Scientific Research Publishing

Open Journal of Applied Sciences, 2025, 15(8), 2277-2290
https://www.scirp.org/journal/ojapps
ISSN Online: 2165-3925
ISSN Print: 2165-3917

Explicit Artificial Intelligence Timetable Generator for Colleges and Universities

Lawrence A. Farinola, Mahougnon B. M. Assogba

Department of Software Engineering, Faculty of Architecture and Engineering, Rauf Denktas University, Mersin, Türkiye
Email: lawrence.farinola@rdu.edu.tr, montfortassogba2@gmail.com

How to cite this paper: Farinola, L.A. and Assogba, M.B.M. (2025) Explicit Artificial Intelligence Timetable Generator for Colleges and Universities. *Open Journal of Applied Sciences*, **15**, 2277-2290.
https://doi.org/10.4236/ojapps.2025.158151

Received: July 19, 2025
Accepted: August 12, 2025
Published: August 15, 2025

Abstract

Manual timetable preparation in colleges and universities is often time-consuming, error-prone, and inefficient, especially with increasing student and course complexity. This paper proposes an Explicit Artificial Intelligence-based Timetable Generator that leverages a Genetic Algorithm to automatically generate conflict-free, optimized academic schedules. The system incorporates user-defined constraints and priorities such as room availability, lecturer-course assignments, and course clash avoidance. Extensive testing was conducted using real-world institutional data to evaluate the system's effectiveness. The proposed approach significantly reduced scheduling time, eliminated course conflicts, and improved allocation efficiency across multiple views, including per lecturer, per student, and per classroom. The results demonstrate the system's scalability and practical viability for higher education environments.

Keywords

Genetic Algorithm, AI Timetabling, Automatic Scheduling, Constraint Satisfaction, Optimization

1. Introduction

Timetable construction is a fundamental administrative task in educational institutions, aimed at efficiently assigning courses, lecturers, and students to specific time slots and available resources. However, this task becomes significantly complex due to numerous constraints such as lecturer preferences, classroom availability, and avoidance of scheduling conflicts among courses. Manually creating such timetables is not only time-consuming and labor-intensive but also prone to human error, often leading to course overlaps, room double-bookings, or inefficient allocations.

The timetable problem is formally recognized as an NP-complete problem [1], indicating that no known algorithm can solve all instances optimally in polynomial time. This computational difficulty has motivated researchers to explore algorithmic approaches and optimization techniques. Over the years, a wide array of strategies has been developed to automate timetable generation, including graph coloring algorithms [2], neural networks [3], simulated annealing [4], tabu search [5], and constraint-based programming [6]. Despite their effectiveness in specific scenarios, many of these approaches struggle with incorporating complex institutional constraints and achieving generality across diverse academic structures.

Genetic Algorithms (GAs) have emerged as one of the most promising methods for solving timetabling problems due to their flexibility, robustness, and ability to handle large search spaces. GAs are inspired by the principles of natural selection and have been successfully applied to university course scheduling problems with promising results [7]-[11]. For instance, Srinivasan *et al.* [8] applied an evolutionary algorithm with multiple context reasoning to address constrained university module scheduling. Similarly, Barkha *et al.* [9] and Narang *et al.* [10] introduced active rule-based GA systems capable of handling faculty-specific constraints and classroom allocations efficiently. Ladipo *et al.* [12] demonstrated how a GA-based system can adapt to changing scheduling rules while minimizing course conflicts.

Recent literature surveys and benchmarks emphasize the need for generalizable, user-friendly, and scalable systems. Ceschia and Schaerf [12], for example, provided a detailed classification of educational timetabling problems and proposed benchmarks that accommodate both hard and soft constraints. Mrunmayee *et al.* [13] and Tan *et al.* [14] also surveyed optimization methodologies, identifying hybrid metaheuristics and constraint decomposition as key trends in modern timetable generation systems. More recently, Pounikar *et al.* [15] and the GECCO conference series [16]-[18] have discussed frameworks that combine classical optimization with evolutionary computing to achieve better performance, adaptability, and reduced execution times.

Despite these advances, several challenges remain. Many systems are designed for specific institutions, requiring custom configurations and manual data handling. Some lack user-friendly interfaces or fail to account for dynamic priority settings, such as lecturer availability changes or evolving student enrollment data. Thus, there is a clear need for a generic, explicit, AI-driven timetable generation system that can support flexible constraints, integrate administrator priorities, and deliver multi-view outputs suitable for students, lecturers, and departments.

This research addresses these needs by proposing an Explicit Artificial Intelligence Timetable Generator built upon a genetic algorithm. The system is designed to:

- Automatically generate conflict-free schedules across multiple constraints and user-defined priorities;
- Accommodate real-world institutional settings including departments, pro-

grams, and resource allocations;

- Provide a user-friendly interface for administrators to dynamically input, modify, and manage scheduling data;
- Deliver timetables in multiple views (by course, lecturer, classroom, or student group);
- Ensure scalable and adaptable solutions across different educational institutions.

By combining evolutionary optimization with explicit user control, this research aims to demonstrate how automated timetable systems can achieve both efficiency and institutional flexibility.

