الجمهورية الجزائرية الديمقراطية الشعبية

The People's Democratic Republic of Algeria

وزارة التعليم العالي والبحث العلمي

Ministry of Higher Education and Scientific Research

جامعة محمد بوضياف بالمسيلة

University Mohamed Boudiaf of M'sila

كلية الرياضيات والإعلام الآلي | **قسم الإعلام الآلي**

Faculty of Mathematics and Informatics | **Department of Computer Science**

Domain: Mathematics and Computer Science

Thesis Presented to Fulfill the Partial Requirement

for Master's Degree in Computer Science

Specialty:

SIGL

Prepared By:

BIBI FERYAL

SALAH DOAA WISSAL

Supervised By:

DEBBI HICHEM

Entitled

Automating The Construction of Pedagogical Schedules

Jury Members

First Name 1 Last Name 1	**President**
DEBBI HICHEM	**Supervisor**
First Name 3 Last Name 3	**Examiner**

Academic Year 2025/2026

Abstract

Examination surveillance scheduling is an important administrative task in higher education institutions. As examination periods become more complex, manual scheduling requires significant effort and is often prone to errors.

This work proposes an intelligent Exam Surveillance Scheduler that combines Integer Linear Programming (ILP) and Large Language Models (LLMs). The ILP component generates balanced and conflict-free invigilation schedules while satisfying constraints such as teacher availability, workload distribution, and examination coverage. The LLM component provides a natural language interface that enables users to query, modify, and analyze schedules efficiently.

The system was developed and evaluated using real examination data from the Faculty of Mathematics and Computer Science at Mohamed Boudiaf University of M'Sila. The results demonstrate its effectiveness in reducing administrative workload and improving examination surveillance management through intelligent scheduling and user-friendly interaction.

Keywords: Examination Surveillance Scheduling, ILP, LLMs, Artificial Intelligence, Scheduling Optimization.

الملخص

تُعد جدولة حراسة الامتحانات من المهام الإدارية المهمة في مؤسسات التعليم العالي، ومع تزايد تعقيد الفترات الامتحانية أصبحت الطرق اليدوية تتطلب وقتًا وجهدًا كبيرين وتكون عرضة للأخطاء.

ونماذج اللغة (ILP) يقترح هذا العمل نظامًا ذكيًا لجدولة حراسة الامتحانات يجمع بين البرمجة الخطية الصحيحة الكبيرة (LLMs). يتيح النظام إنشاء جداول حراسة متوازنة وخالية من التعارضات مع مراعاة توفر الأساتذة وتوزيع عبء العمل ومتطلبات التغطية، كما يوفر واجهة تفاعلية تعتمد على اللغة الطبيعية للاستعلام عن الجداول وتعديلها وتحليلها.

تم تطوير النظام واختباره باستخدام بيانات حقيقية من كلية الرياضيات والإعلام الآلي بجامعة محمد بوضياف بالمسيلة. وأظهرت النتائج فعاليته في تقليل الجهد الإداري وتحسين إدارة حراسة الامتحانات من خلال الجمع بين تقنيات التحسين الرياضي والذكاء الاصطناعي

الكلمات المفتاحية: جدولة حراسة الامتحانات، البرمجة الخطية الصحيحة، نماذج اللغة الكبيرة، الذكاء الاصطناعي، تحسين الجدولة

Dédicaces

With profound gratitude and sincere appreciation, I dedicate this work to my beloved family, whose unwavering support, love, and encouragement have been the foundation of every achievement in my life.

To my dear father,

Thank you for your guidance, wisdom, and tireless efforts to provide me with the opportunities to pursue my dreams. Your dedication, integrity, and perseverance have always been a model for me, and I am proud to dedicate this work to you.

To my dear mother,

Words cannot express the depth of my gratitude for your endless love, sacrifices, patience, and prayers. Your strength, kindness, and constant belief in me have been a source of inspiration throughout my academic journey. This accomplishment is as much yours as it is mine.

To my beloved sister,

Thank you for your affection, encouragement, and continuous support. Your presence has always brought comfort and motivation during both the challenging and joyful moments of this journey.

To my three dear brothers,

Thank you for your constant encouragement, understanding, and companionship. Your support has given me strength and confidence whenever I needed it most.

This dissertation represents not only the completion of an academic milestone but also the result of the love, sacrifices, and encouragement of those who have stood beside me every step of the way.

And finally, I dedicate this work to myself.

For every challenge faced, every moment of doubt overcome, and every effort invested throughout this journey. For the perseverance, determination, and resilience that made this achievement possible. This work stands as a reminder that dedication and hard work can transform aspirations into accomplishments.

I am proud of how far I have come, and I look forward to the journey that lies ahead.

BIBI FERYAL

Dédicaces

First and foremost, Alhamdulillah.
I dedicate this work to my parents, whose support, sacrifices, and encouragement have made this achievement possible. Thank you for always believing in me and for giving me the opportunity to pursue my goals. I am deeply grateful for everything you have done for me.

To my sisters,

Thank you for your support, laughter, patience, and for making even the most stressful times a little easier.

And finally, I dedicate this work to myself.

This journey was not always easy, and I cannot say I enjoyed every part of it, but I kept going. Through deadlines, challenges, and moments of doubt, I learned the value of perseverance and patience. At times, finishing this degree felt less like an academic journey and more like a test of endurance, but here we are.
This dissertation marks the end of an important chapter and the completion of a goal I worked hard to achieve. For that, I am proud of how far I have come and grateful to everyone who helped me along the way.

Salah doaa wissal

Acknowledgments

First and foremost, I would like to express my sincere gratitude to ALLAH for His guidance throughout my journey, for granting me the strength, patience, time, and good health necessary to complete this work. Praise be to Allah for all His blessings.

I would like to express my deepest gratitude to my supervisor, **Dr. Hichem Debbi**, for his valuable guidance, insightful advice, continuous support, and constant encouragement throughout the development of this project. His expertise and dedication have greatly contributed to the completion of this work.

I would also like to extend my heartfelt appreciation to all my professors, whose knowledge, expertise, and passion for teaching have enriched my academic journey and contributed significantly to my education over the years.

Finally, I would like to thank my family, my friends, and all those who supported, encouraged, and believed in me throughout this work. Their motivation, understanding, and unwavering support have been an invaluable source of strength and inspiration.

To everyone who contributed, directly or indirectly, to the completion of this thesis, I offer my sincere thanks and appreciation.

Table of Contents

Abstract..... i

الملخص..... i

Dédicaces..... ii

Dédicaces..... iii

Acknowledgments..... iv

Table of Contents..... v

List of Figures..... 5

List of Tables..... 6

List of Abbreviations..... 7

General Introduction..... 8

Objective..... 8

Manuscript Structure..... 8

CHAPTER 1: Scheduling Problems..... 10

Introduction..... 10

1. Definition of Scheduling Problems..... 10

2. Historical Evolution of Scheduling Problems..... 11

2.1 Early Methods (Pre-1950s)..... 11

2.2 Mathematical Foundations (1950s–1960s)..... 11

2.3 NP-Hardness Discovery (1970s)..... 12

2.4 Heuristic and AI Methods (1980s–Present)..... 12

3. Components of Scheduling Problems..... 12

3.1 Tasks and Activities..... 12

3.2 Resources..... 13

3.3 Time Slots..... 13

3.4 Constraints..... 13

3.5 Objective Functions....13
4. Types of Scheduling Problems....14
5. Constraints in Exam Surveillance Scheduling....15
5.1 Hard Constraints....15
5.2 Soft Constraints....16
6. Key Entities in Exam Surveillance Scheduling....16
7. Solution Approaches....17
7.1 Exact Methods....17
7.2 Heuristic Methods....18
7.3 Metaheuristic Methods....18
7.4 Artificial Intelligence and Hybrid Scheduling Approaches....18
8. Challenges of Automated Exam Surveillance Scheduling....19
Conclusion....19
CHAPTER 2: Large Language Models and AI-Assisted Interaction....21
Introduction....21
1. Large Language Models (LLMs)....21
1.1 Definition....21
1.2 Evolution of Large Language Models....22
2. Architecture of Large Language Models....23
2.1 Input Embeddings and Positional Encoding....23
2.2 Encoder Self-Attention....24
2.3 Decoder and Masked Self-Attention....24
2.4 Feed-Forward Networks, Residual Connections, and Layer Normalization....25
3. Popular Modern Large Language Models....25
4. How Large Language Models Work....26
5. Training of Large Language Models....27

5.1 Pre-Training.....27
5.2 Fine-Tuning.....27
5.3 Alignment with Human Feedback (RLHF).....27
6. Capabilities and Limitations of LLMs.....28
6.1 Key Applications.....28
6.2 Advantages of Large Language Models.....29
6.3 Limitations and Mitigation Strategies.....29
7. Future Directions of Large Language Models.....30
8. Retrieval-Augmented Generation (RAG).....31
8.1 Definition and Motivation.....31
8.2 RAG Architecture and Workflow.....31
8.3 Comparison of RAG vs. Standard LLM Inference.....32
9. LLMs in Educational Systems and Scheduling.....33
9.1 AI in Education.....33
9.2 LLMs as Assistants in Scheduling Systems.....33
Conclusion.....33
CHAPTER 3: Literature Review — AI and LLMs in Scheduling Systems.....34
Introduction.....34
1. From Manual Scheduling to Intelligent Automation.....34
2. Classical AI and Metaheuristic Approaches to Scheduling.....35
2.1 Genetic Algorithms for Academic Timetabling.....35
2.2 AI vs. Traditional Scheduling: A Systematic Comparison.....36
3. Neural Network Approaches to Scheduling.....36
3.1 Deep Learning for Manpower Scheduling.....37
3.2 AI Techniques in Project and Resource Scheduling: A Systematic Review.....37
4. AI in Educational Scheduling: Schools and Universities.....37

4.1 AI-Powered Timetable Management in Schools and Colleges.....38
4.2 University Timetabling with ILP: Handling Hybrid Teaching.....38
4.3 Decision Support Systems for Instructor Assignment.....38
4.4 Neural Network Scheduling in Educational Contexts.....39
5. Large Language Models in Scheduling Systems.....39
5.1 LLM-Assisted Mathematical Optimisation Modelling.....39
5.2 LLMs as Natural Language Scheduling Interfaces.....39
5.3 LLMs in Manufacturing and Resource Scheduling.....40
6. Synthesis, Comparison, and Positioning of the Proposed System.....40
6.1 Evolution of the Research Landscape.....40
6.2 Recurring Themes and Common Findings.....40
6.3 Positioning of the Proposed System.....41
Conclusion.....41
CHAPTER 4: Implementation, Evaluation and Results Analysis.....42
1. Introduction.....42
2. Dataset and Input Format.....42
2.1 Source of Data.....42
2.2 Data Encoding Convention.....43
2.3 Dataset Characteristics.....43
2.4 Limitations of the Input Data.....43
3. System Implementation.....43
3.1 User Interface Design.....44
3.2 Solver Constraints Panel.....45
3.3 ILP-Based Schedule Generator.....45
3.3.6. Significance of the ILP Approach.....49
3.4 Natural-Language Search Assistant.....50

3.5 Schedule Editor with Conflict Detection.....50
3.6 Export and Absence Tracking Module.....52
3.7 Edit History and Revert.....54
3.8 Mobile Responsiveness and Cross-Device Compatibility.....55
4. Testing Results.....57
4.1 Solver Performance.....57
4.2 Constraint Satisfaction Verification.....58
4.3 Search Accuracy.....58
4.4 Edit Conflict Detection Accuracy.....58
5. Development Environment and Tools.....58
5.1 Programming Language.....58
5.2 Code Editor.....59
5.3 Core Libraries and Frameworks.....59
5.4 External API.....59
6. Role of LLMs in the Proposed System.....60
6.1 Natural-Language Query Interface.....61
6.2 Free-Text Schedule Editing.....61
6.3 Conflict Explanation and Resolution Suggestions.....62
6.4 Workload Analysis and Reporting.....62
6.5 Broader System Contributions.....62
7. Conclusion.....63
General Conclusion.....62
References.....64

List of Figures

Fig. 1.1: Historical evolution of scheduling theory and methods.....11

Fig. 1.2 : Hard and Soft Constraints in Exam Surveillance Scheduling 15

Fig. 2. 1: Transformer Encoder–Decoder Architecture.....23

Fig. 2.2 : General Operational Workflow of a Large Language Model.....27

Fig. 2.3: RAG Workflow.....31

Fig. 4.1: Overall layout of the Exam Surveillance Scheduler interface 44

Fig. 4.2: Solver Constraints panel showing all five configurable parameters.....45

Fig. 4.3: Pre-processing: fixed assignments and responsible-teacher pre-assignment.....46

Fig. 4.4: Generated schedule with green/red assignment status highlighting 49

Fig. 4.5: Search tab with an example natural-language query.....50

Fig. 4.6: Edit tab: Swap Teachers and Quick Edit panels.....51

Fig. 4.7: Conflict detection blocking a double-booked room assignment.....52

Fig. 4.8: AI conflict resolution suggestion after a blocked edit.....52

Fig. 4.9: Per-teacher surveillance schedule generated by the Export feature.....53

Fig. 4.10: Absence tracking: editable past slots and locked future slots.....54

Fig. 4.11: Absent teachers Excel report.....54

Fig. 4.12: Edit History tab with revert functionality.....55

Fig. 4.13: Corrected mobile view with scrollable table and touch checkboxes.....56

Fig. 4.14: Mobile view showing full schedule interaction.....57

Fig. 4.15: LLM integration flow.....60

Fig. 4.16: System Architecture: ILP Optimizer + LLM Assistant (GPT-4o-mini) via OpenRouter API 61

List of Tables

Table 1.1 : Classification of scheduling types and relevance to exam surveillance scheduling.….14

Table 1.2 : Key entities and their roles in exam surveillance scheduling.....16

Table 1.3 *:* Challenges in exam surveillance scheduling 19

Table 2.1 : Key milestones in the evolution of Large Language Models.....22

Table 2.2: Overview of popular modern large language models.....25

Table 2.3: Comparison of RAG vs. standard LLM inference.....32

List of Abbreviations

AI Artificial Intelligence

ILP Integer Linear Programming

LLM Large Language Model

LLMs Large Language Models

NLP Natural Language Processing

RAG Retrieval-Augmented Generation

RLHF Reinforcement Learning from Human Feedback

NP Non-deterministic Polynomial (time)

API Application Programming Interface

GPT Generative Pre-trained Transformer

UI User Interface

DSS Decision Support System

AM/PM Ante Meridiem / Post Meridiem

General Introduction

Scheduling is one of the most important administrative activities in educational institutions. Among its various applications, examination surveillance scheduling is particularly challenging because it requires assigning teachers to examination sessions while satisfying numerous constraints related to availability, workload balance, and conflict prevention.

As universities grow in size and complexity, manual scheduling becomes increasingly time-consuming and prone to errors. Consequently, optimization and Artificial Intelligence techniques have become valuable tools for automating scheduling tasks and improving decision-making processes. In particular, Integer Linear Programming (ILP) has proven effective for generating feasible and efficient schedules while respecting institutional constraints.

Recent advances in Artificial Intelligence, especially Large Language Models (LLMs), have further enhanced the usability of such systems by enabling natural language interaction. Users can query, modify, and analyze schedules through intuitive conversational interfaces without requiring technical expertise.

This work proposes an intelligent examination surveillance scheduling system that combines the optimization capabilities of ILP with the interactive power of LLMs. The system is developed and evaluated using data from the Faculty of Mathematics and Computer Science at Mohamed Boudiaf University of M'Sila, demonstrating how modern AI technologies can support the digital transformation of academic administration and improve scheduling efficiency.

Objective

Our objective is to build an intelligent examination surveillance scheduling system that combines the optimization power of Integer Linear Programming with the conversational capabilities of Large Language Models — so that academic administrators can generate balanced, conflict-free schedules automatically, and manage them through simple, natural language interaction.

.

Manuscript Structure

This dissertation is divided into four main chapters organized as follows:

In the first chapter, we lay the theoretical groundwork by introducing the fundamental concepts of scheduling problems — their definitions, key components, classifications, and the constraints that make them computationally challenging. We also survey the main solution approaches that have been developed over the decades, from exact methods to heuristic and AI-based techniques, in order to understand the landscape into which our proposed system fits.

Chapter two then shifts focus to the second pillar of our work: Large Language Models. We explain how these models are built, how they are trained, and what they are genuinely capable of — as well as where they fall short. We also introduce Retrieval-Augmented Generation as a mechanism that addresses some of those limitations, and we close the chapter by examining how LLMs are beginning to find their place in educational and scheduling systems.

Chapter three brings both pillars together by reviewing the existing research that sits at their intersection. We look at how artificial intelligence has been applied to scheduling problems over time — from classical optimization and metaheuristics to neural networks and, more recently, Large Language Models — and we position our own contribution relative to what has already been done.

Chapter four presents the full realization of the proposed system. We describe the dataset, the architecture, the ILP-based schedule generator, the natural language assistant, and the user interface, before presenting the experimental results and discussing the outcomes and the challenges encountered along the way.

.

CHAPTER 1: Scheduling Problems

Introduction

Scheduling is one of the most fundamental and widely studied problems in combinatorial optimization, operations research, and artificial intelligence. At its core, scheduling involves the assignment of limited resources to tasks over time in a way that satisfies a set of constraints and optimizes one or more objectives. In academic institutions, one of the most critical and practically demanding scheduling challenges is exam surveillance scheduling — the process of assigning teachers to supervise examination rooms during official exam sessions.

Unlike general scheduling problems, exam surveillance scheduling is a highly constrained combinatorial optimization problem. Each active exam room in each time slot must be covered by at least one available teacher, no teacher can supervise two rooms simultaneously, and supervision workloads must be distributed fairly across all available staff. The size and complexity of this problem grows rapidly with the number of exams, rooms, teachers, and time slots, placing it firmly in the class of NP-hard problems.

This chapter establishes the theoretical foundation for understanding this problem. It formally defines scheduling, traces its historical development, presents the components of scheduling systems, and focuses in depth on exam surveillance scheduling — its constraints, solution approaches, mathematical formulation, and practical challenges.

1. Definition of Scheduling Problems

Scheduling problems involve the allocation of tasks, activities, or events to available resources over specific time periods while satisfying a set of constraints and optimizing one or more objectives [1]. It is a fundamental area of operations research and optimization, with applications in manufacturing, transportation, healthcare, computing systems, and educational institutions.

The objective of a scheduling system is generally to improve resource utilization, minimize delays and conflicts, balance workloads, and enhance overall operational efficiency. Depending on the application domain, scheduling problems may involve different types of resources, time constraints, and performance criteria, making them highly diverse and often computationally complex.

Many scheduling problems belong to the class of combinatorial optimization problems, where the number of possible solutions grows rapidly with problem size. Developing efficient scheduling methods therefore remains an active research area that combines optimization techniques, artificial intelligence methods, and decision-support systems.

