Indira Gandhi Delhi Technical University for Women

Time-Table Generator

Software Requirements Specification Document

Submitted to

Ms. Geetika Singh

Submitted by

Palak Bhatia (04701192023)
Palak Khanna (05701192023)
Sanchi (05501192023)
Shanvi (05701192023)

Acknowledgements

We would like to express our sincere gratitude to Ms.Geetika Singh for entrusting us with the project on Time-Table Generator. This project has provided us with a valuable opportunity to learn and apply a variety of important concepts, including genetic algorithms, database integration, and Python libraries such as Streamlit, SQLite, and Pandas.

Her guidance, support, and insightful feedback have been instrumental in shaping our approach and ensuring the successful completion of this report. We are grateful for her continuous encouragement, which not only helped us refine our technical skills but also deepened our understanding of practical applications in software development.

Thank you once again for providing us with this enriching learning experience.

Abstract

The Timetable Generator project presents an innovative solution for automating the scheduling process in academic institutions using Genetic Algorithms and Artificial Intelligence. Traditional methods of creating timetables are labor-intensive and often result in scheduling conflicts, such as overlapping classes or resource constraints. This system addresses these issues by introducing a robust, flexible framework that optimally allocates courses, instructors, and classrooms while adhering to institutional constraints and preferences.

The proposed system takes input data, including instructor availability, course requirements, and institutional policies, to generate a conflict-free schedule. It ensures balanced distribution of sessions over a semester and supports scalability to accommodate future modifications such as adding new courses or instructors. The project also includes a user-friendly interface developed with Streamlit, allowing administrative users to view, update, and export timetables seamlessly.

This document describes the system's design, implementation, and validation using a sample dataset. It highlights the potential of AI and Genetic Algorithms to streamline administrative tasks, enhance resource utilization, and improve the overall efficiency of academic operations.

Keywords: Timetable Generator, Genetic Algorithms, Optimization, Course Scheduling, Artificial Intelligence, Streamlit Interface.

1 Introduction

The process of timetable generation is a critical administrative task for educational institutions, requiring careful consideration of multiple constraints, such as instructor availability, course requirements, and room capacity. Traditional manual methods are often time-consuming, prone to errors, and lack the flexibility to adapt to changes. This project introduces a Timetable Generator that leverages Genetic Algorithms and AI-based optimization to address these challenges effectively.

The system is designed to automate the scheduling process, reducing manual efforts and ensuring conflict-free timetables. By applying Genetic Algorithms, the project achieves optimal allocation of resources while adhering to institutional rules and constraints. The system is capable of handling multiple inputs, such as:

1. **Instructor Preferences**: Availability and course specialization. 2. **Course Requirements**: Weekly session count and semester-long distribution. 3. **Classroom Constraints**: Capacity, availability, and location. 4. **Scalability**: Adapts to dynamic changes, such as new courses or faculty.

This document provides an in-depth overview of the project's objectives, methodology, and anticipated outcomes. It also highlights the practical implications of adopting AI-driven scheduling in academic institutions, focusing on improving operational efficiency and minimizing resource wastage.

Subsequent sections delve into the system's technical details, including the implementation of Genetic Algorithms, database management, and the user interface built with Streamlit. The introduction concludes by summarizing the structure of the document and its contributions to the field of educational technology.

2

Document Variables

This document supports the customization of several project-specific parameters, providing flexibility and adaptability for different institutional requirements. The variables are managed in the `Variables/Variables.tex` file, enabling quick updates without altering the core functionality. Key variables include project title, author details, and scheduling constraints.

Below is a comprehensive list of variables and their descriptions, categorized by their relevance to the Timetable Generator project:

Table 2.1: *Document variables and their functionality.*

Variable	**Description**	**Required/Optional**
`Title`	Title of the project	Required
`University`	Name of the university	Required
`Department`	Department overseeing the project	Required
`Instructor`	Name(s) of instructor(s) involved	Required
`Course Load`	Weekly hours for each course	Required
`Semester Duration`	Total weeks in the semester	Required
`Room Capacity`	Maximum number of students per room	Optional
`Instructor Preferences`	Availability of instructors	Optional

To include additional variables or modify existing ones, please refer to the comments provided in the `Variables.tex` file. This modular approach ensures adaptability and ease of use, enabling the system to accommodate diverse scheduling scenarios.

