# CHAPTER 1

## INTRODUCTION

### Introduction to the Project

A timetable is a key administrative component in any educational institution. Creating one is typically a complex and demanding process. Coordinators or administrators must consider various factors during scheduling, which becomes particularly stressful in colleges due to multiple departments, diverse subjects, and faculty members with varying designations. They must carefully allocate courses, classrooms, and faculty based on subject requirements, and patiently construct an efficient timetable to ensure an organized schedule and smooth functioning of the institution. Despite the increasing automation of administrative tasks in colleges, lecture timetables are still often prepared manually due to the inherent complexities of the process.

Timetable scheduling problem is inherently complex due to the numerous interrelated factors and constraints, including teaching plans, courses, instructors, administrative and teaching classes, classrooms, and time slots. In universities, the complexity increases significantly during semester scheduling, as it must accommodate multiple student batches, various student groups, elective subjects (with different groups choosing different electives), and common mandatory courses taken by all the students. It is essential to ensure that there are no clashes among student groups, faculty members, and lecture halls during the scheduling process.

Timetable scheduling is widely recognized as a combinatorial optimization problem and a constraint satisfaction problem where the goal is to find a solution that satisfies a predefined set of constraints. These constraints are generally categorized as hard and soft constraints. Hard constraints are mandatory and must be satisfied for a timetable to be feasible, such as preventing classes from clashing in the same room or a teacher having more than one class simultaneously. Soft constraints, on the other hand, represent preferences or desired conditions that ideally should be met but are not strictly required for feasibility. Meanwhile violating soft constraints is acceptable, but an optimal timetable minimizes such violations.

This project aims to address the difficulties of generating timetables by providing an automated scheduling system. Automating the process helps save time, avoid the complexity of manual management, and reduce documentation work. The goal is to generate timetables that are more accurate, precise, and free of human errors. An automated system ensures up-to-date and accurate information.

To achieve automatic timetable generation, the project proposes the use of algorithms such as genetic, heuristic, and resource scheduling. Genetic algorithms (GAs) are frequently applied to timetabling problems, which are known to be NP-hard optimization problems, because GAs are effective for finding optimal or near-optimal feasible solutions among a complex set of variables and constraints. They are based on natural selection and evolution principles and are known for their robustness in solving complex combinatorial problems. Heuristic approaches are also commonly used in timetabling, either independently or as components within algorithms like GAs, often focusing on scheduling the most constrained elements first. Resource scheduling is another algorithmic approach listed for addressing these problems. These algorithms incorporate various strategies aimed at improving the efficiency, scalability and reliability of the timetable generation process.

The system will generate the timetables based on various inputs, including the number of subjects and teachers, teacher workload, semester details, and subject priorities. Additional necessary inputs include faculty details, subject details (including name and code), workload based on faculty designation, faculty and subject allotment based on time slots, and details of theory and lab courses handled by each faculty. Classroom or room details, including availability and capacity, are also crucial inputs. By relying on these inputs and utilizing optimization algorithms, the system will generate possible timetables for the working days.

In conclusion, the proposed solution aims to streamline the scheduling process, enhance accuracy, and offer a more efficient, scalable, and reliable approach to timetable generation, ultimately contributing to smoother academic operations and a better teaching-learning experience.

### Introduction to Technology Used

The Web-Based Automatic Timetable Scheduler relies on a combination of modern technologies to ensure efficiency, accuracy, and scalability. The primary programming language used in development is Python, which offers extensive libraries that support data processing, optimization, and user interface creation. Python frameworks like Django are utilized for backend development, enabling secure database management and seamless interaction between different system components. For handling and storing timetable data, a relational database is integrated, allowing efficient retrieval and management of faculty schedules, subject allotments, and classroom assignments.

To ensure optimal timetable generation, the system employs Genetic Algorithms (GA), which mimic natural selection principles to refine scheduling solutions iteratively. Additionally, heuristic methods and resource scheduling techniques are used to prioritize tasks and manage faculty workload distribution efficiently. The user interface is designed using HTML, CSS, and JavaScript, ensuring accessibility across multiple devices and providing an interactive visual representation of generated timetables. The combination of these technologies ensures that the timetable scheduler not only automates scheduling tasks but also optimizes educational resource allocation, reducing human errors and streamlining academic operations.