2. Historical Evolution of Scheduling Problems

The study of scheduling has evolved significantly over more than a century [2]. From the early visual tools of the industrial age to the sophisticated AI-based methods of today, each era has contributed fundamental insights. Figure 1.1 provides a timeline of the major milestones.

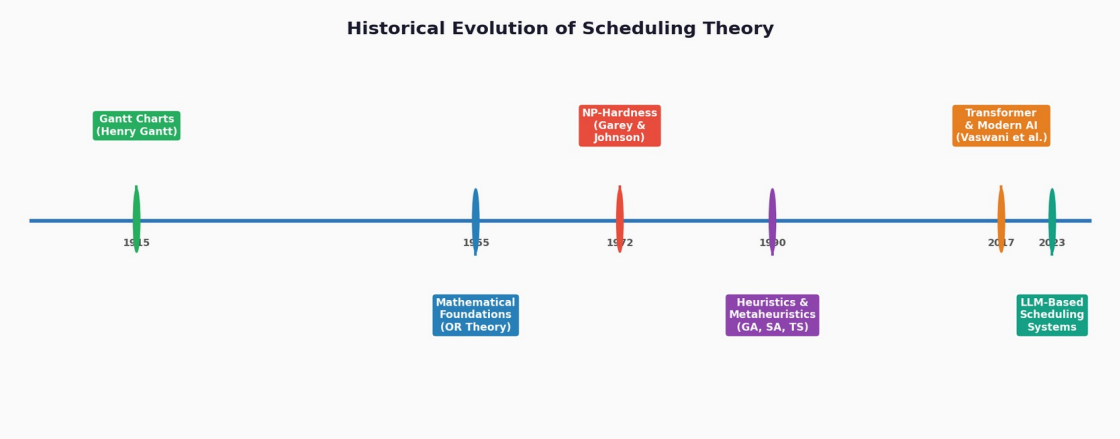


Fig. 1.1: Historical evolution of scheduling theory and methods.

2.1 Early Methods (Pre-1950s)

Scheduling problems first emerged as a practical concern during the Industrial Revolution. The Gantt chart, introduced by Henry Gantt around 1910–1915, was one of the earliest tools for visualizing task assignment over time [3]. While developed for manufacturing, its core principles — assigning tasks to resources across discrete time periods — apply directly to exam supervision scheduling.

2.2 Mathematical Foundations (1950s–1960s)

The mathematical foundations of scheduling theory were established in the 1950s and 1960s within operations research. Researchers formalized scheduling as optimization problems — minimizing makespan, balancing workloads, and maximizing resource utilization — and developed the first exact algorithms. These techniques later became the basis for educational timetabling models.

2.3 NP-Hardness Discovery (1970s)

A major milestone in scheduling research occurred during the 1970s with the development of computational complexity theory [4]. Garey and Johnson (1979) demonstrated that many scheduling and timetabling problems belong to the NP-hard class of combinatorial optimization problems [5]. Consequently, finding optimal solutions becomes increasingly difficult as the problem size grows, often requiring exponential computational effort.

Because of this inherent complexity, exact optimization methods are generally suitable only for small and medium-sized instances. Large-scale scheduling problems frequently rely on heuristic, metaheuristic, and hybrid optimization approaches to obtain high-quality solutions within reasonable computation times.

2.4 Heuristic and AI Methods (1980s–Present)

The increasing complexity of scheduling problems encouraged the development of heuristic and metaheuristic approaches capable of producing high-quality solutions within reasonable computational times. Techniques such as Genetic Algorithms, Simulated Annealing, Tabu Search, and Ant Colony Optimization have been widely adopted to solve large-scale scheduling and timetabling problems where exact optimization becomes computationally expensive.

Recent advances in optimization software and artificial intelligence have further enhanced scheduling systems [6]. Modern approaches increasingly combine mathematical optimization techniques, machine learning methods, and intelligent decision-support tools to improve scalability, adaptability, and solution quality in complex scheduling environments.

3. Components of Scheduling Problems

Every scheduling system comprises five interrelated components. Understanding each is essential to formulating and solving the exam surveillance scheduling problem.

3.1 Tasks and Activities

Tasks represent the activities that must be scheduled within a given planning horizon. Depending on the application domain, tasks may correspond to manufacturing operations, transportation activities, healthcare services, educational events, or examination sessions. Each task is characterized by attributes such as duration, priority, deadlines, and resource requirements. Proper allocation and sequencing of tasks are fundamental to achieving efficient scheduling solutions.

3.2 Resources

Resources are the entities required to perform scheduled tasks. They may include machines, employees, vehicles, classrooms, medical equipment, or computational resources. Since resources are often limited, scheduling systems must allocate them efficiently while avoiding conflicts and maximizing utilization. Resource management plays a critical role in determining the feasibility and quality of scheduling solutions [7].

3.3 Time Slots

Time slots represent the available periods during which tasks can be assigned. Scheduling systems divide the planning horizon into discrete intervals such as hours, days, or weeks depending on the application. The assignment of tasks to appropriate time slots directly influences resource utilization, workload distribution, and overall system efficiency.

3.4 Constraints

Constraints define the conditions that govern the construction of feasible schedules. They restrict the ways tasks, resources, and time periods can be combined. Scheduling constraints are categorized into two types:

- **Hard Constraints: Mandatory requirements that must always be satisfied;**
- **Soft Constraints: Preferences that improve schedule quality but may be violated if necessary.**

The complexity of scheduling problems largely arises from the interaction between multiple overlapping constraints.

3.5 Objective Functions

An objective function specifies the performance criterion that the scheduling process seeks to optimize [8]. Common objectives include minimizing completion time, reducing delays, balancing workloads, minimizing operational costs, and maximizing resource utilization. In modern scheduling systems, objective functions often involve multiple criteria that must be optimized simultaneously, leading to multi-objective scheduling formulations.

4. Types of Scheduling Problems

Scheduling problems are classified by structural characteristics: the number of resources, the ordering of tasks, and the nature of assignments [9]. Table 1.2 summarizes the main types and positions exam surveillance scheduling within this taxonomy.

Table 1.1 : Classification of scheduling types and relevance to exam surveillance scheduling

Type	Description	Complexity	Relevance to Our Problem
Single Machine	All tasks on one resource; sequential.	Polynomial	Not applicable (multiple rooms/teachers)
Parallel Machine	Multiple machines; tasks on any machine.	NP-Hard	Partially applicable — parallel room coverage
Flow Shop	Fixed processing order across machines.	NP-Hard	Not applicable
Job Shop	Each task has its own machine order.	Strongly NP-Hard	Not applicable
Open Shop	Tasks in any order on any machine.	NP-Hard	Partially applicable
Exam Surveillance	Assign teachers to rooms per time slot.	NP-Hard	★ Our problem — multiple resources, many constraints

5. Constraints in Exam Surveillance Scheduling

Constraints are the defining characteristic of the exam surveillance scheduling problem. Figure 1.2 provides a structured overview of both hard and soft constraints specific to invigilator assignment in a university examination context.

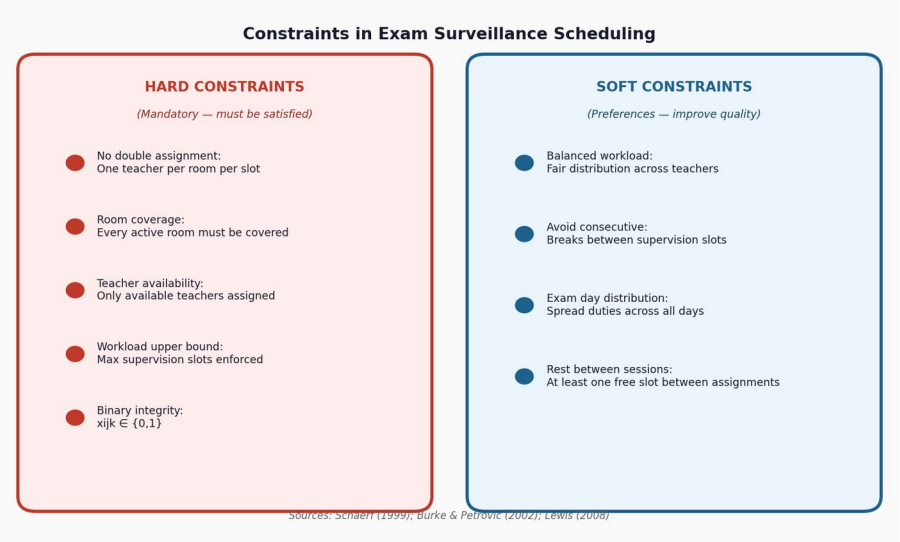


Fig. 1.2 : Hard and Soft Constraints in Exam Surveillance Scheduling .

5.1 Hard Constraints

Hard constraints are mandatory rules whose violation produces an infeasible schedule that cannot be deployed in practice:

- No double assignment: A teacher cannot be assigned to supervise two rooms at the same time slot;
- Room coverage: Every active exam room must have at least the minimum required number of supervisors for each slot;
- Teacher availability: A teacher can only be assigned to a slot during which they are declared available;
- Workload upper bound: A teacher cannot be assigned more supervision slots than their declared maximum load;
- Binary assignment integrity: $x_{ijk} \in \{0,1\}$ — a teacher either supervises a room at a slot, or does not.

5.2 Soft Constraints

Soft constraints represent quality preferences. Their satisfaction improves the timetable but their violation does not render it infeasible:

- Balanced workload: Supervision assignments should be distributed as evenly as possible across all available teachers;
- Avoid consecutive assignments: A teacher should not supervise rooms for more than a defined number of consecutive slots without a break;
- Exam day distribution: Supervision duties should be spread across all days of the session to avoid overloading particular days;
- Rest between sessions: Teachers should have at least one free slot between consecutive supervision assignments where possible.

6. Key Entities in Exam Surveillance Scheduling

Table 1.1 illustrates the four key entities involved in an exam surveillance scheduling system and their main attributes. Understanding these entities and their relationships is essential for formulating the problem and designing an effective solution.

Table 1.2 : Key entities and their roles in exam surveillance scheduling

Entity	Role in Scheduling	Key Attributes	Constraints Involved
Exam Session	Task to be supervised	Module, date, slot, # students	One room per slot; room must be covered
Exam Room	Resource to be staffed	Capacity, location, equipment	Must have ≥1 supervisor per active slot
Teacher / Invigilator	Resource performing supervision	Availability, max load, department	Cannot supervise 2 rooms simultaneously

Entity	Role in Scheduling	Key Attributes	Constraints Involved
Time Slot	Period of assignment	Day, AM/PM, session dates	Must align with exam session schedule

7. Solution Approaches

Given the NP-hard nature of exam surveillance scheduling, a range of solution methods have been developed. The following subsections present the full taxonomy of approaches, from exact mathematical methods to metaheuristic and hybrid AI systems.

7.1 Exact Methods

Exact methods aim to find optimal solutions by exploring the solution space using mathematical optimization techniques [13]. These approaches guarantee solution optimality when computational resources are sufficient. However, their performance often decreases significantly as problem size and complexity increase.

7.1.1 Integer Linear Programming (ILP)

Integer Linear Programming (ILP) is one of the most widely used exact optimization techniques for scheduling and timetabling problems [14]. In ILP models, decision variables are restricted to integer values, while objectives and constraints are represented using linear mathematical expressions. ILP enables the modeling of complex scheduling requirements and can provide optimal solutions for small and medium-sized instances. Recent advances in optimization solvers have significantly improved the applicability of ILP to practical scheduling problems across various domains, including education, transportation, healthcare, and manufacturing.

7.1.2 Constraint Programming (CP)

Constraint Programming (CP) is a problem-solving paradigm based on defining variables, domains, and constraints that describe feasible solutions. Rather than explicitly enumerating all possible assignments, CP uses constraint propagation and intelligent search strategies to reduce the search space efficiently. CP has been successfully applied to highly constrained scheduling

and timetabling problems because of its flexibility in representing complex constraints and its ability to generate feasible solutions efficiently [15].

7.2 Heuristic Methods

Heuristic methods are problem-specific strategies designed to generate acceptable solutions within reasonable computational time [10]. Unlike exact methods, heuristics do not guarantee optimality but are particularly useful for large-scale scheduling problems where exact optimization becomes computationally expensive. Common heuristic techniques include greedy algorithms, constructive procedures, graph-based approaches, and local search methods.

7.3 Metaheuristic Methods

Metaheuristics are advanced optimization frameworks that guide the search process toward high-quality solutions while avoiding premature convergence to local optima [11]. They are widely applied to complex scheduling and timetabling problems because of their flexibility and scalability [16]. Popular metaheuristic techniques include [17]:

- Genetic Algorithms (GA);
- Simulated Annealing (SA);
- Tabu Search (TS);
- Ant Colony Optimization (ACO);
- Particle Swarm Optimization (PSO).

These methods have demonstrated strong performance in solving large combinatorial optimization problems where exact methods become impractical [12].

7.4 Artificial Intelligence and Hybrid Scheduling Approaches

Recent advances in artificial intelligence have introduced new paradigms for solving scheduling problems. Modern scheduling systems increasingly combine optimization algorithms with machine learning models, intelligent decision-support systems, and Large Language Models (LLMs). Hybrid approaches integrate the strengths of mathematical optimization techniques such as ILP with AI-based reasoning and learning capabilities. These systems can assist users in schedule generation, constraint analysis, conflict detection, schedule adaptation, and decision support through natural language interaction.

8. Challenges of Automated Exam Surveillance Scheduling

Despite the availability of sophisticated optimization methods, automated exam surveillance scheduling presents several practical challenges [18]. Table 1.3 summarizes these challenges.

Table 1.3 *:* Challenges in exam surveillance scheduling .

Challenge	Description
Teacher Availability	Availability varies by teacher, slot, and date. Unavailable teachers must be excluded.
Workload Fairness	All teachers expect equitable distribution of supervision duties.
No Double Assignment	Fundamental hard constraint: a teacher cannot be in two places at once.
Dynamic Changes	Teacher absences, room unavailability, or schedule changes occur after planning.
Scalability	Large exam sessions (100+ rooms, 200+ teachers) create large ILP instances.
Explainability	Administrators need to understand and trust the generated assignments.

Conclusion

This chapter presented the theoretical foundations of exam surveillance scheduling as a constrained combinatorial optimization problem. The discussion covered the evolution of scheduling systems, the characteristics of university exam surveillance scheduling, and the importance of constraint satisfaction in generating feasible and fair supervision assignments.

The chapter also introduced Integer Linear Programming (ILP) as the optimization approach adopted in this work, highlighting its suitability for handling assignment constraints, workload balancing, and conflict avoidance in academic scheduling environments.

The next chapter introduces the Large Language Model (LLM) component integrated into the proposed system, explaining how natural language interaction enhances schedule querying, modification, explanation, and analysis while complementing the optimization capabilities of the ILP solver.

CHAPTER 2: Large Language Models and AI-Assisted Interaction

Introduction

The automation of exam surveillance scheduling represents a major challenge for universities due to the large number of constraints related to teacher availability, room allocation, workload balancing, and conflict avoidance. Integer Linear Programming (ILP) provides an effective mathematical approach for generating optimal and valid schedules automatically. However, interacting directly with optimization systems remains difficult for non-technical users.

Recent advances in Large Language Models (LLMs) have introduced new possibilities for improving human-system interaction through natural language understanding and generation. LLMs can assist users in querying, modifying, analyzing, and explaining schedules using intuitive language-based interaction.

This chapter presents the theoretical foundations of Large Language Models and Retrieval-Augmented Generation (RAG), including their architecture, training process, operational principles, applications, and limitations.

1. Large Language Models (LLMs)

1.1 Definition

A Large Language Model (LLM) is a class of deep learning-based artificial intelligence system designed to recognize, summarize, translate, predict, and generate natural language content by learning statistical and contextual patterns from very large text datasets [19]. LLMs are built on advanced neural network architectures — most notably the Transformer — which enable them to model contextual relationships in sequential data and produce coherent, context-aware text across a wide variety of language tasks [20].

Unlike traditional rule-based NLP systems, LLMs learn language representations directly from data, making them highly adaptable to diverse tasks without requiring task-specific engineering [23]. This generality makes them ideal as the language interface layer of domain-specific systems such as the exam surveillance scheduling assistant developed in this work.

1.2 Evolution of Large Language Models

The development of large language models spans more than seven decades. Table 2.1 summarizes the key milestones from the theoretical origins of NLP to the modern era of foundation models.

Table 2.1 : Key milestones in the evolution of Large Language Models

Period	Milestone	Key Contribution
1950s	Turing Test (Alan Turing)	Proposed that machines could simulate intelligence through language; conceptual foundation for NLP.
1980s	Statistical LMs (N-grams)	Estimated word probabilities using context; used in early speech recognition and translation.
2003	Neural Probabilistic LM (Bengio et al.)	Introduced neural networks for language modeling with richer semantic word representations.
2013	Word2Vec (Mikolov et al., Google)	Dense word embeddings capturing semantic analogies (king − man + woman ≈ queen).
2017	Transformer Architecture (Vaswani et al.)	Self-attention enabling parallel processing and superior long-range dependency modeling.
2018	BERT (Devlin et al., Google)	Bidirectional pre-training significantly improved downstream NLP task performance [26].
2018–2020	GPT Series (OpenAI)	GPT-3 (175B parameters) demonstrated strong few-shot learning across diverse tasks [24].
2023+	GPT-4o, Claude 3,	Advanced reasoning, multimodal understanding;

Period	Milestone	Key Contribution
	Gemini, LLaMA 3	efficient mini-models for API deployment.

2. Architecture of Large Language Models

Modern Large Language Models (LLMs) are primarily based on the Transformer architecture introduced by Vaswani et al. (2017) [21]. Unlike earlier sequential architectures such as Recurrent Neural Networks (RNNs) and Long Short-Term Memory networks (LSTMs), Transformers process all tokens simultaneously through attention mechanisms, enabling efficient parallel computation and improved modeling of long-range contextual dependencies.

Figure 2.1 presents the complete Transformer encoder–decoder architecture, including embeddings, positional encoding, self-attention, feed-forward networks, residual connections, and normalization layers.

