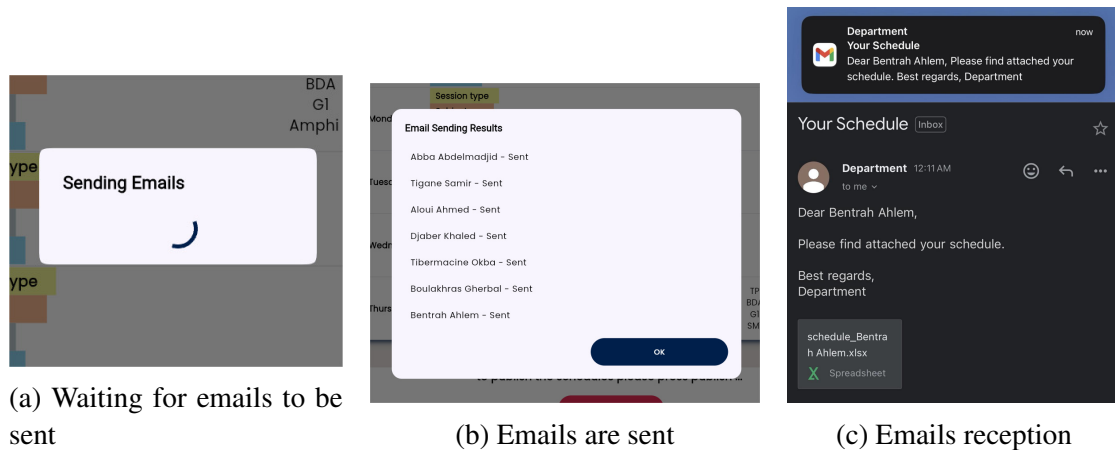
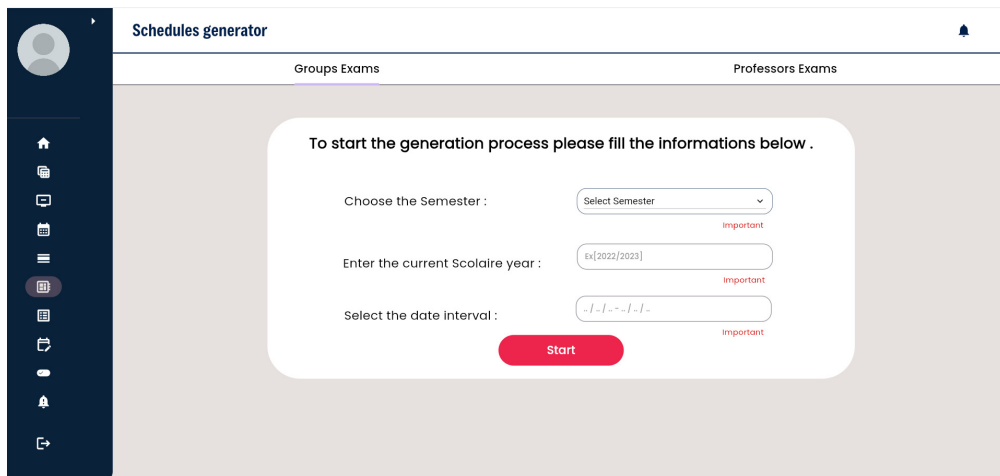


Figure 4.33. Email sending process

Université Mohamed Khider - Biskra Faculty of Exact Sciences, Natural and Life Sciences Computer Science Departement Schedule of the second semester 2022/2023									
		08:00 - 09:30	09:40 - 11:10	11:20 - 12:50	13:10 - 14:40	14:50 - 16:20	16:30 - 18:00		
Sunday	Subject								
	Level								
	Professor								
Monday	Subject		LECTURE PGGPU				LECTURE MR		
	Level		M1 IVA - Sec 1/G1				M1 IA - Sec 1/G1		
	Professor		Babahenini Med Chaouki				Babahenini Med Chaouki		
Tuesday	Subject								
	Level								
	Professor								
Wednesday	Subject	TD SE 1		TP SE 1					
	Level	L2 SI - Sec 1/G7		L2 SI - Sec 1/G1					
	Professor	Babahenini Med Chaouki		Babahenini Med Chaouki					
Thursday	Subject			LECTURE RS		TP PGGPU			
	Level			L3 SI - Sec 1/G7		M1 IVA - Sec 1/G1			
	Professor			Babahenini Med Chaouki		Babahenini Med Chaouki			
	Room			Amphi A		SM7			

Figure 4.34. Professor schedule excel format

Exam: Same process as the semester scheduling, but with specifying the starting and ending dates. The figure 4.35 illustrates the selection fields of the necessary details (semester, semester year and date interval) before starting the exam schedule generation.



The screenshot shows a web application titled "Schedules generator". It has a dark blue sidebar on the left with various navigation icons. The main content area is divided into two tabs: "Groups Exams" (active) and "Professors Exams". A white rounded rectangle contains the following form fields:

- Choose the Semester :** A dropdown menu with "Select Semester" and a red "Important" label below it.
- Enter the current Scolaire year :** A text input field containing "Ex[2022/2023]" and a red "Important" label below it.
- Select the date interval :** A date range input field with a red "Important" label below it.

A red "Start" button is positioned below the date interval field.

Figure 4.35. Exam schedule generator

The admin select the start date and end date of the exams session as shown in figure 4.36.

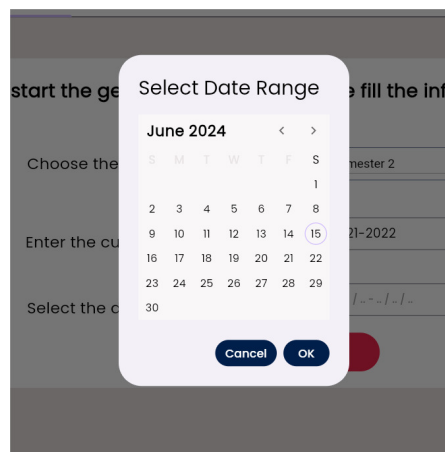


Figure 4.36. Choosing the exam interval dates

After filling out the necessary details and pressing start button, the generated exam groups schedules will be displayed as shown in the figure 4.37.

Schedules generator

Groups Exams | Professors Exams

Level : LI Section : 1 Speciality : SI

Day	Details	08:30 - 10:00	10:30 - 12:00	12:30 - 14:00	14:30 - 16:00
Saturday 2024-06-15	Subject Proctors Room				Structure M 2 Ramdani Mohamed Amphi H
Sunday 2024-06-16	Subject Proctors Room		Algèbre 2 Berrouisse Nassima Amphi 2		
Monday 2024-06-17	Subject Proctors Room			Proba-Stat Soukeur Abdessalam Amphi H	Outill Prog Youkana Imene Amphi 1
Tuesday 2024-06-18	Subject Proctors Room			Physique 2 Belamri Djamel Amphi 2	

Figure 4.37. The generated exam schedule for groups

After the admin verifies the generated group schedule, he presses confirm schedule, which results in the schedule data being inserted into the database. The professor proctoring schedule will be displayed as shown in the figure 4.38.

2024-06-15

2024-06-15	Subject Section Room			
Sunday 2024-06-16	Subject Section Room			
Monday 2024-06-17	Subject Section Room	BDA 1 B13		
Tuesday 2024-06-18	Subject Section Room		AABD 1 B5	
Wednesday 2024-06-19	Subject Section Room			
Thursday 2024-06-20	Subject Section Room			

to publish the schedules please press publish ...

publish

Figure 4.38. The generated proctoring schedule for professors

After pressing on the publish button, the professors will be notified by an email with their proctoring schedule are attached in their excel file format as shown in the figures 4.39 and 4.40.



	A	B	C	D	E	F	G
1							
2				Université Mohamed Khider - Biskra Faculty of Exact Sciences, Natural and Life Sciences Computer Science Departement			
3							
4							
5							
6	Professor:	Babahenini Med Chaouki					
7	Department:	Computer Science					
8							
9							
10				08:30 - 10:00	10:30 - 12:00	12:30 - 14:00	14:30 - 16:00
11	Saturday	Subject					
12		Section					
13		Room					
14	Sunday	Subject		MR			
15		Section		M1 IA - Sec 1			
16		Room		B8			
17	Monday	Subject					
18		Section					
19		Room					
20	Tuesday	Subject			RS	PGGPU	
21		Section			L3 SI - Sec 1	M1 IVA - Sec 1	
22		Room			Amphi A	B13	
23	Wednesday	Subject					
24		Section					
25		Room					
26	Thursday	Subject					
27		Section					
28		Room					
29							

Figure 4.39. The generated proctoring schedule excel format

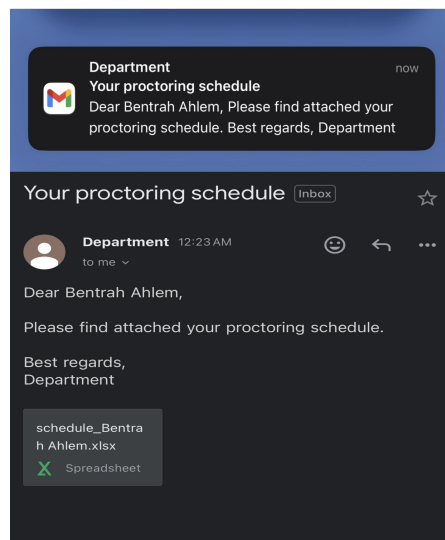


Figure 4.40. Email reception

- **Manipulate schedules:** To manage schedules, the admin must first select the specific level, option, and group name as shown in figure 4.41.

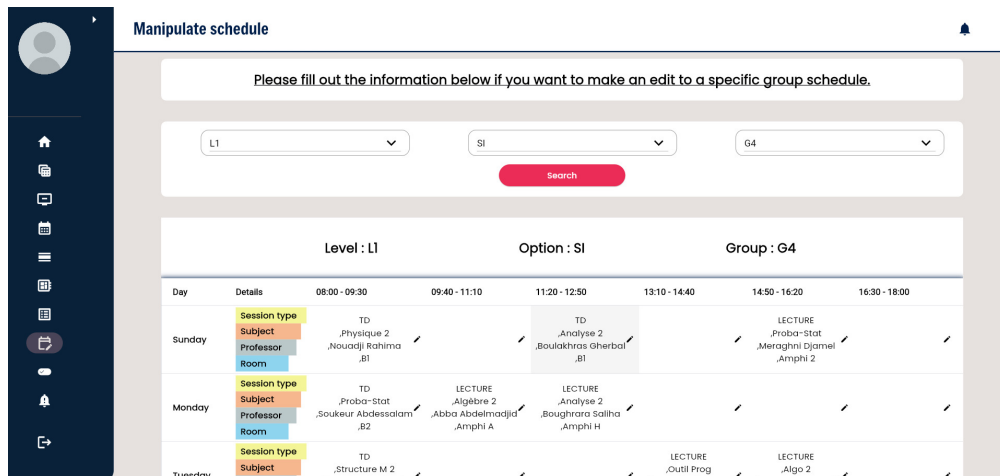


Figure 4.41. Manipulate schedule

The admin can edit, add, or delete sessions through the dialog fields shown in figure 4.42.



(a) Add session

(b) Edit or delete session

Figure 4.42. Manipulation options

- **Graduation project assignment:** This interface is divided into two part, one for the bachelor graduation assignment and the other for the masters. Allowing the admin to upload a file with specific format and data. After confirming the process, the result will appear.

Bachelor: By pressing on the select button that it is represented in the figure 4.43.