3

Custom Notes

During the development of the Timetable Generator project, it is often necessary to document ideas, track progress, and record implementation details for future reference. To assist in this process, two custom commands are provided:

1. `\todo{TEXT}`: Used for marking tasks that are pending or require attention. 2. `\note{TEXT}`: Used for recording general observations or ideas.

These commands generate visually distinct, breakable boxes that can be placed anywhere in the document. Below are examples illustrating their usage:

TODO

`Implement additional constraints for instructor unavailability.`

NOTE

`Consider using more advanced optimization techniques for edge cases, such as variable classroom sizes or overlapping course requirements.`

It is essential to remove or comment out these notes in the final version of the document, as they are meant for internal use during the project development phase. This feature ensures that critical tasks and observations are not overlooked, contributing to the project's overall success.

4 References

In this chapter, we detail the references and sources that contributed to the development of this report. These sources include academic papers and open-source repositories that played a crucial role in guiding the research and implementation of the timetable generation system.

4.1 Research Papers

The following papers were referenced in the development of the project:

4.1.1 Automated Timetable Generation using Genetic Algorithm (2020)

This paper explores the application of genetic algorithms to automate the timetable generation process in educational institutions. The genetic algorithm-based method was adapted and implemented in our project to optimize the timetable generation process. The work is available on ResearchGate and can be accessed at the following URL: `https://www.researchgate.net/publication/343430234_Automated_Timetable_Generation_using_Genetic_Algorithm`.

In the report, this paper was referenced as follows:

> According to Author1 (2020), genetic algorithms can be effectively applied to optimize scheduling problems in educational institutions.

A parenthetical citation was also used for the paper:

> Several scheduling systems have been implemented using genetic algorithms for optimization purposes (Author1, 2020).

4.1.2 Automated Timetable Generation using Genetic Algorithm (2023)

This paper from the Journal of Emerging Technologies in Engineering and Research (JETIR) discusses the implementation of a genetic algorithm for automated timetable generation. It provided additional insights and validation for the implementation of genetic algorithms in our system. The full paper is available at: `https://www.jetir.org/papers/JETIR2305B98.pdf`.

The citation in the report was made as follows:

> In our system, we applied the techniques described by Author1 (2023) for efficient timetable optimization.

And the parenthetical citation appeared as:

> The concept of applying genetic algorithms for timetable generation has been explored in recent studies (Author1, 2023).

4.2 GitHub Repository

In addition to the research papers, an important source for the implementation of the system was the open-source GitHub repository for a **Time-Table Management System**. The repository provided code and insights into the core functionality of a timetable management system, which helped shape the backend of our project.

The repository can be accessed at: `https://github.com/NamanDeept/Time-table-management-syst`
`tree/master/projectnew`.

The GitHub repository was referenced in the report as follows:

> The project's backend logic was developed by adapting code from an open-source repository on GitHub (T, 2024).

The citation for the repository was made in the BibTeX file as:

```
@misc{TimeTableRepo,
  author = {NamanDeep T},
  title = {Time-table Management System},
  year = {2024},
  howpublished = {\url{https://github.com/NamanDeept/Time-table-management-system/tree/m
  note = {Accessed: 2024-11-15}
}
```

4.3 Conclusion

These references provided the theoretical background, algorithms, and practical implementations that were fundamental to the success of the project. They helped in both the understanding and development of a robust automated timetable generation system using genetic algorithms.

