

Project Report: Timetable Generation System

1. Abstract

The Timetable Generation System is an advanced automated solution designed to solve the complex optimization problem of academic scheduling. Utilizing a custom implementation of a Genetic Algorithm (GA), the system generates conflict-free schedules that respect teacher availability, room constraints, and subject priorities. This report details the architectural design, algorithmic logic, and technical implementation of the system, showcasing its ability to streamline administrative workflows and enhance educational efficiency.

2. Introduction

In modern educational institutions, creating a balanced and efficient timetable is a daunting task. Administrators must juggle multiple variables, including diverse subject requirements, finite classroom resources, and individual teacher schedules. Manual methods are often error-prone, leading to scheduling overlaps and inefficient resource utilization. Over the years, the need for an automated system has grown, and this project addresses that need by using Artificial Intelligence to find optimal solutions in a vast search space.

3. Problem Statement

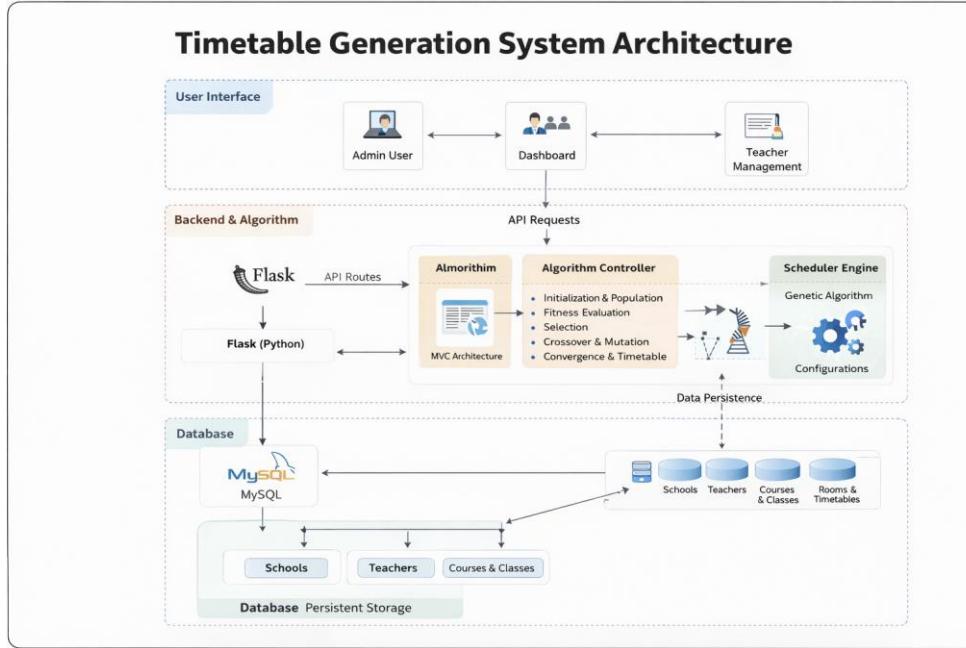
Timetable generation involves managing hard and soft constraints. Hard constraints, such as avoiding teacher and room conflicts, must always be satisfied. Soft constraints, like prioritizing core subjects and evenly distributing lectures, improve the overall quality of the schedule.

4. Technical Stack

- Backend Framework: Flask (Python) – Handles API routing.
- Database Management: MySQL – Stores institutional data.
- Algorithmic Foundation: Genetic Algorithm – Custom implementation.
- Frontend Technologies: HTML5, CSS3, JavaScript – Dashboard for management.

5. System Architecture

The application follows the Model-View-Controller (MVC) architectural pattern.



Data Flow:

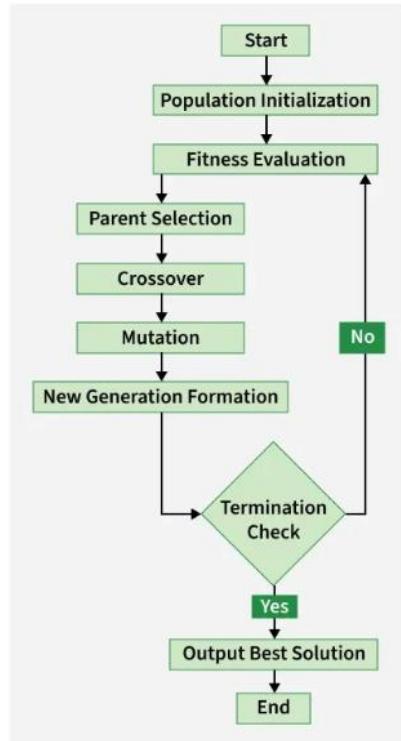
1. Input: Admin enters details about faculty and rooms.
2. Processing: Flask backend initializes the Genetic Algorithm engine.
3. Optimization: The GA performs generations of evolution.
4. Persistence: The best-fit timetable is stored in MySQL.
5. Output: Result rendered on dashboard.

6. Detailed System Design: Database Schema

The database design is centralized around the 'schools' table to support multi-tenancy. Key tables include:

- Schools: Stores credentials and global parameters.
- Course & Class: Defines academic structure.
- Teacher: Faculty records.
- Subject: Links subjects to faculty and credits.
- Room: Stores space details like 'Lecture Hall' or 'Practical Lab'.
- Timeslot: Defines time intervals.
- Timetable: Final mapping output.

7. Algorithmic Logic: Genetic Algorithm



- Initialization: Population of schedules generated.
- Fitness Evaluation: Scored with penalties for overlaps.
- Selection: Elite parents chosen for next generation.
- Crossover & Mutation: Creating diversity in schedules.
- Convergence: Repeats until maximum generations reached.

8. Key System Features

- Multi-School Capability: Isolated data management.
- Dynamic Scheduling: Immediate regeneration on variable change.
- Priority Optimization: Weighted core subject allocation.
- Validation Suite: Data integrity checks.

9. Conclusion & Future Scope

The system demonstrates the power of Genetic Algorithms in solving complex scheduling. It produces higher quality schedules efficiently.