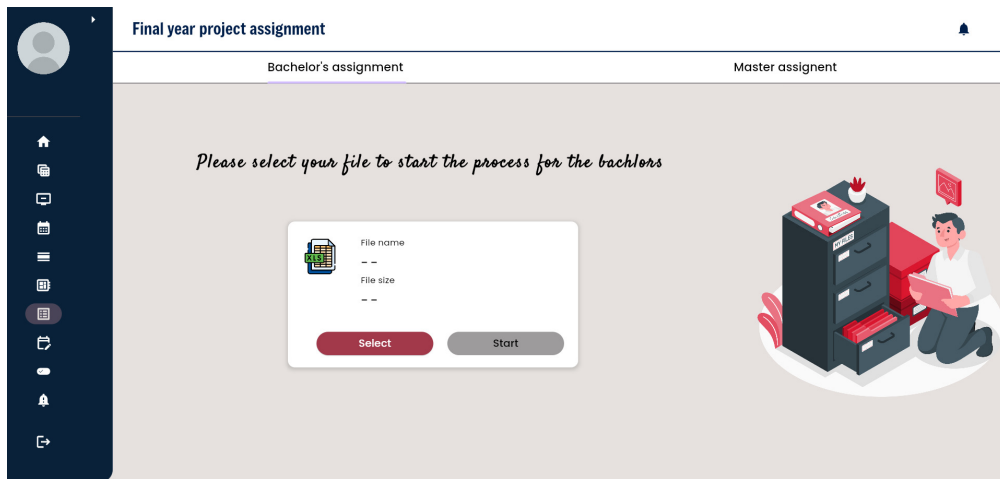


Figure 4.43. Bachelors project assignment

The admin start the bachelors final year project assignment by uploading the excel file as shown in the figure 4.44, which contains the groups of four students and their 5 desired choices. Starting from the most desirable choice to the least one.

ID	group	Name	MoyG_11	MoyG_12	moyG	Name.1	MoyG_11.1	MoyG_12.1	moyG.1	Name.2	MoyG_11.2	MoyG_12.2	moyG.2	Name.3	MoyG_11.3	MoyG_12.3	moyG.3	moyG_grou	choice 1	choice 2	choice 3	choice 4	choice 5	affected choi
1	7	KHARFALLA	12.6	16.85		MEKHEL Af	15.79	14.68											31	19	3	10	29	
2	10	SALHI AII	12.21	17.67		SEDRATI AP	16.52	17.07		TABBI BACI	12.43	13.87							29	35	1	30	3	
3	2	BENAOUI A	12.14	14.72		BENBRAKFA	16.65	10.77		BERBAKH H	17.09	17.86		BOUABIDA A	13.88	13.98			17	1	39	3	7	
4	4	GHANEM C	14.33	14.07		GHESBIA M	11.94	15.95											28	36	8	9	9	
5	8	MILOUDI N	15.52	12.26															4	40	9	38	25	
6	5	GHEDD Issa	17.07	13.87		HADEF Naf	10.03	14.09		HAFAYED A	13.30	14.57							38	4	40	30	25	
7	9	MOHAMME	11.88	11.44		MOUAKI BF	17.25	16.16		NOLAR Elm	12.77	13.25							38	30	1	40	12	
8	6	HEUIS Islam	15.5	10.1		HENNI Abd	14.96	14.63		KAZAR Lou	13.46	10.94							6	9	2	37	1	
9	3	BOUAICHA	14.2	10.28		ELGUERRI N	10.52	10.44		GASMI Mah	17.74	13.68							4	33	19	9	11	
10	11	YOUSSEF Ad	10.86	11.35		ZEGHADIA S	12.85	14.69											30	3	25	12	34	
11	1	ABDELOUAI	13.7	12.77		ASSASSI Bri	10.8	11.89		ASSASSI Sa	11.21	10.3							38	32	36	6	32	
12	12	BEN AMEUF	13.72	11.69															38	17	4	28	31	

Figure 4.44. Bachelors excel file

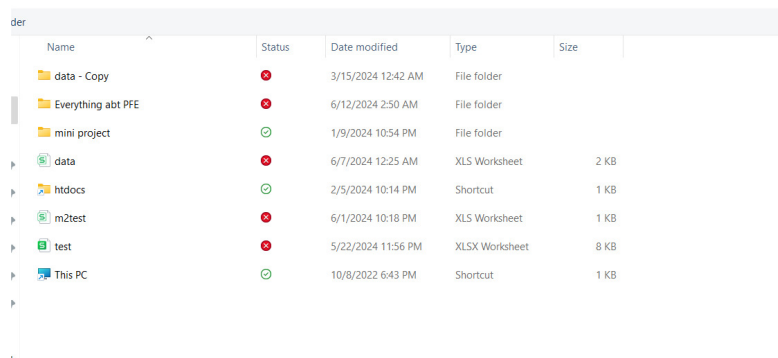
To enable the administrator to choose an Excel file, the file picker interface is called as seen in the following Dart code excerpt.

```
pickfile() async {
  FilePickerResult? result = await FilePicker.platform.pickFiles(
    type: FileType.custom,
    allowMultiple: false,
    allowedExtensions: ['.xls', 'xlsx', 'csv'],
  );

  if (result != null && result.files.isNotEmpty) {
    final fileBytes = result.files.first.bytes;
    final filesize = result.files.first.size;
    final filesizeInKB = filesize / 1024;
    final fileName = result.files.first.name;
    setState(() {
      filename = fileName;
      filesize = "${filesizeInKB.toStringAsFixed(2).toString()} KB";
      start = true;
    });
    await uploadFile( fileName, fileBytes!);
  }
}
```

Figure 4.45. Pick file code

A file chooser will be pop up to allow the admin to select only the excel files format as it is illustrated in the figure 4.46.



Name	Status	Date modified	Type	Size
data - Copy	✖	3/15/2024 12:42 AM	File folder	
Everything abt PFE	✖	6/12/2024 2:50 AM	File folder	
mini project	✔	1/9/2024 10:54 PM	File folder	
data	✖	6/7/2024 12:25 AM	XLS Worksheet	2 KB
htdocs	✔	2/5/2024 10:14 PM	Shortcut	1 KB
m2test	✖	6/1/2024 10:18 PM	XLS Worksheet	1 KB
test	✖	5/22/2024 11:56 PM	XLSX Worksheet	8 KB
This PC	✔	10/8/2022 6:43 PM	Shortcut	1 KB

Figure 4.46. Bachelors project assignment file chooser

After selecting, the admin can start the file upload process by pressing the 'Start' button, as shown in figure 4.47.

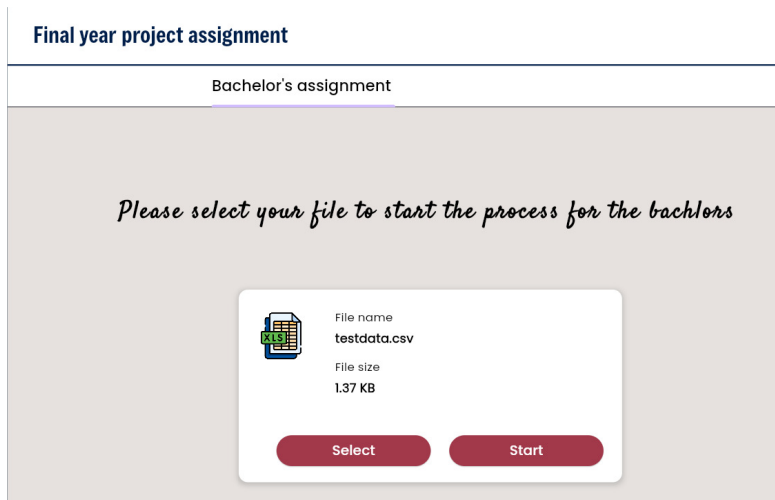


Figure 4.47. Selection of the file

This action triggers the *runAlgorithm* function in the Flutter application, which sends the file to the Flask server. The code for this function is illustrated in Figure 4.48.

```
Future<void> runAlgorithm(String fileName, Uint8List fileBytes) async {
  var uri = Uri.parse("http://localhost:5000/upload");

  var request = http.MultipartRequest('POST', uri)
    ..files.add(http.MultipartFile.fromBytes(
      'file',
      fileBytes,
      StreamedResponse response
    ))
    ..Type: StreamedResponse;
  var response = await request.send();
  if (response.statusCode == 200) {
    print("File uploaded successfully.");
  } else {
    print("File upload failed with status code: ${response.statusCode}");
  }
}
```

Figure 4.48. Upload code

The resulted assignment will be displayed as the figure 4.49. By clicking the confirm button, a notification will be sent to all the concerned part and emails will be also sent to all the professor for informing them as seen in the figure 4.50.

ID	Name Students	Group's Average	Choice 1	Choice 2	Choice 3	Choice 4	Choice 5	Affected Choice
1	ABDELOUHAB Dounia ASSASSI Brahim ASSASSI Salah eddine	11.78	38	32	36	6	32	33 - Biometric Authentication Technology
2	BENAOUI Assia BENBRAKIA Mohamed zakaria BERSAKHI Isam BOUABID Abdelattar	14.50	17	1	39	3	7	17 - Web Application Frameworks
3	BOUAKICH Djilane ELGUERNI Mohamed seddik GASMI Mahdi tairi eddine	12.88	4	33	19	9	11	33 - Smart Waste Management System
4	GHANEMI Oussama GHERBIA Meriem	14.08	28	36	8	9	9	28 - Cryptocurrency Analysis
GHERD Issam								18 - Virtual-Contentful Desktop

Figure 4.49. Displaying the assignment result

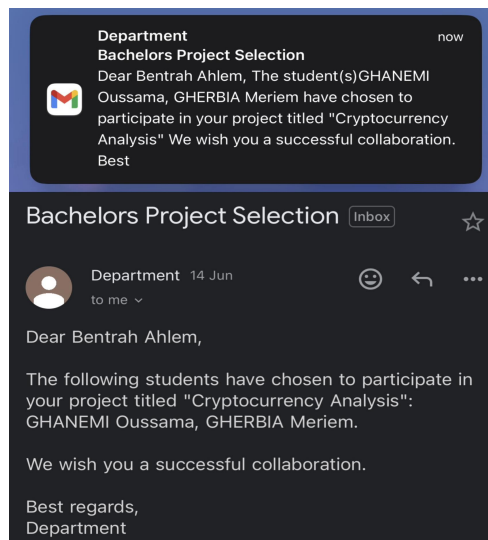


Figure 4.50. Email reception after the affection

Master: Master final year project assignment works in the same process as the bachelors assignment, the only difference is that the uploaded excel file contains groups of just 3 students max. The admin presses the select button as shown in 4.51, which results in a pop-up file picker to allow him to pick the desired Excel file.

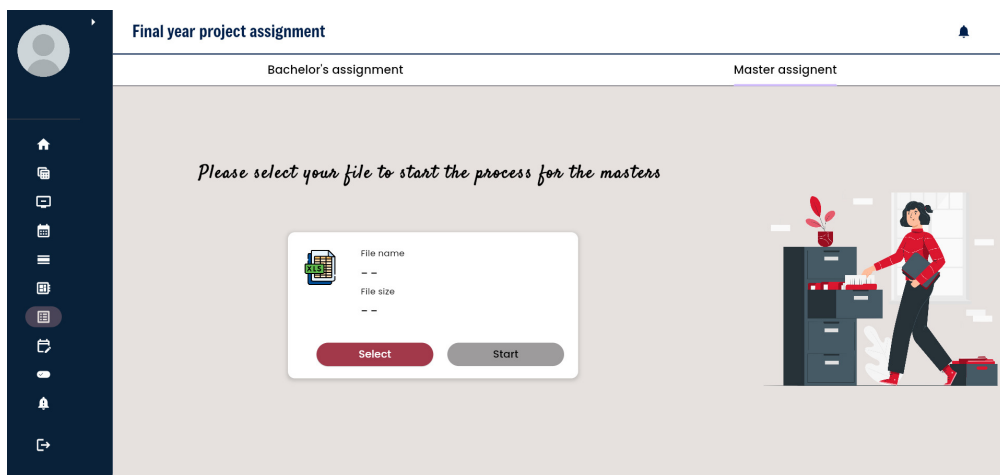


Figure 4.51. Masters project assignment

After the selection, the admin can start the assignment process by clicking on the start button that is illustrated in the figure 4.52.