# CHAPTER 2

## LITERATURE SURVEY

### Introduction

Timetable scheduling is an NP-hard problem, meaning it involves a high level of computational complexity, and finding an exact solution is often infeasible within a reasonable time for large instances. Therefore, the goal is typically to find an optimal or near-optimal solution within a complex set of variables and constraints. As a result, extensive research has been conducted on applying optimization techniques to address this challenge.

This literature review explores various existing approaches to automated timetable generation, with a particular focus on Genetic Algorithms (GAs). GAs have gained popularity due to their effectiveness in solving NP-hard optimization problems like timetabling. They operate on the principles of natural selection and evolution, using mechanisms such as selection, crossover, and mutation to iteratively improve solutions. Their ability to efficiently explore large and complex search spaces makes them well-suited for generating feasible and optimized timetables under multiple constraints.

### Literature Survey

Mittal et al. [1] addressed the timetable scheduling problem by proposing the use of Genetic Algorithms to automate and optimise the scheduling process. Their observation says that manual scheduling is time-consuming and prone to errors due to complexity of satisfying various constraints. The authors classify these constraints into two categories: hard constraints such as avoiding conflicts while assigning the classes and rooms; and soft constraints, such as minimizing idle gaps for students lectures and distributing teaching assignments evenly across the week. The authors applied Genetic Algorithms where the algorithm initialises the population of guesses, then three operators are applied- selection, crossover and mutation to create an optimal timetable. The system was tested with a real data within the author’s institution. The results demonstrated significant improvements in both efficiency and accuracy compare to manual scheduling. Although the study succeeded in creating a more efficient alternative to manual scheduling, it lacked mechanisms for dynamic adaptability and deeper real-world constraint handling.

Abdelhalim et al. [2] introduced a Utilization-based Genetic Algorithm (UGA) to solve the university course timetabling problem with a focus on optimizing space utilization alongside satisfying scheduling constraints. Their approach employed a two-dimensional chromosome representation, mapping events to room-timeslot pairs, and generated the initial population using heuristic methods: Largest Degree First (LD) and Largest Enrollment First (LE), the authors proposed a utilization-based crossover operator aimed at reallocating underutilized events and a targeted mutation strategy employing local search to optimize room occupancy. A composite fitness function weighted towards occupancy rates, frequency rates, and minimizing scheduling gaps was used to evaluate solutions. The system was tested on real data from Alexandria University the dataset comprised of 337 events, 32 rooms and 131 professors also, the system was tested against two benchmark datasets from the International Timetabling Competition (ITC 2007). The Results demonstrated significant improvements in space utilization, reduced scheduling hours, and better overall timetable quality compared to manual scheduling. However, the study identified gaps in dynamic real-time adaptability and slight computational overhead for medium-sized problems.

Colorni et al. [3] proposed a Genetic Algorithm approach to solve the highly constrained school timetabling problem. Their method used a two-dimensional matrix representation of teachers and timeslots, with a hierarchical objective function focusing first on feasibility, second on management rules, and finally on individual teacher preferences. This method uses customized genetic operators, like a row-based crossover and order-k mutation, along with a genetic repair (filtering) process to correct infeasibilities. The integration of local search significantly enhanced timetable quality, achieving a substantial reduction in cost compared to manual scheduling and simulated annealing. However, the approach faced scalability challenges due to the computational complexity of the repair process and lacked mechanisms for dynamic timetable adjustments.

Othman et al. [4] presented a Genetic Algorithm approach for university course timetabling, ensuring all hard constraints were satisfied while minimizing soft constraint violations. Each chromosome encoded subject, section, instructor, time, and room data. The method employed rank-based selection, single-point crossover, and random mutation, with exclusiveness to retain the best solutions across generations. A fitness function was designed to normalize and weigh both hard and soft constraint violations, covering instructor time preferences, room assignments, course prerequisites, and overload control. Tested on real data from the University of Jordan, the algorithm achieved zero hard constraint violations and significantly reduced instructor overload and scheduling conflicts. However, the model lacked real-time adaptability, sensitivity to constraint weights, and user interaction capabilities, limiting its flexibility and scalability in dynamic environments.

