**PYTHON TIMETABLE GENERATOR**

**MINOR PROJECT REPORT**

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF

**BACHELOR OF TECHNOLOGY**  
(Computer Science and Engineering)



Submitted By: Submitted To:  
 Kirti (17032456767) Dr. Harjot Kaur

**Department of Computer Science and Engineering  
 Guru Nanak Dev University, Regional Campus**,

Gurdaspur, India

**September, 2025**

# Acknowledgement

I would like to express my deep gratitude to my project supervisor, **Dr. Harjot Kaur, Department of Computer Science and Engineering, Guru Nanak Dev University Regional Campus, Gurdaspur**, for her invaluable guidance, encouragement, and continuous support throughout this mini project. Her valuable suggestions and motivation helped me in successfully completing the work on **“Timetable Generator using CSP”**.

Her expert suggestions, timely feedback, and technical insights have been instrumental in the successful completion of this project.

I would also like to extend my heartfelt thanks to the entire **faculty and staff of the Department of Computer Science and Engineering** for providing the necessary facilities, environment, and resources for the execution of this work.

Lastly, I express my deep appreciation to my **family and friends** for their constant encouragement, motivation and moral support throughout this journey.

**SIGNATURE OF STUDENT**

Kirti (17032456767)

**Certificate**

This is to certify that the mini project entitled “Timetable generator using CSP” has been carried out by **Kirti** (roll no: 17032456767) under the supervision of **Dr. Harjot Kaur, Department of Computer Science and Engineering, Guru Nanak Dev University,** as a part of the partial fulfillment of the requirements for the degree of Bachelor of Technology (Computer Science and Engineering), Semester 3.

**SUPERVISOR:** DR. HARJOT KAUR

**SIGNATURE OF STUDENT**  
Kirti (17032456767)

**DATE:**

This is certifying that the above statement made by candidate is correct to the best of my knowledge.

# 3. Abstract

The project, titled **“Timetable Generator using CSP (Constraint** **Satisfaction Problem)”**, focuses on developing an intelligent system that automates the generation of academic timetable using **the Constraint Satisfaction Problem (CSP)** approach. In most educational institutions, timetable scheduling is performed manually, which is time-consuming, error-prone, and difficulty to modify. This project provides a computational solution that generates conflict-free timetables automatically, satisfying all given constraint such as teachers availability, subject limitations, and time slot restrictions.

The main objectives of this project are:

1. To design a timetable generation system that applies CSP logic and backtracking algorithm.

2. To eliminate overlapping of subjects and teachers during scheduling.

3. To allow users to input data such as teachers, subjects, and available time slots through a graphical user interface (GUI).

4. To display the generated timetable in a user-friendly format.

5. To improve the accuracy and efficiency of academic scheduling.

The methodology involves defining the problem in terms of variable, domains, and constraints, followed by applying CSP-solving techniques using Python programming. The system is implemented using Tkinter for GUI creation and logical algorithms for constraint checking.

The results demonstrate that the system efficiently generates valid and non-overlapping timetables within seconds. The applications of this project lies in its ability to assist schools, colleges, and universities in managing their schedules automatically, thereby saving time and minimizing human errors. The project showcases the effective use of artificial intelligence principles in solving real-world academic management problem.

**3.1 Problem Statement**

Creating a timetable for an educational institution is a complex and time-consuming process. Each teacher, subject, and class must be assigned appropriate slots while ensuring that there are no overlaps. The task becomes even more complicated when constraints such as teacher availability, subject credit hours, and room capacity are considered.

Traditionally, this process is done manually by administrative staff, which often results in scheduling conflicts, unbalanced workloads, and frequent modifications. Any small change, such as a teacher’s leave or subject adjustment, requires reworking large portions of the timetable.

Therefore, there is a clear need for an **automated, intelligent, and constraint-based timetable generation system** that can efficiently allocate resources, handle constraints, and produce valid timetables without human errors.

