Web-Based Automatic Timetable Scheduler for Colleges

1st Gaurav G Alva
Dept. of CSE (Data Science)
Vivekananda College of Engineering & Technology, Puttur
Karnataka, India
gauravalva.me@gmail.com

2nd Prapthi J P
Dept. of CSE (Data Science)
Vivekananda College of Engineering & Technology, Puttur
Karnataka, India
prapthijaineera@gmail.com

3rd Supreetha N S
Dept. of CSE (Data Science)
Vivekananda College of Engineering & Technology, Puttur
Karnataka, India
4vp22cd058@vcetputtur.ac.in

4th Harshit M Naik
Dept. of CSE (Data Science)
Vivekananda College of Engineering & Technology, Puttur
Karnataka, India
4vp22cd022@vcetputtur.ac.in

5th Chaithanya D
Assistant Professor, Dept. of CSE (Data Science)
Vivekananda College of Engineering & Technology, Puttur
Karnataka, India
chaithanyad.cd@vcetputtur.ac.in

***Abstract*—Timetable generation in educational institutions is a complex task involving numerous constraints, such as faculty availability, classroom assignments, subject load, and session continuity. Manual scheduling methods are often inefficient, error-prone, and difficult to scale, especially in multi-department and multi-semester academic environments. The Web-Based Automatic Timetable Scheduler for Colleges addresses this challenge by providing an intelligent, web-based timetable generation system using Genetic Algorithms. Built using Python and the Streamlit framework, the system allows administrators to input essential scheduling parameters including faculty details, subject details, classroom availability, and slot preferences. The core of the application employs a Genetic Algorithm to evolve feasible solutions, guided by a fitness function that prioritizes conflict resolution, slot continuity and balanced faculty workloads. Both hard constraints and soft constraints are incorporated to ensure practical and optimal scheduling. The generated timetables are structured to prevent common scheduling issues such as double allotment for teachers and same subject clashes, also provides scalability to accommodate elective subjects, lab sessions, and multi-semester academic structures. The system outputs user-friendly, downloadable schedules and allows for administrative oversight and adjustments. This, demonstrates the potential of evolutionary computation in solving NP-hard optimization problem of timetable scheduling, offering a scalable, accurate, and efficient solution to a long-standing administrative challenge in academic institutions.**

***Index Terms*—timetable scheduler, genetic algorithm**

I. Introduction

Timetable scheduling is a critical yet complex task in educational institutions, involving the optimal allocation of courses, faculty, and rooms under numerous hard and soft constraints. As institutional size and scheduling requirements grow, traditional manual methods become increasingly inefficient, error-prone, and difficult to scale [1]. The problem is inherently NP-Hard, making it impractical to solve using exact algorithms for large instances. Consequently, heuristic and metaheuristic approaches have been widely adopted. Among these, Genetic Algorithms (GA) have shown significant promise due to their adaptive nature and effectiveness in exploring large solution spaces. GA-based systems evolve populations of candidate solutions through operations such as selection, crossover, and mutation to produce conflict-free and near-optimal timetables [2]. In this study, we propose an improved GA-based timetable generation system designed to address these issues. Our approach emphasizes better constraint satisfaction, integration of user preferences, and a scalable design suitable for institutional deployment.