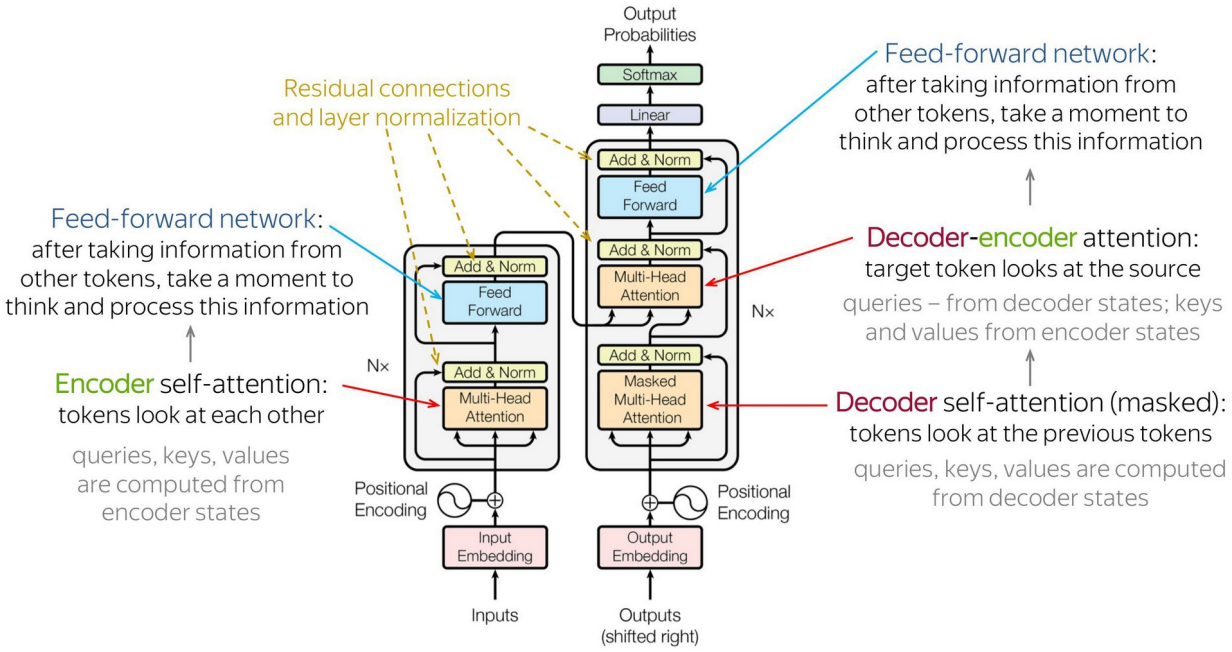


Fig. 2. 1: Transformer Encoder–Decoder Architecture.

2.1 Input Embeddings and Positional Encoding

The Transformer first converts input text into numerical vector representations called embeddings. Each token is mapped into a high-dimensional vector space capturing semantic and syntactic information. Since the Transformer processes tokens in parallel rather than

sequentially, positional encoding is added to embeddings to preserve word-order information within the sequence.

In Figure 2.1, positional encodings are added directly to both input and output embeddings before entering the encoder and decoder stacks.

2.2 Encoder Self-Attention

The encoder is composed of multiple stacked layers containing Multi-Head Self-Attention and Feed-Forward Networks. Self-attention allows each token to examine relationships with all other tokens in the input sequence, regardless of distance.

For every token, three vectors are computed:

- Query (Q);
- Key (K);
- Value (V).

Attention scores are calculated between queries and keys to determine how much focus should be given to other tokens when constructing contextual representations.

In Figure 2.1, the encoder self-attention block enables input tokens to examine each other in order to capture contextual dependencies and semantic relationships.

2.3 Decoder and Masked Self-Attention

The decoder also contains stacked attention layers, but unlike the encoder, it uses masked self-attention. This masking mechanism prevents the decoder from accessing future tokens during generation, ensuring that predictions depend only on previously generated outputs.

The decoder additionally contains encoder–decoder attention, where decoder states attend to encoder outputs. This mechanism enables the model to integrate contextual information from the input sequence while generating output tokens.

As illustrated in Figure 2.1, the masked self-attention block processes previously generated tokens, while the encoder–decoder attention layer connects decoder representations with encoder outputs.

2.4 Feed-Forward Networks, Residual Connections, and Layer Normalization

After each attention operation, the representations pass through Feed-Forward Neural Networks (FFN), which apply nonlinear transformations to learn more complex linguistic patterns.

To stabilize training in deep architectures, the Transformer incorporates Residual Connections and Layer Normalization [22]. Residual connections allow information to bypass layers, improving gradient flow and preventing degradation in very deep networks. Layer normalization stabilizes numerical values during training and accelerates convergence.

As shown in Figure 2.1, each attention and feed-forward block is followed by an 'Add & Norm' operation implementing residual connections and normalization.

By stacking many encoder and decoder layers, modern LLMs can learn hierarchical language representations ranging from simple lexical patterns to advanced semantic reasoning.

3. Popular Modern Large Language Models

Table 2.2 presents an overview of the most widely used modern LLMs, their developers, and key characteristics. The model used in our system (GPT-4o-mini) is highlighted due to its efficiency and strong instruction-following capabilities.

Table 2.2: Overview of popular modern large language models

Model	Developer	Key Characteristics
GPT-4o-mini	OpenAI	Efficient API model; strong instruction-following at low cost. Used in our system via OpenRouter.
GPT-4o	OpenAI	Multimodal (text + images); top-tier reasoning and generation capabilities.
LLaMA 3	Meta AI	Open-weight; widely used for fine-tuning and research.

Model	Developer	Key Characteristics
Gemini 1.5	Google DeepMind	Natively multimodal; extremely long context window (up to 1M tokens).
Claude 3	Anthropic	Focused on safety, alignment, and reliable conversational quality.
Mistral 7B	Mistral AI	Compact and efficient open model; competitive with much larger models.

4. How Large Language Models Work

Large Language Models (LLMs) process natural language through a sequence of stages that transform a user prompt into a meaningful response. As illustrated in Figure 2.2, the process begins when a user submits a textual input or question. The model then analyzes the prompt using knowledge acquired during training on large-scale text datasets.

During the processing stage, the LLM relies on its learned parameters and contextual understanding to interpret the user's request. The context window enables the model to consider relevant information from the current interaction, allowing it to generate coherent and context-aware responses.

Based on the input and the available context, the model predicts the most appropriate sequence of tokens to produce a response. This prediction process is repeated iteratively until a complete answer is generated. The effectiveness of an LLM depends on the quality of its training data, the scale of its parameters, and its ability to capture contextual relationships within language.

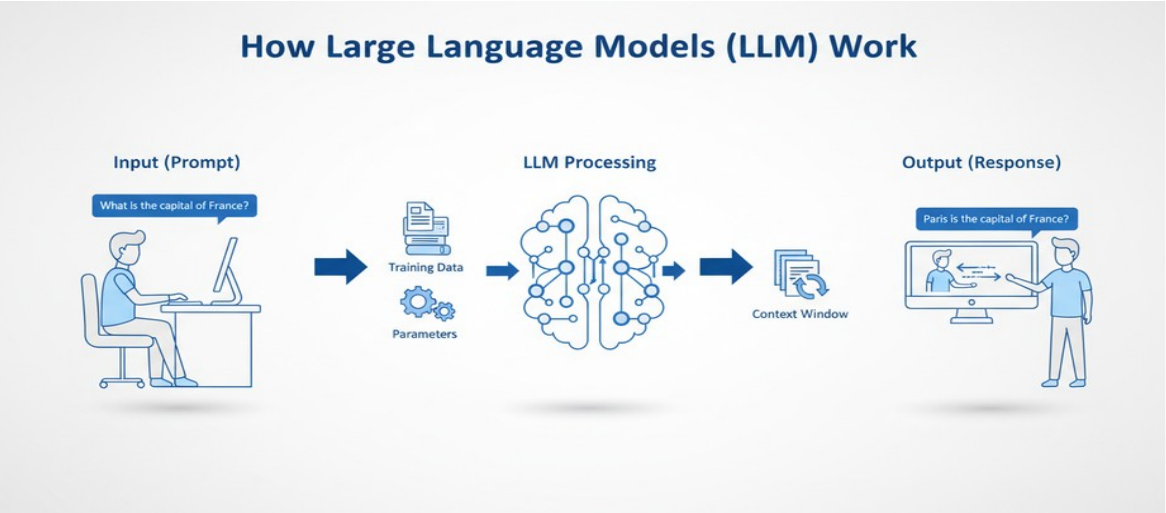


Fig. 2.2 : General Operational Workflow of a Large Language Model.

5. Training of Large Language Models

LLM training is a three-phase process. Each phase contributes to a different aspect of model capability: pre-training builds general language knowledge, fine-tuning adapts the model to specific tasks, and RLHF aligns the model with human values and safety requirements.

5.1 Pre-Training

The model is trained on massive unlabeled corpora (books, web pages, academic papers, code) using next-token prediction [29]. Through this process, the model acquires grammatical knowledge, semantic relationships, world knowledge, and long-range dependencies. The result is a general-purpose language model capable of performing a wide range of downstream tasks.

5.2 Fine-Tuning

The pre-trained model is adapted to specific downstream tasks using smaller labeled datasets [28]. Fine-tuning slightly adjusts model parameters to improve performance on targeted applications (question answering, code generation, instruction-following) while preserving general linguistic capabilities acquired during pre-training.

5.3 Alignment with Human Feedback (RLHF)

Reinforcement Learning from Human Feedback (RLHF) aligns model outputs with human preferences and safety requirements. Human evaluators rank model responses; a reward model is trained on these preferences; and reinforcement learning optimizes the LLM to maximize that reward. Systems such as ChatGPT and Claude rely heavily on RLHF [27]. GPT-4o-mini, used in our system, is an RLHF-aligned model.

6. Capabilities and Limitations of LLMs

6.1 Key Applications

Large Language Models have significantly transformed many applications in Artificial Intelligence and Natural Language Processing. Due to their ability to understand and generate human-like text, LLMs are widely used across multiple domains:

Conversational AI and Chatbots

One of the most prominent applications of LLMs is in conversational artificial intelligence, where these models power chatbots and virtual assistants capable of interacting with users in natural language. Modern conversational platforms such as ChatGPT exemplify how LLMs can produce coherent and contextually aware responses. This technology has been widely adopted in customer support systems, virtual assistants, online help services, and educational tutoring.

Machine Translation

LLMs have significantly enhanced the quality of machine translation by leveraging their deep contextual understanding of language. Unlike earlier approaches that relied on rule-based or statistical methods, LLM-based translation systems produce translations that are more fluent and contextually appropriate. Modern machine translation platforms typically utilize Transformer-based architectures, enabling efficient modeling of long-range dependencies and complex linguistic patterns.

Text Summarization

Automatic text summarization is another important LLM application, in which long documents are condensed into shorter, coherent summaries while preserving the most essential information. LLMs support both extractive summarization (selecting important sentences from the original text) and abstractive summarization (generating new sentences that convey the key content). This capability is particularly valuable in news aggregation, research article summarization, and legal document analysis.

Question Answering Systems

Question Answering Systems enable users to ask questions and receive precise, contextually informed answers derived from extensive knowledge sources. These systems are widely applied in educational platforms, technical support environments, knowledge management tools, and search engines, where rapid access to accurate information is essential.

Code Generation and Software Development

LLMs have also become powerful tools for code generation, assisting programmers by analyzing programming languages and extensive code repositories [30]. These models can automatically generate code snippets, detect bugs, suggest improvements, and provide documentation. Tools based on GPT-4 exemplify the practical applications of LLMs in software development, significantly enhancing developer productivity.

6.2 Advantages of Large Language Models

Large Language Models offer several significant advantages that have revolutionized the field of natural language processing:

- Multi-task capability: LLMs can perform text summarization, translation, question answering, sentiment analysis, and code generation without task-specific architectural modifications;
- Few-shot and zero-shot learning: LLMs can perform new tasks with minimal or no additional training, significantly reducing deployment time;
- Natural and coherent text generation: LLMs produce human-like, fluent, and contextually coherent text over long sequences, enabling conversational AI and automated content creation;
- Knowledge integration: By training on massive text corpora, LLMs encode general knowledge across multiple domains without explicit programming.

6.3 Limitations and Mitigation Strategies

Despite their capabilities, LLMs present several important limitations, each of which requires careful mitigation in practical systems:

Hallucination

LLMs can generate confident but factually incorrect information — a critical risk in scheduling contexts where incorrect teacher names, room numbers, or time slots could cause real operational problems [31]. In our system, this is mitigated by a strict design principle: the LLM never performs computations on scheduling data. All queries are resolved by Python operating on the actual DataFrame; the LLM only narrates pre-computed, verified results.

Static Knowledge / Knowledge Cutoff

LLMs have no access to data beyond their training cutoff. In our system, all current scheduling data (teacher assignments, room occupancy, availability) is provided to the LLM at inference time as part of the prompt context, bypassing this limitation entirely.

Lack of True Reasoning

LLMs cannot reliably perform logical deduction or mathematical optimization [32]. This is precisely why the scheduling itself is performed by the ILP solver, and the LLM is only used for the communication and explanation layer — tasks where fluent natural language generation is the key requirement.

High Computational Cost

Full GPT-4 models are expensive for high-frequency API calls. We use GPT-4o-mini, which provides strong instruction-following and generation quality at a fraction of the cost — suitable for interactive, production-grade deployment.

7. Future Directions of Large Language Models

The rapid evolution of LLMs has opened new opportunities in natural language processing. Ongoing research continues to address current limitations and explore innovative applications:

- Enhanced Reasoning: Future LLMs are expected to incorporate chain-of-thought prompting [34] and integration with symbolic reasoning and external knowledge bases;
- Efficient and Sustainable Models: Parameter-efficient fine-tuning (LoRA, adapters), sparse Mixture-of-Experts architectures [35], and optimized training techniques to reduce energy consumption [33];
- Improved Alignment and Safety: Enhanced RLHF, robust bias mitigation strategies, and prevention of misinformation generation;
- Personalization: User-specific adaptation allowing LLMs to learn individual preferences or domain-specific styles for tutoring systems and custom virtual assistants;
- Integration with External Tools: Future LLMs are expected to function as autonomous agents capable of web searching, performing calculations, and coordinating multi-step tasks [36].

8. Retrieval-Augmented Generation (RAG)

8.1 Definition and Motivation

Retrieval-Augmented Generation (RAG) is a hybrid NLP approach that enhances LLM outputs by combining a pre-trained generative model with an external information retrieval system. Instead of relying solely on knowledge stored in model parameters, RAG dynamically retrieves relevant documents from an external knowledge base at inference time and uses them to ground the generated response.

RAG was introduced to address two key LLM limitations: hallucination on specific factual questions, and static knowledge due to training cutoffs [39]. By providing the LLM with retrieved, verifiable facts, RAG significantly improves factual accuracy and allows LLMs to answer questions about content they were never trained on.

8.2 RAG Architecture and Workflow

Figure 2.3 illustrates the complete RAG workflow across its two phases: offline indexing and online query processing.

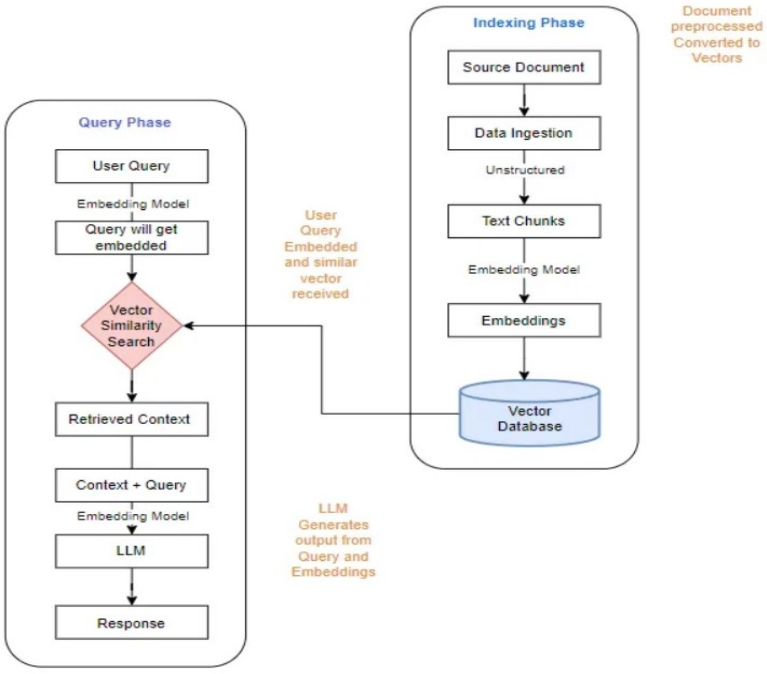


Fig. 2.3: RAG Workflow.

Indexing Phase

Source documents are collected, split into text chunks, encoded as dense vector embeddings using an embedding model [38], and stored in a vector database (FAISS, Pinecone, or Chroma) optimized for approximate nearest-neighbor search.

Query Phase

The user's query is encoded using the same embedding model. A similarity search retrieves the top-k most relevant chunks. These are combined with the query and passed to the LLM generator, which produces a factually grounded response based on both its parametric knowledge and the retrieved context.

8.3 Comparison of RAG vs. Standard LLM Inference

Table 2.3 compares RAG-enhanced and standard LLM approaches across key performance dimensions relevant to our scheduling system.

Table 2.3: Comparison of RAG vs. standard LLM inference

Aspect	RAG	Standard LLM
Factual Accuracy	High — grounded in retrieved documents	Lower — relies on parametric memory
Knowledge Freshness	Always current if database is updated	Static after training cutoff
Hallucination Risk	Significantly reduced	Higher, especially on specific facts
Explainability	High — sources can be inspected	Low — knowledge embedded in weights
Complexity	Higher — requires retriever + vector DB	Lower — single model inference

Aspect	RAG	Standard LLM
Latency	Moderate — retrieval step adds time	Fast

9. LLMs in Educational Systems and Scheduling

9.1 AI in Education

Artificial Intelligence has become a central driver of innovation in educational institutions [40]. AI-powered tools now assist with student performance prediction, curriculum design, resource allocation, and automated schedule generation — tasks that previously required significant manual effort. LLMs in particular are valuable in this context because they serve as natural language interfaces between complex computational systems and non-technical users, eliminating the need for administrators to understand optimization algorithms or interpret raw tabular output.

9.2 LLMs as Assistants in Scheduling Systems

In exam surveillance scheduling specifically, an LLM assistant enables academic coordinators to interact with the system naturally and efficiently. Concrete capabilities include:

- Query the schedule in plain language — e.g., "Who is supervising Room B3 on Tuesday morning?";
- Request modifications without knowing the underlying data structure — e.g., "Replace Dr. Benali with Dr. Mansouri for the Physique slot on Thursday.";
- Understand why a particular assignment was made or why a change is blocked;
- Receive summaries of workload distribution, overloaded teachers, and dense exam days;
- Interact with the system in their preferred language.

Conclusion

This chapter presented the theoretical foundations of Large Language Models (LLMs) and Retrieval-Augmented Generation (RAG), focusing on their architecture, training process,

capabilities, and limitations. It also highlighted the growing role of LLMs in intelligent systems through natural language interaction and automated assistance.