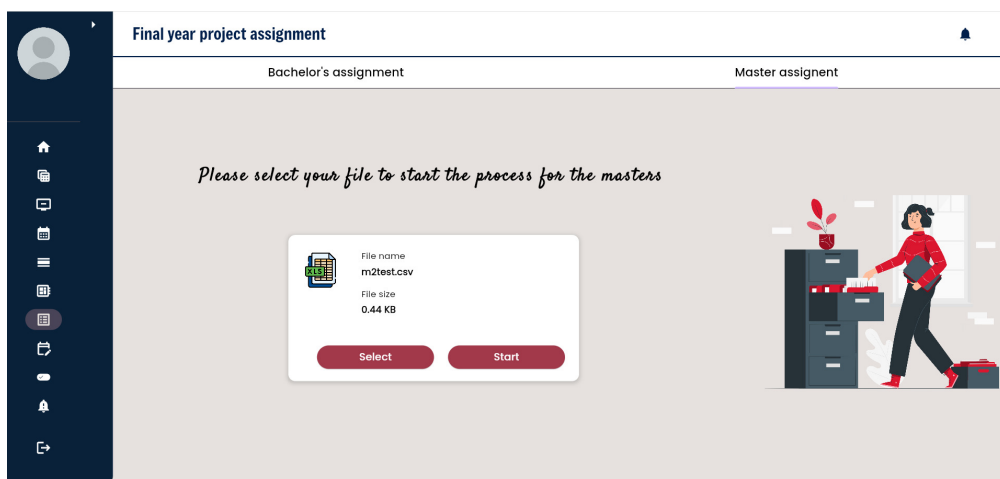
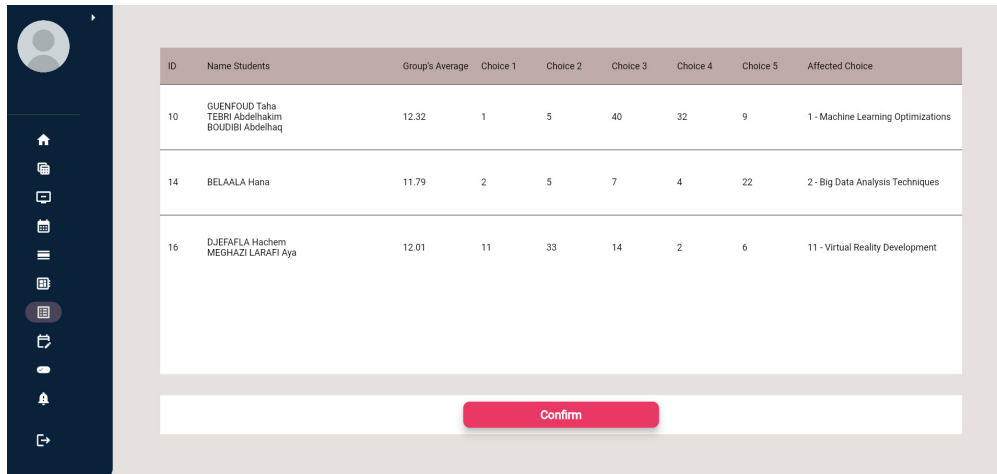


Figure 4.52. Selection of the master file

Whenever the assignment process is completed, the resulted assignment project groups will be displayed as the figure 4.53 presents.



ID	Name Students	Group's Average	Choice 1	Choice 2	Choice 3	Choice 4	Choice 5	Affected Choice
10	GUENFOUD Taha TEBRI Abdelhakim BOUDIBI Abdelhaq	12.32	1	5	40	32	9	1 - Machine Learning Optimizations
14	BELAALA Hana	11.79	2	5	7	4	22	2 - Big Data Analysis Techniques
16	DJEFAFLA Hachem MEGHAZI LARAFI Aya	12.01	11	33	14	2	6	11 - Virtual Reality Development

Figure 4.53. Displaying the masters assignment result

After pressing the confirm button, all the concerned parties will be notified. For example, the professor will be alerted via email, as presented in figure 4.54.

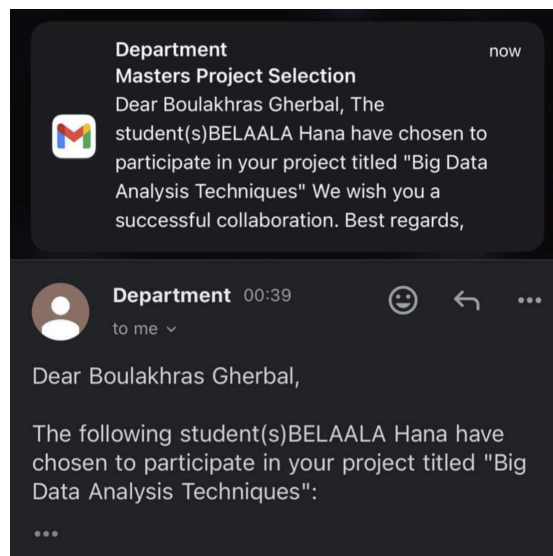


Figure 4.54. Email reception after the assignment

- **Manipulate department data:** The interface shown in figure 4.55 contains a menu for manipulating department data, covering a comprehensive range of administrative functions. Subsequent figures from 4.56 to 4.78 display the visual interfaces available to the admin for managing different aspects of the department. Each fig-

ure features a table that contains relevant data for managing levels, options, sections, groups, students, professors, subjects, rooms, users, bachelor's graduation projects, and master's graduation projects, accompanied by manipulation options for adding, editing, and deleting this information. These interfaces are tailored to enable efficient and streamlined management across all department functions.

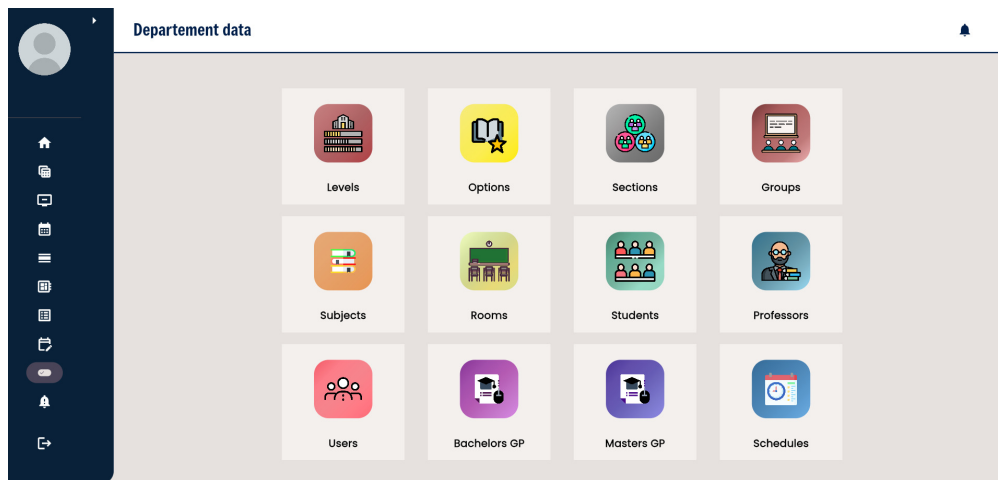


Figure 4.55. Manipulation department data main menu

The figure 4.56 displays the 'Level' table, which includes detailed information such as the name of each level, as well as counts of students, groups, and options associated with each level.

Levels				
Level Name	Options Count	Groups Count	Students Count	Action
L1	1	12	30	⋮
L2	1	8	30	⋮
L3	1	7	30	⋮
M1	5	5	1	⋮
M2	5	5	102	⋮

Figure 4.56. Level manipulation

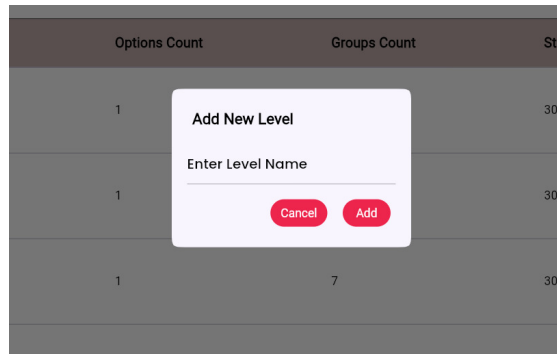


Figure 4.57. Add level

The table presented in figure 4.58 provides a full overview of the options that are available. It includes information on the name of the options and the associated levels.

Options			
Option Id	Option Name	Level	Action
1	SI	L1, L2, L3	⋮
2	IA	M1, M2	⋮
3	SIOD	M1, M2	⋮
4	RTIC	M1, M2	⋮
5	IVA	M1, M2	⋮

Figure 4.58. Option manipulation

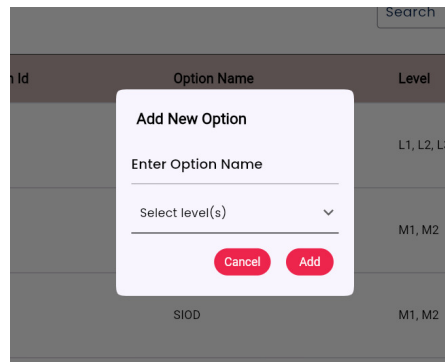


Figure 4.59. Add option

As seen in figure 4.60, this table provides department sections, listing each by name alongside corresponding level and option.

Sections					
Search <input type="text" value="Search"/> + Add					
Section ID	Section Name	Level	Option	Action	
1	Sec 1	L1	SI	⋮	
2	Sec 2	L1	SI	⋮	
3	Sec 1	L2	SI	⋮	
4	Sec 1	L3	SI	⋮	

Figure 4.60. Section manipulation

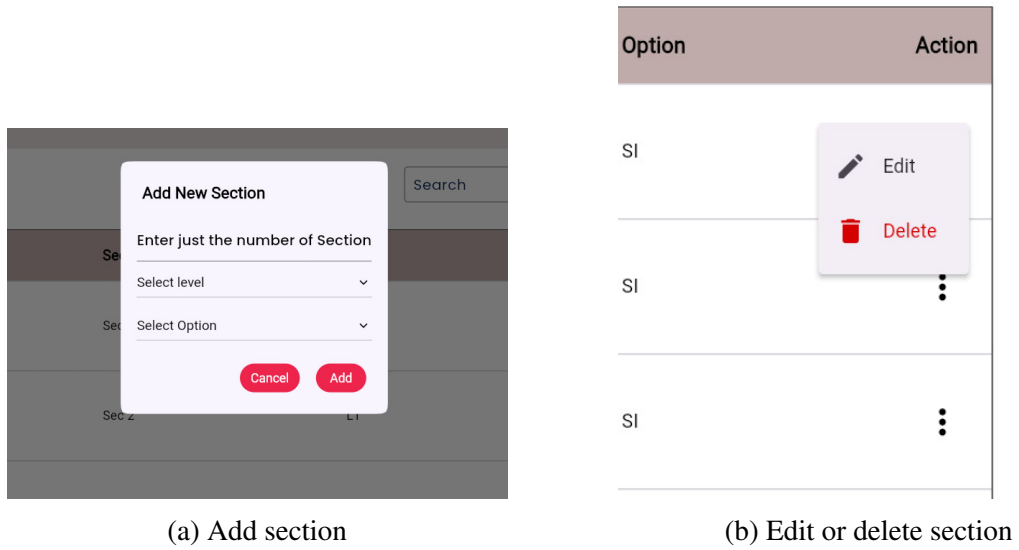


Figure 4.61. Edit, delete or add section

The 'Groups' table, as illustrated in figure 4.62, arranges group names along with their corresponding level, option, and section.

Groups					
Search + Add					
Group ID	Group Name	Level	Option	Section	Action
1	G1	L1	SI	Sec 1	⋮
2	G2	L1	SI	Sec 1	⋮
3	G3	L1	SI	Sec 1	⋮
4	G4	L1	SI	Sec 1	⋮

Figure 4.62. Groups manipulation

Figure 4.63. Add group

This table, which is shown in figure 4.64, contains information on each individual student record, including name, matriculation details, and associations with particular groups, options, and levels. This information helps to provide a comprehensive academic and demographic picture of the student body.

Students

+ Add

registration number	Subject Name	Level	Option	Section	Group	Email	Birthday	Action
UN07012023232335273002	ATMANE NAOUFEL MOUATEZ	L1	SI	1	G1	null	null	⋮
UN07012023232335231908	AZOUZ DOUAA	L1	SI	1	G1	null	null	⋮
UN07012023232335276502	AZZOUZ KAMEL EDDINE	L1	SI	1	G1	null	null	⋮
UN07012023232335305806	BENYAHIA MOHAMED WASSIM	L1	SI	1	G1	null	null	⋮

Figure 4.64. Students manipulation

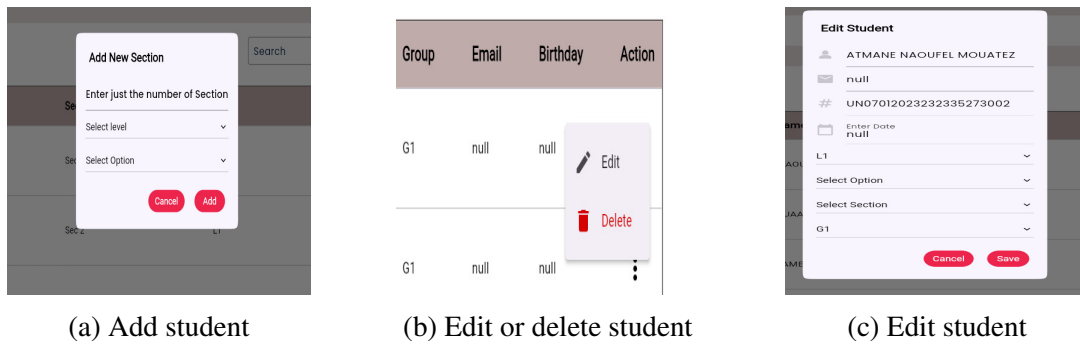


Figure 4.65. Edit, delete or add student

The 'Professors' table in figure 4.66 presents professors names along with the subjects they specialize in and their emails.