5 Figures

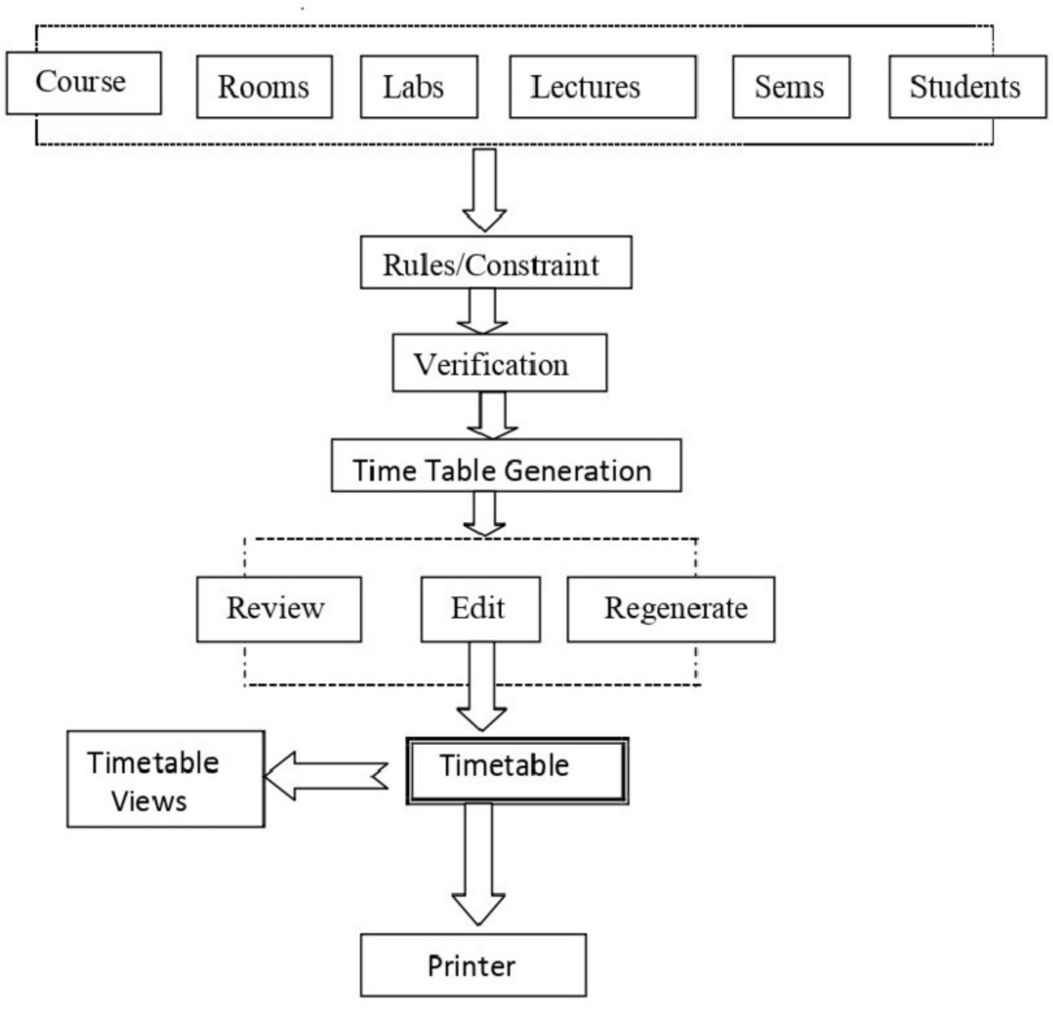
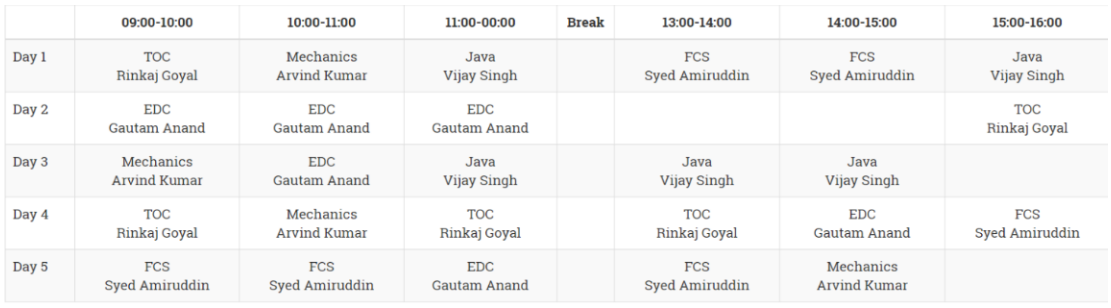