II. Literature Survey

Various approaches have been proposed to address the timetabling problem, ranging from graph-based models and constraint satisfaction to heuristic and metaheuristic algorithms. Among them, GA have stood out due to their adaptive nature and ability to explore large solution spaces effectively. Customized genetic operators, such as row-based crossover and order-k mutation, have been employed to tackle highly constrained school timetabling problems. These are often combined with local search and a repair mechanism to correct infeasibility [3]. Constraints that must be satisfied during scheduling are typically classified into hard and soft constraints. These include minimizing idle gaps between student lectures and ensuring an even distribution of teaching assignments across the week [4]. Scalability and real-time adaptability have also been addressed using GA. A system employing roulette wheel and tournament selection, single-point crossover, and mutation—combined with a weighted fitness function to evaluate constraint violations—achieved 95% conflict resolution and 90% resource utilization on real-world university data [5]. GA has also been combined with Dynamic Programming (DP) to solve joint course timetables. While GA is used to optimize time slot assignments through swap-based mutation and a repair mechanism, DP is employed for classroom allocation to reduce seat wastage and enhance utilization. This hybrid approach has outperformed methods such as Particle Swarm Optimization, Ant Colony Optimization, and the Producer-Scrounger method across metrics like fitness and occupancy rate [6]. Another technique utilizes dynamic chromosomes that adapt to varying course loads across departments and semesters. Scheduling variables are encoded as tuples of course, time, and room. This system applies roulette wheel selection, two-point crossover, and optimized mutation to achieve up to 93% conflict-free timetables in under five minutes [7]. A hybrid approach integrating GA with graph coloring has also been proposed. Courses are modelled as graph nodes and conflicts as edges to prevent time slot overlaps. Each timetable is represented as an array of genes, where each gene includes course, instructor, room, and time slot. A penalty-based fitness function is used to evaluate violations of hard and soft constraints [8]. These studies highlight the effectiveness of GA-based and hybrid approaches in resolving complex timetabling constraints. The research continues to evolve toward more efficient, scalable, and real-time systems that can meet the dynamic needs of educational institutions.

III. Existing System

Timetable generation has long been a complex scheduling problem due to the presence of multiple hard and soft constraints, including room availability, faculty workload, and subject-hour distribution. Many institutions still rely on manual scheduling, where teachers or coordinators physically construct timetables while considering institutional constraints and limitations. This process is time-consuming, error-prone, and difficult to scale for large academic environments. Several studies have applied GA in various forms—such as constraint-weighted models [9], hybrid approaches [6], and preference-based scheduling [10]. Despite these advancements, limitations remain in terms of dynamic adaptability, real-time editing, and scalability across departments [11].

A. Limitations of Existing System

- Limited support for Soft Constraints such as faculty preferences, load balancing.
- Minimal User interaction, especially in systems without web-based or collaborative interfaces.
- Lack of real-time adaptability to handle sudden changes such as room unavailability.

IV. Proposed Methodology

The proposed method employs a GA to generate conflict-free and optimized academic timetable based on institutional constraints and faculty preferences.

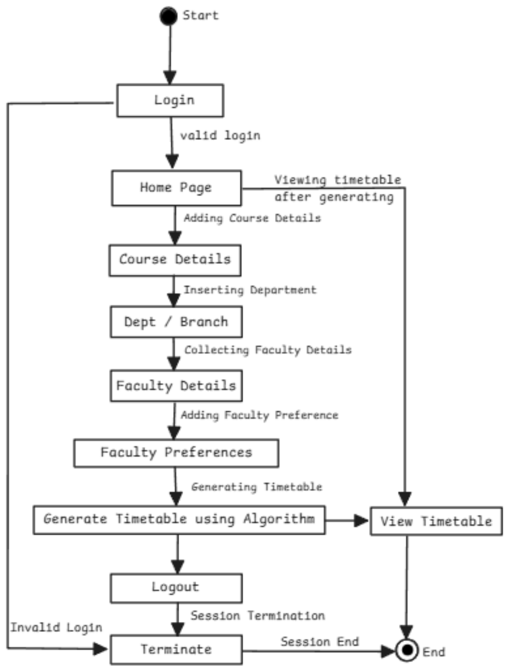


Fig. 1. Overview of the proposed system of GA timetable generation process.

Figure 1. Flowchart of the proposed timetable generation system. The process begins with user login and navigates through stages including course, department, faculty detail collection, and faculty preference inputs. Upon submission, the system uses a genetic algorithm to generate an optimized timetable. Users can then view the generated timetable or terminate the session. Invalid logins and session terminations are handled with proper flow control.