Professors				
Profesor ID	Professor Name	Subject	Email	Action
54	Ababsa Tarek	Réseaux, Sec	null	⋮
1	Abba Abdelmadjid	Algèbre 2	nadasakercss@gmail.com	⋮
32	Abba Khadidja	Proba-Stat	null	⋮
50	Abdelli Belgacem	cloudBigD, POO	null	⋮
40	Abbas Diabari	Algo 2, TI	null	⋮

Figure 4.66. Professors manipulation

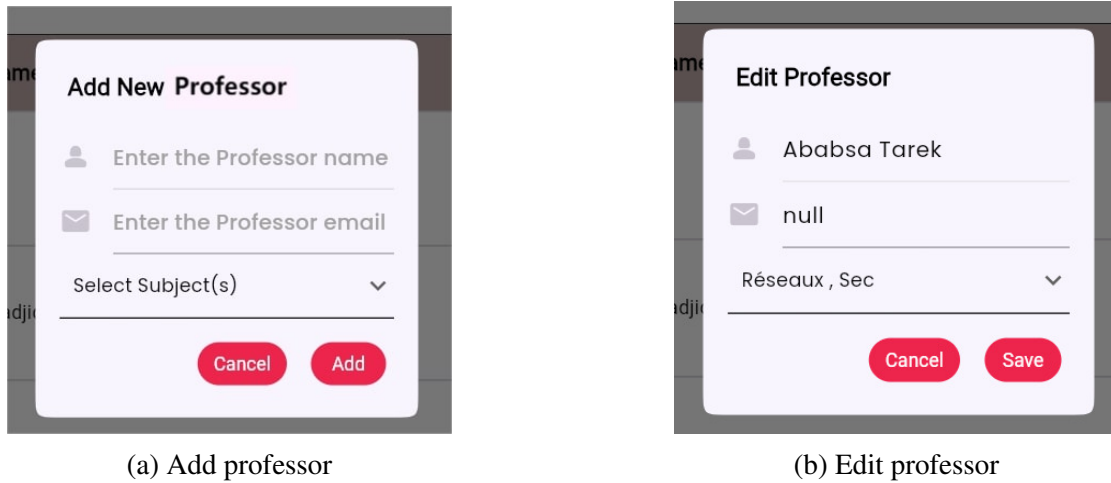


Figure 4.67. Edit, delete or add professor

Figure 4.68 presents a table that outlines department subjects, along with their name, type, associated level, option and the professor in charge.

Subjects						
Subject ID	Subject Name	Subject Type	Level	Option	Professor in charge	Action
2	Algo 2	TP, TD, LECTURE	L1	SI	Bennoui Hammadi	⋮
4	Outil Prog	TP, LECTURE	L1	SI	Berghida Meryem	⋮
10	DAW	TP, LECTURE	L2	SI	Aloui Ahmed	⋮
11	BD	TP, TD, LECTURE	L2	SI	Belouaar Hocine	⋮
12	...	TP, LECTURE	L2	SI	...	⋮

Figure 4.68. Subjects manipulation

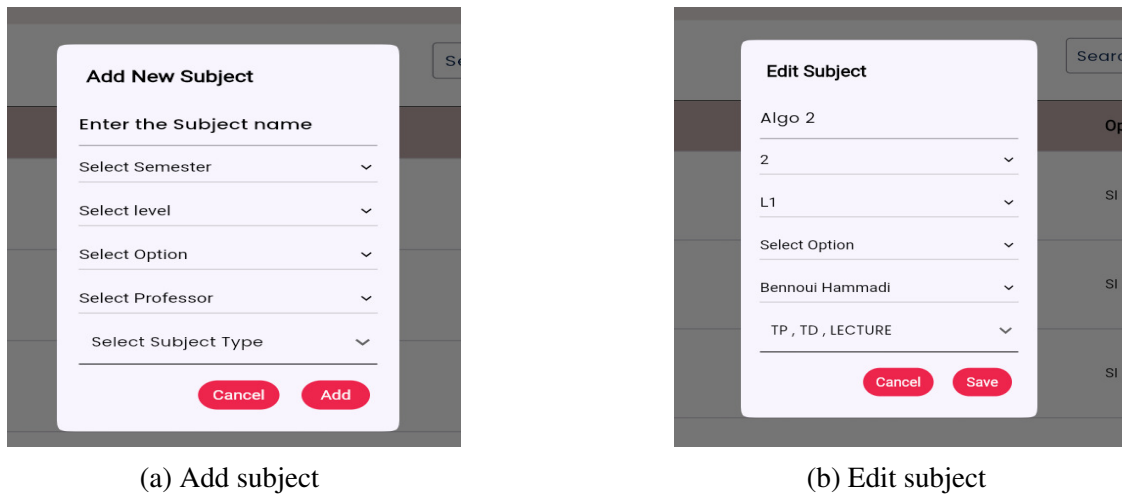


Figure 4.69. Edit, delete or add professor

”Rooms” table, which is shown in figure 4.70, lists the department resources (rooms), including their names and types (Lecture, tutorial or Lab work).

Rooms			
Room ID	Room Name	Room Type	Action
1	B1	TD	⋮
2	B2	TD	⋮
3	B3	TD	⋮
4	B4	TD	⋮

Figure 4.70. Rooms manipulation

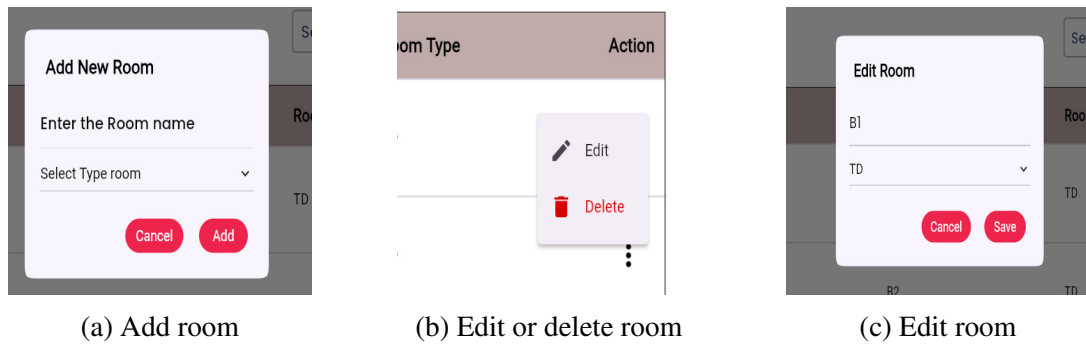


Figure 4.71. Edit, delete or add room

Figure 4.72 displays the 'Users' table, which classifies all system users according to their roles (student, teacher, admin), and associates them with their username and password.

Users				
Name	Username	Password	User Role	Action
ATMANE NAOUFEL MOUATEZ	74440304	37006367	student	⋮
Tigane Samir	38033089	41353302	teacher	⋮
Babahenini Med Chaouki	49173454	49034190	admin	⋮
Bouguettiche Amina	64743513	89635464	teacher	⋮

Figure 4.72. Users manipulation

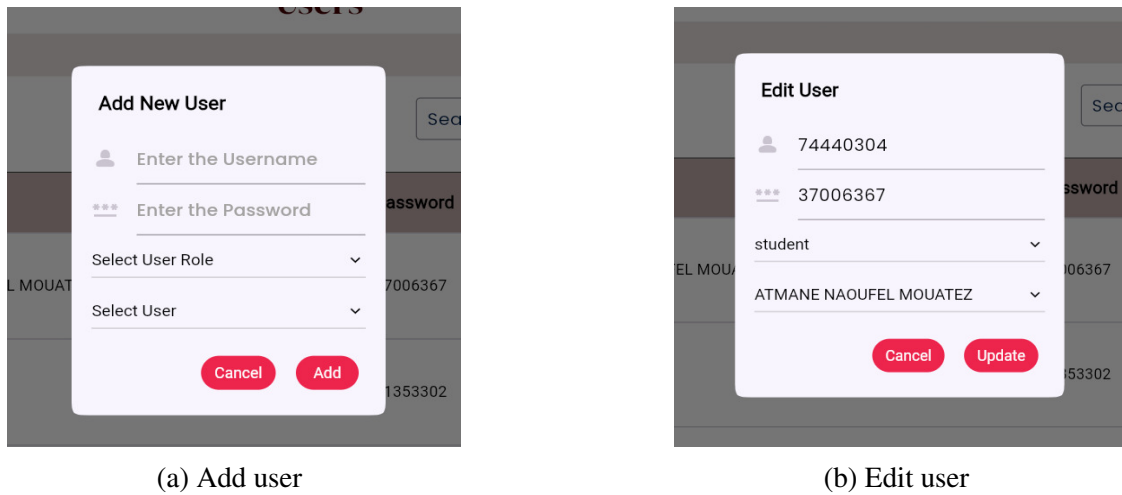


Figure 4.73. Edit, delete or add user

Two tabs are shown in the display in Figure 4.74: one for projects and another for assignments (figure 4.76), primarily related to bachelor's degrees. Each project's title, supervising professors, and project's description are listed under the 'Projects' tab. The affected assignment project to each group or person are listed under the 'Assignments' tab.

Graduation Projects		Graduation Assignments		
<input type="text" value="Search"/> + Add Delete 				
Project ID	Project Name	Description	Professor	Action
1	Machine Learning Optimizations		Bitam Salim	⋮
2	Big Data Analysis Techniques		Boulakhras Gherbal	⋮
3	Advanced Networking Solutions		Abdelli Belgacem	⋮
4	Cloud Computing Advancements		Djerou Leila	⋮

Figure 4.74. Bachelors final year project manipulation

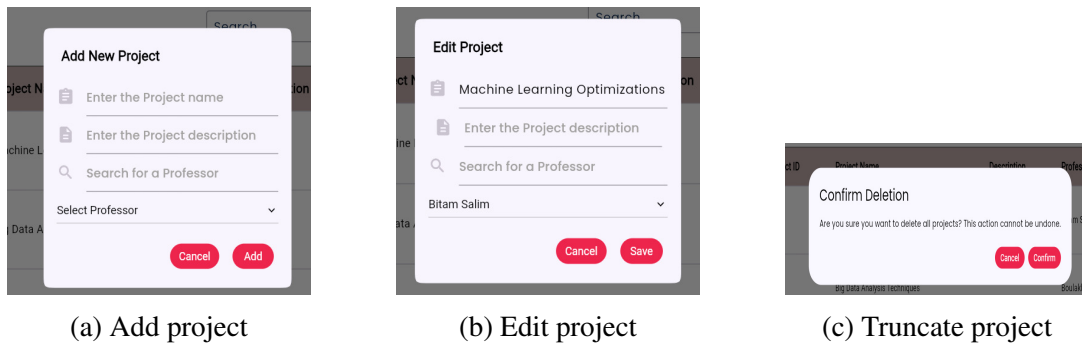


Figure 4.75. Edit, truncate or add project

*

Graduation Projects		Graduation Assignments							
ID	Name Students	Group's Average	Choice 1	Choice 2	Choice 3	Choice 4	Choice 5	Affected Choice	Action
1	ABDELOUHAB Dounia ASSASSI Brahim ASSASSI Salah-eddine	11.78	38	32	36	6	1	32 - Biometric Authentication Technology	⋮
2	BENAOUI Assia BENBRAIKHA Mohamed zakaria BENBRAIKHA Isam BOUABID Abdessattar	14.50	17	1	39	3	7	17 - Web Application Frameworks	⋮
3	BOUAICHA Djihane ELGUENI Mohamed seddik GASMI Mahdi taki eddine	12.88	4	33	19	9	11	33 - Smart Waste Management System	⋮
4	GHANEMI Oussama GHERBI A Mervem	14.08	28	36	8	9	9	28 - Cryptocurrency Analysis	⋮
5	GHERD Issam HADEEF Mohamed badis HAFAYED Abdelaziz	13.84	38	4	40	39	25	38 - Voice-Controlled Desktop Assistant	⋮
6	HEUS Islam HENNI Abdelkader	13.27	6	9	2	37	1	6 - Cybersecurity Threat Analysis	⋮

Figure 4.76. Bachelors final year project assignment manipulation

The interface, which focuses on master’s degrees, is separated into two tabs as shown in Figure 4.77: one for projects and another for assignments (figure 4.78). The ‘Projects’ tab offers a listing project associated by their name, description and the professor in charge. The ”Assignment” tab display the affected assignment project of groups .