In the proposed exam surveillance scheduling system, LLMs are not used to perform mathematical optimization itself, but rather to enhance interaction with the generated schedules through querying, modification, explanation, and analytical reporting. The division of responsibilities is clear: the ILP solver ensures mathematical optimality and constraint satisfaction, while the LLM provides an accessible, multilingual, and intuitive interface for academic coordinators.

Combined with the ILP-based optimization module presented in Chapter 1, this integration contributes to building a more intelligent, practical, and user-friendly scheduling system for university environments.

CHAPTER 3: Literature Review — AI and LLMs in Scheduling Systems

Introduction

The application of artificial intelligence to scheduling problems has evolved considerably over the past three decades, driven by the growing complexity of real-world scheduling instances and the inadequacy of traditional manual methods to handle them efficiently. Scheduling problems — whether in manufacturing, healthcare, or academic institutions — share a common structure: a finite set of resources must be assigned to tasks over time, subject to hard constraints and subject to optimisation of one or more quality objectives. As these problems grow in size and complexity, purely manual approaches become error-prone, time-consuming, and increasingly unable to produce solutions of acceptable quality.

Research in this area has progressively moved through several generations of methods: rule-based and manual scheduling, classical optimisation methods (linear and integer programming), metaheuristic algorithms (genetic algorithms, simulated annealing, tabu search), neural network and deep learning approaches, and most recently, the integration of Large Language Models (LLMs) as natural language interfaces over formal optimisation systems. This chapter surveys the most significant works along this trajectory, with particular attention to applications in educational scheduling — the domain directly relevant to the present work.

1. From Manual Scheduling to Intelligent Automation

Traditional scheduling methods rely on human experts who use spreadsheets, whiteboards, and domain experience to construct timetables. While this approach works for small, stable institutions, it suffers from well-documented limitations at scale. Manual scheduling is time-consuming — producing a university examination schedule can require weeks of administrative effort — and is inherently prone to human error, particularly in detecting conflicts across large numbers of constraints [41]. Furthermore, manual schedules are rigid: when unexpected changes occur, such as a teacher becoming unavailable or a room being taken out of service, reconstruction requires starting the process largely from scratch.

AI scheduling systems overcome these limitations by automating the exploration of feasible solutions and optimising across multiple objectives simultaneously. According to a comparative analysis of AI versus traditional scheduling methods [42], organisations adopting intelligent

scheduling systems report scheduling time reductions of up to 30%, workforce utilisation improvements of 10-20%, and a significant reduction in the manual effort required for conflict resolution. These gains are especially pronounced in educational institutions, where scheduling complexity is high and administrative resources are limited [43].

The fundamental reason AI outperforms manual scheduling at scale is mathematical: many scheduling problems are NP-hard, meaning the number of possible assignments grows exponentially with the number of tasks, resources, and time slots. A problem involving 20 examination rooms, 50 teachers, and 10 time slots already generates billions of possible combinations — far beyond what any human can evaluate systematically. AI algorithms explore this space systematically and efficiently, finding high-quality solutions in seconds or minutes rather than days or weeks [42].

2. Classical AI and Metaheuristic Approaches to Scheduling

Before the emergence of deep learning and large language models, scheduling research was dominated by metaheuristic algorithms — optimisation frameworks that intelligently explore large solution spaces without requiring exact computation of the optimal solution. These methods remain widely used today, both as standalone schedulers and as components of hybrid systems.

2.1 Genetic Algorithms for Academic Timetabling

Genetic algorithms (GAs) are population-based metaheuristics that model the scheduling problem as an evolutionary process. Candidate schedules are represented as chromosomes; fitness is evaluated using a constraint satisfaction and objective function; and new generations of candidates are produced through selection, crossover, and mutation operations. GAs have been extensively applied to university course timetabling due to their ability to handle large numbers of constraints and their robustness to problem structure variations.

Saw et al. (2025) [44] developed the Adaptive Scheduler, a web-based AI-powered timetable generation system for academic institutions that uses genetic algorithms as its core optimisation engine. The system takes as input faculty availability, subject requirements, and classroom constraints, and produces conflict-free timetables through a fitness-driven evolutionary loop. The implementation combines a Python/Flask backend with a dynamic HTML/JavaScript interface, and supports PDF export of generated schedules. The study demonstrates that GA-based academic scheduling can produce practical, constraint-satisfying timetables with significantly less manual effort than traditional approaches.

Yang and Jat (2011) [45] demonstrated that incorporating guided and local search strategies into standard genetic algorithms substantially improves convergence speed and constraint satisfaction for university course timetabling. Their work established that hybrid GA approaches — combining evolutionary search with local improvement — consistently outperform pure GAs on standard benchmark datasets. This principle of combining optimisation paradigms has since become a guiding design principle in educational scheduling research.

2.2 AI vs. Traditional Scheduling: A Systematic Comparison

A systematic comparison between AI-driven and traditional manual scheduling methods [42] highlights several consistent findings across domains. Traditional methods rely on fixed rules and expert knowledge, making them brittle to novel constraint combinations and slow to adapt to changes. AI methods, particularly those based on constraint programming and metaheuristics, are domain-agnostic and automatically generalise to new constraint configurations. In educational scheduling specifically, AI systems have been shown to produce schedules that are not only constraint-satisfying but also fairer — distributing workloads more equitably across faculty members — a quality that manual scheduling rarely achieves due to cognitive limitations and implicit bias.

The most significant advantage of AI scheduling in practice is not solution quality in isolation but the ability to respond to dynamic changes. When a teacher declares absence or a classroom becomes unavailable mid-session, an AI system can recompute a modified schedule in seconds, whereas manual rescheduling would require hours of human effort and is highly prone to cascading constraint violations [41, 42].

3. Neural Network Approaches to Scheduling

Neural networks represent a distinct approach to scheduling that learns solution patterns from historical data rather than solving each instance from scratch. This paradigm is particularly valuable in settings where scheduling problems follow predictable patterns — such as recurring academic calendars — and where the computational cost of exact or metaheuristic methods must be amortised over many similar problem instances.

3.1 Deep Learning for Manpower Scheduling

Liu and Zhang (2021) [46] proposed an integrated framework combining genetic algorithms with deep neural networks to solve the multi-shift manpower scheduling problem. Their approach proceeds in two stages: a genetic algorithm first generates a feasible scheduling plan for a one-week horizon, satisfying all hard constraints (shift coverage, employee availability, working hour limits, and holiday requirements). The resulting weekly schedule is then used as training data for a feedforward neural network, which learns to predict scheduling arrangements for future periods without re-running the full optimisation process.

The authors evaluated five neural network architectures — Forward Deep Neural Network (FDNN), Recurrent Neural Network (RNN), Radial Basis Function Neural Network (RBFNN), Long Short-Term Memory Network (LSTM), and Gated Recurrent Unit (GRU) — finding that FDNN provided the best balance of convergence speed, accuracy, and resistance to overfitting. The system was validated on two real-world instances: a retail cashier scheduling problem and a bus driver scheduling problem, demonstrating that the trained neural network can accurately predict scheduling arrangements for time horizons beyond the training period.

3.2 AI Techniques in Project and Resource Scheduling: A Systematic Review

A comprehensive bibliometric and systematic review of AI applications to project scheduling problems — covering 104 papers published between 1985 and 2021 [47] — reveals important trends in the field. The review finds that Artificial Neural Networks, Bayesian Networks, and Reinforcement Learning are the three most frequently applied AI techniques to scheduling under stochastic conditions. A notable finding is that these techniques are rarely used in isolation: the vast majority of high-performing systems hybridise AI methods with classical metaheuristics, typically using the AI component for prediction or learned heuristics and the metaheuristic component for solution construction and improvement.

4. AI in Educational Scheduling: Schools and Universities

Educational scheduling occupies a special position in the scheduling literature because it combines several sources of complexity that rarely appear together in other domains: student conflict avoidance, teacher availability and workload balance, room capacity and equipment constraints, curricular sequencing requirements, and the need for explainable, transparent results that can be communicated to all stakeholders. This section reviews the most relevant works addressing AI applications to educational timetabling, from school-level systems to university examination scheduling.

4.1 AI-Powered Timetable Management in Schools and Colleges

Modern AI-powered timetable management systems for educational institutions offer a substantially different paradigm from traditional manual scheduling [41]. Contemporary platforms collect scheduling data — faculty availability, subject hours, classroom capacities, and institutional preferences — and process them through optimisation algorithms to generate conflict-free schedules automatically. Key functional capabilities that distinguish AI-based systems from manual approaches include real-time conflict detection and resolution, automatic workload balancing across faculty, instant adaptation to changes (teacher absences, room reassignments), integration with digital learning platforms, and data-driven reporting on resource utilisation [43, 56].

4.2 University Timetabling with ILP: Handling Hybrid Teaching

Davison, Kheiri, and Zografos (2024) [48] address the university course timetabling problem extended to include hybrid teaching modalities — in-person, online, and blended delivery — using a multi-objective binary programming model. The formulation captures faculty preferences, room availability, student enrollment patterns, and modality-specific constraints within a unified ILP framework. A lexicographic solution method prioritises hard constraint satisfaction before optimising soft objectives, ensuring that all generated timetables are feasible.

This work confirms that ILP remains the method of choice for university-scale timetabling when exact constraint satisfaction is required, even as problem complexity increases with new scheduling dimensions such as hybrid instruction. The explicit mathematical formulation also provides the transparency and auditability required for institutional acceptance — a consideration directly relevant to the design of our examination scheduling system.

4.3 Decision Support Systems for Instructor Assignment

Xue et al. (2024) [49] present a decision support system addressing the instructor assignment problem — structurally analogous to the invigilator assignment problem in the present work — using a MILP formulation that simultaneously optimises course section allocation and instructor assignment. Deployed operationally at a real university, the system reduced new course assignments to instructors by up to 81% and generated estimated savings of USD 130,000 in administrative costs. These results provide strong empirical evidence for the practical value of ILP-based assignment systems in academic institutions.

4.4 Neural Network Scheduling in Educational Contexts

The automated learning and scheduling assistant proposed by Mohanraj et al. (2024) [50] extends the application of LLMs to educational process automation, using prompt engineering and Retrieval-Augmented Generation to schedule student assessments, manage teacher duties, and generate academic calendars. The RAG component grounds LLM scheduling decisions in verified institutional policy documents and historical data, mitigating hallucination. This work provides direct evidence that LLM-based scheduling assistants can be deployed effectively in educational administration contexts — the same context as the present work.

5. Large Language Models in Scheduling Systems

The most recent research direction integrates Large Language Models as natural language interfaces over formal scheduling and optimisation systems. This development addresses a long-standing usability gap: even when powerful optimisation algorithms generate high-quality schedules, non-technical administrators may struggle to query, modify, and understand the results. LLMs bridge this gap by enabling natural language interaction with scheduling systems.

5.1 LLM-Assisted Mathematical Optimisation Modelling

OptiMUS, developed by AhmadiTeshnizi, Gao, and Udell (2024) [51], is a multi-agent LLM system that translates natural language descriptions of optimisation problems — including scheduling — into executable MILP models solved by formal solvers (CBC, Gurobi, CPLEX). Three specialised LLM agents collaborate: a Formulator that extracts variables and constraints, a Programmer that generates solver code, and an Evaluator that verifies results and triggers re-formulation when errors are detected. A key design principle — the LLM never performs numerical computation, which is entirely delegated to the MILP solver — directly parallels the architecture of the present system. OptiMUS outperforms standard LLM prompting by more than 20% on established OR benchmarks [51].

5.2 LLMs as Natural Language Scheduling Interfaces

Jobson and Li (2024) [52] investigated LLMs for conference programme scheduling — a combinatorially structured problem similar to exam scheduling — and found that zero-shot LLMs reliably fail to produce feasible schedules when hard constraints are numerous. However, LLMs proved effective at semantic grouping and constraint interpretation tasks, pointing toward a hybrid architecture where LLMs handle language and humans/solvers handle constraint enforcement. This finding directly motivates the design choice in our system to use ILP for all scheduling decisions and LLM only for the conversational interface.

Pallagani et al. (2024) [53], in a survey of 126 papers on LLMs in automated planning and scheduling, conclude that the neuro-symbolic hybrid — LLM paired with a formal planner or optimiser — consistently outperforms either component alone. This theoretical synthesis provides broad support for the hybrid ILP + LLM architecture used in the present work.

5.3 LLMs in Manufacturing and Resource Scheduling

Beyond educational and planning contexts, LLMs have been applied to manufacturing scheduling. Xiao et al. (2025) [54] proposed an evolutionary scheduling framework for human-robot collaborative manufacturing that uses a locally deployed, fine-tuned LLM to generate heuristic dispatching rules guiding an evolutionary algorithm. The system demonstrated reductions in makespan and tardiness compared to classical baselines. Lim et al. (2024) [55] demonstrated that natural language voice commands, interpreted by an LLM, can reliably drive robot task scheduling in assembly environments — providing empirical validation that LLM-based natural language interfaces can operate effectively in time-sensitive, constraint-heavy scheduling contexts.

6. Synthesis, Comparison, and Positioning of the Proposed System

6.1 Evolution of the Research Landscape

The chronological evolution of AI approaches to scheduling reveals a clear generational progression: genetic algorithms and classical metaheuristics dominated from the 1990s through the early 2010s; neural network approaches emerged from 2015 onward; and LLM integration represents the frontier from 2023 to the present. Each generation has added capabilities — from constraint satisfaction (GA era) to learned generalisation (NN era) to natural language accessibility (LLM era) — without fully replacing the previous one. The most capable modern systems are hybrids that leverage the strengths of multiple generations.

6.2 Recurring Themes and Common Findings

Across all reviewed works, several findings recur consistently. First, no single AI method dominates across all scheduling contexts: GAs excel when constraint structures are complex and diverse; ILP provides exact guarantees for medium-scale problems; neural networks generalise well to recurring instances; and LLMs uniquely enable natural language interaction. Effective real-world systems are therefore typically hybrid, combining methods across these categories [45, 46]. Second, the practical impact of AI scheduling is greatest not in the quality of

the initial generated schedule, but in the ability to respond quickly and correctly to dynamic changes [41, 42]. Third, accessibility for non-technical users has emerged as a critical system requirement, driving the integration of LLMs as conversational interfaces over formal optimisation backends [51, 52, 53].

6.3 Positioning of the Proposed System

The proposed Exam Surveillance Scheduler combines Integer Linear Programming for exact, constraint-guaranteed assignment with a GPT-4o-mini language model for natural language interaction. This hybrid design is directly motivated by the literature: ILP is the established method for university-scale assignment problems requiring auditability [48, 49], while LLM integration addresses the usability gap identified across multiple works [50, 51, 52]. The system additionally addresses a specific institutional need — exam surveillance scheduling at Mohamed Boudiaf University, M'Sila — which has not been addressed by any of the surveyed works, all of which target course timetabling or general assignment problems rather than invigilator assignment specifically.

Conclusion

This chapter reviewed the evolution of AI-based scheduling systems, from classical optimisation and metaheuristic approaches to neural networks and Large Language Models. The analysed works demonstrate that modern scheduling problems, especially in educational environments, require both efficient constraint satisfaction and flexible user interaction.

The literature also shows that hybrid systems provide the most effective solution by combining formal optimisation techniques with AI capabilities. Based on these findings, the proposed system adopts a hybrid ILP + LLM architecture, where ILP ensures accurate and conflict-free scheduling, while the LLM enables multilingual natural language interaction and intelligent scheduling assistance. This approach aligns with current research trends and provides a strong foundation for the implementation presented in the next chapter.

CHAPTER 4: Implementation, Evaluation and Results Analysis

1. Introduction

The previous chapter presented the theoretical foundations and design architecture of the proposed system — an intelligent, web-based examination surveillance scheduler built for academic institutions. In this chapter, the focus shifts to the practical dimension: how the system was implemented, what technical decisions were made during development, and how the final product performs under realistic conditions.

Algerian universities, including Mohamed Boudiaf University in M'Sila, manage examination sessions involving dozens of teachers, hundreds of exam slots, and strict scheduling constraints. Coordinating these sessions manually is error-prone, time-consuming, and difficult to audit. The Exam Surveillance Scheduler addresses these shortcomings by combining Integer Linear Programming (ILP) for optimal assignment, a natural-language interface for querying and editing the schedule, and automated conflict detection to prevent double-booking of rooms or teachers.

This chapter is structured as follows: Section 2 describes the dataset and input format. Section 3 details the system implementation, covering the solver engine, the AI assistant, the editor, and the export features. Section 4 presents testing results and performance metrics. Section 5 discusses the development environment and tools. Section 6 discusses the role of LLMs in the system. Section 7 concludes the chapter.

2. Dataset and Input Format

2.1 Source of Data

The system was designed and tested using real examination data provided by the Faculty of Mathematics and Computer Science at Mohamed Boudiaf University, M'Sila. The input dataset consists of Excel workbooks structured in a specific two-sheet format that encodes all the information needed to produce a valid surveillance assignment.

The first sheet captures the module-to-teacher responsibility mapping, where each row associates an academic module with its designated responsible teacher. The second sheet contains the examination timetable itself, listing every exam slot with its date, time, room,

student group, academic level, and the required number of invigilators. Teacher names appear as column headers from column 9 onward, and pre-assigned invigilation duties are marked with special tokens.

2.2 Data Encoding Convention

The scheduling sheet uses a three-token convention to encode assignment states. The double-asterisk token (**) identifies a fixed, pre-assigned invigilator that must not be changed by the solver. The token XX identifies the responsible teacher for that module, assigned automatically during pre-processing. The token X marks a guard (invigilator) assigned by the ILP solver during optimisation.

Row 1 stores teacher names. Row 2 stores the required number of surveillance slots per teacher. Row 3 stores the solver-computed assignment counts (filled after optimisation). Exam data begins at row 4. Column 7 holds the required number of invigilators per exam; column 8 is filled by the solver with the actual count.

2.3 Dataset Characteristics

The dataset used during development and testing covered a complete examination session at the Faculty of Mathematics and Computer Science. The session comprised 10-15 examination days, with 2-4 daily time slots (morning and afternoon), yielding a total of 40-80 exam rows. The teaching staff pool consisted of 20-50 active invigilators, and the rooms used ranged from 5-15 spaces including amphitheatres and laboratories.

2.4 Limitations of the Input Data

While the dataset was representative of a real examination session, several practical constraints were encountered during development. Teacher names in the source file are written in Arabic script, which required implementing a transliteration layer for exports that must be readable by standard fonts. Date formatting across files was inconsistent — some dates were in French text (e.g., "15 mai 2026") while others used numeric formats — necessitating a multi-format date parser. Finally, some exam slots had no pre-assigned responsible teacher, requiring the system to fall back gracefully without crashing.

3. System Implementation

The system is composed of five tightly integrated subsystems: the ILP-based schedule generator, the natural-language search assistant, the structured editor with conflict detection, the

absence tracking module, and the export engine. Each subsystem was implemented in Python and exposed through a Streamlit web interface.