2. Research Methodology

This research adopts a Genetic Algorithm (GA) to develop an automatic and constraint-aware course timetable generator for academic institutions. The GA was selected due to its strong performance in solving NP-complete optimization problems such as timetable scheduling [7] [8] [11]. Unlike deterministic methods that struggle with high-dimensional constraint spaces, GAs can efficiently explore large search spaces and find near-optimal solutions within a reasonable time.

GA's suitability for this task is grounded in its ability to:

- Evolve feasible schedules over generations through crossover and mutation,
- Avoid local optima using population diversity,
- Handle multiple hard and soft constraints effectively,
- And allow priority-based configurations during the evolution process.

Prior literature has also demonstrated the efficacy of GAs in similar domains, showing their adaptability to real-world constraints like classroom availability, faculty preferences, and course overlaps [9]-[11].

2.1. Soft Constraints Used in Building the Genetic Algorithm

The GA in this study was implemented to satisfy the following soft constraints:

- Each lecture must be assigned to only one classroom or laboratory.
- No classroom or lab is assigned more than one lecture at a given time.

These constraints were incorporated in the fitness function, which penalizes conflicts and rewards feasible, balanced solutions.

2.2. System Implementation

The system was implemented using a three-tier architecture, with front-end user interfaces, server-side logic, and a relational database. The hardware and software requirements are modest, ensuring accessibility and ease of deployment in low-resource environments.

Hardware Requirements:

- Minimum of 500 MB hard disk space,
- Pentium II Intel (R) Celeron (R) N4000 CPU or higher,
- At least 125 MB RAM.

Software Stack:

- Operating System: Windows Vista or later,
- Local Server: XAMPP (includes Apache, MySQL, PHP),
- Web Browsers: Google Chrome, UC Browser, Microsoft Edge, etc.

Programming Languages Used:

- HTML and CSS (for UI/UX design),
- JavaScript (for client-side logic),
- PHP (for server-side scripting),
- MySQL (for backend database management).

The application allows administrators to input institutional data such as courses, instructors, classrooms, and constraints. The GA then generates optimized timetables based on these inputs and user-defined priorities. The system supports multiple views (e.g., by semester, lecturer, or room), improving accessibility and communication.

3. Software Architecture, Findings, and Discussion

3.1. System Architecture Overview

Figure 1 presents the architecture of the Explicit (Semi) Automatic Timetable Generator, composed of three core components: input modules, a manual priority-setting module (processing), and an output interface for viewing the generated timetable. The system was designed for administrative use in higher education institutions, allowing staff to manage and automate the complex task of scheduling.

The system administrator is responsible for setting up and maintaining the timetable. After logging in, the administrator can input classroom data, faculty structures, academic programs, lecturers, and course-related information such as codes, titles, and credit units. This input phase is flexible and allows for corrections via "append," "remove," and "modify" functions. Once all data is provided, the administrator can proceed to the priority-setting module to manually assign courses to available timeslots and venues before generating a conflict-free timetable.

3.2. Detailed System Functionality and Interface Overview

This section presents the core components of the system from the perspective of the administrator—the primary user of the Explicit (Semi) Automatic Timetable Generator. The interface is designed to support a structured, step-by-step process for inputting, managing, and generating academic timetables in a flexible and semi-automated manner.

3.2.1. Login and Home Page Interface

Upon launching the system, the administrator is greeted with a login screen (Figure 2) that ensures only authorized personnel gain access to timetable creation and editing features. Upon successful authentication, the administrator is directed

to the home dashboard (Figure 3(a)), which presents the three main modules of the system:

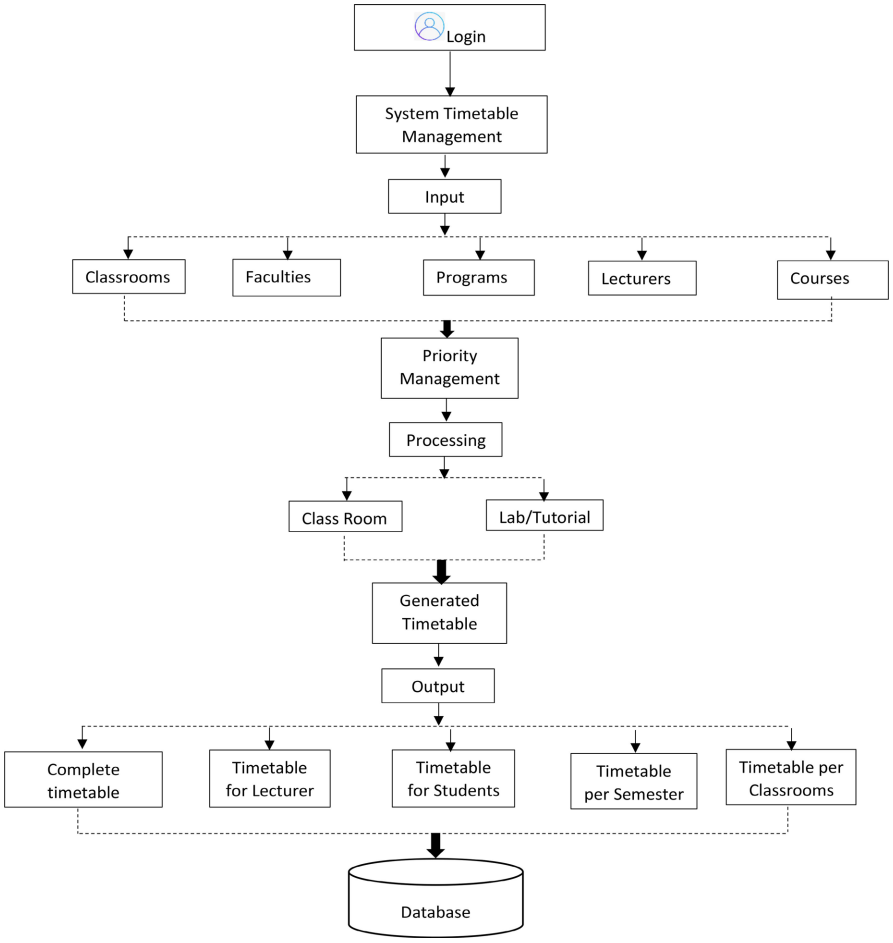


Figure 1. The architecture of explicit (Semi) automatic timetable generator.

1) System Timetable Management
2) Priority Management
3) Generated Timetable Display

These modules guide the administrator through the logical workflow of data input, manual priority setting, and viewing of output results. Figure 3(b) illustrates the detailed view of the System Timetable Management module, where the administrator interacts with core timetable parameters. The administrator is responsible for inputting the following key attributes:

- Course Code: Unique identifier for each course.
- Course Title: Full name of the course.
- Credit Units: Numerical credit weight of each course.
- Assigned Classroom: Preferred or assigned room for delivery.
- Laboratory Usage: Specification if the course requires a lab facility.

These parameters are essential for generating an accurate, conflict-free timetable that meets institutional and departmental scheduling constraints.

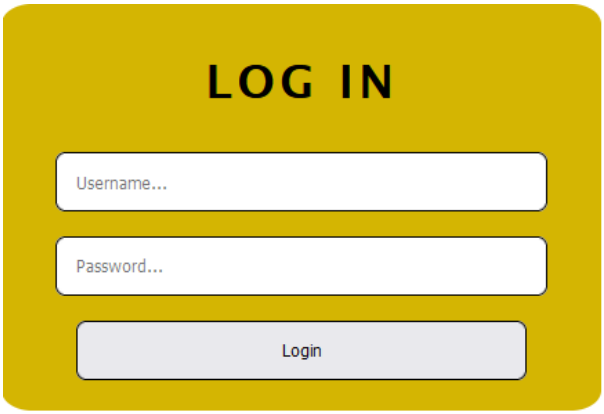
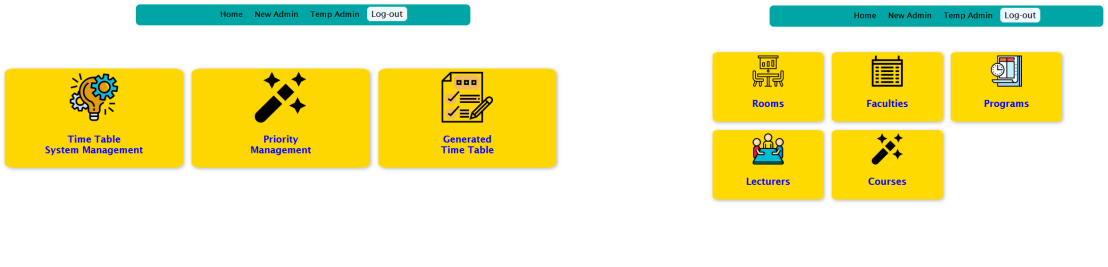


Figure 2. The login interface of the system.



(a) (b)

Figure 3. (a) The Home page of the system, (b) The Home page of the system Timetable Management.

3.2.2. System Timetable Management (Input Interface)

This is the primary data entry section (Figure 3(b)) where all entities relevant to scheduling are configured. The administrator uses this module to build the structural foundation of the timetable. The interface supports CRUD operations—Create, Read, Update, and Delete—via three primary buttons: Append, Remove, and Modify.

The input components are categorized as follows:

- **Classroom Management**

(Figure 4): Administrators input room numbers, seating capacities, locations, and facility types (e.g., lecture hall or laboratory). This ensures classroom allocation is based on actual spatial constraints.

- **Faculty and Department Data**

(Figure 5, Figure 6): The administrator defines academic units, including faculties and departments/programs. This hierarchical structure enables course-to-program associations and departmental-level filtering.

- **Lecturer Profiles**

(Figure 7): Each lecturer's name, designation, title, credentials, email, and assigned courses are entered into the system. This database is critical to avoid scheduling conflicts and ensure workload balance.

- **Course Database**

(Figure 8): This includes detailed metadata such as course code, title, semester

offered, credit units, prerequisite courses, target student group(s), program affiliation, required class hours, and preferred resources.

Each input screen offers form validation and field-specific instructions to guide users, reducing the possibility of input errors that might result in scheduling conflicts or illogical allocations.