A. Problem Modeling

The objective is to generate a timetable that assigns all classes (theory and lab) to valid time slots such that:

- Hard constraints are satisfied:
  - No overlapping classes for the same semester.
  - No faculty member is assigned to multiple classes simultaneously.
  - Labs span exactly two consecutive periods within daily limits.
- Soft constraints are optimized:
  - Faculty preferences for specific time slots are honored.
  - Lab sessions avoid crossing breaks like recess or lunch.

B. Genetic Algorithm (GA)

GA are metaheuristic optimization techniques inspired by the principles of natural selection. They are designed to efficiently explore large and complex solution spaces. GA utilize flexible and adaptive frameworks, making them particularly effective in dynamic environments. In this work, a GA is employed to automatically construct class schedules that satisfy hard constraints and soft constraints. A self-adaptive approach further enhances its effectiveness by allowing the algorithm to adjust parameters and strategies dynamically based on problem-specific feedback during execution.

C. Population Initialization

Population initialization involves generating a diverse set of candidate timetables, each representing a unique assignment of classes to time slots. In this implementation, initialization is purely random to encourage diversity and avoid early stagnation.

D. Chromosome Representation

A chromosome represents a list of scheduled class instances, with each gene encoding one scheduled class, including its assigned day, period, faculty, and type (lab or theory). This structure supports both hard and soft constraint evaluations.
Each chromosome encodes a full academic timetable as a list of Scheduled Class genes. Each gene comprises:
Semester ID, Course ID, Faculty ID, Day of the week (Monday–Saturday), Start period (1–7), Duration (1hr for theory, 2hr for lab).

E. Fitness Evaluation

Fitness evaluation quantifies the quality of a timetable by using penalty-based scoring system. Hard constraints incur high penalties to ensure feasibility, while soft constraints guide the optimization toward preferred scheduling patterns. Hard constraints must be strictly enforced, while soft constraints are preferred but flexible.
A penalty-based fitness function is used. The total fitness of a chromosome is the negative sum of penalties for constraint violations:

- Hard Constraint Penalties:
  - Semester or faculty time conflicts: +1000
  - Invalid lab durations or boundaries: +500
- Soft Constraint Penalties:
  - Not scheduled in preferred slots: +10
  - Labs scheduled across breaks: +5

The fitness is calculated as:

$$Fitness = -\sum (\text{penalties}) \tag{1}$$

(1) A higher fitness score (closer to 0 or positive) indicates a better timetable.

F. Selection

Selection identifies high-quality timetables for reproduction. This implementation uses tournament selection, where groups of chromosomes compete and the fittest are selected, encouraging stronger candidates to propagate.

G. Crossover

Crossover recombines two parent timetables by exchanging time assignments beyond a randomly chosen point, preserving order and structure while introducing variation into the offspring.

H. Mutation

Mutation introduces random changes in a chromosome by reassigning new time slots to individual classes with a small probability. This maintains diversity in the population and enables broader search of the solution space.

I. Evolution and Termination

Termination occurs either when a feasible timetable meeting all hard constraints is found or after a predefined number of generations. This ensures bounded computation time. The GA proceeds for a predefined number of generations or until an optimal chromosome ($fitness\ \geq 0$) is found. In each generation:

1) Fitness scores are updated.
2) Parents are selected.
3) Crossover and mutation are applied.
4) A new population is formed.
5) The best individual is tracked.

Early stopping is triggered if a feasible solution is found before the maximum generation count.

V. Design and Implementation

A. System Architecture

The application is structured using a modular three-layer architecture:

- **Frontend**: A Streamlit-based web interface that facilitates user interaction through a simple and intuitive UI.
- **Backend**: Developed using Python, the backend manages application logic and communicates with a PostgreSQL database for persistent data storage `database.py`.
- **GA Module**: The `genetic_algorithm.py` script is responsible for generating and optimizing academic timetables using a Genetic Algorithm.