Graduation Projects		Graduation Assignments			
<input type="text" value="Search"/> <input type="button" value="+ Add"/> <input type="button" value="Delete"/> 					
Project ID	Project Name	Description	Type	Professor	Action
1	Machine Learning Optimizations		NORMAL	Bitam Salim	⋮
2	Big Data Analysis Techniques		NORMAL	Boulakhras Gherbal	⋮
3	Advanced Networking Solutions		NORMAL	Abdelli Belgacem	⋮
4	Cloud Computing Advancements		NORMAL	Djerou Leila	⋮

Figure 4.77. Masters final year project manipulation

Graduation Projects		Graduation Assignments							
<input type="text" value="Search"/> <input type="button" value="+ Add"/> <input type="button" value="Delete"/> 									
ID	Name Students	Group's Average	Choice 1	Choice 2	Choice 3	Choice 4	Choice 5	Affected Choice	Action
10	GUENFOUD Taha TEBRI Abdelhakim BOUDIBI Abdelhaq	12.32	1	5	40	32	9	1 - Machine Learning Optimizations	⋮
14	BELAALA Hana	11.79	2	5	7	4	22	2 - Big Data Analysis Techniques	⋮
16	DJEFAPLA Hachem MEGHAZI LABAFI Aya	12.01	11	33	14	2	6	11 - Virtual Reality Development	⋮

Figure 4.78. Masters final year project assignment manipulation

5. Student Interfaces

The student can consult his calendar, his exams calendar, and receive notifications after any updates may apply to his schedule.

- **Calendar:** The figure 4.79 represent the student calendar interface of his personal semester schedule.

Day	Details	08:00 - 09:30	09:40 - 11:10	11:20 - 12:50	13:10 - 14:40	14:50 - 16:20	16:30 - 18:00
Sunday	Session type Subject Professor Room	TD Physique 2 Ouamane Samia B5	TP Outill Prog Djerou Leila SM8		TP Algo 2 Belouar Hocine SM3	LECTURE Proba-Stat Meraghni Djamel Amphi 2	
Monday	Session type Subject Professor Room		LECTURE Algèbre 2 Abba Abdelmadjid Amphi A	LECTURE Analyse 2 Bouhtrara Saliha Amphi H		TD Structure M 2 Bourekache Samir B11	
Tuesday	Session type Subject Professor Room			TD Analyse 2 Boulakhras Gherbal B10	LECTURE Outill Prog Berghida Merjem Amphi 2	LECTURE Algo 2 Bennoui Hammadi Amphi 1	
Wednesday	Session type Subject Professor Room	LECTURE TIC Rezeg Khaled Amphi H	TD Algèbre 2 Chaouech Khaouane Nassima B10		LECTURE Physique 2 Baazouzi Mourad Amphi 2	TD Algo 2 Fakraoui Farah B12	
Thursday	Session type Subject Professor	TD Proba-Stat Soukeur Abdessalam		TP Outill Prog Merizig Abdelhak		LECTURE Structure M 2 Bourekache Samir	

Figure 4.79. Student calendar

- **Exams Calendar:** In exams session, the student can consult their own exams schedule using this interface as shown in figure 4.80.

Day	Details	08:00 - 09:30	10:30 - 12:00	12:30 - 14:00	14:30 - 16:00
Saturday 2024-06-15	Subject Proctors Room		Algèbre 2 Bouziane Nadjat Amphi 1		
Sunday 2024-06-16	Subject Proctors Room		Analyse 2 Hamdi Soumia Amphi 2		
Monday 2024-06-17	Subject Proctors Room				Physique 2 Ouamane Samia Amphi A
Tuesday	Subject Proctors Room				
Wednesday	Subject Proctors Room				
Thursday 2024-06-13	Subject Proctors Room		Algo 2 Tigane Samir Amphi 2		

Figure 4.80. Student exams calendar

- **Notifications:** The figure 4.81 presents how the notifications appeared to the student.

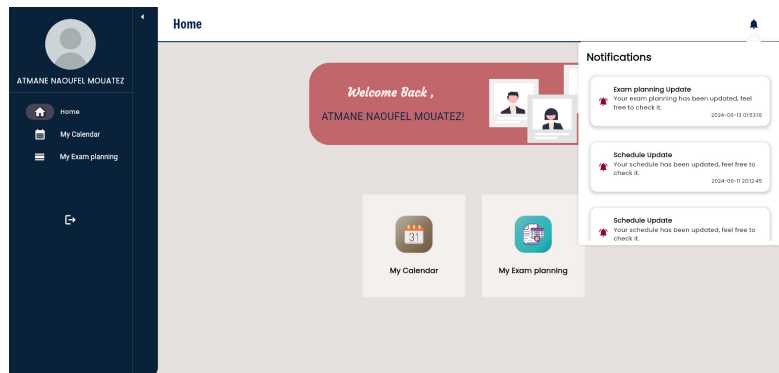


Figure 4.81. student notifications

4.4.2 Mobile application

In this section, we will present the interfaces of the mobile application. Since we have already explained all the functionalities in the web part, here we will focus on the changes and differences in the mobile app interfaces design. Additionally, we did not include the admin in the mobile application since some of his functionalities are complex and require high computational power, which is better suited for the web application.

1. **Login** The next figure 4.82 represents the shared login page of the mobile application, which contains username and password fields. If the username or password is incorrect, an alert message will appear to the user. If not, the user will login successfully to his personal page.

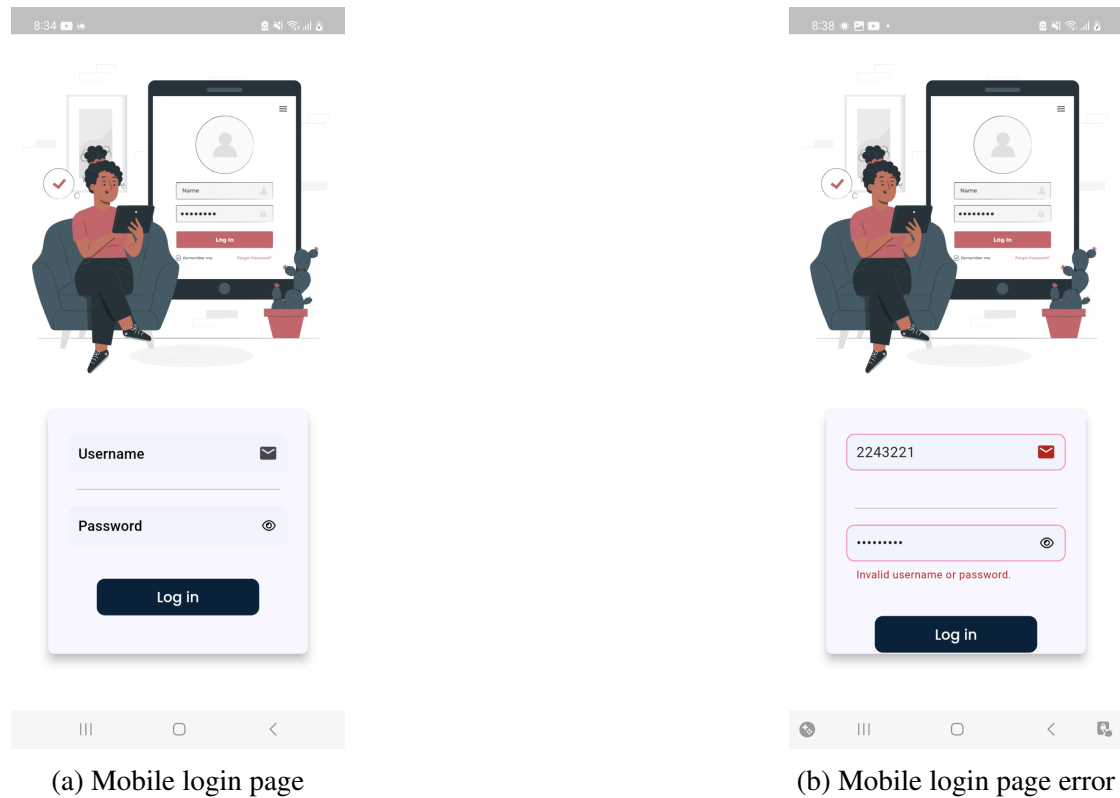


Figure 4.82. Mobile login

2. Professor Interfaces

This section describes the several features that our system offers to professors. This consists of the opportunity to request different academic sessions, schedules, proctor calendars, and a thorough view of personal profiles.

Professor profile: As the figure 4.83 illustrates, this interface displays the professor profile, which contain his personal information such as email address and name.

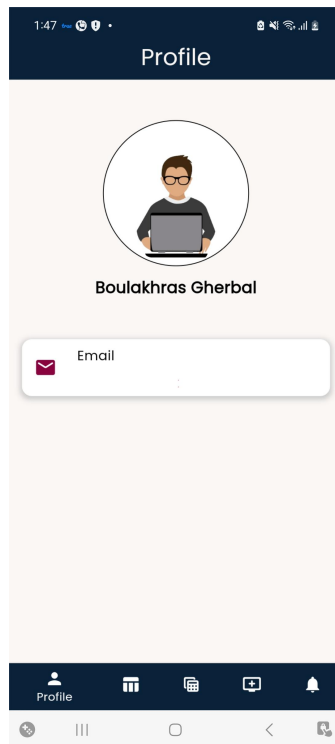


Figure 4.83. Professor profile

Professor personal schedule: The professor can consult his own schedule using this interface by navigating through the days to delay each day's schedule, as shown in the figure 4.84.

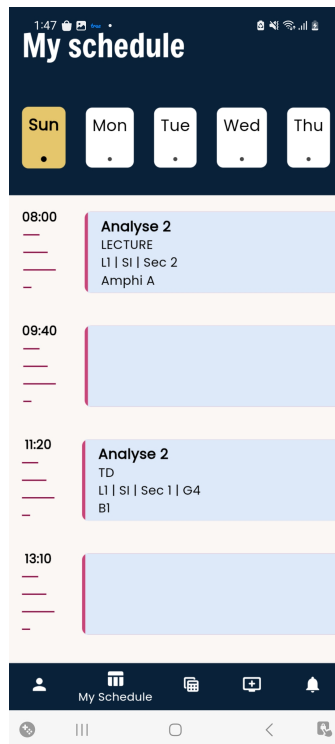


Figure 4.84. Professor schedule

Professor proctoring calendar: the professor uses this interface to consult his proctoring schedule as presented in the following figure 4.85.