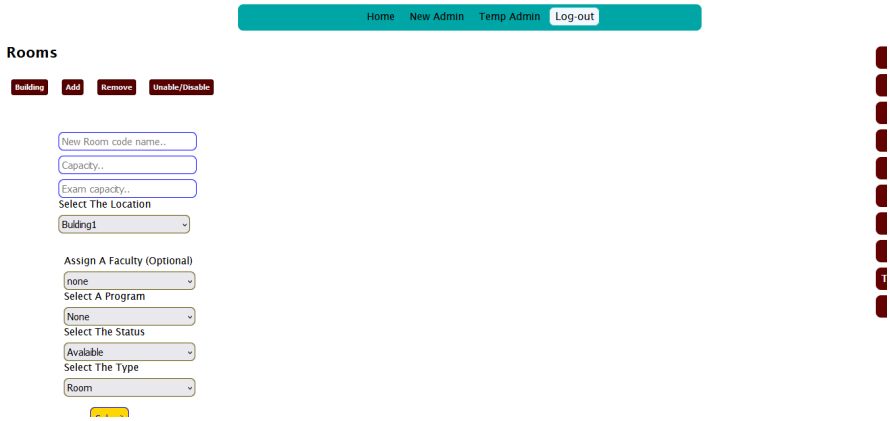


Figure 4. A classroom input section.

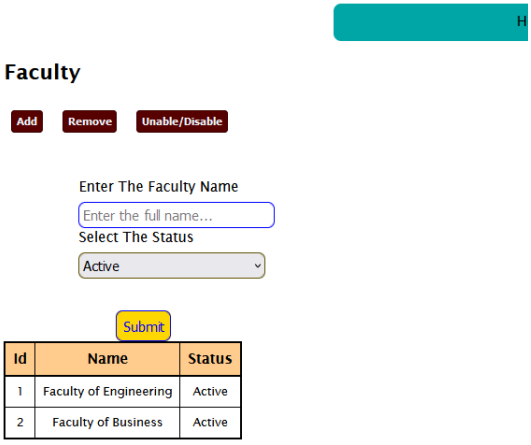


Figure 5. A faculty input section.

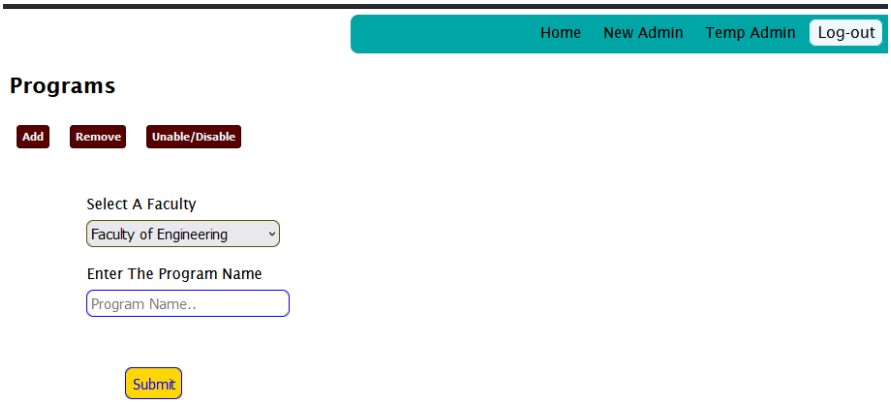


Figure 6. A program input section.

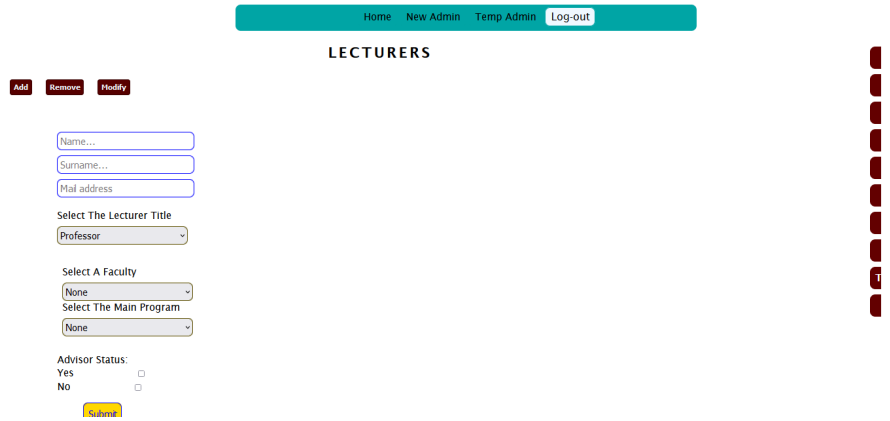


Figure 7. A lecturer input section.

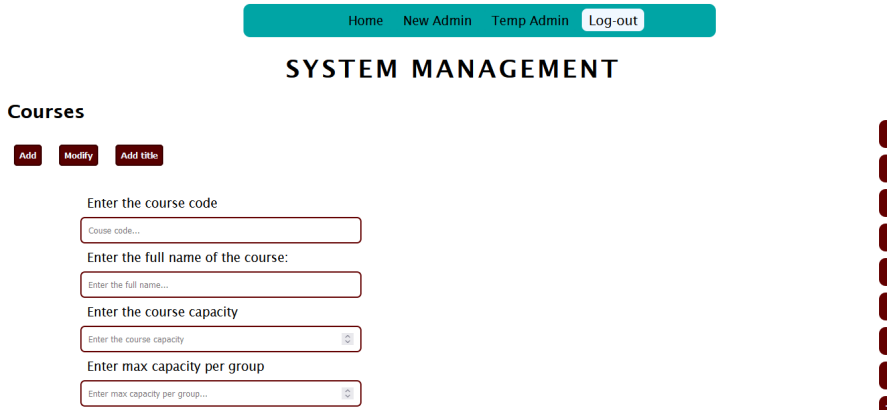


Figure 8. Course input section.