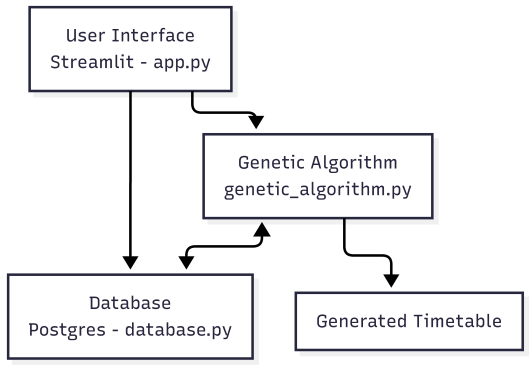


Fig. 2. System architecture of web application.

B. User Interface

A Streamlit-based web interface allows users to input data and visualize the generated timetable.

- **Authentication**: User registration and login with password hashing.
- **Entity Management**: Forms to add/edit departments, semesters, faculty, and courses.
- **Mappings**: Assign theory/lab courses to faculty.
- **Preferences**: Set preferred slots per faculty.
- **GA Parameters**: Set population size, generations, mutation, and crossover rates.
- **Output Display**: Tabular timetable display.

C. Data Storage

Data is retrieved from structured inputs containing semester, courses, faculty assignments, and preferences.

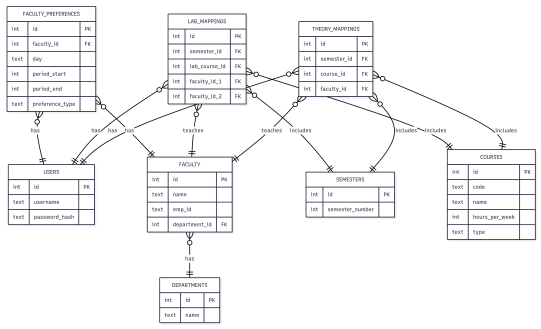
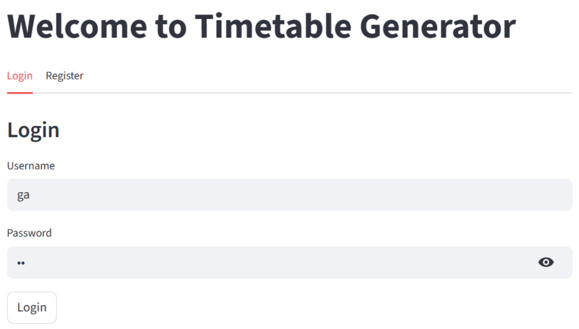


Fig. 3. Database schema illustrating the relationships between core entities such as departments, courses, faculty, and timetables.

VI. Result and Analysis

A. Login Page

The user authentication interface allows registered users to securely log in to access the timetable generation system.



Welcome to Timetable Generator

Login Register

Login

Username

ga

Password

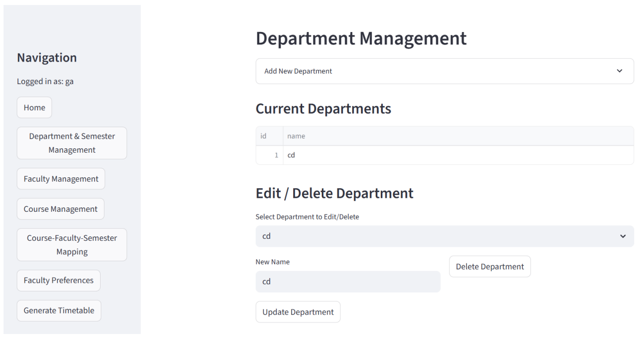
••

Login

Fig. 4. Login interface for user authentication.

B. Add Department Details

This interface enables users to add and manage academic departments for course and faculty mapping.