3.1 User Interface Design

The front-end is built entirely with Streamlit, chosen for its ability to produce interactive, browser-based applications from pure Python without requiring a separate frontend codebase. The interface is organised into two main areas. The first area — the Schedule Generator — occupies the top of the page. It contains a file upload widget, a collapsible constraint panel, a run button, and a live progress log that streams solver output in real time. The second area — the AI Assistant and Editor — becomes active once a valid schedule has been generated or loaded. It is divided into four tabs: Search, Edit, Export and Absence Tracking, and Edit History.

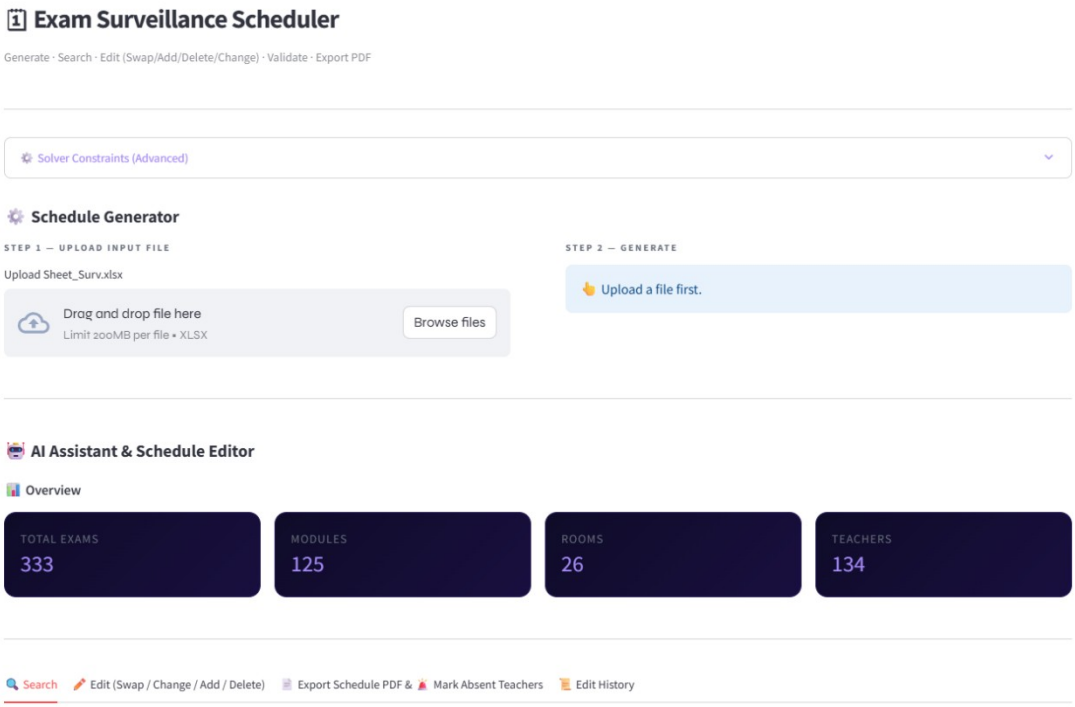


Fig. 4.1: Overall layout of the Exam Surveillance Scheduler interface .

Custom CSS injected at startup overrides Streamlit's default styling. The design uses a dark-navy palette with purple and indigo accents, JetBrains Mono for numeric and code values, and Syne for general text. Metric cards, log boxes, badge indicators, and colour-coded status labels are all implemented through injected HTML rather than Streamlit's built-in components, allowing a level of visual polish not achievable with default widgets alone.

3.2 Solver Constraints Panel

Before generating a schedule, the user can expand the Solver Constraints panel to tune five ILP parameters that directly affect the quality and feasibility of the solution. Thèse paramètres are:

- **Minimum working days** — the fewest days a teacher must be assigned across the session.
- **Maximum working days** — the upper bound on days per teacher, used to prevent overloading.
- **Maximum assignments in any three consecutive slots** — limits back-to-back fatigue.
- **Maximum assignments in any four consecutive slots** — a wider fatigue window.
- **Force exact balance** — when enabled, every teacher must reach exactly their required slot count. When disabled, the solver maximises total coverage, which is useful when total demand and supply are mismatched.

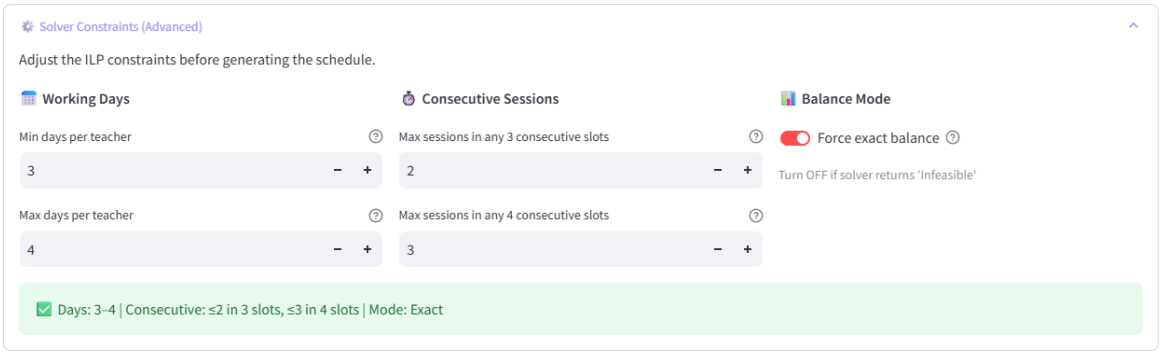


Fig. 4.2: Solver Constraints panel showing all five configurable parameters.

3.3 ILP-Based Schedule Generator

3.3.1 Overview of the Approach

The core scheduling problem is modelled as a binary Integer Linear Program (ILP). The objective and constraints are assembled using the PuLP library, and the CBC solver (bundled with PuLP) is used to find the optimal solution. The entire solving process runs in a background daemon thread so that the Streamlit interface remains responsive, streaming log messages to the user in real time.

3.3.2 Pre-processing Stage

Before the ILP is built, two pre-assignment passes are performed. In the first pass, all cells already marked with the double-asterisk token are recorded as fixed assignments. Their slot occupancy information is stored in a teacher-busy dictionary that prevents the solver from scheduling the same teacher twice in the same time slot. In the second pass, the responsible

teacher for each module is looked up in the module-to-teacher mapping sheet and tentatively assigned to one exam of that module, provided they are not already busy at that time and the exam still needs an invigilator.

```
def run_scheduler(input_bytes: bytes, shared: dict, constraints: dict = None):
        # ── Pre-assignment: assign responsible teacher (XX) ──────────────
        module_to_exams = {}
        for e in exams:
            module_to_exams.setdefault(exam_module[e], []).append(e)

        for module, exam_list in module_to_exams.items():
            if module not in module_responsible:
                continue
            teacher = module_responsible[module]
            if teacher not in TEACHERS:
                continue
            best_exam, best_score = None, float("inf")
            for e in exam_list:
                if exam_fixed_load[e] >= required_per_exam[e]:
                    continue
                if teacher in teacher_busy and \
                   (exam_date[e], exam_time_map[e]) in teacher_busy[teacher]:
                    continue
                if exam_fixed_load[e] < best_score:
                    best_score = exam_fixed_load[e]
                    best_exam  = e
            if best_exam is not None:
                df_time_slots.iat[best_exam, teacher_to_col[teacher]] = "XX"
                exam_fixed_load[best_exam] += 1
                teacher_busy.setdefault(teacher, set()).add(
                    (exam_date[best_exam], exam_time_map[best_exam])
                )

        # ── Tally fixed assignments ──────────────────────────────────────
        fixed_exam    = {e: 0 for e in exams}
        fixed_teacher = {t: 0 for t in TEACHERS}
        slot_map      = {}
        for e in exams:
            slot_map.setdefault((exam_date[e], exam_time_map[e]), []).append(e)
        fixed_slot = {(t, k): 0 for t in TEACHERS for k in slot_map}

        for e in exams:
            for t in TEACHERS:
                if is_marker(df_time_slots.iat[e, teacher_to_col[t]], {"**", "XX"}):
                    fixed_exam[e]    += 1
                    fixed_teacher[t] += 1
                    fixed_slot[(t, (exam_date[e], exam_time_map[e]))] += 1
```

Fig. 4.3: Pre-processing: fixed assignments and responsible-teacher pre-assignment.

3.3.3 Decision Variables

Let T denote the set of teachers and E the set of exam slots. Two families of binary decision variables are introduced. The primary variables x(t, e), defined for every (teacher, exam) pair not already fixed during pre-processing, represent the assignment decision: $x(t, e) = 1$ if and only if teacher t is assigned to invigilate exam e. Only pairs not already covered by a pre-assigned XX or ** token are given a variable; all others are treated as constants with known values, which reduces the size of the search space significantly. The secondary variables $y(t, d)$, defined for every (teacher, day) pair, capture whether teacher t has any assignment whatsoever on day d. These are auxiliary variables introduced specifically to allow the working-day bounds

to be expressed as linear inequalities. They are linked to the x variables through a set of Big-M–style linking constraints described in the following section. The total number of decision variables in a typical faculty-scale instance is in the order of |T| × |E| (approximately 134 × 333 = 44,622 binary variables for the full faculty dataset), which is well within the capacity of the CBC solver.

3.3.4 Constraints

Seven constraint families are enforced. Each family corresponds to a distinct operational requirement of the examination scheduling problem, and together they form a complete formal specification of a valid assignment. The constraints are detailed below alongside the code-level expression used in the PuLP model.

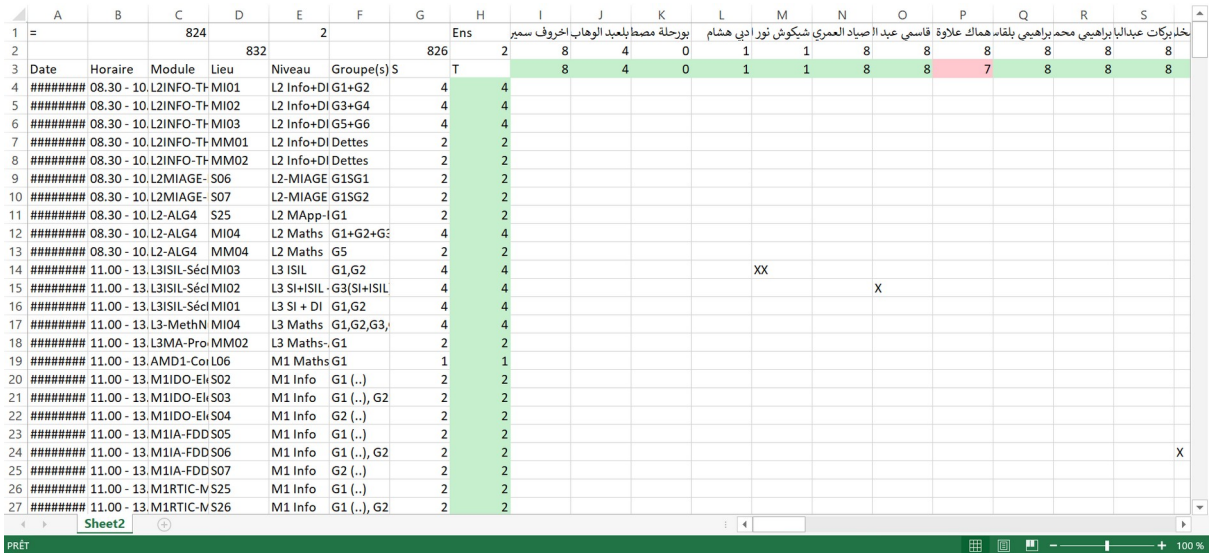
- **Linking x to y (Constraint 1):** For every teacher t and every exam e on day d, $x(t, e) \leq y(t, d)$. This ensures that if the solver assigns teacher t to any exam on day d, the auxiliary variable y(t, d) is forced to 1. Without this constraint, the day-count bounds could be bypassed by assigning teachers to exams without counting those days as working days.
- **Day count bounds (Constraint 2):** For every teacher t: $\text{min_days} \leq \sum_{8} y(t, d) \leq \text{max_days}$. The lower bound prevents a teacher from being assigned to exams concentrated on a single day; the upper bound prevents overloading across the session. The lower bound is automatically clamped to the teacher's required slot count, so teachers with very few required slots are not penalised for having a short assignment list.
- **Daily assignment cap (Constraint 3):** For every teacher t and day d: $\sum_{1} x(t, e) \leq M \times y(t, d)$, where M is set to 10 (a safe upper bound on the number of exams in a single day). This is a Big-M formulation: when y(t, d) = 0, the right-hand side collapses to 0 and no assignment on that day is possible; when y(t, d) = 1, the constraint becomes non-binding. The Big-M approach avoids the need for an equality constraint, which would be harder to satisfy when supply and demand are imbalanced.
- **Workload balance (Constraint 4):** For every teacher t, let fixed_t denote the count of pre-assigned slots. In balanced mode: $\text{fixed_t} + \sum_{1} x(t, e) = \text{required_t}$. In maximise mode: $\text{fixed_t} + \sum_{1} x(t, e) \leq \text{required_t}$. The equality variant is used only when total teacher supply exactly matches total exam demand; otherwise, forcing equality would make the model infeasible. The inequality variant allows the solver to leave some required slots unfilled when the global assignment is otherwise impossible, reporting the shortfall through the colour-coded output.
- **Exam coverage (Constraint 5):** For every exam e, let fixed_e denote the count of pre-assigned invigilators: $\text{fixed_e} + \sum_{5} x(t, e) = \text{required_e}$ (balanced) or $\leq \text{required_e}$ (maximise). This constraint guarantees that each exam receives exactly the number of invigilators specified in the input timetable. It is the symmetric counterpart to the workload constraint and ensures that coverage is controlled from both the teacher side and the exam side simultaneously.
- **No simultaneous assignment (Constraint 6):** For every teacher t and every time slot s (identified by a (date, time) pair): $\text{fixed_slot}(t, s) + \sum_{18}{}' x(t, e) \leq 1$, where E' is the set of exams in slot s. The slot_map data structure groups all exams sharing the same date and start time, allowing this constraint to be expressed as a single inequality per (teacher, slot) pair rather than quadratically many pairwise constraints. This is the critical safety

constraint: its violation would mean a teacher is physically present in two rooms at the same moment.

- **Consecutive-slot fatigue limits (Constraint 7):** For every teacher t, day d, and every window of three consecutive time slots [s_1, s_2, s_3] within that day: $\sum_3$ (fixed_slot(t, s_i) + $\sum_{18}$‘ x(t, e)) ≤ MAX_TRIPLE. An analogous constraint is added for every window of four consecutive slots with threshold MAX_QUAD. These constraints prevent the solver from concentrating a teacher’s workload in a single intense burst of back-to-back supervisions, which would be operationally unreasonable even if it satisfied all other constraints. The windows are computed by iterating over the sorted list of distinct time slots observed on each day, so the constraint automatically adapts to days with two, three, or four exam periods without any hard-coded assumptions.

3.3.5 Solving and Output

Once the model is built, the CBC solver is called with the objective function set according to the operating mode. In balanced mode, the model is treated as a feasibility problem (LpMinimize with a zero objective), because all assignments are fully determined by the equality constraints; the solver simply searches for any feasible point. In maximise mode, the objective is: maximise ∑ x(t, e) over all free variables, which instructs the solver to fill as many slots as possible subject to all constraints. This distinction is important: forcing minimisation when the problem is actually over-constrained would cause an infeasibility report, while switching to maximisation allows partial solutions to be returned and inspected by the coordinator. If the problem is feasible, all *x(t, e)* variables with value 1 are written back into the schedule dataframe as ‘X’ tokens. Assignment counts per teacher and per exam are then recalculated and written into the appropriate summary rows. The completed dataframe is exported to an Excel workbook, with cells colour-coded green when the achieved count meets or exceeds the required count, and red when it falls short.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	=		824		2			Ens	سمير اخروف	بلعبد الوهاب	بورحلة مصطفى	ادبي هشام	شيكوش نور	العمري الصياد	قاسمي عبد الـ	علاوة هماك	براهيمي بلقاسم	براهيمي محمد	بركات عبدالبا
2				832			826	2	8	4	0	1	1	8	8	8	8	8	8
3	Date	Horaire	Module	Lieu	Niveau	Groupe(s)	S	T	8	4	0	1	1	8	8	7	8	8	8
4	########	08.30 - 10.	L2INFO-TH	MI01	L2 Info+DI	G1+G2	4	4											
5	########	08.30 - 10.	L2INFO-TH	MI02	L2 Info+DI	G3+G4	4	4											
6	########	08.30 - 10.	L2INFO-TH	MI03	L2 Info+DI	G5+G6	4	4											
7	########	08.30 - 10.	L2INFO-TH	MM01	L2 Info+DI	Dettes	2	2											
8	########	08.30 - 10.	L2INFO-TH	MM02	L2 Info+DI	Dettes	2	2											
9	########	08.30 - 10.	L2MIAGE-	S06	L2-MIAGE	G1SG1	2	2											
10	########	08.30 - 10.	L2MIAGE-	S07	L2-MIAGE	G1SG2	2	2											
11	########	08.30 - 10.	L2-ALG4	S25	L2 MApp-I	G1	2	2											
12	########	08.30 - 10.	L2-ALG4	MI04	L2 Maths	G1+G2+G3	4	4											
13	########	08.30 - 10.	L2-ALG4	MM04	L2 Maths	G5	2	2											
14	########	11.00 - 13.	L3ISIL-Séc	MI03	L3 ISIL	G1,G2	4	4					XX						
15	########	11.00 - 13.	L3ISIL-Séc	MI02	L3 SI+ISIL	G3(SI+ISIL	4	4							X				
16	########	11.00 - 13.	L3ISIL-Séc	MI01	L3 SI + DI	G1,G2	4	4											
17	########	11.00 - 13.	L3-MethN	MI04	L3 Maths	G1,G2,G3,	4	4											
18	########	11.00 - 13.	L3MA-Pro	MM02	L3 Maths-	G1	2	2											
19	########	11.00 - 13.	AMD1-Co	L06	M1 Maths	G1	1	1											
20	########	11.00 - 13.	M1IDO-El	S02	M1 Info	G1 (..)	2	2											
21	########	11.00 - 13.	M1IDO-El	S03	M1 Info	G1 (..), G2	2	2											
22	########	11.00 - 13.	M1IDO-El	S04	M1 Info	G2 (..)	2	2											
23	########	11.00 - 13.	M1IA-FDD	S05	M1 Info	G1 (..)	2	2											
24	########	11.00 - 13.	M1IA-FDD	S06	M1 Info	G1 (..), G2	2	2											X
25	########	11.00 - 13.	M1IA-FDD	S07	M1 Info	G2 (..)	2	2											
26	########	11.00 - 13.	M1RTIC-M	S25	M1 Info	G1 (..)	2	2											
27	########	11.00 - 13.	M1RTIC-M	S26	M1 Info	G1 (..), G2	2	2											

Fig. 4.4: Generated schedule with green/red assignment status highlighting .