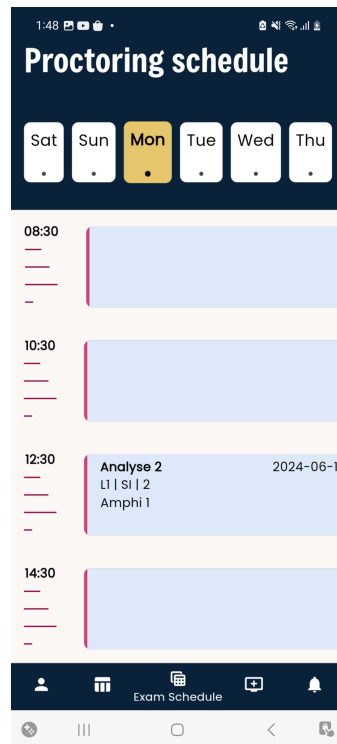
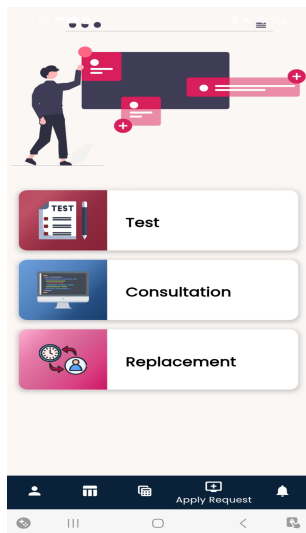
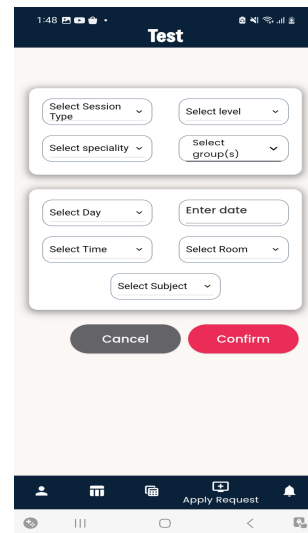


Figure 4.85. Professor proctoring calendar

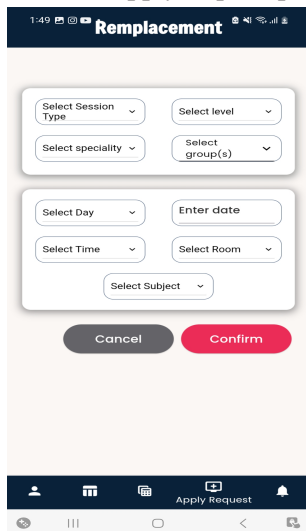
Apply request: Same process as the web application, the professor can schedule a test, replacement and consultation sessions through the phone and that by filling out the session details. The figure 4.86 illustrates the changes in mobile interfaces design.



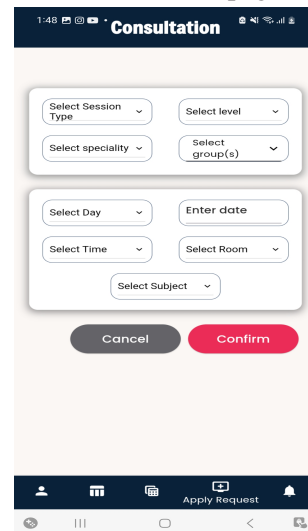
(a) Mobile apply request page



(b) Mobile test page



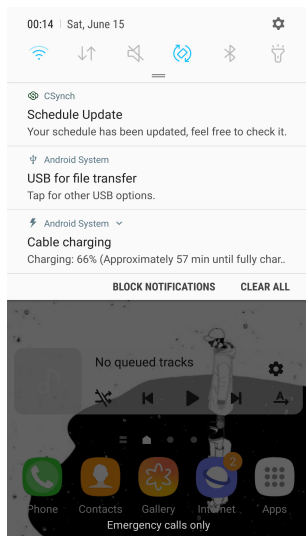
(c) Mobile replacement page



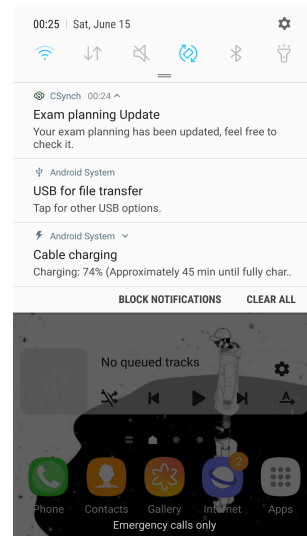
(d) Mobile consultation page

Figure 4.86. Mobile Apply request option

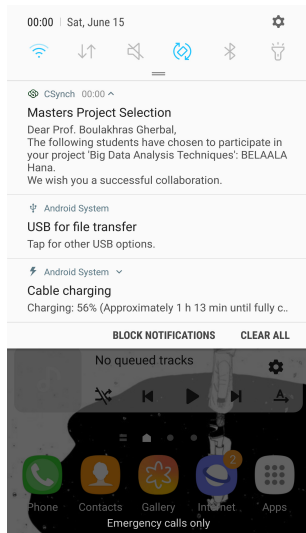
Professor Notifications: If any updates happened, whether schedule updates, exam schedule updates, or the graduation project was assigned, the professor was notified. The figure 4.87 represents the potential notifications that the professor can receive.



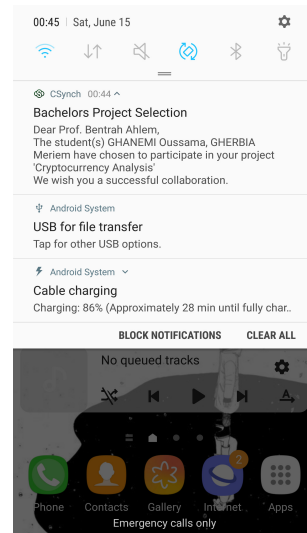
(a) Schedule updates notification



(b) Proctoring schedule update notification



(c) Masters assignment notification



(d) Bachelors assignment notification

Figure 4.87. Professor notification

3. Student interfaces

This section offers students a comprehensive range of resources to help them manage their academic journey, including personal profiles, semester and exam schedules, and notifications regarding assignments and schedule changes.

Student profile: As the figure 4.88 illustrates, this interface displays the student profile, which contain his level, option, group, section and his email address.

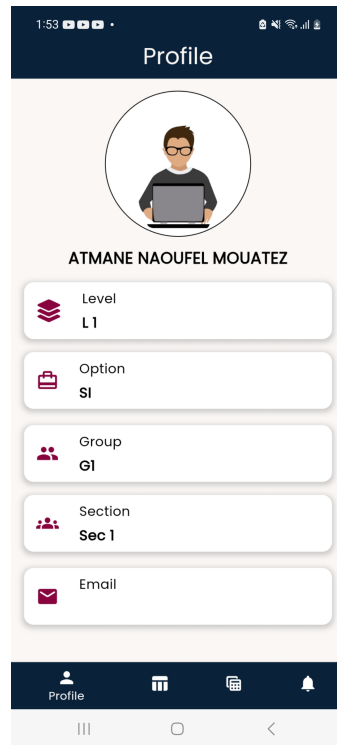


Figure 4.88. Student profile

Student calendar: Same as the professor, The student can consult his schedule using this interface by navigating through the days to delay each day's schedule, as shown in the figure. 4.89.

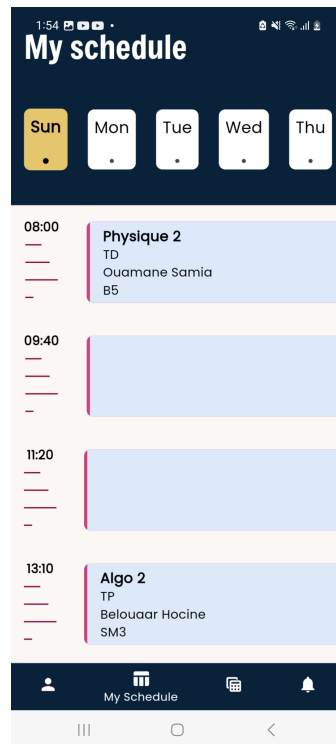


Figure 4.89. Student schedule

Student exam calendar: the student use this interface to consult his exam schedule as presented in the figure 4.90.

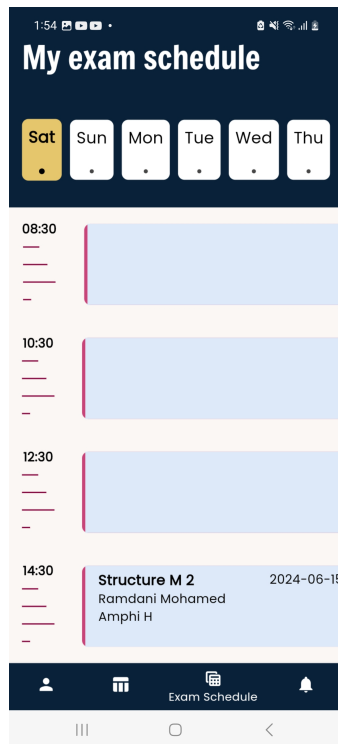
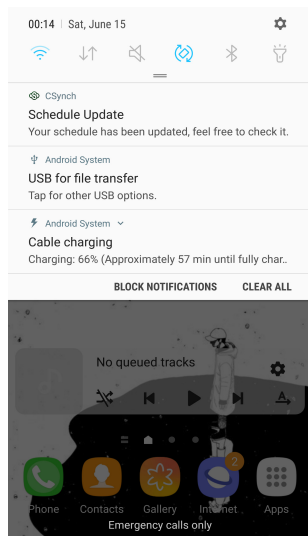
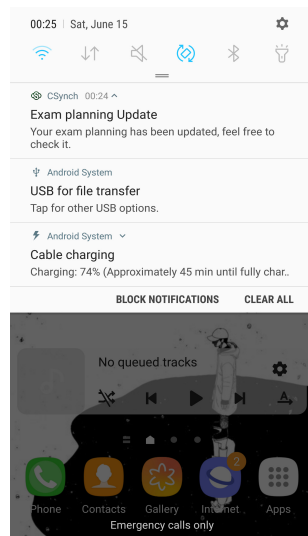


Figure 4.90. Student exam calendar

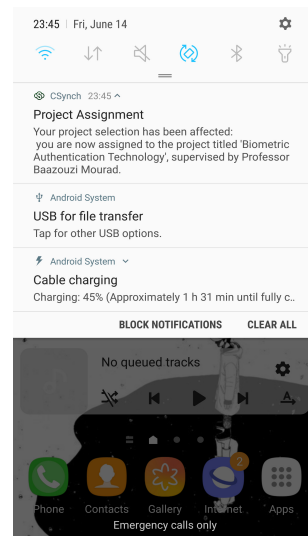
Student notifications: If any updates happened, whether schedule updates, exam schedule updates, the graduation project was assigned or sessions are add, the student will be notified. The figure 4.91 represent the potential notifications that the students can receive.



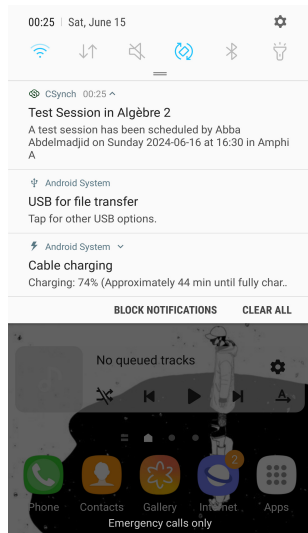
(a) Schedule updates notification



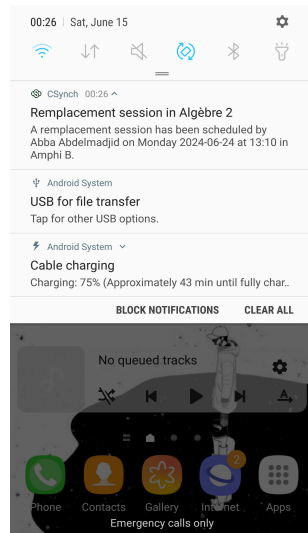
(b) Exam schedule update notification



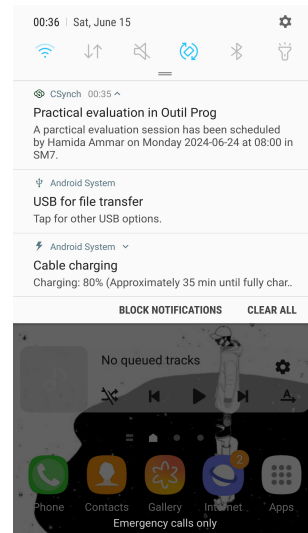
(c) Project assignment notification



(d) Scheduling test notification



(e) Scheduling replacement notification



(f) Scheduling consultation notification

Figure 4.91. Student notification

4.5 Results of the genetic algorithm scheduling

This section presents the technical details and results related to the implementation of the genetic algorithm (GA) used in the proposed scheduling system for the department.

4.5.1 Genetic algorithm configuration

The tables 4.2 and 4.3 details the parameters used in the courses and exams scheduling algorithms, respectively. Each parameter is chosen based on its crucial role in optimizing the algorithm's efficiency and effectiveness. The settings reflect a strategy designed to maximize solution quality while managing computational complexity.