Navigation

Logged in as: ga

Home

Department & Semester Management

Faculty Management

Course Management

Course-Faculty-Semester Mapping

Faculty Preferences

Generate Timetable

Department Management

Add New Department

Current Departments

id	name
1	cd

Edit / Delete Department

Select Department to Edit/Delete

cd

New Name

cd

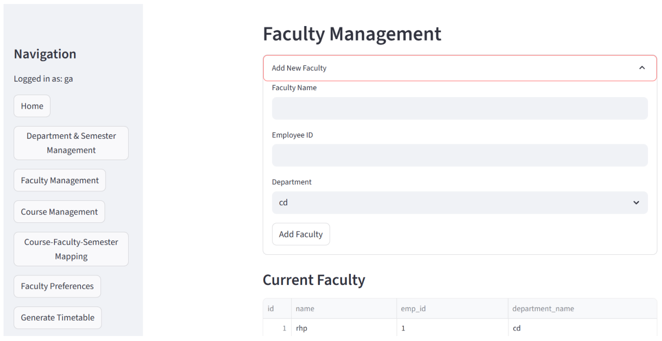
Delete Department

Update Department

Fig. 5. Form to add department details into the system.

C. Add Faculty Details

Faculty details and preferences are added to ensure proper allocation during timetable generation.



Navigation

Logged in as: ga

Home

Department & Semester Management

Faculty Management

Course Management

Course-Faculty-Semester Mapping

Faculty Preferences

Generate Timetable

Faculty Management

Add New Faculty

Faculty Name

Employee ID

Department

cd

Add Faculty

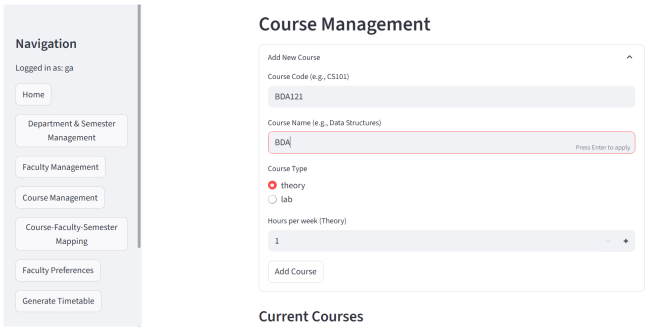
Current Faculty

id	name	emp_id	department_name
1	rhp	1	cd

Fig. 6. Interface to add faculty information and configure slot preferences.

D. Add Coourse Details

Course details are added and mapped to respective class to ensure proper allocation during timetable generation.



Navigation

Logged in as: ga

Home

Department & Semester Management

Faculty Management

Course Management

Course-Faculty-Semester Mapping

Faculty Preferences

Generate Timetable

Course Management

Add New Course

Course Code (e.g., CS101)

BDA121

Course Name (e.g., Data Structures)

BDA

Press Enter to apply

Course Type

[✓ theory]

lab

Hours per week (Theory)

1

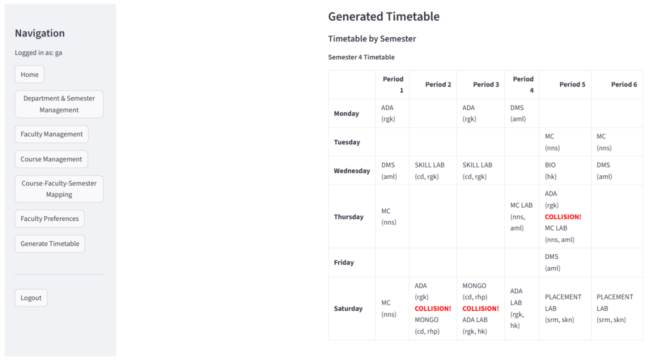
Add Course

Current Courses

Fig. 7. Course entry interface and mapping subjects to semesters.

E. Generating Timetable

The final timetable is displayed in a structured format, showing class schedules for each semester.