3.2.3. Priority Management (Processing Interface)

Once all relevant data is collected, the administrator navigates to the Priority Management module—the heart of the timetable configuration process. In this interface, the administrator begins by selecting available classrooms or laboratories from a searchable dropdown menu (Figure 9). The administrator then proceeds to assign courses manually into time slots (Figure 10), using visual scheduling blocks that represent different days of the week and hours of the day.

The system highlights scheduling conflicts in real-time, allowing the administrator to resolve them before finalizing the timetable. This semi-automated flexibility ensures human oversight while supporting efficient organization. Once all course allocations are completed, the administrator saves the configuration. The system then processes the data and automatically generates five distinct timetable views.

3.2.4. Timetable Output Interface

The system provides structured and filtered timetable views that cater to different stakeholders:

- **Complete Timetable View** (Figure 11): A holistic view that encompasses all courses across all departments and programs.

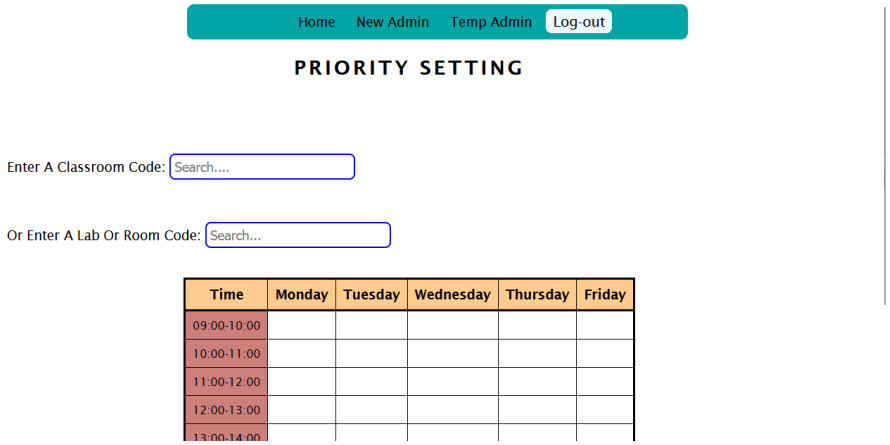


Figure 9. Priority setting per classroom/laboratory.

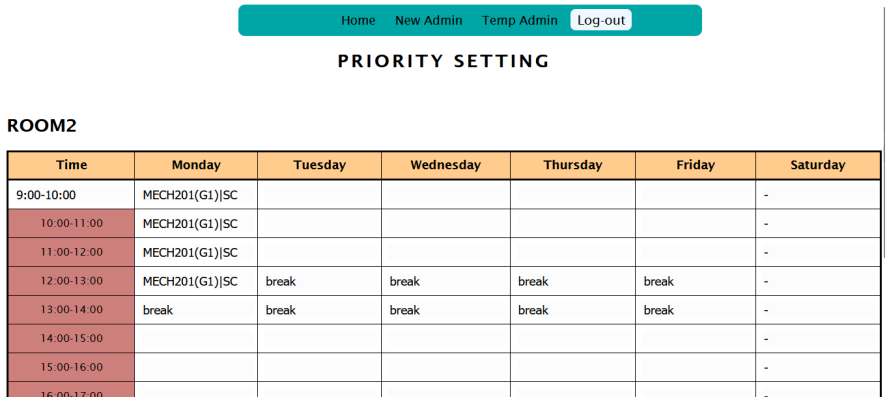


Figure 10. Priority setting for courses per classroom.

- **Lecturer-specific Timetable** (Figure 12): Enables instructors to quickly identify their teaching schedules.
- **Student Group Timetable** (Figure 13): Organized by year, department, or cohort, helping students track their weekly schedule.
- **Semester Timetable** (Figure 14): Focused view based on academic terms (e.g., Fall, Spring), assisting in administrative planning.
- **Classroom Timetable** (Figure 15): Lists all bookings for each room, aiding in space management and avoiding double-booking.

Each view is auto-generated upon saving the schedule in the Priority Management module and can be navigated interactively via the GUI. This segmentation of output enhances usability and ensures that every stakeholder accesses the information most relevant to him or her.

3.3. System Implementation and User Feedback

To evaluate real-world applicability, the system was deployed in the Department of Computer Science at [Institution Name], where administrative staff and academic coordinators used it to construct actual semester timetables. Feedback was collected from 8 academic administrators and 15 faculty members via structured

interviews and a usability questionnaire based on the SUS (System Usability Scale).

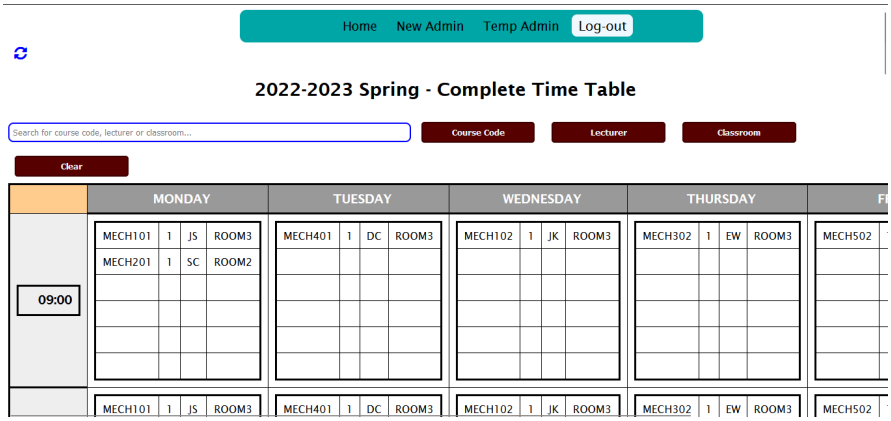


Figure 11. Result 1: Complete timetable section.