**3.2 Objectives**

The main objectives of this project, **“Timetable Generator using CSP”**, are as follows:

1. **To automate the process** of timetable generation using Python programming.
2. **To apply CSP (Constraint Satisfaction Problem) techniques** for scheduling tasks efficiently.
3. **To ensure conflict-free allocation** of subjects, teachers, and time slots using constraints.
4. **To develop a user-friendly interface (GUI)** using Tkinter for data input and output visualization.
5. **To minimize time and human effort** in the timetable creation process.
6. **To provide flexibility** for modifications and easy updates to the generated timetable.

These objectives make the system reliable, accurate, and adaptable for real academic use.

**3.3 Methodology Followed**

The project follows a structured approach using the principles of the **Constraint Satisfaction Problem (CSP)** and **Backtracking Algorithm**.

1. **Requirement Analysis:**  
   The first step was to identify key requirements — such as teacher names, subjects, number of classes, and available time slots.
2. **System Design:**
   * The timetable problem was represented as a CSP model where:
     + Variables = Subjects or Classes
     + Domains = Possible teachers or time slots
     + Constraints = No teacher or subject repetition in the same slot
   * A **flowchart and system diagram** were created to visualize the scheduling logic.
3. **Algorithm Implementation:**
   * The **Backtracking Algorithm** was used to generate valid combinations that satisfy all constraints.
   * If a constraint was violated (like teacher overlap), the system backtracked to try another assignment.
4. **GUI Design:**
   * The **Tkinter** library in Python was used to create an easy-to-use interface.
   * Users can enter subjects, teachers, and slots and generate the timetable by clicking a button.
5. **Testing and Output Generation:**
   * The generated timetables were tested for accuracy and constraint satisfaction.
   * Output was displayed in a grid/table format within the GUI.

This methodology ensures that the timetable generated is valid, non-overlapping, and visually clear.

**3.4 Key Findings and Outcomes**

After implementation and testing, the project achieved the following outcomes:

* The timetable generation process was **fully automated** and produced **error-free results**.
* The system generated a valid timetable in **seconds**, compared to hours of manual scheduling.
* **CSP and backtracking** techniques efficiently handled all constraints and combinations.
* The **GUI interface** made the system accessible to non-technical users.
* The project demonstrated how **AI-based logical modeling** can solve real administrative challenges.

The generated timetables were tested for multiple scenarios (different subjects and teachers), and the system maintained consistency and correctness in all cases.

**3.5 Significance and Applications**

The **Timetable Generator using CSP** holds significant value for schools, colleges, and universities. It reduces the workload of faculty and administrative staff and ensures accurate and balanced timetables.

**Practical applications include:**

* **Academic institutions:** Automatic generation of lecture schedules for different departments or semesters.
* **Examination departments:** Slot allocation for exams with teacher invigilation planning.
* **Corporate training:** Scheduling of sessions based on trainer availability and topics.

Beyond practical use, the project also demonstrates the **power of Artificial Intelligence** and **Constraint Programming** in problem-solving. It enhances understanding of key AI concepts like search space, constraint satisfaction, and optimization.

In summary, this project provides an efficient, flexible, and intelligent solution for timetable management while serving as a strong example of how theoretical computer science can be applied to real-world administrative systems.

**4. Introduction**

**4.1 Background and Context**

In any educational institution, creating a timetable is one of the most essential yet challenging administrative tasks. It involves assigning subjects to specific time slots and teachers in a way that avoids overlapping, ensures balanced workloads, and fulfills institutional requirements. When done manually, this process often becomes time-consuming and error-prone, especially when handling large numbers of classes, subjects, and teachers.

Traditional timetable creation relies on human effort and intuition, which can lead to conflicts such as the same teacher being assigned to two classes at the same time or unbalanced distribution of lectures throughout the week. Additionally, any change in teacher availability or subject allocation requires revising the entire timetable, consuming even more time.

With advancements in **computer science and artificial intelligence**, these scheduling problems can be modeled using computational techniques. One of the most effective approaches is the **Constraint Satisfaction Problem (CSP)** method. In this approach, the problem is represented as a set of variables, their possible values (domains), and the constraints that define which combinations are valid. The CSP framework allows the computer to explore different assignments efficiently while maintaining all constraints.