Han et al. [5] proposed a hybrid approach called POGA-DP, combining Genetic Algorithms (GA) with Dynamic Programming (DP) to solve the University Course Scheduling Problem (UCSP), particularly for complex joint-course timetables. The GA component optimizes time slot assignments using a swap-based mutation with a repair mechanism, while DP allocates classrooms to minimize seat wastage and improve utilization. The method achieved significant improvements in scheduling quality (up to 46.99%) and reduced classroom usage by 29.27% compared to standard GAs. Tested on data from Beijing Forestry University, the approach outperformed GA, Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), and Producer–Scrounger Method (PSM) across multiple metrics, including fitness and occupancy rate. However, the model lacks real-time adaptability, parameter tuning, cross-institutional testing, and practical integration into university systems, limiting its deployment readiness.

Ghiridhar et al. [6] proposed a Genetic Algorithm-based system for generating conflict-free timetables tailored to the regulations of A P J Abdul Kalam Technological University. Their approach introduced a conflict operator within each chromosome to dynamically detect and resolve scheduling clashes during evolution. The system addressed both hard constraints, such as subject-hour limits and lab continuity, and soft constraints, like balanced faculty workloads. Mutation was performed through a conflict-aware local descent strategy, outperforming traditional random mutation methods. Implemented as a web application using React and Flask, the model successfully generated valid timetables for six real student batches. However, the system lacked real-time adaptability, multi-objective optimization, and scalability testing across larger institutional datasets, limiting its broader applicability.

Vignesh et al. [7] proposed a Genetic Algorithm-based solution to address inefficiencies in manual academic timetable generation by introducing a novel chromosome structure. Each chromosome encoded multiple genes containing course, faculty, frequency, group, and room data, allowing a more granular and adaptable representation of timetable data. The system incorporated GA operations including one-point crossover and standard mutation, with the fitness function evaluating constraint satisfaction. Inputs and outputs were managed using XML and HTML formats, respectively, enhancing usability. The model effectively handled both hard constraints and soft constraints, thus successfully generating timetables for Information Technology and Computer Engineering departments. Despite these strengths, the system was limited in scope, lacking scalability across multiple departments or large institutions. Furthermore, faculty preferences and dynamic constraints such as elective course selection or live updates were not addressed.

Mahlous et al. [8] introduced a Genetic Algorithm for student timetabling that prioritizes individual preferences such as free days, preferred class times, and grouping with friends. The algorithm uses a binary chromosome representation to encode student-class assignments and applies customized multi-point crossover, mutation, and repair operators. A simulated annealing-based function enhances soft constraint satisfaction while preserving feasibility. The model also integrates parallel processing, fitnesscaching, and adaptive mutation rates to improve execution time and solution quality. The system achieved over 90% preference satisfaction with no hard constraint violations. However, it is limited to optimizing student allocations for fixed schedules, lacks a user-facing interface, and has not been deployed in real-time institutional settings.

Paramatmuni et al. [9] proposed a web-based timetable generation system using Genetic Algorithms, optimized for scalability and real-time adaptability. The system employs roulette wheel and tournament selection, single-point crossover, and mutation, with a weighted fitness function evaluating hard and soft constraint violations. It achieved 95% conflict resolution and 90% resource utilization on real-world university data, while maintaining execution times under 11 seconds for large datasets. The system is implemented using Django for the backend and HTML/CSS for the frontend, the system outputs schedules in Excel format. Despite its efficiency, the model requires manual weight tuning, offers limited support for collaborative user input, and shows reduced performance under extreme resource constraints.

Febrita et al. [10] proposed a modified Genetic Algorithm (GA) enhanced with fuzzy time windows to optimize high school timetable scheduling. The approach categorizes subjects by cognitive demand and uses fuzzy logic to assign more mentally intensive subjects (e.g., mathematics) to earlier time slots. Chromosomes represent weekly schedules with 210 genes, and a fuzzy-guided mutation prioritizes replacing low-satisfaction time slots. A 2D crossover and a repair mechanism ensure constraint satisfaction and diversity. Despite demonstrating improved fitness scores and convergence compared to standard Gas, the system lacks subject frequency control per day, real-time adaptability, multi-department scalability, and a deployable user interface, limiting its application in broader institutional settings.