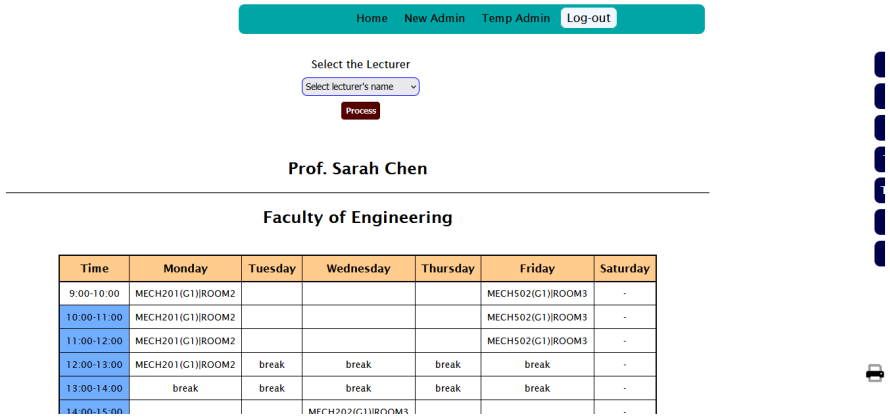


Figure 12. Result 2: Timetable per lecturer section.

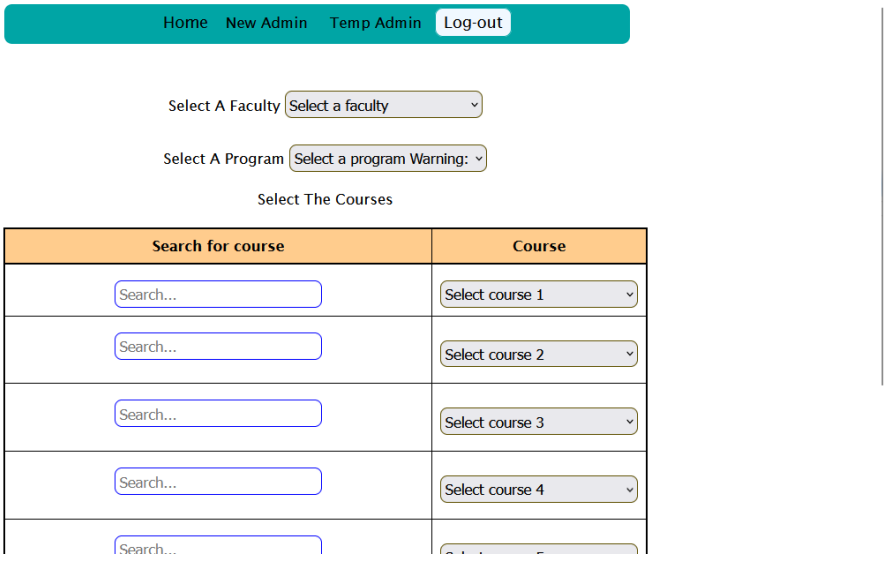
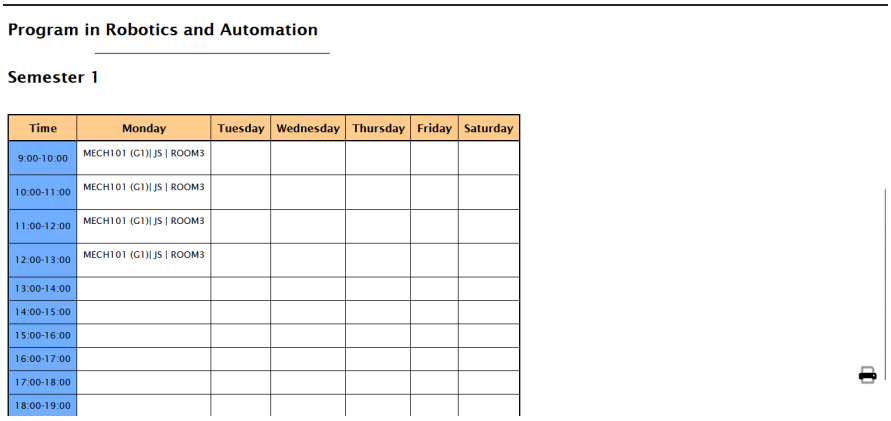


Figure 13. Result 3: Timetable per student section.