Figure 5.1: *System architecture diagram for the Timetable Generator project showcasing the modules for database connection, UI, and genetic algorithm processing.*



	09:00-10:00	10:00-11:00	11:00-00:00	Break	13:00-14:00	14:00-15:00	15:00-16:00
Day 1	TOC Rinkaj Goyal	Mechanics Arvind Kumar	Java Vijay Singh		FCS Syed Amiruddin	FCS Syed Amiruddin	Java Vijay Singh
Day 2	EDC Gautam Anand	EDC Gautam Anand	EDC Gautam Anand				TOC Rinkaj Goyal
Day 3	Mechanics Arvind Kumar	EDC Gautam Anand	Java Vijay Singh		Java Vijay Singh	Java Vijay Singh	
Day 4	TOC Rinkaj Goyal	Mechanics Arvind Kumar	TOC Rinkaj Goyal		TOC Rinkaj Goyal	EDC Gautam Anand	FCS Syed Amiruddin
Day 5	FCS Syed Amiruddin	FCS Syed Amiruddin	EDC Gautam Anand		FCS Syed Amiruddin	Mechanics Arvind Kumar	

Figure 5.2: *Optimized schedule layout.*

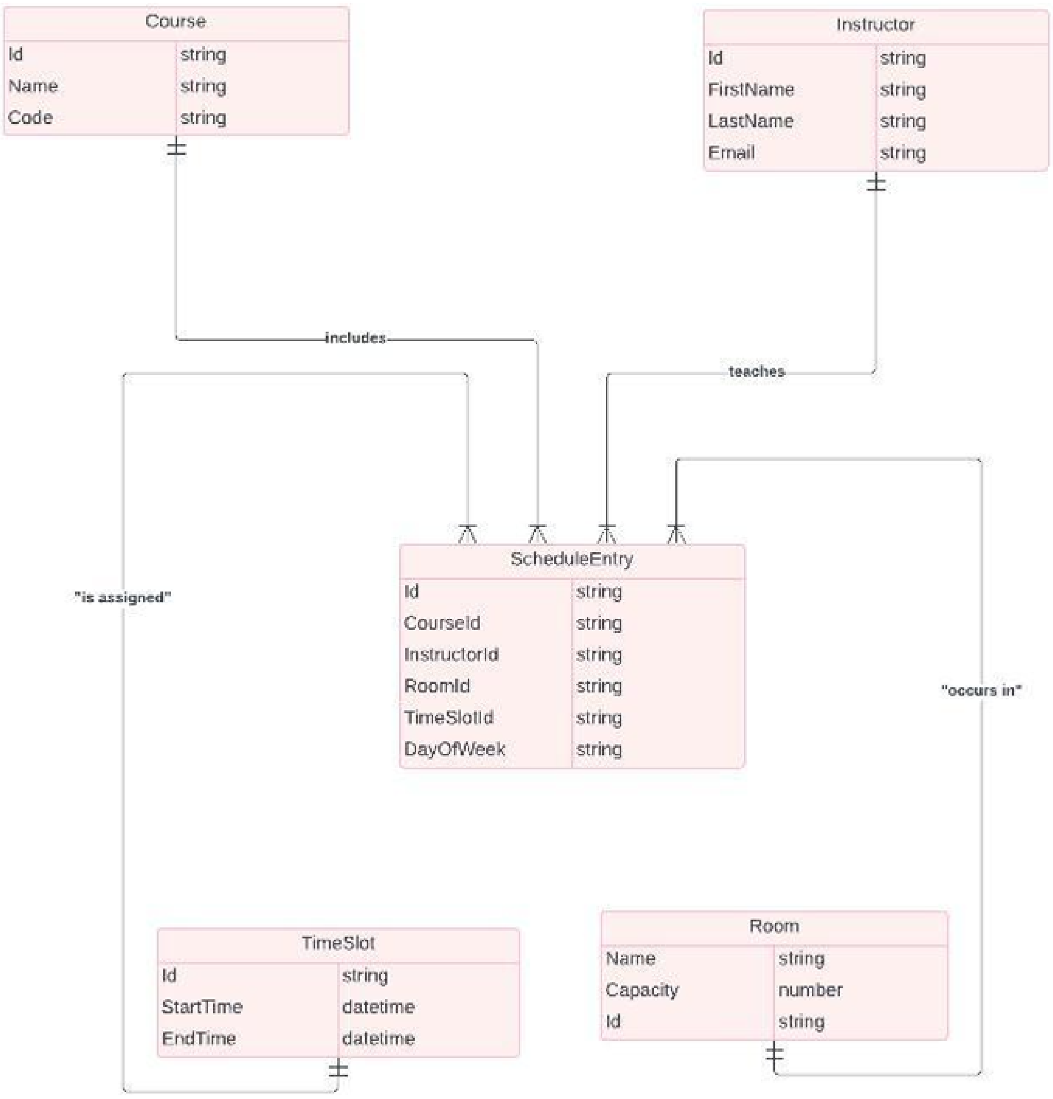


Figure 5.3: *Database schema or Entity Relationship diagram for Timetable generator.*