This project, titled **“Timetable Generator Using CSP”**, applies this principle to automatically generate timetables. It integrates the CSP concept with **Python programming** and a **Tkinter-based GUI**, enabling users to easily input data and visualize the generated timetable. The approach not only ensures accuracy and efficiency but also demonstrates the practical use of theoretical computer science concepts in solving real-world problems.

**4.2 Relevance and Motivation**

The motivation behind this project arises from the increasing need for automation in academic and administrative systems. Manual timetable generation consumes several working hours of faculty or administrators every semester and is prone to human errors. A **computerized timetable generator** can significantly reduce this workload while ensuring logical and conflict-free scheduling.

The **relevance** of this project lies in its ability to integrate artificial intelligence techniques—specifically CSP—with programming to build a smart solution that can adapt to different constraints like teacher availability, subject load, and class timing. CSP-based scheduling systems are already being applied in various fields like airline scheduling, exam timetables, and resource allocation. Implementing such a system in an educational context demonstrates how real-world problems can be modeled and solved computationally.

Additionally, this project enhances understanding of **AI search techniques, problem modeling, and backtracking algorithms**. It helps bridge the gap between theoretical concepts studied in computer science and their practical implementation.

In short, this project was chosen because it provides a realistic and educational application of CSP while solving a genuine, recurring problem faced in academic environments. It promotes both **technical learning** and **practical efficiency**, making it a valuable contribution for students, teachers, and institutions.

**5. Methodology / System Design**

The proposed system utilizes the Constraint Satisfaction Problem (CSP) approach to generate feasible and conflict-free timetables. The Python ‘constraint’ library is used to define and solve constraints between subjects, teachers, and time slots. The system’s design includes three main modules: Input Module, CSP Solver, and Output Display.

The process includes the following steps:

1. Input of subjects, teachers, and number of periods per week.

2. Definition of days and period timings.

3. Application of CSP constraints using AllDifferent and custom teacher conflict constraints.

4. Solving the CSP to assign subjects to available time slots.

5. Displaying the final timetable using Tkinter GUI.

The overall architecture of the system is represented through the following block diagram. It show the flow of information from user input to timetable generation through various modules.

User Input GUI

(Tkinter Interface)

* Enter subjects
* Assign teachers
* Set days & timing

Data Processing Unit

(Input Validation Layer)

* Stores subjects, teachers

and period data

* Prepare CSP variables

Constraint Satisfaction Solver

(python-constraint libraray)

* Define constraints:  
  - No teacher conflicts

- No subject overlap

- Balanced period allocation

* Generate valid timetable slot

Timetable generation

(output formatting &GUI display)

* Display in tabular form
* Highlight breaks

As shown in the figure above, the system consists of three main modules – the input Module, CSP Solver, and Output Module. The input module takes subject and teacher data, the CSP solver applies constraint logic, and the output module displays the generated timetable in a tabular GUI format.

**6. Implementation / Experimental Setup**

6.1 Hardware Configuration:

The project was developed and tested on a standard personal laptop with the following hardware configuration:

**Component Specification**

Processor Intel core i5 10th gen

RAM

Storage

Operating system

Display

The system does not require high-end hardware. Any modern computer or laptop capable of running Python 3 and Tkinter is sufficient.

No internet connection is required after installation.

Execution time for generating a complete timetable is usually under 2 second.

**6.2 Simulation Environment**

The project was implemented using the Python programming language, which provides powerful built-in-libraries for logic and GUI development.

# 7. Results and Discussion

**7.1 Results**

The Timetable Generator system was tested with different sets of teacher-subject combinations and time slots to evaluate the effectiveness of the scheduling logic. The system generated conflict-free timetables in all test cases.

Below is a sample output generated for a single class with 5 subjects and 4 teachers across 5 days and 5 time slots per day:

📊 Sample Generated Timetable (Class A)

# 

Fig 1: GUI Screenshot showing the generated timetable for class A

**7.2 Observations and Interpretations**

* The system successfully assigned all subjects to available time slots without violating constraints (no teacher is scheduled for two subjects in the same slot).
* Each subject appears multiple times per week, meeting lecture requirements.
* Teachers are not overloaded and do not appear in two places at once.
* The timetable varies each time it's generated (when using randomized assignment), showing flexibility.
* Execution time is fast—most timetables are generated in less than 2 seconds.