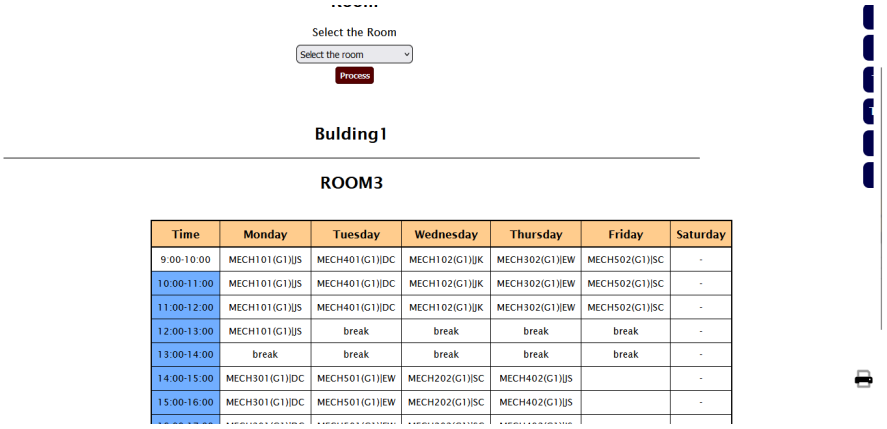


Program in Robotics and Automation

Semester 1

Time	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
9:00-10:00	MECH101 (G1)|JS | ROOM3					
10:00-11:00	MECH101 (G1)|JS | ROOM3					
11:00-12:00	MECH101 (G1)|JS | ROOM3					
12:00-13:00	MECH101 (G1)|JS | ROOM3					
13:00-14:00						
14:00-15:00						
15:00-16:00						
16:00-17:00						
17:00-18:00						
18:00-19:00						

Figure 14. Result 4: Timetable per semester section.



Select the Room

Select the room

Process

Bulding1

ROOM3

Time	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
9:00-10:00	MECH101(G1)|JS	MECH401(G1)|DC	MECH102(G1)|JK	MECH302(G1)|EW	MECH502(G1)|SC	-
10:00-11:00	MECH101(G1)|JS	MECH401(G1)|DC	MECH102(G1)|JK	MECH302(G1)|EW	MECH502(G1)|SC	-
11:00-12:00	MECH101(G1)|JS	MECH401(G1)|DC	MECH102(G1)|JK	MECH302(G1)|EW	MECH502(G1)|SC	-
12:00-13:00	MECH101(G1)|JS	break	break	break	break	-
13:00-14:00	break	break	break	break	break	-
14:00-15:00	MECH301(G1)|DC	MECH501(G1)|EW	MECH202(G1)|SC	MECH402(G1)|JS		-
15:00-16:00	MECH301(G1)|DC	MECH501(G1)|EW	MECH202(G1)|SC	MECH402(G1)|JS		-

Figure 15. Result 4: timetable per classroom section.

Positive feedback included:

- *Ease of use*: Administrators found the interface intuitive and appreciated the manual override flexibility.
- *Efficiency*: Generating a complete timetable was significantly faster compared to manual or spreadsheet-based methods.
- *Accuracy*: Participants reported that the system successfully reduced scheduling conflicts, particularly for shared resources like labs.
- Challenges and areas for improvement:
- *Manual burden*: Some users felt that the manual assignment phase was still time-consuming and suggested a hybrid auto-assignment feature.
- *Visual layout*: Faculty requested clearer visual grouping for day/time slots in the output.
- *Export features*: The lack of direct export to Excel or PDF was identified as a practical limitation.

3.4. Discussion and Lessons Learned

The deployment confirmed that the system could effectively streamline the scheduling process. However, it also revealed the trade-off between control and auto-

mation. Users valued the ability to manually assign priorities but noted that for larger institutions, a fully automated or semi-automated backend (e.g., via optimization algorithms like GA or CSP) would be more scalable.

Participants expressed interest in integrating features such as:

- Timetable conflict alerts during entry,
- Notification system for lecturers and students,
- Access control levels for department-level users.

Overall, the system proved its utility in mid-sized academic environments and laid the groundwork for iterative enhancement based on real feedback. Future work will focus on integrating optimization algorithms and mobile accessibility, in direct response to user suggestions.

4. Conclusions

This study demonstrates that the use of Genetic Algorithm (GA) techniques is effective and practical in solving complex timetable scheduling problems. The approach adopted in this paper not only proves to be successful in generating feasible and equitable schedules for students, but also shows promise for broader applications in other time-sensitive and resource-constrained environments.

The automated timetable system developed here simplifies a process that is traditionally tedious and error-prone when done manually—especially using spreadsheets, which often lead to scheduling conflicts and inconsistencies. By integrating essential information such as classrooms, faculties, programs, lecturers, and course details into a centralized database (MySQL server), the system efficiently generates conflict-free timetables. The use of GA allows for dynamic optimization, producing near-optimal solutions based on defined constraints and institutional requirements.

One of the major challenges encountered during development was designing a system capable of generating feasible timetables for higher education institutions, considering the diversity of constraints. However, the automated system significantly reduces the time and effort required, offers a more organized approach, and allows for easy identification and resolution of scheduling clashes.

Although the system is now operational and practical for deployment, ongoing testing and refinement are necessary to ensure robustness and accuracy. Particular attention should be given to improving database constraints and triggers to enhance flexibility and adaptability in varied contexts. Currently, data entry can be labor-intensive, especially for large-scale institutions. Future improvements will aim to streamline this process and minimize repetitive manual input.

Furthermore, implementing Genetic Algorithms effectively requires careful constraint formulation and parameter tuning. Despite the wealth of literature available, the practical implementation of GAs can be challenging. For this reason, GAs should be integrated into computer science curricula to help students understand their mechanics and applications in real-world problems. Teaching this algorithmic technique can deepen comprehension while promoting innovative so-

lutions in domains like automated scheduling.