Gore et al. [11] proposed an automated timetable scheduling system using a Genetic Algorithm to address the limitations of manual timetable generation in educational institutions. The model generates schedules by encoding each class with course, instructor, and room data, using crossover and mutation operators to evolve conflict-free solutions. A fitness function penalizes overlaps and lecture overloads, and the final timetables are exported in Excel format for user access. The system includes GUI modules for faculty and course input, and generates individualized timetables. While being effective at basic conflict resolution, the system does not account for advanced constraints such as faculty preferences, load balancing, or real-time adaptability. Additionally, it lacks scalability testing on large datasets and offers no support for multi-objective optimization or collaborative scheduling environments.

### Summary of Literature Survey

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl.no.** | **Title** | **Author** | **Methodology** | **Pros & Cons** |
| [1] | Automatic timetable generation using genetic algorithm. | Dipesh Mittal, Hiral Doshi, Mohammed Sunasra, Renuka Nagpure. |  |  |
| [2] | A utilization-based genetic algorithm for solving the university timetabling problem (uga). | Esraa A. Abdelhalim, Ghada A. El Khayat | Utilization-based genetic algorithm: focused on optimizing space utilization alongside satisfying scheduling constraints | Limited to pre-semester plannings i.e., the algorithm does not support dynamic timetable changes. |
| [3] | A genetic algorithm to solve the timetable problem. | Alberto Colorni, Marco Dorigo, Vittorio Maniezzo | Genetic Algorithm with customized genetic operators like row-based crossover, order-k mutation, and a filter. | The system does not provide dynamic adjustments. |
| [4] | A novel genetic algorithm technique for solving university course timetabling problems. | Othman M. K. Alsmadi, S. Za'er, Dia I. Abu-Al-Nadi, Alia Algsoon | Genetic algorithm which involves rank based selection, with a generation limit of the fitness ≥ 0.99. | No dynamic changes can be made. Prioritization of classes are not allowed. |
| [5] | Gradual Optimization of University Course Scheduling Problem Using Genetic Algorithm and Dynamic Programming. | Xu Han, Dian Wang | Hybrid approach which is a combination of Dynamic Programming and Genetic algorithm. | Improved scheduling quality by 46.99% and reduced classroom use by 29.27%. |
| [6] | TIMETABLE GENERATION USING GENETIC ALGORITHM FOR BATCHES UNDER APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY | Ghiridhar S, Sachin A, Edwin M.T, Unnikrishnan K.N | Genetic Algorithm with conflict operator; conflict-aware local descent mutation; implemented via React + Flask web app. | Resolves both hard and soft constraints.  Lacks real-time adaptability |
| [7] | Solving time-table scheduling problem by novel chromosome representation using Genetic algorithm. | Paresh M Chauhan, Kashyap B Parmar, Mahendra B Mendapara | GA with novel chromosome (course, faculty, group, room info) one-point crossover and standard mutation, I/O is managed using XML/HTML interface. | Limited to few departments.No support for faculty preferences or live updates |
| [8] | Student timetabling genetic algorithm accounting for student preferences. | Ahmed Redha Mahlous, Houssam Mahlous | GA with binary chromosome, simulated annealing based function and adaptive mutation. | Efficient execution with parallelism.  Focused only on student allocation |
| [9] | Smart Timetable Generation using Genetic Algorithm. | Sahith Siddharth Paramatmuni, Dumpala Yashwanth Reddy, Elakurthi Sai Spoorthi, Akhil Dharani, K. Venkatesh Sharma | Web-based GA with roulette/tournament selection, weighted fitness function; implemented in Django; outputs Excel files. | High conflict resolution.  Reduced performance under resource constraints |
| [10] | Modified genetic algorithm for high school time-table scheduling with fuzzy time window. | Ruth Ema Febrita, Wayan Firdaus Mahmudy | Modified GA with fuzzy time windows, cognitive load-based slot assignment, fuzzy-guided mutation. | Innovative use of fuzzy logic.  No control over subject frequency |
| [11] |  |  |  |  |
| [12] |  |  |  |  |

**Table 2.1 Observations of Literature Survey**

### Comparison with Existing Systems

Automated timetable scheduling has been the subject of extensive research, with numerous approaches leveraging various optimization algorithms. Traditional manual scheduling methods are prone to inefficiencies, leading to scheduling conflicts, workload imbalances, and increased administrative effort. To address these issues, several advanced computational techniques have been proposed, primarily focusing on Genetic Algorithms (GA) due to their ability to efficiently generate optimized timetables under complex constraints.