💡 Interpretation:  
The CSP-based algorithm is effective in modeling real-world scheduling scenarios and ensures conflict-free and balanced distribution of periods across a week. It demonstrates how theoretical AI models can solve practical administrative problems.

**7.3 Limitations and Challenges Faced**

While the system performed well for small- to medium-sized datasets, a few limitations and challenges were observed:

1. 🧠 Algorithm Scalability:  
 With an increase in the number of subjects, classes, and constraints, the backtracking algorithm may become slower due to exponential growth in combinations.

2. ⚠️ No Custom Constraint UI:  
 Currently, constraints like "teacher unavailable at certain hours" or "preferred slots" are hardcoded and not customizable by the user.

3. 📂 No Export Feature:  
 The generated timetable is viewable on the GUI only; there is no current option to export to Excel, PDF, or print formats.

4. 👥 Multi-Class Timetables:  
 The current version supports a single class. Extending it to handle multiple classes simultaneously requires more complex logic and additional constraint handling.

5. 📅 No Breaks or Holidays:  
 Lunch breaks or holidays are not considered in the current timetable layout, which may be essential in real-world scenarios.

**7.4 Summary**

The results of this project indicate that the CSP-based approach provides an efficient and reliable method for generating academic timetables. The system fulfills its primary objectives and demonstrates significant potential for further development and deployment in educational settings. Future improvements can help overcome current limitations and make the system scalable for institutional use.

**8. Conclusion and Future Scope**

**8.1 Conclusion:**

The Timetable Generator Using CSP project successfully demonstrated how artificial intelligence principles—particularly Constraint Satisfaction Problems and backtracking algorithms—can be applied to solve real-world scheduling challenges in academic environments. The system automated the process of assigning subjects, teachers, and time slots in a way that eliminates conflicts and satisfies all predefined constraints.

The key findings and achievements of the project are as follows:

* A functional and interactive timetable generator was implemented using Python and Tkinter.
* The system used a CSP model where variables, domains, and constraints were effectively defined.
* A backtracking algorithm was successfully integrated to solve the scheduling problem with constraint checking at each step.
* The generated timetables were accurate, conflict-free, and visually well-formatted.
* The project significantly reduced the time and effort required to generate a schedule compared to manual methods.

This solution met all the original objectives of the project, including automated generation, constraint handling, conflict prevention, and ease of use through a GUI. The system was tested with multiple inputs and performed as expected across all cases.

Overall, the project not only served its practical purpose but also showcased how theoretical AI concepts like CSPs can be translated into real, usable applications.

**8.2 Future Work:**

While the current version of the system fulfills the basic requirements of timetable generation, there are several areas for future enhancement and expansion:

1. 📁 Data Storage & Export:  
 Add functionality to export the generated timetable as a PDF, Excel file, or image. This would allow users to print or share timetables easily.

2. 📅 Multi-Class Support:  
 Extend the system to handle multiple classes or departments simultaneously, each with its own set of subjects and teachers.

3. 📶 Dynamic Constraint Handling:  
 Allow users to define custom constraints such as room assignments, teacher preferences, or unavailable time slots.

4. 👩‍🏫 Teacher Load Balancing:  
 Incorporate logic to ensure balanced distribution of lecture hours among faculty members based on workload preferences or contracts.

5. 📲 Web-Based Version:  
 Develop a web-based or mobile version using frameworks like Flask, Django, or React to improve accessibility and scalability.

6. 🧠 Use of Advanced AI Techniques:  
 Explore more advanced CSP solvers or optimization algorithms like Genetic Algorithms, Simulated Annealing, or SAT solvers for large-scale problems.

By implementing these future improvements, the system can evolve from a prototype mini-project to a robust, full-scale academic scheduling tool adaptable to real institutional requirements.