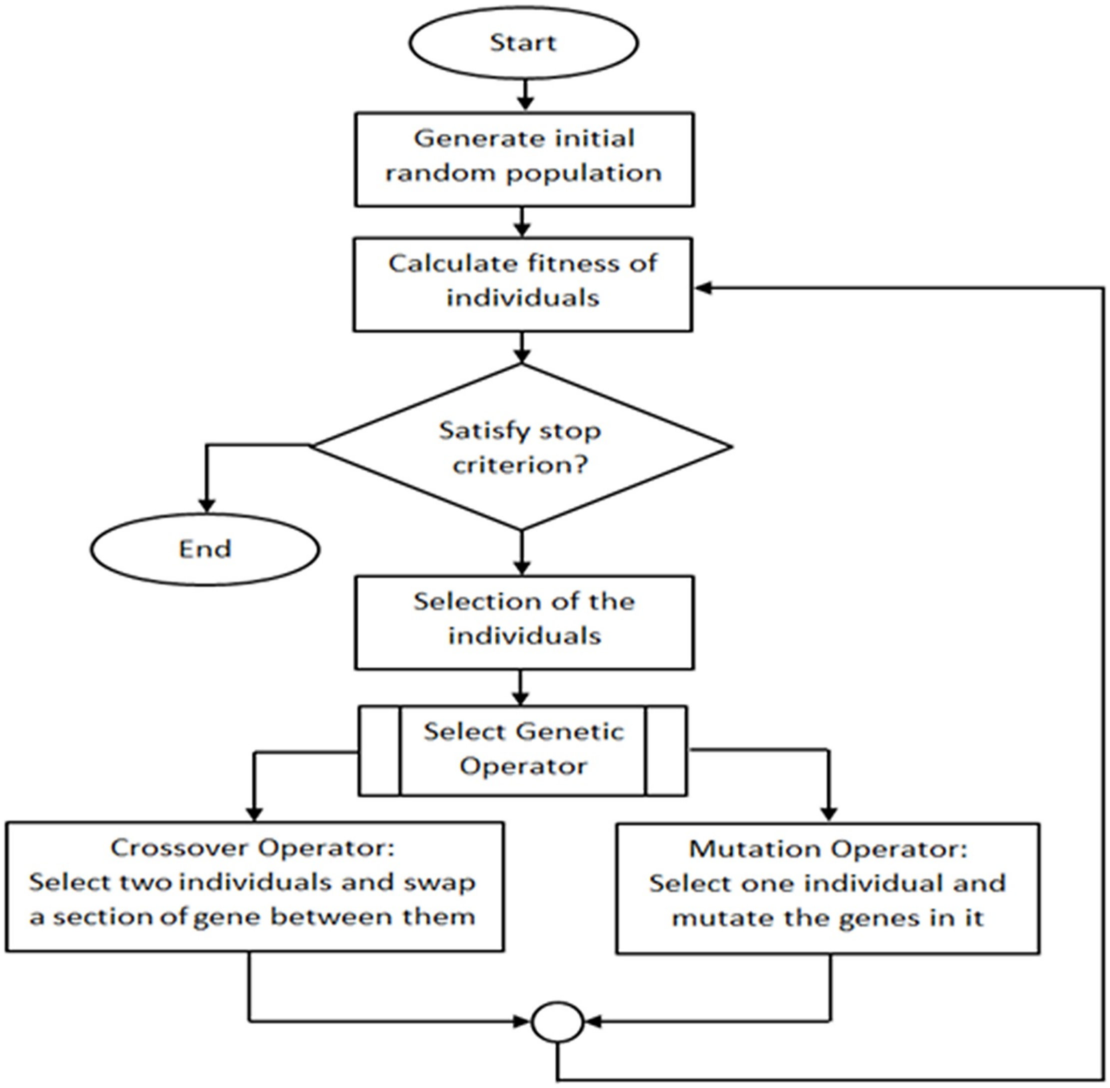

Start
Generate initial random population
Calculate fitness of individuals
Satisfy stop criterion?
End
Selection of the individuals
Select Genetic Operator
Crossover Operator: Select two individuals and swap a section of gene between them
Mutation Operator: Select one individual and mutate the genes in it

Figure 5.4: *The meta heuristic genetic algorithm (GA) is based on the natural selection process that falls under the umbrella category of evolutionary algorithms (EA). Genetic algorithms are typically utilized for generating high-quality solutions.*

6

Tables

Tables play a vital role in presenting your findings effectively. In this chapter, we delve into various techniques for conveying information through tables, using different environments available in this template. Although defining tables in LaTeX may appear complex, using this template makes the process more straightforward.

6.1 Weekly Schedule Table

The primary table for the timetable generator project is the weekly schedule table, shown in Table 6.1, displaying courses, instructors, and allocated time slots.

Table 6.1: *Weekly timetable schedule generated for a 5-day week.*