In summary, the application of Genetic Algorithms in timetable generation provides a strong foundation for building intelligent, efficient, and user-centered academic scheduling systems. With further development, such systems can play a significant role in modern educational administration.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Cooper, T.B. and Kingston, J.H. (1996) The Complexity of Timetable Construction Problems. In: Burke, E. and Ross, P., Eds., *Practice and Theory of Automated Timetabling*, Springer, 281-295. https://doi.org/10.1007/3-540-61794-9_66

[2] Neufeld, G.A. and Tartar, J. (1974) Graph Coloring Conditions for the Existence of Solutions to the Timetable Problem. *Communications of the ACM*, **17**, 450-453. https://doi.org/10.1145/361082.361092

[3] Yu, T.L. (1990) Time-Table Scheduling Using Neural Network Algorithms. 1990 *IJCNN International Joint Conference on Neural Networks*, San Diego, 17-21 June 1990, 279-284. https://doi.org/10.1109/ijcnn.1990.137582

[4] Downsland, K.A. (1996) Simulated Annealing Solutions for Multi-Objective Scheduling and Timetabling. In: Rayward-Smith, V.J., Osman, I.H., Reeves, C.R. and Smith, G.D., Eds., *Modern Heuristic Search Methods*, Wiley, 155-166.

[5] Hertz, A. (1991) Tabu Search for Large Scale Timetabling Problems. *European Journal of Operational Research*, **54**, 39-47. https://doi.org/10.1016/0377-2217(91)90321-l

[6] Abbas, A.M. and Tsang, E.P.K. (2001) Constraint-Based Timetabling-A Case Study. *Proceedings ACS/IEEE International Conference on Computer Systems and Applications*, Beirut, 25-29 June 2001, 67-72. https://doi.org/10.1109/aiccsa.2001.933953

[7] Kehinde, W. and Micheal, A. (2019) Automatic Timetable Generation Using Genetic Algorithm. *Computer Engineering and Intelligent Systems*, **10**, 23-26.

[8] Srinivasan, D., Seow, T.H. and Xu, J.X. (2002) Automated Time Table Generation Using Multiple Context Reasoning for University Modules. *Proceedings of the* 2002 *Congress on Evolutionary Computation. CEC*'02 (*Cat. No.*02*TH*8600), Honolulu, 12-17 May 2002, 1751-1756. https://doi.org/10.1109/cec.2002.1004507

[9] Barkha, N., Ambika, G. and Rashmi, B. (2013) Use of Active Rules and Genetic Algorithm to Generate the Automatic Timetable. *International Journal of Advances in Engineering Sciences*, **3**, 143-182.

[10] Narang, B., Gupta, A. and Bansal, R. (2013) Use of Active Rules and Genetic Algorithm to Generate the Automatic Time-Table. *International Journal of Advances in Engineering Sciences*, **3**, 40-44.

[11] Ladipo, W.K., Bamidele, A.O. and Olalekan, A.M. (2019) Automatic Timetable Generation Using Genetic Algorithm. *International Journal of Applied Information Systems*, **12**, 1-3.

[12] Ceschia, S. and Schaerf, A. (2022) Educational-Timetabling Problems, Benchmarks, and State-of-the-Art Results. *European Journal of Operational Research*, **296**, 1-13. https://doi.org/10.1016/j.ejor.2022.07.011

[13] Rane, M.V., Apte, V.M., Nerkar, V.N., Edinburgh, M.R. and Rajput, K.Y. (2021) Automated Timetabling System for University Course. 2021 *International Conference on Emerging Smart Computing and Informatics* (*ESCI*), Pune, 5-7 March 2021, 328-334. https://doi.org/10.1109/esci50559.2021.9396906

[14] Tan, J.S., Goh, S.L., Kendall, G. and Sabar, N.R. (2021) A Survey of the State-Of-The-Art of Optimisation Methodologies in School Timetabling Problems. *Expert Systems with Applications*, **165**, Article ID: 113943. https://doi.org/10.1016/j.eswa.2020.113943

[15] Pounikar, A., Bhandage, H., Dalvi, N., Borade, T. and Lokhande, S.H. (2023) Survey Paper on Automatic Timetable Generator. *International Journal of Advanced Research in Science*, *Communication and Technology*, **3**, 620-623. https://doi.org/10.48175/ijarsct-8016

[16] Rowe, J.E. (2007) Genetic Algorithm Theory. *Proceedings of the* 9*th Annual Conference Companion on Genetic and Evolutionary Computation*, London, 7-11 July 2007, 3585-3608. https://doi.org/10.1145/1274000.1274125

[17] Duque, T.S.P.C., Goldberg, D.E. and Sastry, K. (2008) Improving the Efficiency of the Extended Compact Genetic Algorithm. *Proceedings of the* 10*th Annual Conference on Genetic and Evolutionary Computation*, Atlanta, 12-16 July 2008, 467-468. https://doi.org/10.1145/1389095.1389181

[18] Tinós, R., Przewozniczek, M., Whitley, D. and Chicano, F. (2023) Genetic Algorithm with Linkage Learning. *Proceedings of the Genetic and Evolutionary Computation Conference*, Lisbon, 15-19 July 2023, 981-989. https://doi.org/10.1145/3583131.3590349