Existing systems employing GA-based approaches demonstrate improvements in scheduling efficiency. For instance, some models utilize utilization-based genetic algorithms, optimizing space usage while satisfying hard constraints. Others integrate hybrid methods, combining Genetic Algorithms with Dynamic Programming or Fuzzy Logic to refine results further. While these approaches successfully minimize violations of scheduling constraints, they often lack real-time adaptability and user interactivity. Many of these solutions focus on pre-semester timetable generation, requiring manual adjustments for dynamic changes.

### Proposed System

The Web-Based Automatic Timetable Scheduler is designed to address the complexities and inefficiencies associated with manual scheduling in educational institutions. The system automates the process using Genetic Algorithms (GA) and heuristic optimization techniques to generate conflict-free and optimized timetables while satisfying predefined constraints. The primary objective is to ensure an efficient, scalable, and user-friendly solution that accommodates multiple departments, diverse faculty preferences, and dynamic scheduling adjustments.

The system will leverage constraint satisfaction principles, categorizing conditions into hard constraints (mandatory requirements, such as preventing timetable clashes and ensuring faculty availability) and soft constraints (preferences like distributing lectures evenly and minimizing student gaps). Unlike existing scheduling solutions, which often require manual intervention for modifications, the system incorporates real-time adaptability, allowing administrators to make necessary changes dynamically.

### Objectives

The primary objective of the Web-Based Automatic Timetable Scheduler is to streamline the scheduling process for schools and colleges by automating timetable generation through optimization techniques. The system aims to eliminate inefficiencies and human errors in manual scheduling while ensuring scalability and adaptability to diverse academic environments.

The key objectives of the proposed system include:

* To develop an intelligent scheduling system that generates conflict-free academic timetables based on faculty workload, subject allocations, and classroom availability.
* To implement a structured approach to accommodate hard constraints and soft constraints.
* To utilize Genetic Algorithms, heuristic techniques, and resource scheduling approaches to optimize timetable allocation efficiently.
* Enabling faculty and administrators to adjust schedules dynamically through an interactive user interface, ensuring flexibility in handling last-minute changes.
* To design an intuitive web-based platform with role-based access control for administrators, faculty, and students, allowing easy navigation and interaction.
* To ensure that the system is scalable and can efficiently handle timetable generation for colleges with multiple departments and diverse elective course.
* To minimize manual scheduling efforts by automating data processing, faculty assignments, and subject allocations while maintaining scheduling efficiency.
* To implement secure authentication mechanisms to protect sensitive academic data.

# CHAPTER 3

## REQUIREMENT SPECIFICATION AND ANALYSIS

### 3.1 Introduction

An efficient automated timetable scheduling system requires well-defined specifications to ensure smooth implementation. This chapter details the functional, user interface, integration, software, and hardware requirements that govern the development of this system.

### 3.2 Functional Requirements

The system must fulfil several key functional requirements to support automatic timetable generation effectively:

#### 3.2.1 Timetable Generation

* The system automatically generates conflict-free schedules, considering faculty workload, subject allotment, and room availability.
* Allow adjustments based on faculty preferences (such as preferred teaching slots).

#### 3.2.2 Constraint Handling

**Hard Constraints** (must be strictly satisfied):

* + No clashes between classes in the same room.
  + No instructor assigned to multiple classes at the same time.
  + Courses assigned according to department requirements.

**Soft Constraints** (should be optimized but can be violated if necessary):

* Minimizing gaps in students’ schedules.
* Distributing lectures evenly throughout the week.

#### 3.2.3 User Access Control

* The system implements role-based access for administrators, faculty, and students.
* Administrators have the ability to modify schedules, while faculty can request changes and students can view their timetable.

#### 3.2.4 Data Input & Management

**Faculty Details**: Names, designations, availability.

**Course Details**: Subject names, codes, types (lecture/lab), credit hours.

**Classroom Allocation**: Room availability and capacity.

**Semester and Student Batch Details**: Number of students, group divisions, elective choices.

#### 3.2.5 Optimization Algorithms

The system incorporates genetic algorithms, heuristic approaches, and resource scheduling techniques to efficiently generate optimized timetables.