Table 4.2. Parameter settings for courses scheduling algorithm

Parameter	Value
Population Size	40
Crossover Rate	0.9
Mutation Rate	0.1
Generations	500

Table 4.3. Parameter settings for exams planning algorithm

Parameter	Value
Population Size	50
Crossover Rate	0.3
Mutation Rate	0.1
Generations	300

- **Optimal execution time:** With the previously mentioned configurations, the exam planning algorithm takes around 3 minutes to finish, while the courses scheduling algorithm takes approximately 5 minutes to execute. These durations reflect the optimal execution times attainable, successfully maintaining a balance between rapid results and optimal scheduling outcomes for both courses and exams.
- **Configuration selection:** The chosen configurations were determined after testing various settings. This process involved evaluating different combinations of population sizes, crossover rates, and mutation rates to identify the most effective configurations for our needs.
- **Population size and Computational Efficiency:** Increasing the population size was initially considered to potentially enhance the quality of results. However, it was found that larger population sizes did not improve outcome quality but significantly

increased computational demands. This led to the decision to operate with a smaller population size with a relatively larger number of generations, which proved more effective in achieving the desired results without undue computational expense.

4.5.2 Obtained results

This section covers the results derived from applying the NSGA-II algorithm. Clearly state the numerical results and the graphical displays of the algorithm's performance for each of the different objectives.

4.5.2.1 Courses scheduling algorithm

The results of the courses scheduling algorithm are represented through various tables and plots, each corresponding to different objective.

The table 4.4 represents the results of minimizing courses scheduling conflicts over generations.

Table 4.4. Results for Minimizing courses scheduling conflicts.

	Total Number of conflicts	Conflicts re-solved	Percentage achieved
Values	258	258	100%

Where:

- Total number of conflicts: The initial count of courses scheduling conflicts identified.
- Conflicts resolved: Number of conflicts successfully resolved or minimized through the scheduling process.
- Percentage achieved: Shows the success rate in reducing conflicts by a percentage.

The plot in the figure 4.92 shows the minimization of conflicts over generations.

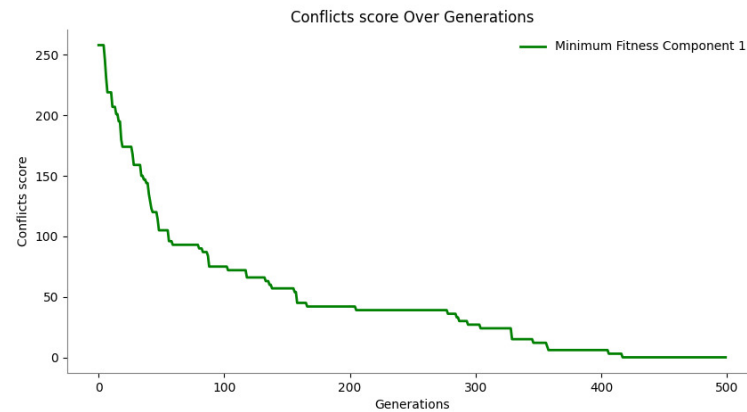


Figure 4.92. Courses conflict reduction across generations

The table 4.5 represents the results for the of maximizing professors sessions consolidation score.

Table 4.5. Results for Professors sessions consolidations

	Total Number of Profs	Profs Scheduled for Max 3 Days	Percentage Achieved
Values	77	53	68.83%

Where:

- Total number of professors: Indicates the total number of professors involved in the scheduling process.
- Professors scheduled for Max 3 Days: Number of professors who were scheduled for a maximum of three days, aiming to consolidate their sessions for efficiency.
- Percentage achieved: Reflects the success rate of achieving the scheduling goal for professors as a percentage of the total.

Figure 4.93 shows the trend of maximizing the professor session consolidation score across various generations.

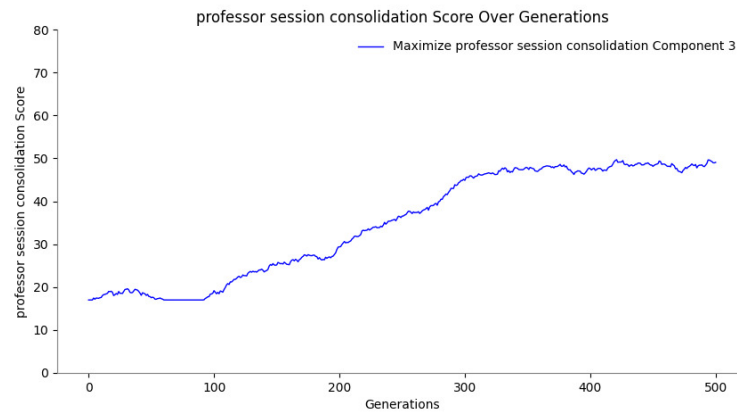


Figure 4.93. professor session consolidation Score maximization across generations

The table 4.6 presents the results for minimizing the subject order penalty in the scheduling algorithm.

Table 4.6. Results for Subject Sessions Order

	Total Number of Subject-Group Pairs	Subject-Group Pairs Not Respecting Order	Percentage Achieved
Values	218	62	28.44%

Where:

- Total Number of Subject-Group Pairs: Indicates the total count of pairings between subjects and groups.
- Subject-Group Pairs Not Respecting Order: The number of pairs that did not follow the prescribed scheduling order.
- Percentage Achieved: Reflects the percentage of subject-group pairs that complied with the intended order.

The figure 4.94 below illustrates the distribution and trends of subject-group pair order compliance over various generations.

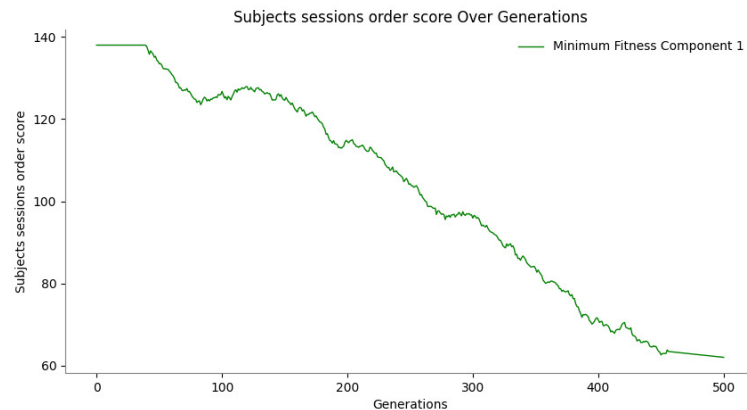


Figure 4.94. Subject sessions order score reduction across generations.

The table 4.7 illustrates the results for maximizing the number of sessions allocated to professors within the scheduling system.

Table 4.7. Results for maximizing required professor sessions number score.

	Total Number of Profs	Profs with Required Sessions number	Percentage Achieved
Values	77	65	84.33%

Where :

- Total Number of Profs: Represents the total number of professors involved in the scheduling process.
- Profs with Required Sessions Number: Indicates how many professors received the planned number of teaching sessions, meeting the maximum requirements which is six sessions total.
- Percentage Achieved: Reflects the percentage of professors who received their required number of sessions.

The figure 4.95 illustrates the improvements in assigning the required number of sessions to professors across different generations.

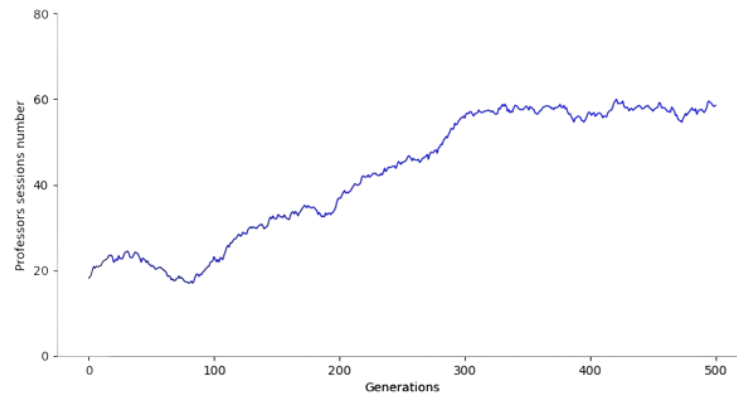


Figure 4.95. professors required number of sessions score across generations

4.5.2.2 Exams planning algorithm

The results of the exams planning algorithm are represented through various tables and plots, each corresponding to different objective.

The table 4.8 represent the results for the objective of minimizing exams scheduling conflicts.

Table 4.8. Results for minimizing conflicts.

	Total Number of conflicts	Conflicts re-solved	Percentage achieved
Values	96	96	100%

Where:

- Total number of conflicts: The initial count of scheduling conflicts identified.
- Conflicts resolved: Number of conflicts successfully resolved or minimized through the exams scheduling algorithm.
- Percentage achieved: Shows the success rate in reducing conflicts by a percentage. The plot in the figure 4.96

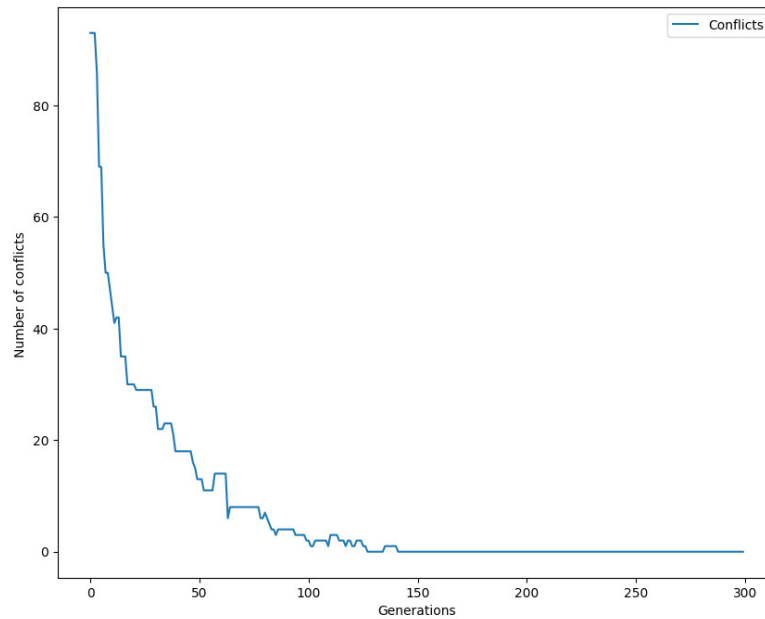


Figure 4.96. Conflicts reduction across generations

The table 4.9 outlines the results for the objective of maximizing the scheduling of one exam per day for each group . Where:

Table 4.9. Results of maximizing single exam per day

	Total Number of Section-Day Pairs	Section-Day Pairs Respecting the Objective	Percentage Achieved
Values	54	39	72.22%

- Total number of section-day pairs: Represents the total combinations of sections and days possible.
- Total number of section-day pair respecting the objective: Shows the count of pairs where only one exam per section per day is scheduled.
- Percentage achieved: Indicates the proportion of section-day pairs that meet the objective, providing a measure of scheduling effectiveness.

The plot in Figure 4.97 shows the progression of the "one exam per day score" across 100 generations.

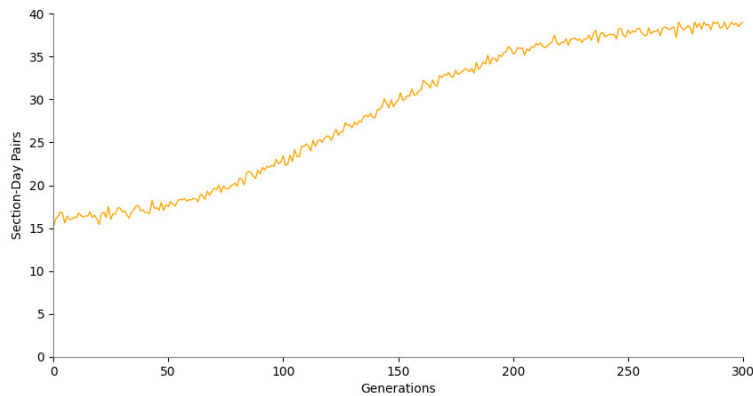


Figure 4.97. One exam per day score across generations

Presented in the table 4.10 are the outcomes related to the objective of minimizing the variance in proctoring distribution among professors.