Day	**10:00-11:00**	**11:00-12:00**	**12:00-1:00**	**1:00-2:00**
Monday	Database	Discrete Math	Software Eng.	Communication
Tuesday	Numerical Method	Database	Discrete Math	Software Eng.
Wednesday	Communication	Numerical Method	Database	Discrete Math
Thursday	Software Eng.	Communication	Numerical Method	Database
Friday	Discrete Math	Software Eng.	Communication	Numerical Method

6.2 Instructor-Course Assignment Table

The following table in Table 6.2 outlines the assignment of instructors to courses.

Table 6.2: *Instructor assignments for each course.*

Course	**Instructor**
Database Management Systems	Dr. A. Sharma
Discrete Structures	Dr. B. Singh
Software Engineering	Dr. C. Verma
Numerical Method	Dr. D. Patel
Communication	Dr. E. Kumar

6.3 Slot Preferences Table

To view instructor preferences for scheduling, refer to the slot preferences table in Table 6.3.

Table 6.3: *Preferred time slots for instructors.*

Instructor	**Preferred Slot**
Dr. A. Sharma	Morning
Dr. B. Singh	Morning
Dr. C. Verma	Afternoon
Dr. D. Patel	Any
Dr. E. Kumar	Afternoon

7 Lists

Creating lists in LaTeX is straightforward, offering various options to suit your needs. Here are examples relevant to the Timetable Generator project.