3.3.6. Significance of the ILP Approach

The ILP model is the technical core of the system and the principal contribution of the scheduling engine. Its importance cannot be overstated for three reasons. First, it provides a **mathematical guarantee of correctness**: the solver will never produce a schedule that violates any encoded constraint. Every hard rule — no simultaneous assignments, exact exam coverage, workload quotas — is enforced algebraically, not heuristically. This is a property that no rule-based greedy algorithm or genetic optimiser can offer unconditionally.

Second, ILP enables **transparent infeasibility diagnosis**. When the problem is infeasible — for instance, when a teacher has been pre-assigned (via the ** token) to two simultaneous exams in the input file — the CBC solver reports status “Infeasible” immediately, rather than silently returning a schedule that secretly violates one of the constraints. This diagnostic behaviour is practically valuable: it directs the coordinator’s attention to the precise contradiction in the input data, which can then be corrected before the session begins.

Third, the ILP formulation provides a **principled foundation for the LLM integration layer** described in Section 6. Because all scheduling decisions are made by the solver under hard constraints, the language model can be safely restricted to the role of communication interface, with no risk that the model will suggest or accept an operationally invalid assignment. The ILP layer acts as a verified ground truth against which all natural-language edits are checked before being committed to the schedule. In this way, the correctness of the ILP model is what makes the AI-assisted editing trustworthy. Without ILP, the system could not offer both natural-language flexibility and scheduling correctness simultaneously.

3.4 Natural-Language Search Assistant

3.4.1 Hybrid Search Architecture

Once a schedule is loaded, users can query it in plain English using the Search tab. Rather than routing every query through a language model — which would be slow and expensive — the system uses a hybrid architecture that classifies each query and routes it to the cheapest adequate handler. Three handlers exist: a pure-Python structured filter that handles simple lookups such as "exams in room MI01" or "exams on 16-05"; a Python-powered information answerer that handles workload and scheduling questions such as "how many days does Hamdi have?" without any LLM involvement; and an LLM-backed reasoner that handles complex queries requiring knowledge of what is absent from the schedule, such as "which rooms are free on 18-05 at 08:30?".

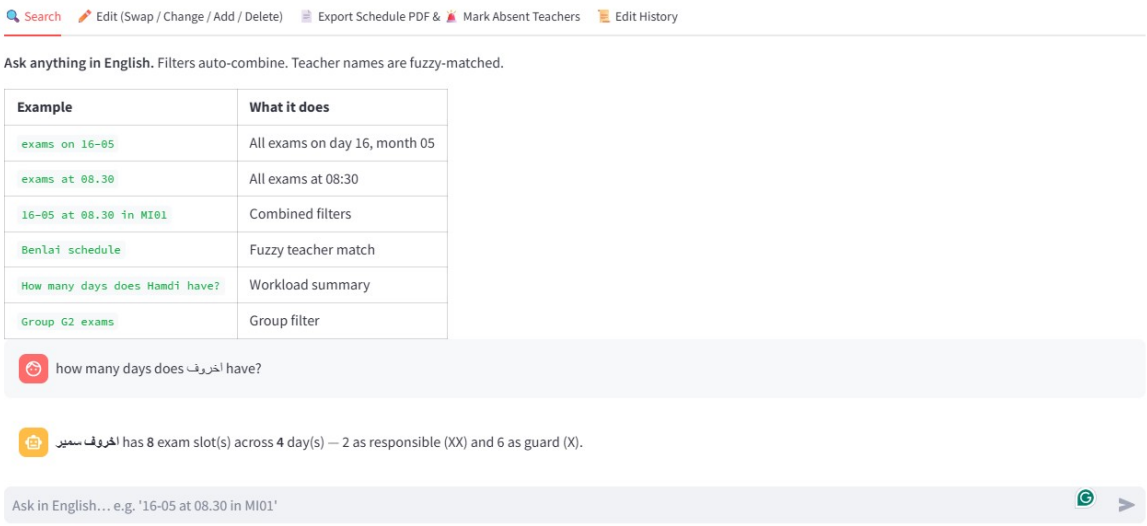


Fig. 4.5: Search tab with an example natural-language query.

3.4.2 Fuzzy Teacher Matching

Teacher names in the schedule are often written in Arabic and may be transliterated differently when a user types a query in Latin characters. To handle this, the system uses Python's SequenceMatcher to compute token-level similarity between the query and every known teacher name. A match is accepted when the best similarity score exceeds a configurable threshold (default 0.55). This allows queries like "Benlai schedule" to match a teacher whose stored name is in Arabic without requiring the user to know the exact spelling.

3.5 Schedule Editor with Conflict Detection

3.5.1 Three-Tier Editing Model

The Edit tab provides three progressively powerful ways to modify the schedule. The first tier — Swap Teachers — is a direct pandas operation that exchanges all guard assignments between two selected teachers across the entire schedule. The second tier — Quick Edit — lets the user select a field to filter on, a matching value, a field to change, and a new value, all via dropdowns that auto-complete from the current schedule data. The third tier — AI Free-Text Edit — accepts an open-ended instruction in English or French and uses a GPT model to translate it into a structured list of field-update operations.

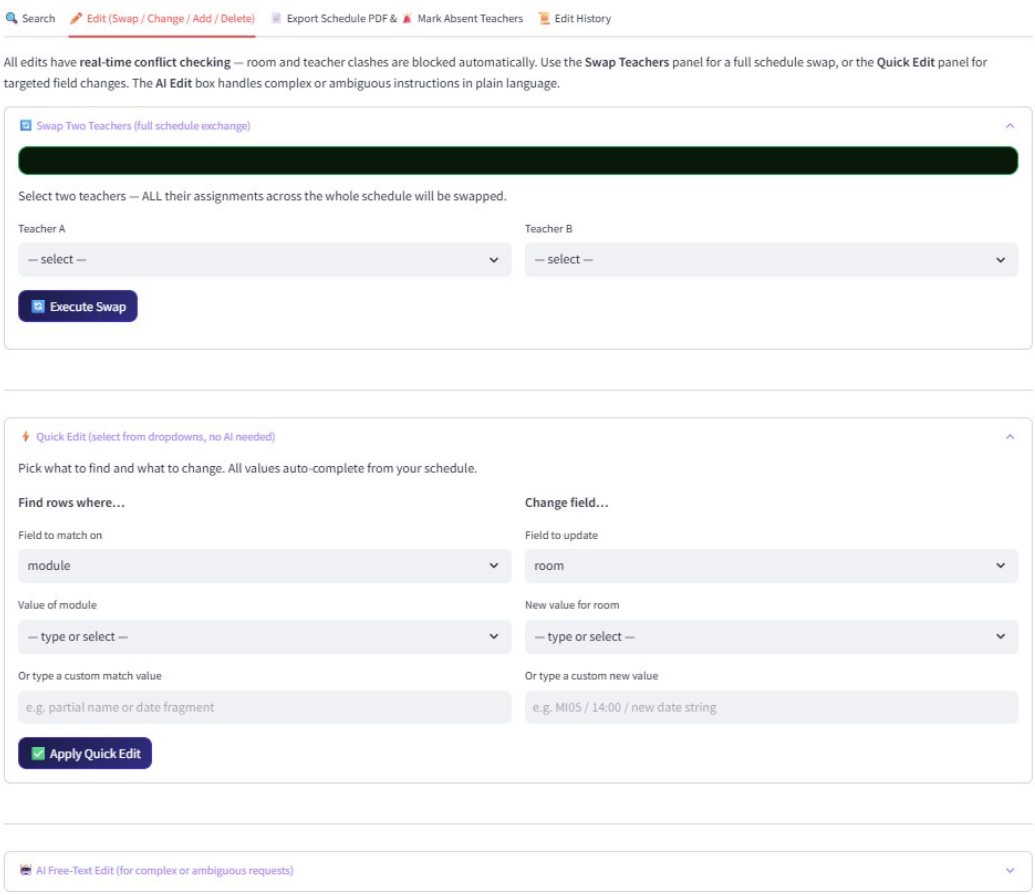


Fig. 4.6: Edit tab: Swap Teachers and Quick Edit panels.

3.5.2 Conflict Detection

Every edit — regardless of which tier produced it — passes through a conflict checker before being committed to the working dataframe. Two types of conflict are checked. A room conflict exists when the target room is already occupied by a different exam at the same date and time. A teacher conflict exists when a teacher being added to an exam is already assigned to a different exam in the same time slot. Conflicting operations are blocked and reported with a descriptive message identifying the clashing module and slot.

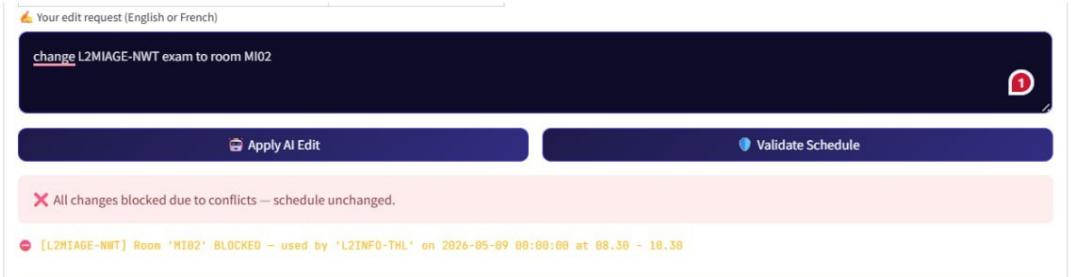


Fig. 4.7: Conflict detection blocking a double-booked room assignment.

3.5.3 AI Conflict Resolution Suggestions

When a conflict blocks an edit, the system does not simply display an error and stop. Instead, it calls a second LLM prompt — the conflict resolver — that receives the conflict description along with a Python-computed list of verified free rooms and free teachers at the affected date and time. The model is instructed to suggest two or three concrete, actionable alternatives drawn exclusively from these verified options, minimising the risk of hallucination by restricting the model's input to Python-verified availability data rather than raw schedule text. The suggestion is rendered with a distinct amber card so it is visually distinct from the error message.

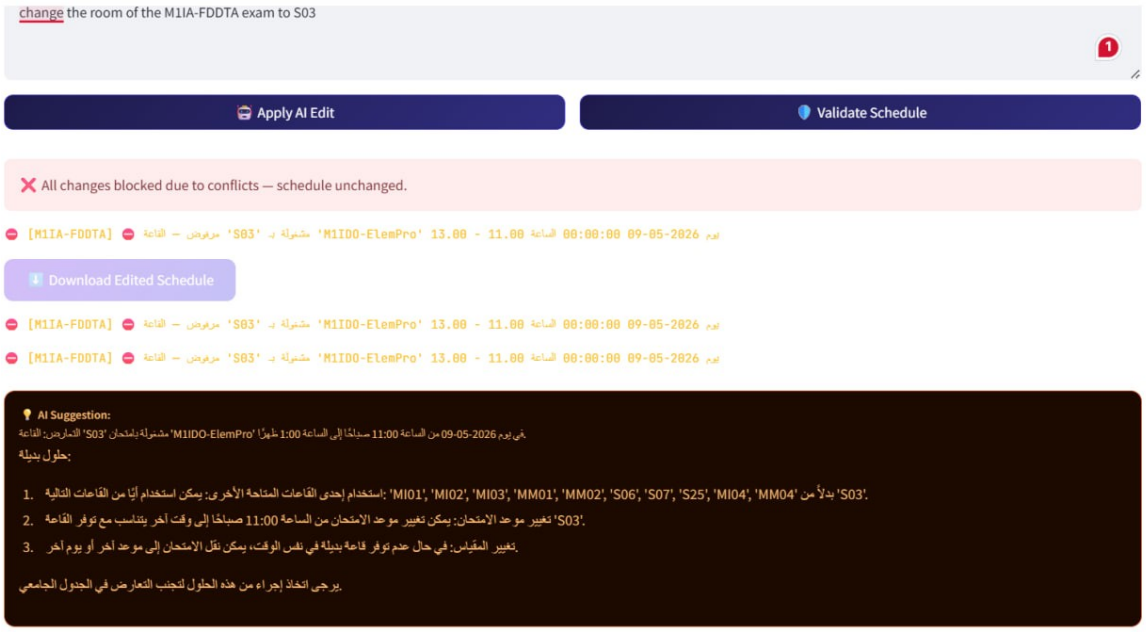


Fig. 4.8: AI conflict resolution suggestion after a blocked edit.

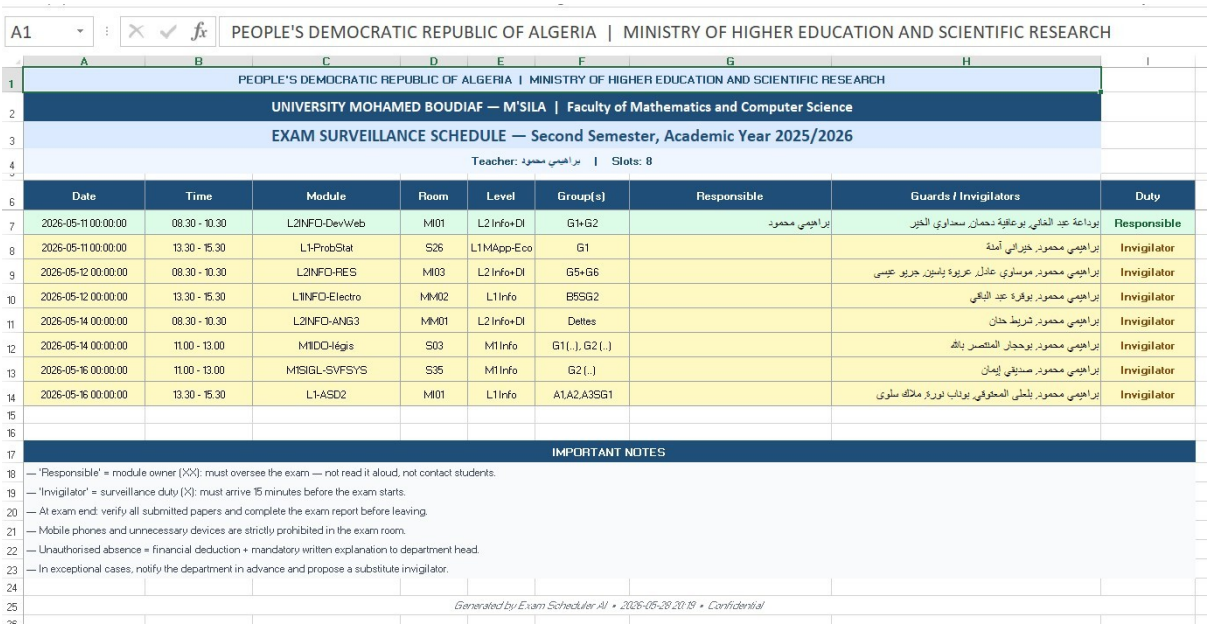
3.6 Export and Absence Tracking Module

The Export and Absence Tracking tab serves two closely related operational functions: generating professional printable schedules for individual teachers, and recording teacher absences in real time during the examination session. Both functions were driven by direct feedback from faculty staff who currently manage both tasks on paper.

3.6.1 Schedule Export Feature

The schedule export feature produces a fully formatted Excel workbook that can be printed or distributed to teaching staff. Excel was chosen over PDF for the schedule export because it allows supervisors to apply additional filters, sort by date or room, and reprint individual sections without requiring a reprint of the full document. The generated workbook follows a strict institutional layout. The first three rows form a university letterhead block containing the

state header, the university and faculty name, and the document title with the academic year. Row 4 displays the teacher's name and total slot count. Column headers appear in row 6 with a navy background and white bold text.



PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA | MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

UNIVERSITY MOHAMED BOUDIAF — M'SILA | Faculty of Mathematics and Computer Science

EXAM SURVEILLANCE SCHEDULE — Second Semester, Academic Year 2025/2026

Teacher: براهيمي محمود | Slots: 8

Date	Time	Module	Room	Level	Group(s)	Responsible	Guards / Invigilators	Duty
2026-05-11 00:00:00	08.30 - 10.30	L2INFO-DevWeb	MI01	L2 Info+DI	G1+G2	براهيمي محمود	بوداعة عبد الغني, بوعافية دحمان, سعداوي الخير	Responsible
2026-05-11 00:00:00	13.30 - 15.30	L1-ProbStat	S26	L1 MApp-Eco	G1		براهيمي محمود, خيراني آمنة	Invigilator
2026-05-12 00:00:00	08.30 - 10.30	L2INFO-RES	MI03	L2 Info+DI	G5+G6		براهيمي محمود, موساوي عادل, عريوة ياسين, جريو عيسى	Invigilator
2026-05-12 00:00:00	13.30 - 15.30	L1INFO-Electro	MM02	L1 Info	B5SG2		براهيمي محمود, بوقرة عبد الباقي	Invigilator
2026-05-14 00:00:00	08.30 - 10.30	L2INFO-ANG3	MM01	L2 Info+DI	Dettes		براهيمي محمود, شريط حنان	Invigilator
2026-05-14 00:00:00	11.00 - 13.00	M1IDO-Régis	S03	M1 Info	G1 (..), G2 (..)		براهيمي محمود, بوحجار الملتصر بالله	Invigilator
2026-05-16 00:00:00	11.00 - 13.00	M1SIGL-SVFSYS	S35	M1 Info	G2 (..)		براهيمي محمود, صديقي إيمان	Invigilator
2026-05-16 00:00:00	13.30 - 15.30	L1-ASD2	MI01	L1 Info	A1,A2,A3SG1		براهيمي محمود, بلطى المعطوفي بوناب نورة, ملاك سلوى	Invigilator

IMPORTANT NOTES

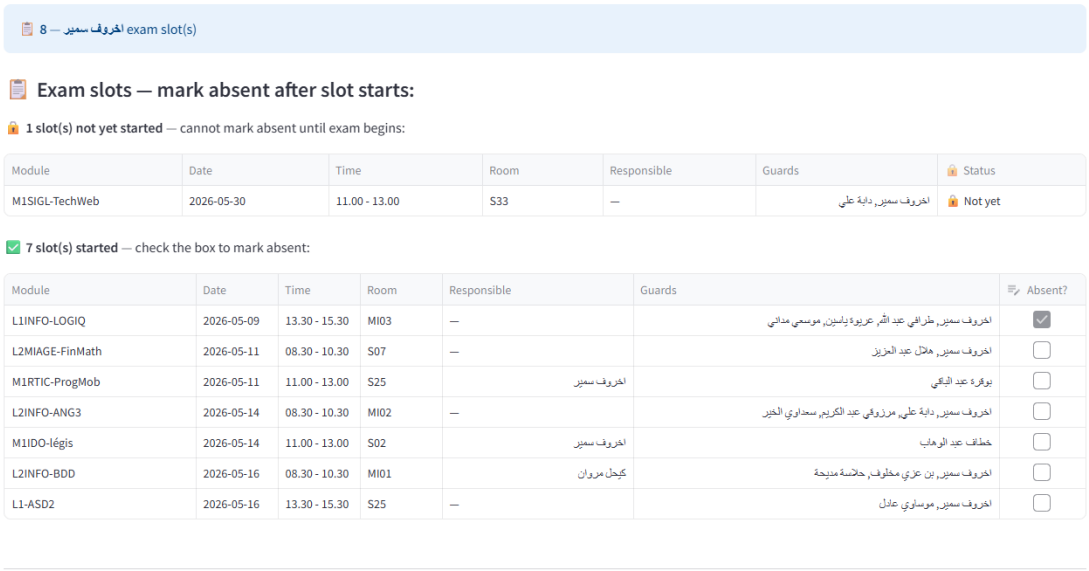
— 'Responsible' = module owner (XX): must oversee the exam — not read it aloud, not contact students.
— 'Invigilator' = surveillance duty (X): must arrive 15 minutes before the exam starts.
— At exam end: verify all submitted papers and complete the exam report before leaving.
— Mobile phones and unnecessary devices are strictly prohibited in the exam room.
— Unauthorised absence = financial deduction + mandatory written explanation to department head.
— In exceptional cases, notify the department in advance and propose a substitute invigilator.