Generated Timetable

Timetable by Semester

Semester 4 Timetable

	Period 1	Period 2	Period 3	Period 4	Period 5	Period 6
Monday	ADA (rgk)		ADA (rgk)	DMS (aml)		
Tuesday					MC (nns)	MC (nns)
Wednesday	DMS (aml)	SKILL LAB (cd, rgk)	SKILL LAB (cd, rgk)		BIO (hk)	DMS (aml)
Thursday	MC (nns)			MC LAB (nns, aml)	ADA (rgk) **COLLISION!** MC LAB (nns, aml)	
Friday					DMS (aml)	
Saturday	MC (nns)	ADA (rgk) **COLLISION!** MONGO (cd, rhp)	MONGO (cd, rhp) **COLLISION!** ADA LAB (rgk, hk)	ADA LAB (rgk, hk)	PLACEMENT LAB (srm, skn)	PLACEMENT LAB (srm, skn)

Fig. 8. Final generated timetable displayed in a tabular view.

VII. Conclusion

This paper presents a web-based solution for automating the complex task of timetable scheduling in academic institutions. By integrating a user-friendly Streamlit interface, a structured PostgreSQL database, and a Genetic Algorithm-based optimization engine, the system effectively generates efficient timetables that respect faculty preferences and institutional constraints. The modular design ensures scalability, adaptability, and ease of maintenance. Experimental results demonstrate that the proposed system significantly reduces manual effort while improving scheduling accuracy.

References

[1] B. S. Prabhakaran, M. Shanmugam, C. B. Karunakaran, B. Murugesan, and D. Manimaran, "Automatic timetable generator using genetic algorithm," in *6th International Conference on Intelligent Computing (ICIC-6 2023)*. Atlantis Press, 2023, pp. 84–89.

[2] O. M. K. Alsmadi, S. Za'er, D. I. Abu-Al-Nadi, and A. Algsoon, "A novel genetic algorithm technique for solving university course timetabling problems," in *International Workshop on Systems, Signal Processing and their Applications, WOSSPA*. IEEE, 2011, pp. 195–198.

[3] A. Colorni, M. Dorigo, and V. Maniezzo, "A genetic algorithm to solve the timetable problem," *Politecnico di Milano, Milan, Italy TR*, pp. 90–060, 1992.

[4] D. Mittal, H. Doshi, M. Sunasra, and R. Nagpure, "Automatic timetable generation using genetic algorithm," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 4, no. 2, pp. 245–248, 2015.

[5] S. S. Paramatmuni, D. Y. Reddy, E. S. Spoorthi, A. Dharani, and K. V. Sharma, "Smart timetable generation using genetic algorithm," *Macaw International Journal of Advanced Research in Computer Science and Engineering*, vol. 10, no. 1s, pp. 204–215, 2024.

[6] X. Han and D. Wang, "Gradual optimization of university course scheduling problem using genetic algorithm and dynamic programming," *Algorithms*, vol. 18, no. 3, p. 158, 2025.

[7] G. Alnowaini and A. A. Aljomai, "Genetic algorithm for solving university course timetabling problem using dynamic chromosomes," in *2021 International Conference of Technology, Science and Administration (ICTSA)*. IEEE, 2021, pp. 1–6.

[8] M. Assi, B. Halawi, and R. A. Haraty, "Genetic algorithm analysis using the graph coloring method for solving the university timetable problem," *Procedia Computer Science*, vol. 126, pp. 899–906, 2018.

[9] E. A. Abdelhalim and G. A. El Khayat, "A utilization-based genetic algorithm for solving the university timetabling problem (uga)," *Alexandria Engineering Journal*, vol. 55, no. 2, pp. 1395–1409, 2016.

[10] A. R. Mahlous and H. Mahlous, "Student timetabling genetic algorithm accounting for student preferences," *PeerJ Computer Science*, vol. 9, p. e1200, 2023.

[11] R. E. Febrita and W. F. Mahmudy, "Modified genetic algorithm for high school time-table scheduling with fuzzy time window," in *2017 International Conference on Sustainable Information Engineering and Technology (SIET)*. IEEE, 2017, pp. 88–92.