Table 4.10. Results for Minimizing the Proctoring Distribution Variance

	Initial variance score	Final variance score	Percentage Achieved
Values	5	0.95	81%

Where:

- Initial variance score: Represents the variance score at the beginning of the exams scheduling.
- Final variance score: shows the variance score at the end of the process, demonstrating the outcome to distribute proctoring duties more equitably.
- Percentage achieved: calculates the percentage change from the initial variance score to the final variance score.

The plot in the figure 4.98 illustrates the progressive reduction in proctoring distribution variance across successive generations

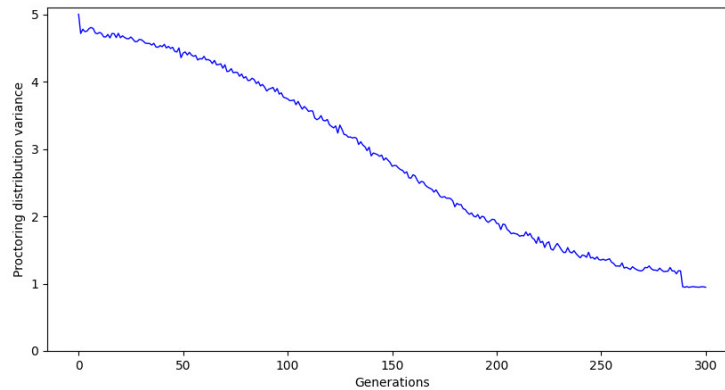


Figure 4.98. Proctoring distribution variance across generations

4.6 Conclusion

This chapter covered the deployment and implementation details of our scheduling system. The system architecture and development tools that enable reliable web and mobile applications are presented. The specific features related to each platform were described, emphasizing user-friendly layout and effective system interactions. In order to address changing educational needs, future improvements will incorporate cutting-edge features.

4.7 Future enhancements

As we strive to continuously evolve and improve our app, we have identified several key areas for development that will enhance user experience and functionality. The possible enhancements are:

- Add graduation scheduling.
- Provide the offline mode that enable access to schedules without an internet connection to ensure continuous accessibility.
- Provide multi-language interface options to enhance usability for all users.

- Virtual classroom integration like Zoom for seamless transitions between scheduling and participating in virtual classrooms.

General conclusion

General conclusion

Throughout this project, we have explored the scheduling problem of pedagogical activities, a complex task that needs to be handled automatically. The manual approach to scheduling has proven to be time-consuming and tiring task, especially as the number of students grows.

To address this challenge, we developed a new software solution that utilizes genetic algorithms to automate and optimize the scheduling of different pedagogical activities, namely courses, tests, exams, and graduation project assignment. Additionally, this solution is deployed across multiple platforms (web and mobile), enhancing usability for students, professors, and administrative staff responsible for planning various tasks.

Our experiments have demonstrated that genetic algorithms is well-suited for handling the stochastic nature of scheduling, capable of managing multiple objectives and constraints. By simulating natural selection, our algorithm generates high-quality schedules that significantly reduce conflicts, improve resource utilization, and satisfy diverse constraints. The iterative nature of genetic algorithms facilitates continuous improvement, ultimately producing schedules that are not only feasible but also optimized according to academic criteria.

The obtained results underscore the significant potential of these algorithms to enhance operational efficiency and user satisfaction. Departments, administrators, and students alike can benefit from this application, which streamlines the scheduling process, reduces administrative burdens, and improves the overall educational experience.

In conclusion, this solution provides a robust foundation for future applications in educational scheduling. Potential enhancements could include more sophisticated multi-criteria optimization, real-time scheduling adjustments, and improved user interface designs. Furthermore, the principles and methodologies developed here could be adapted for other complex scheduling environments beyond academia.

In summary, the successful implementation of this application highlights the transformative potential of genetic algorithms in university scheduling. This approach effectively meets the immediate practical needs of educational institutions by optimizing complex scheduling tasks, thereby enhancing operational efficiency and streamlining processes on a daily basis.

Bibliography

- [1] Wix Support. Wix bookings: Scheduling your classes, 2022. Accessed: 2024-03-14.
- [2] Maurice Comlan. Groups of algorithms in machine learning, 2023. Accessed: 2024-06-25.
- [3] Meriem Bahi and Mohamed Batouche. Deep learning for ligand-based virtual screening in drug discovery. In *Proceedings of the 2018 3rd International Conference on Pattern Analysis and Intelligent Systems (PAIS)*, October 2018.
- [4] Phung and Rhee. A high-accuracy model average ensemble of convolutional neural networks for classification of cloud image patches on small datasets. *Applied Sciences*, 9:4500, 10 2019.
- [5] Vikram Singh. Understanding transformers: A beginner's guide to the basics and applications. Assistant Manager - Content, July 2023. Updated on July 10, 2023 11:00 IST.
- [6] G. Tuncel. *A Heuristic Rule-Based Approach for Dynamic Scheduling of Flexible Manufacturing Systems*, chapter Dynamic Scheduling of Flexible Manufacturing Systems. InTechOpen, 2007.
- [7] Adrian Gepp and Phil Stocks. A review of procedures to evolve quantum algorithms. *School of Information Technology, Bond University*, May 29 2008.
- [8] Borhan Kazimipour, Xiaodong Li, and Kai Qin. A review of population initialization techniques for evolutionary algorithms. In *Proceedings of the 2014 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, July 2014.

-
- [9] Crossover in genetic algorithm, 2023. Last Updated: 10 March, 2023.
- [10] GeeksforGeeks. MI – convergence of genetic algorithms, 2021. Accessed: 2024-06-18.
- [11] Cambridge Dictionary. Schedule. [Online; accessed March 14, 2024].
- [12] Patrick Weaver. Originally presented: myprimavera conference. 4 - 6 April 2006, Hyatt, Canberra, 2006. Director, Mosaic Project Services Pty Ltd.
- [13] R. I. Khabipov. Automation of the creation of the schedule at the university: Mathematical model. *Utopía y Praxis Latinoamericana*, 25(Esp.7):251–256, 2020. Received: 03 August 2020, Accepted: 07 September 2020.
- [14] Mark Wallace. *Constraint Programming*. IC-Parc, Imperial College, LONDON SW7 2AZ, September 1995. Contact address: IC-Parc, William Penney Laboratory, Imperial College, LONDON SW7 2AZ. Email: mgw@doc.ic.ac.uk.
- [15] An introduction to linear programming problems with some real-life applications. *European Journal of Mathematics and Statistics*, April 2022. Article in press.
- [16] Ahmed Wasfy and Fadi A. Aloul. Solving the university class scheduling problem using advanced ilp techniques. 2007.
- [17] Fred Glover. Tabu search-part i. *ORSA Journal on Computing*, 1(3):190, 1989.
- [18] Teotia Tanya Tawhid Abdalrahman and Elmiligi Haytham. Machine learning for optimizing healthcare resources. In *Machine Learning, Big Data, and IoT for Medical Informatics*, chapter Chapter 13, pages 215–239. Academic Press, 2021.
- [19] Mohammadali Geranmehr, Mohammad Reza Nikoo, Ghazi Al-Rawas, Khalifa Al-Jabri, and Amir H. Gandomi. Application of metaheuristic algorithms in optimal design of sewer collection systems. In *Comprehensive Metaheuristics: Algorithms and Applications*, chapter Chapter 8, pages 153–161. Academic Press, City, Country, 2023.
- [20] Guang-Feng Deng and Woo-Tsong Lin. Ant colony optimization-based algorithm for airline crew scheduling problem. *Expert Systems with Applications*, 38(5):5787–5793, 2011.

-
- [21] D. Merkle, M. Middendorf, and H. Schmeck. Ant colony optimization for resource-constrained project scheduling. *IEEE Transactions on Evolutionary Computation*, 6(4):333–346, 2002.
- [22] Yavuz Eren, İbrahim B. Küçükdemiral, and İlker Üstoğlu. Optimization in renewable energy systems recent perspectives. *Chapter 2 - Introduction to Optimization*, pages 27–74, 2017.
- [23] Ikram Hamadache and Mohamed Arezki Mellal. Design optimization of a car side safety system by particle swarm optimization and grey wolf optimizer. In *Nature-Inspired Computing Paradigms in Systems*, pages 15–24. 2021.
- [24] Sk. Imran Hossain, M.A.H. Akhand, M.I.R. Shuvo, N. Siddique, and H. Adeli. Optimization of university course scheduling problem using particle swarm optimization with selective search. *Expert Systems with Applications*, 127:9–24, Aug 2019.
- [25] Achini Kumari Herath. Genetic algorithm for university course timetabling problem. M.sc. thesis, University of Mississippi, Mississippi, USA, 5 2017. Available at eGrove, Electronic Theses and Dissertations, University of Mississippi Graduate School.
- [26] Abdullah Elen. A complete solution to exam scheduling problem: A case study. *Journal of Scientific Reports-A*, 2022. Received Date: 14 October 2021, Accepted Date: 30 March 2022.
- [27] Qiong Liu and Ying Wu. Supervised learning. In *Encyclopedia of Machine Learning*, pages 921–922. Springer, 2012.
- [28] Chonghyo Joo, Hyukwon Kwon, Junghwan Kim, Hyungtae Cho, and Jaewon Lee. Machine-learning-based optimization of operating conditions of naphtha cracking furnace to maximize plant profit. *Computer Aided Chemical Engineering*, 52:1397–1402, 2023.
- [29] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

-
- [30] Manjuan Duan, Ethan Hill, and Michael White. Generating disambiguating paraphrases for structurally ambiguous sentences. In *Proceedings of the Conference*, Columbus, OH 43210, USA, 2023. Department of Linguistics, The Ohio State University.
- [31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008, 2017.
- [32] Florentin Woergoetter and Bernd Porr. Reinforcement learning. *Scholarpedia*, 3:1448, 2008.
- [33] Thomas Bartz-Beielstein, Jürgen Branke, Jörn Mehnen, and Olaf Mersmann. *Overview: Evolutionary Algorithms*, volume 2 of *Schriftenreihe CIplus*. Publisher Name, 2015.
- [34] Aseel Abdalfattah. Genetic algorithms. Computer Engineering Department, An-Najah National University, 2020. November 26, 2020.
- [35] J. H. Holland. *Adaptation in Natural and Artificial Systems*. MIT Press, 1975.
- [36] Chaturangi Shyalika. Population initialization in genetic algorithms: An insight to genetic algorithms — part ii, 1 2019.
- [37] J. Carr. *An Introduction to Genetic Algorithms*. 2014.
- [38] V Sowmyashri. Fitness functions in genetic algorithms: Evaluating solutions. *Medium*, Jan 2024.
- [39] Aravind Seshadri. A fast elitist multiobjective genetic algorithm: Nsga-ii, 2001. Detailed discussion on the implementation and theory of NSGA-II.
- [40] Georg Beyer and Kalyanmoy Deb. On self-adaptive features in real-parameter evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 5(3):250–270, June 2001.

-
- [41] Lucid Software Inc. Uml use case diagram tutorial. <https://www.lucidchart.com/pages/uml-use-case-diagram>, 2023.
- [42] Visual Paradigm International Ltd. What is class diagram? <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-class-diagram/>, 2023.
- [43] Visual Paradigm International Ltd. What is sequence diagram? <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/>, 2023.
- [44] Python Software Foundation. What is python? executive summary. <https://www.python.org/doc/essays/blurb/>, 2023. Accessed: 2024-06-07.
- [45] DEAP Project. *DEAP Documentation*. Accessed: 2024-06-07.
- [46] The Matplotlib development team. *Matplotlib: Visualization with Python*. Accessed: 2024-06-07.
- [47] GeeksforGeeks. Python random module, 2024. Accessed: 2024-06-07.
- [48] GeeksforGeeks. Python pandas tutorial, 2024. Accessed: 2024-06-07.
- [49] PythonBasics. What is flask python?, 2024. Accessed: 2024-06-07.
- [50] MDN Web Docs. Cross-origin resource sharing (cors), 2024. Accessed: 2024-06-07.
- [51] ngrok Documentation. Ngrok platform. <https://ngrok.com/>, 2024. Accessed: 2024-06-07.
- [52] File Picker Package Documentation. File picker. https://pub.dev/packages/file_picker, 2023. Accessed: 2024-06-07.
- [53] THE PHP Group. What is php? <https://www.php.net/manual/en/intro-what-is.php>, 2001-2022. Accessed: 2024-06-07.