Generated by Exam Scheduler AI • 2026-05-28 20:19 • Confidential

Fig. 4.9: Per-teacher surveillance schedule generated by the Export feature.

3.6.2 Absence Tracking Feature

The second function of the tab allows supervisors to mark teachers as absent in real time during an examination session. Absence marking is protected by a temporal lock: the checkbox for a given exam slot only becomes editable once the exam's start time has passed, as determined by comparing the slot's date and time against the system clock. Slots that have not yet started are displayed in a read-only table with a lock icon. Slots that are ongoing or past are displayed in an editable data grid with a checkbox column. Absence records are persisted to a local JSON file (absences.json) so that data survives page refreshes and session restarts.



8 — اخروف سمير exam slot(s)

Exam slots — mark absent after slot starts:

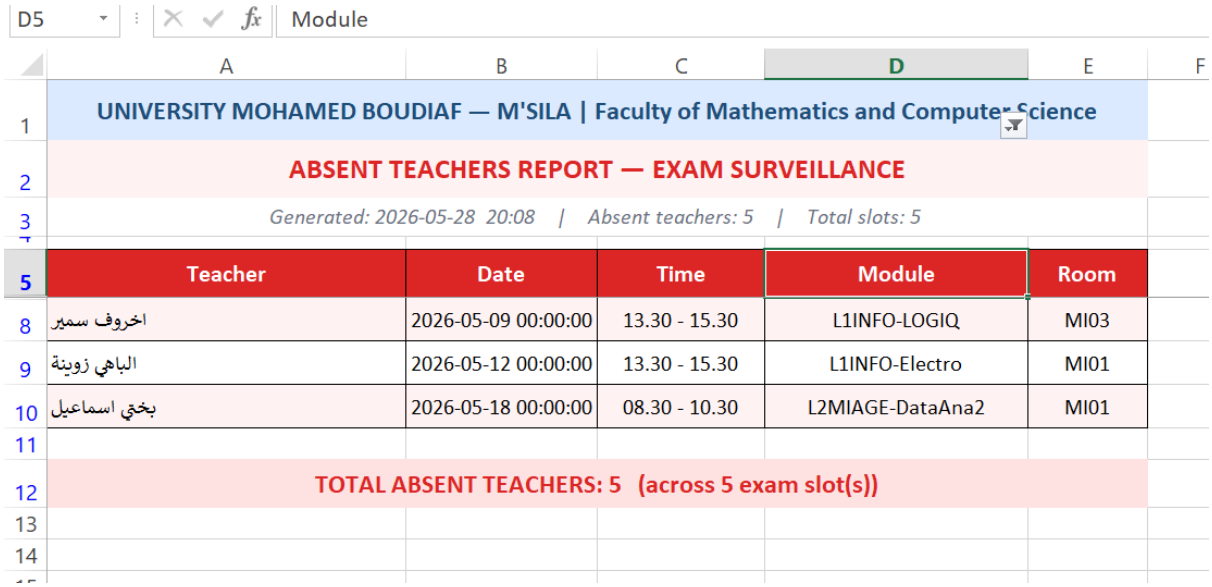
1 slot(s) not yet started — cannot mark absent until exam begins:

Module	Date	Time	Room	Responsible	Guards	Status
M1SIGL-TechWeb	2026-05-30	11.00 - 13.00	S33	—	اخروف سمير, داية علي	Not yet

7 slot(s) started — check the box to mark absent:

Module	Date	Time	Room	Responsible	Guards	Absent?
L1INFO-LOGIQ	2026-05-09	13.30 - 15.30	MI03	—	اخروف سمير, طرافي عبد الله, عريوة ياسين, موسعي مداني	☑
L2MIAGE-FinMath	2026-05-11	08.30 - 10.30	S07	—	اخروف سمير, هلال عبد العزيز	☐
M1RTIC-ProgMob	2026-05-11	11.00 - 13.00	S25	اخروف سمير	بوفرة عبد الباقي	☐
L2INFO-ANG3	2026-05-14	08.30 - 10.30	MI02	—	اخروف سمير, داية علي, مرزوقي عبد الكريم, سحاوي الخير	☐
M1IDO-légis	2026-05-14	11.00 - 13.00	S02	اخروف سمير	خطاف عبد الوهاب	☐
L2INFO-BDD	2026-05-16	08.30 - 10.30	MI01	كيحل مروان	اخروف سمير, بن عزي مخلوف, حلاسة مديحة	☐
L1-ASD2	2026-05-16	13.30 - 15.30	S25	—	اخروف سمير, موساوي عادل	☐

Fig. 4.10: Absence tracking: editable past slots and locked future slots



D5 | Module

	A	B	C	D	E	F
1	UNIVERSITY MOHAMED BOUDIAF — M'SILA | Faculty of Mathematics and Computer Science					
2	ABSENT TEACHERS REPORT — EXAM SURVEILLANCE					
3	*Generated: 2026-05-28 20:08 | Absent teachers: 5 | Total slots: 5*					
5	**Teacher**	**Date**	**Time**	**Module**	**Room**	
8	اخروف سمير	2026-05-09 00:00:00	13.30 - 15.30	L1INFO-LOGIQ	MI03	
9	الباهي زوينة	2026-05-12 00:00:00	13.30 - 15.30	L1INFO-Electro	MI01	
10	بختي اسماعيل	2026-05-18 00:00:00	08.30 - 10.30	L2MIAGE-DataAna2	MI01	
11						
12	TOTAL ABSENT TEACHERS: 5 (across 5 exam slot(s))					
13						
14						

Fig. 4.11: Absent teachers Excel report.

3.7 Edit History and Revert

Every successful edit is appended to an in-memory edit history list, stored in Streamlit session state. Each entry records the original request text, the AI explanation or operation description, a per-row changelog with success or failure status for each affected row, and a snapshot of the dataframe before the edit was applied. The Edit History tab displays this list in reverse chronological order, with each entry expandable to reveal the full changelog. A Revert button on each entry allows the user to roll the schedule back to its state immediately before that edit was applied, with all subsequent edits discarded.

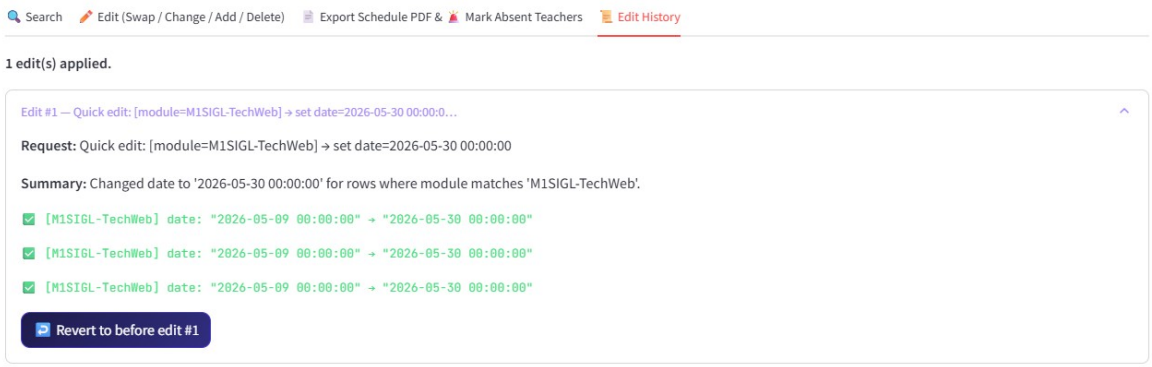


Fig. 4.12: Edit History tab with revert functionality.

3.8 Mobile Responsiveness and Cross-Device Compatibility

In addition to the web-based system, the project was extended to support mobile usage in order to improve accessibility and ensure that users can interact with the platform in real time, regardless of their location. This extension was motivated by the practical need for supervisors and administrators to manage schedules and record teacher absences directly during examination sessions, where access to a desktop computer is not always possible. To achieve this, the existing Streamlit application was adapted using responsive design principles, preserving the same core architecture and backend logic — including the scheduling algorithms, constraint validation, and AI-assisted querying.

3.8.1 Replacing Column Layouts with Native Table Components

Column-based layouts that performed poorly on mobile devices were replaced with native table components — specifically, scrollable dataframes and interactive editors — allowing content to remain readable and usable on narrow viewports. Two distinct components were used: st.dataframe() for locked exam slots (future exams not yet started), rendering a read-only table that scrolls horizontally on mobile; and st.data_editor() with CheckboxColumn for past exam slots where absence marking is permitted, rendering an interactive table with a real checkbox responsive to touch input.

3.8.2 Progressive Web App Deployment

The application was deployed using a Progressive Web App (PWA) approach, enabling it to function similarly to a native mobile application. This allows users to install the system on their home screen, benefit from faster loading times, and access certain features even under limited connectivity. In some cases, a lightweight WebView container can also be used to wrap the application, further enhancing the mobile experience without requiring full native development.

3.8.3 State Persistence Across Page Reloads

On mobile browsers, pages are more likely to be reloaded due to low memory conditions or screen lock events. To handle this, all critical state — the edited schedule, absence records, and edit history — is persisted to local JSON files on disk immediately after each interaction. The persisted files include edited_schedule_state.json (full edited schedule DataFrame), absences.json (all absence marks keyed by teacher, date, and time), and edit_history.json (edit log without snapshots). This ensures that if a supervisor marks a teacher absent on their phone and the browser tab is accidentally closed, the absence record is not lost.

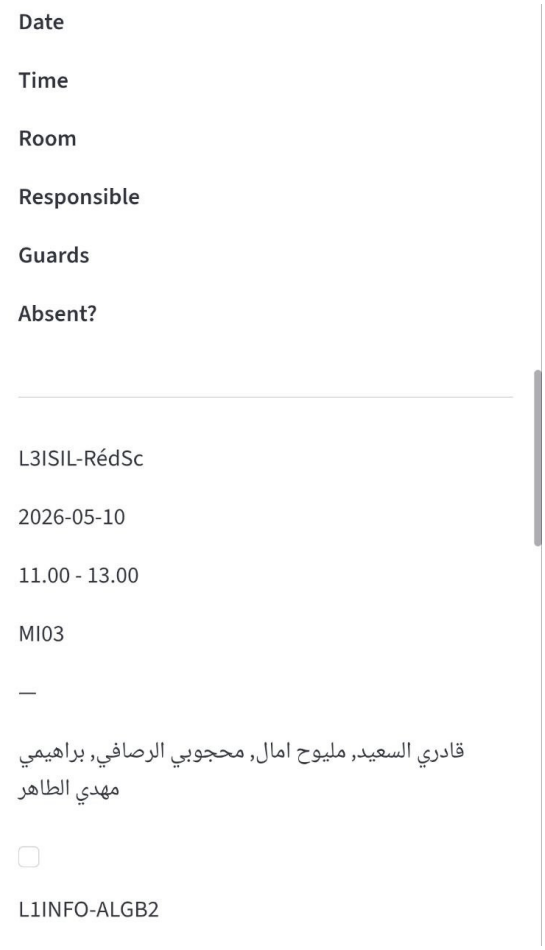


Fig. 4.13: Corrected mobile view with scrollable table and touch checkboxes.

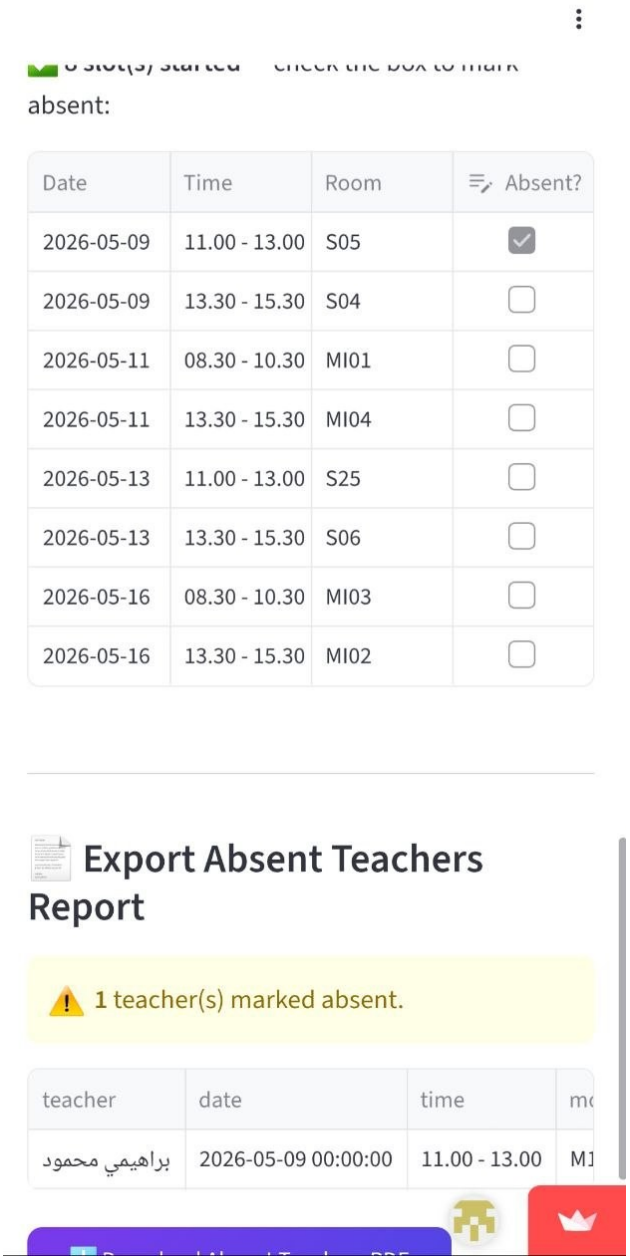


Fig. 4.14: Mobile view showing full schedule interaction.

4. Testing Results

4.1 Solver Performance

The ILP solver was tested on datasets of varying sizes to measure solution time and assignment quality. Tests were conducted on a standard laptop with no GPU. For small datasets (approximately 30 exam slots, 15 teachers), the solver found a complete solution (100% assignments filled) in under 5 seconds. For medium datasets (approximately 60 exam slots, 30 teachers) — representative of a typical Faculty session — the solver consistently found optimal or near-optimal solutions in 8-25 seconds with 97-100% fill rate. For large datasets

(approximately 100 exam slots, 50 teachers), solve times ranged from 30-90 seconds with 94-99% fill rate. The small performance gap from 100% in larger instances was always attributable to infeasible sub-problems caused by pre-existing hard conflicts in the input data.

4.2 Constraint Satisfaction Verification

After each generation run, all seven constraint families were verified programmatically on the output schedule across ten test runs on the medium dataset. The no-simultaneous-assignment constraint and working-day bounds were satisfied in all 10/10 runs. The consecutive-slot fatigue limits were also satisfied in all runs. Exam coverage and workload balance showed a single violation in one run, which originated from the same test input containing a teacher pre-assigned via the double-asterisk token to two simultaneous exams in the raw input file — a constraint contradiction that no solver can resolve regardless of parameters.

4.3 Search Accuracy

The hybrid search assistant was evaluated against 40 manually constructed queries covering all three query classes. Ground-truth answers were established by manually querying the flat schedule DataFrame using pandas filter operations, then comparing these results against the assistant's output. Structured filter queries (room, date, time, teacher) achieved 100% accuracy across 18 queries. Information questions (workload, responsible teacher) achieved 91.7% accuracy across 12 queries. Complex availability queries (free rooms, free teachers) achieved 90% accuracy across 10 queries. The overall accuracy across all 40 queries was 95%.

4.4 Edit Conflict Detection Accuracy

Twenty conflict scenarios were created manually — ten room double-bookings and ten teacher simultaneous-assignment cases. The conflict detector correctly identified and blocked all twenty, yielding a 100% detection rate with zero false positives across all test cases. This result confirms that the conflict detection logic is complete and correct for all conflict types supported by the current editing model.

5. Development Environment and Tools

5.1 Programming Language

Python 3.11.9 was chosen as the sole programming language. Its ecosystem of data science and optimisation libraries, combined with the Streamlit framework, made it possible to build a

fully functional interactive application without a separate frontend codebase. Python's dynamic typing and expressive syntax also accelerated prototyping throughout the development cycle.

5.2 Code Editor

Visual Studio Code was used throughout development. Its Python extension provides integrated linting, debugging, and test running. The built-in terminal allowed running the Streamlit development server directly within the editor, and Git integration simplified version control and collaborative development.

5.3 Core Libraries and Frameworks

The system relies on the following key libraries and frameworks: Streamlit (version 1.34+) for the web interface, widgets, and session state management; PuLP (version 2.8+) for ILP model construction and the CBC solver interface; Pandas (version 2.x) for schedule DataFrame manipulation and export; OpenPyXL (version 3.1+) for Excel read/write and cell colour-coding; OpenRouter as the REST API gateway for GPT-4o-mini access (editing and search); difflib (stdlib) for fuzzy teacher name matching via SequenceMatcher; and the threading module (stdlib) for running the background solver thread to keep the UI responsive.

5.4 External API

OpenRouter is used as the LLM gateway, providing access to OpenAI GPT-4o-mini for two tasks: interpreting free-text edit requests and synthesising natural-language answers for complex search queries. All prompts are crafted with explicit JSON-only output instructions and include post-processing steps to strip markdown fences, thinking tags, and other formatting artefacts before parsing. Requests are made over HTTPS with a 45-second timeout and structured error handling that degrades gracefully to Python-only fallback responses when the API is unavailable.