7.1 Key Features

The key features of the timetable generator project are listed below:

- Automated schedule generation for weekly classes.
- Allocation of time slots based on instructor preferences.
- Optimization of timetable for minimized conflicts.
- UI for displaying the generated timetable.
- Database support for storing instructor and course information.

7.2 Development Steps

Here are the major steps undertaken in the development process:

1. Define project requirements.
2. Set up database and establish table structures.
3. Implement genetic algorithm for schedule optimization.
4. Develop UI for timetable display.
5. Integrate with Streamlit for visualization.

7.3 Supported Courses and Instructors

The supported courses and their instructors are provided below:

- Database Management Systems - Dr. A. Sharma
- Discrete Structures - Dr. B. Singh
- Software Engineering - Dr. C. Verma
- Numerical Method - Dr. D. Patel

- Communication - Dr. E. Kumar

These lists can be expanded or modified as new requirements emerge.

8

Code Listings

In this chapter, we include the source code relevant to the Timetable Generator project. We use `\begin{listing}` to create a listing with both caption and label, and `\begin{minted}` for code highlighting. Listing 8.1 provides an example of a Python implementation for the genetic algorithm used to optimize the timetable.

The code above demonstrates the use of a genetic algorithm to generate a timetable with random assignments of courses to time slots. The fitness function evaluates conflicts in the generated schedule.

8.1 Database Setup Code

For database interaction, the following code in Listing 8.2 demonstrates how to set up an SQLite database for storing course and instructor data.

This code creates tables for storing courses, instructors, and schedules within an SQLite database. The schema links courses to instructors and time slots.

8.2 UI Display Code using Streamlit

For visualizing the generated timetable, the code in Listing 8.3 shows how to use Streamlit for creating an interactive user interface.

This example shows how the timetable is displayed using Streamlit, providing a simple and interactive UI to visualize the generated timetable.

8.3 Using External Code Files

Instead of embedding code directly, you can also input external code files using the `\inputminted` command. Listing 8.4 demonstrates how to input a Python file for the genetic algorithm from an external file.

This approach is helpful when dealing with larger scripts or when you want to keep your source files organized in separate directories.

```python
import random

# Example genetic algorithm for Timetable Optimization
class Timetable:
    def __init__(self, courses, instructors, time_slots):
        self.courses = courses
        self.instructors = instructors
        self.time_slots = time_slots

    def generate_schedule(self):
        schedule = {}
        for course in self.courses:
            schedule[course] = random.choice(self.time_slots)
        return schedule

    def fitness(self, schedule):
        conflicts = 0
        for course, slot in schedule.items():
            if slot in conflicts:
                conflicts += 1
        return conflicts

# Example use case
courses = ['Database', 'Discrete Math', 'Software Eng.']
instructors = ['Dr. A. Sharma', 'Dr. B. Singh', 'Dr. C. Verma']
time_slots = ['Mon 10-11', 'Mon 11-12', 'Tue 10-11', 'Tue 11-12']

timetable = Timetable(courses, instructors, time_slots)
schedule = timetable.generate_schedule()
print(f"Generated Schedule: {schedule}")
```

Listing 8.1: *Python implementation for the genetic algorithm used in timetable generation.*

```python
import sqlite3

# Database setup for timetable generator
def create_db():
    conn = sqlite3.connect('timetable.db')
    cursor = conn.cursor()

    cursor.execute('''CREATE TABLE IF NOT EXISTS courses (
                        course_id INTEGER PRIMARY KEY,
                        course_name TEXT)''')

    cursor.execute('''CREATE TABLE IF NOT EXISTS instructors (
                        instructor_id INTEGER PRIMARY KEY,
                        instructor_name TEXT)''')

    cursor.execute('''CREATE TABLE IF NOT EXISTS schedules (
                        schedule_id INTEGER PRIMARY KEY,
                        course_id INTEGER,
                        instructor_id INTEGER,
                        time_slot TEXT,
                        FOREIGN KEY(course_id) REFERENCES courses(course_id),
                        FOREIGN KEY(instructor_id) REFERENCES
                        ↪  instructors(instructor_id))''')

    conn.commit()
    conn.close()

create_db()
```