#### 3.2.6 Report Generation & Export

* Timetables should be exportable in multiple formats (Excel, PDF).
* Users should be able to print timetables or share digital copies easily.

### 3.3 User Interface Requirements

For ease of use, the system’s interface should be:

**Intuitive**: Designed for quick navigation and clear data visualization.

**Accessible**: Mobile-friendly for easy timetable access on various devices.

**Graphical**: Interactive visual timetables for better readability.

**Customizable**: Administrator should be able to modify preferences dynamically.

### 3.4 Software Requirements

The successful development and deployment of the Automatic Timetable scheduler rely on a robust and well-defined software environment. These software requirements ensure that the system operates efficiently, remains secure, and supports scalability.

* **Programming Language**

The system is built using Python 3.10, a versatile and powerful programming language widely used for web development. Python's extensive library ecosystem and compatibility with Django make it an ideal choice for this project.

* **Framework**

The Django Framework (version 4.x) is utilized as the primary web development framework. Django's built-in features, such as authentication, ORM (Object-Relational Mapping), and admin panel, streamline the development process and reduce time-to market. Its Model-View-Template (MVT) architecture ensures a clean separation of concerns, making the application easier to scale and maintain. Django's security mechanisms, such as protection against SQL injection and cross-site scripting (XSS), ensure that the platform adheres to industry standards.

* **Web Browser**

The system is designed to be compatible with modern web browsers such as Google Chrome, Mozilla Firefox, and Microsoft Edge. This ensures a consistent user experience across different platforms, including Windows, macOS, and Linux. The use of responsive web design techniques further ensures usability on mobile and tablet devices.

* **Operating System**

The system is platform-independent and can run on any OS that supports Python and Django. Commonly used operating systems include:

* + Linux (Ubuntu): Suitable for development and production due to its stability and extensive community support.
  + Windows and macOS: Suitable for local development and testing.

### 3.6 Hardware Requirements

Processor: Intel Core i5 and above

Processor Speed: 1.9GHz

System Type: 64-Bit Operating System

HDD: 100GB and above

RAM: 4-GB RAM and above

## REFERENCES

[1] Mittal, Dipesh, Hiral Doshi, Mohammed Sunasra, and Renuka Nagpure. "Automatic timetable generation using genetic algorithm. *“International Journal of Advanced Research in Computer and Communication Engineering* 4, no. 2 (2015): 245-248.

[2] Abdelhalim, Esraa A., and Ghada A. El Khayat. "A utilization-based genetic algorithm for solving the university timetabling problem (uga)." *Alexandria Engineering Journal* 55, no. 2 (2016): 1395-1409.

[3] Colorni, Alberto, Marco Dorigo, and Vittorio Maniezzo. "A genetic algorithm to solve the timetable problem." *Politecnico di Milano, Milan, Italy TR* (1992): 90-060.

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

[5] Han, Xu, and Dian Wang. "Gradual Optimization of University Course Scheduling Problem Using Genetic Algorithm and Dynamic Programming." *Algorithms* 18, no. 3 (2025): 158.

[6] Ghiridhar, S., A. Sachin, M. T. Edwin, and K. N. Unnikrishnan. "TIMETABLE GENERATION USING GENETIC ALGORITHM FOR BATCHES UNDER APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY." (2020).

[7] Chauhan, Paresh M., Kashyap B. Parmar, and Mahendra B. Mendapara. "Solving time-table scheduling problem by novel chromosome representation using Genetic algorithm." In *2015 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2015]*, pp. 1-6. IEEE, 2015.

[8] Mahlous, Ahmed Redha, and Houssam Mahlous. "Student timetabling genetic algorithm accounting for student preferences." *PeerJ Computer Science* 9 (2023): e1200.

[9] Paramatmuni, Sahith Siddharth, Dumpala Yashwanth Reddy, Elakurthi Sai Spoorthi, Akhil Dharani, and K. Venkatesh Sharma. "Smart Timetable Generation using Genetic Algorithm."*Macaw International Journal of Advanced Research in Computer Science and Engineering* 10, no. 1s (2024): 204-215.

[10] Febrita, Ruth Ema, and Wayan Firdaus 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)*, pp. 88-92. IEEE, 2017.