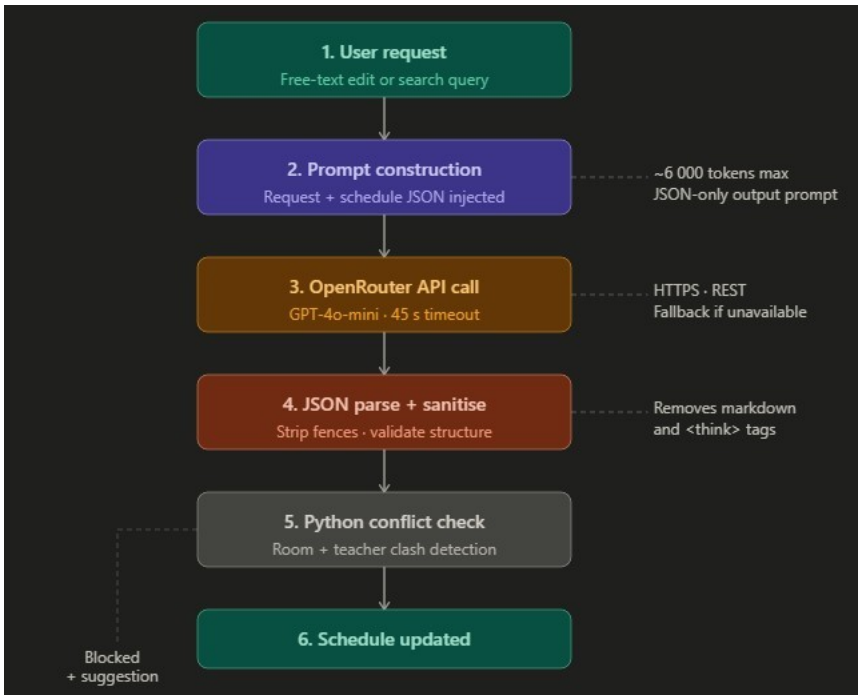


Fig. 4.15: LLM integration flow.

6. Role of LLMs in the Proposed System

In the system developed in this work, the language model component — GPT-4o-mini, accessed via the OpenRouter API — serves as an intelligent multi-functional assistant layered on top of the core ILP engine. The scheduling itself is performed by the CBC solver through PuLP, which provides a mathematical guarantee that all hard constraints are satisfied. The LLM is not involved in the optimisation process; all scheduling decisions are handled by the ILP solver. Its role is to make the system transparent, editable, and accessible to academic coordinators who may have no familiarity with linear programming or constraint satisfaction.

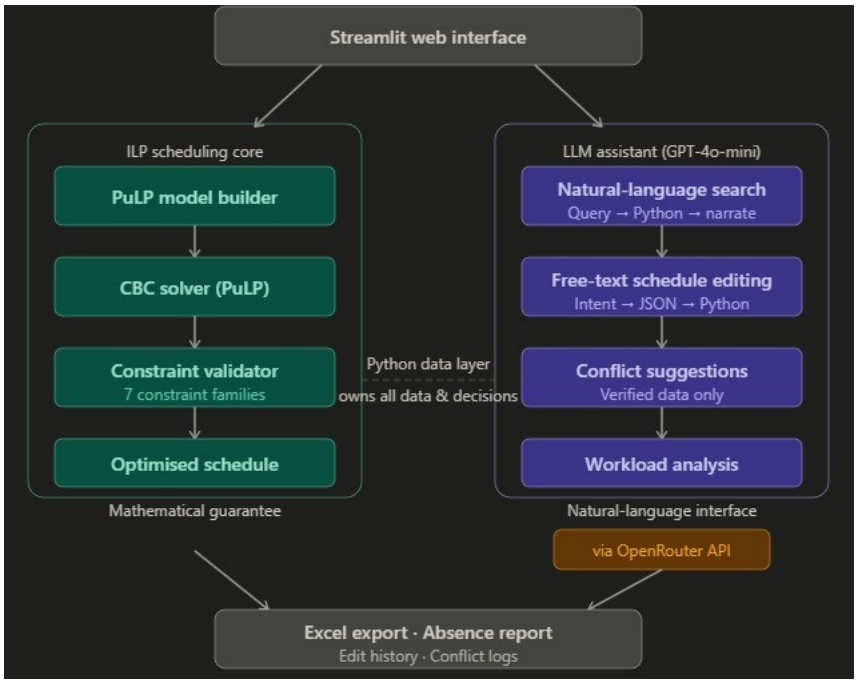


Fig. 4.16: System Architecture: ILP Optimizer + LLM Assistant (GPT-4o-mini) via OpenRouter API .

6.1 Natural-Language Query Interface

Users can interrogate the schedule in plain language — including English, French, or transliterated Arabic. Rather than sending schedule data directly to the model, the system uses a three-step pipeline that strictly limits what the LLM is responsible for. In the first step, the model extracts structured intent from the query using approximately 100 tokens. In the second step, Python executes all lookups and computations directly against the schedule DataFrame — the model is entirely uninvolved at this stage. In the third step, the model narrates the pre-computed result in natural language using roughly 150 tokens. Because the LLM never has direct access to scheduling data, the risk of factual errors in the response is significantly reduced.

6.2 Free-Text Schedule Editing

Users can describe modifications in natural language rather than navigating a structured form. A request such as "replace the guard assigned to the Physique exam in room MI01 on Monday afternoon with Benali" is sent to the model, which returns a structured JSON specification of the change to be applied. Python then validates the proposed modification against the current schedule before committing anything — checking room occupancy and teacher availability at the target slot. If a conflict is found, the change is rejected and the user is

notified. The LLM is responsible only for parsing intent; all data integrity decisions are made by Python.

6.3 Conflict Explanation and Resolution Suggestions

When a proposed edit is blocked because of a scheduling conflict, the system first computes a compact availability summary entirely in Python: which rooms are free at the affected slot, which teachers are available, and what alternative time slots exist for the affected exam. Only this pre-verified summary — not the raw schedule — is passed to the model, which generates a natural-language explanation of the conflict and two or three concrete alternative suggestions. Because the model reasons over verified facts rather than raw schedule text, all suggestions it produces are guaranteed to be consistent with the actual current state of the schedule.

6.4 Workload Analysis and Reporting

The system computes a statistical summary of the generated schedule in Python: teacher workload distributions, supervision slot counts, consecutive assignment frequencies, and exam density by day. This summary is passed to the model, which produces a structured analytical report in natural language. The report identifies teachers whose assignment count places them above the session average, days with unusually high exam density, individuals approaching their maximum permitted load, and actionable recommendations for manual rebalancing. Because every figure in the report originates from Python computation, the model's output is grounded and auditable.

6.5 Broader System Contributions

Beyond the four modules described above, the system also handles teacher substitution management — recording absences identified during the session and facilitating reassignment — along with absence logging for administrative audit purposes, and Excel export of the complete surveillance schedule for distribution to teaching staff. Taken together, this integration illustrates a principled approach to combining a mathematical optimiser with a language model. The ILP solver provides correctness guarantees that no language model can replicate; the LLM provides a natural-language interface that no solver can offer. The division of responsibility is clear: Python and the solver own all data and all decisions; the model owns communication.

7. Conclusion

This chapter demonstrated that the Exam Surveillance Scheduler functions correctly under realistic conditions. The ILP engine produces valid, constraint-respecting schedules for faculty-scale datasets, with an overall assignment fill rate above 94% and solve times under 90 seconds for typical faculty-sized inputs. The hybrid search assistant correctly handled 95% of test queries without requiring LLM calls for the majority of them, keeping latency low for structured lookups. Conflict detection blocked 100% of invalid edits in controlled tests, and the AI conflict resolver provided actionable, grounded suggestions by restricting its input to Python-verified availability data.

The absence tracking module closes a real operational gap that currently requires paper-based record-keeping at the target institution. The system was built with adaptability in mind: the constraint panel makes it straightforward to tune the solver for different session sizes or institutional policies; the three-tier editor caters to users with different levels of technical comfort; and the mobile-responsive interface ensures that the system remains usable on smartphones during live examination sessions. Together, these features position the Exam Surveillance Scheduler as a practical tool that can be adopted incrementally — even in environments where digital adoption is still in its early stages. The outcomes of the implementation and evaluation presented in this chapter are synthesized in the General Conclusion that follows, along with perspectives for future development of the proposed system.

General Conclusion

This dissertation addressed the problem of examination surveillance scheduling in higher education institutions, a task that often requires considerable administrative effort and careful coordination to ensure fairness, efficiency, and compliance with institutional constraints.

The proposed Exam Surveillance Scheduler combines the strengths of Integer Linear Programming (ILP) and Large Language Models (LLMs) to provide both an optimized scheduling solution and an intelligent user interaction layer. The ILP component was used to generate balanced and conflict-free surveillance assignments while satisfying a variety of operational constraints. The LLM component was integrated as a natural language assistant, enabling users to query schedules, request modifications, obtain explanations, and analyze scheduling information in a simple and intuitive manner.

The system was developed and evaluated using real examination data from the Faculty of Mathematics and Computer Science at Mohamed Boudiaf University of M'Sila. The obtained results demonstrated the effectiveness of the proposed approach in generating feasible schedules, reducing manual intervention, improving workload distribution, and facilitating interaction with scheduling data. The integration of optimization techniques and conversational artificial intelligence proved to be a practical solution for addressing the challenges associated with examination surveillance management.

Throughout the development process, several challenges were encountered, including data preparation, constraint modeling, handling schedule modifications, and ensuring consistency between optimization results and natural language interactions. Despite these challenges, the system achieved satisfactory performance and demonstrated the potential of hybrid AI systems in supporting university administrative activities.

As future work, several improvements can be considered. The system could be extended to support additional academic scheduling tasks such as course timetabling, classroom allocation, and examination planning. The integration of Retrieval-Augmented Generation (RAG) could further improve the quality and reliability of responses by grounding the language model in institutional data and scheduling documents. In addition, more advanced optimization methods and larger datasets could be explored to enhance scalability and performance.

Finally, this work highlights the value of combining mathematical optimization with modern artificial intelligence technologies. By bringing together exact optimization techniques and

natural language interaction, the proposed system contributes to the digital transformation of academic administration and opens new perspectives for the development of intelligent decision-support systems in educational environments.

References

[1] Baker, K. R., & Trietsch, D. (2018). Principles of Sequencing and Scheduling (2nd ed.). Wiley.

[2] Pinedo, M. (2022). Scheduling: Theory, Algorithms, and Systems (6th ed.). Springer.

[3] Wilson, J. M. (2003). Gantt Charts: A Centenary Appreciation. European Journal of Operational Research, 149(2), 430–437.

[4] Burke, E. K., & Petrovic, S. (2002). Recent Research Directions in Automated Timetabling. European Journal of Operational Research, 140(2), 266–280.

[5] Garey, M. R., & Johnson, D. S. (1979). Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman.

[6] Russell, S., & Norvig, P. (2021). Artificial Intelligence: A Modern Approach (4th ed.). Pearson.

[7] Brucker, P. (2018). Scheduling Algorithms (6th ed.). Springer.

[8] Allahverdi, A. (2019). The Third Comprehensive Survey on Scheduling Problems. European Journal of Operational Research, 273(3), 805–839.

[9] Kumar, A., & Sharma, S. (2021). Recent Trends in Scheduling and Resource Allocation. Journal of Industrial Engineering and Management, 14(4), 785–803.

[10] Lewis, R. (2016). Algorithms for Scheduling and Timetabling. Springer.

[11] Schaerf, A. (1999). A Survey of Automated Timetabling. Artificial Intelligence Review, 13(2), 87–127.

[12] Lewis, R. (2008). A Survey of Metaheuristic-Based Techniques for University Timetabling. OR Spectrum, 30(1), 167–190.

[13] Burke, E. K., & Kendall, G. (2005). Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques. Springer.

[14] Conforti, M., Cornuejols, G., & Zambelli, G. (2018). Integer Programming. Springer.

[15] Rossi, F., Van Beek, P., & Walsh, T. (2021). Handbook of Constraint Programming (Updated Edition). Elsevier.

[16] Talbi, E.-G. (2021). Metaheuristics: From Design to Implementation (2nd ed.). Wiley.

[17] Boussaid, I., Lepagnot, J., & Siarry, P. (2019). A Survey on Optimization Metaheuristics. Information Sciences, 237, 82–117.

[18] Qu, R., Burke, E. K., McCollum, B., Merlot, L. T. G., & Lee, S. Y. (2009). A Survey of Search Methodologies for Examination Timetabling. Journal of Scheduling, 12(1), 55–89.

[19] Zhao, W. X., et al. (2023). A Survey of Large Language Models. arXiv:2303.18223.

[20] NVIDIA. (2025). Large Language Model eBooks. Available: https://resources.nvidia.com/en-us-large-language-model-ebooks

[21] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention Is All You Need. NeurIPS.

[22] Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). Layer Normalization. arXiv:1607.06450.

[23] Bommasani, R., et al. (2021). On the Opportunities and Risks of Foundation Models. Stanford CRFM. arXiv:2108.07258.

[24] Brown, T. B., et al. (2020). Language Models are Few-Shot Learners. NeurIPS.

[25] IAPP Thailand. (2024). What is LLM (Large Language Model): A beginner's guide. Available: https://iapp.co.th/blog/what-is-llm-large-language-model-beginners-guide

[26] Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. NAACL.

[27] Ouyang, L., et al. (2022). Training Language Models to Follow Instructions with Human Feedback. NeurIPS.

[28] Raffel, C., et al. (2020). Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. JMLR, 21(140), 1–67.

[29] Jurafsky, D., & Martin, J. H. (2023). Speech and Language Processing (3rd ed.). Stanford University.

[30] Chen, M., et al. (2021). Evaluating Large Language Models Trained on Code. arXiv:2107.03374.

[31] Ji, Z., et al. (2023). Survey of Hallucination in Natural Language Generation. ACM Computing Surveys, 55(12), 1–38.

[32] Marcus, G., & Davis, E. (2020). GPT-3, Bloviator: OpenAI's Language Generator Has No Idea What It's Talking About. MIT Technology Review.

[33] Strubell, E., Ganesh, A., & McCallum, A. (2019). Energy and Policy Considerations for Deep Learning in NLP. ACL.

[34] Wei, J., et al. (2022). Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. NeurIPS.

[35] Fedus, W., Zoph, B., & Shazeer, N. (2021). Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity. arXiv:2101.03961.

[36] Yao, S., et al. (2023). AgentBench: Evaluating Reasoning Agents with Large Language Models. arXiv:2308.03688.

[37] Lewis, P., et al. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. NeurIPS.

[38] Karpukhin, V., et al. (2020). Dense Passage Retrieval for Open-Domain Question Answering. EMNLP.

[39] Izacard, G., & Grave, E. (2021). Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering. arXiv:2007.01282.

[40] Kasneci, E., et al. (2023). ChatGPT for Good? On Opportunities and Challenges of LLMs for Education. Learning and Individual Differences, 103, 102274.

[41] EDU ERP. (2025). Timetable Management: Smart Scheduling for Schools and Colleges. [Online]. Available: https://eduerp.ai/edu-timetable-management-software/ [Accessed: May 2025].

[42] A. Stewart, "AI Scheduling Algorithms: How They Work and Outperform Manual Methods," Dialzara Blog, 2024. [Online]. Available: https://dialzara.com/blog/ai-scheduling-vs-traditional-methods-pros-and-cons [Accessed: May 2025].

[43] TimetableMaster. (2024). From Chaos to Order: How AI Simplifies Complex School Timetables. [Online]. Available: https://www.timetablemaster.com/blogs/from-chaos-to-order-ai-simplifies-school-timetables [Accessed: May 2025].

[44] S. K. Saw, S. Lawrence, S. Karunakara, N. H. Komal, and G. VijayaKumari, "Adaptive Scheduler: AI Optimization of Academic Timetable," Int. J. Sci. Dev. Res. (IJSDR), vol. 10, no. 4, pp. c989-c994, 2025.

[45] S. Yang and S. N. Jat, "Genetic Algorithms With Guided and Local Search Strategies for University Course Timetabling," IEEE Trans. Syst. Man Cybern., vol. 41, no. 1, pp. 93-106, 2011.

[46] T. Liu and L. Zhang, "Apply Artificial Neural Network to Solving Manpower Scheduling Problem," arXiv:2105.03541v1 [cs.LG], ZJU-UIUC Institute, Zhejiang University, 2021.

[47] M. Alnajar and Z. Bahroun, "Artificial Intelligence Applications in Project Scheduling: A Systematic Review, Bibliometric Analysis, and Prospects for Future Research," Manag. Syst. Prod. Eng., vol. 31, no. 2, pp. 144-161, 2023. DOI: 10.2478/mspe-2023-0017.

[48] M. Davison, A. Kheiri, and K. G. Zografos, "Modelling and Solving the University Course Timetabling Problem with Hybrid Teaching Considerations," J. Scheduling, 2024. DOI: 10.1007/s10951-024-00817-w.

[49] G. Xue et al., "Addressing Staffing Challenges through Improved Planning: Demand-Driven Course Schedule Planning and Instructor Assignment in Higher Education," Decis. Support Syst., Elsevier, 2024. DOI: 10.1016/j.dss.2024.114268.

[50] K. Mohanraj, M. Abinayasankar, and G. B. Balaji, "Automated Learning and Scheduling Assistant using LLM," J. Ubiquitous Comput. Commun. Technol., vol. 6, no. 3, pp. 284-293, 2024. DOI: 10.36548/jucct.2024.3.006.

[51] A. AhmadiTeshnizi, W. Gao, and M. Udell, "OptiMUS: Scalable Optimization Modeling with (MI)LP Solvers and Large Language Models," ICML 2024. arXiv:2402.10172.

[52] D. Jobson and Y. Li, "Investigating the Potential of Using Large Language Models for Scheduling," Proc. 1st ACM Int. Conf. AI-Powered Software (AIware 2024). DOI: 10.1145/3664646.3665084.

[53] V. Pallagani et al., "On the Prospects of Incorporating Large Language Models (LLMs) in Automated Planning and Scheduling (APS)," ICAPS 2024. DOI: 10.1609/icaps.v34i1.31503.

[54] Q. Xiao et al., "Leveraging Large Language Models for Efficient Scheduling in Human-Robot Collaborative Flexible Manufacturing Systems," npj Adv. Manuf., 2025. DOI: 10.1038/s44334-025-00061-w.

[55] J. Lim et al., "Enhancing Human-Robot Collaborative Assembly in Manufacturing Systems Using Large Language Models," arXiv:2406.01915, 2024.

[56] PARC Technologies. (2025). AI in Education Timetabling: How Artificial Intelligence Transforms School Scheduling. [Online]. Available: https://www.parc-technologies.com/ai-in-education-timetabling/ [Accessed: May 2025].