Listing 8.2: *Database setup for the timetable generator using SQLite.*

```python
import streamlit as st

# Displaying the timetable in Streamlit
def display_timetable(schedule):
    st.title("Generated Timetable")
    for course, time_slot in schedule.items():
        st.write(f"{course}: {time_slot}")

# Example of displaying the generated timetable
display_timetable(schedule)
```

Listing 8.3: *Streamlit code to display the generated timetable.*

Listing 8.4: *External Python file for the genetic algorithm implementation.*

8.4 Long Listings for Larger Code

If the code is too large to fit on one page, you can use the `\begin{longlisting}` environment to automatically break the code across multiple pages. Listing 8.5 shows an example of a long listing for the genetic algorithm.

```python
# Long genetic algorithm code here...
class Timetable:
    def __init__(self, courses, instructors, time_slots):
        self.courses = courses
        self.instructors = instructors
        self.time_slots = time_slots

    def generate_schedule(self):
        # Implement schedule generation with optimization...
        pass

    def fitness(self, schedule):
        # Implement fitness function here...
        pass

    # Additional methods to optimize timetable
```

Listing 8.5: *Long listing of the genetic algorithm code for timetable optimization.*

By using the longlisting environment, your code will span multiple pages, which makes it easier to include large scripts without cutting them off mid-way.

9 Conclusion

The Timetable Generator project presented in this document leverages various AI and optimization techniques, primarily using genetic algorithms, to efficiently generate class schedules for an academic institution. The project aims to solve the challenging problem of creating optimal timetables that minimize conflicts and respect constraints such as instructor availability, course timing, and room allocation.

Throughout the course of this project, we successfully developed a robust genetic algorithm to generate and optimize timetables. The algorithm operates by encoding potential schedules as chromosomes, evaluating their fitness based on conflicts and constraints, and iterating over multiple generations to evolve improved schedules. The primary goal was to minimize conflicts while ensuring that all courses were assigned to appropriate time slots without overloading instructors or rooms.

In addition to the algorithm, the project integrates a relational database system, using SQLite, to store and manage data related to courses, instructors, and schedules. This setup allows for easy retrieval and modification of data, and supports scalability should the timetable need to accommodate more courses, instructors, or time slots in the future.

The project also features a user-friendly interface developed using Streamlit, allowing both administrators and users to interact with the timetable generator. The interface provides a clear and intuitive visualization of the generated timetable, ensuring that users can quickly assess and verify the assigned schedules. This UI enhances the overall usability of the system and enables administrators to add or adjust data as necessary.

Several challenges were encountered during the project, particularly when dealing with scheduling constraints such as room availability and instructor preferences. However, through the application of genetic algorithms, we were able to systematically address these challenges and improve the efficiency of the timetable generation process. The algorithm's ability to evolve over multiple iterations allowed it to adapt to new constraints and provide a more flexible solution.

The implementation of the genetic algorithm also demonstrated the power of evolutionary strategies in solving real-world optimization problems. By modeling the timetable as a genetic population, the system was able to explore a wide solution space and converge towards optimal or near-optimal solutions. This approach proved to be both effective and efficient, even as the size and complexity of the problem increased.

In conclusion, this Timetable Generator project successfully combines optimization algorithms, database management, and user interface design to create a comprehensive solution for generating academic timetables. While the project is already functional, there is potential for further enhancement. Future improvements could include the incorporation of more advanced optimization techniques, such as simulated annealing or machine learning-based approaches, to further reduce conflicts and improve schedule quality. Additionally, the user interface could be expanded to allow for more advanced features, such as automated conflict resolution or real-time updates.

Overall, this project provides a solid foundation for automating timetable generation in educational institutions and can be extended to other domains where optimization and scheduling are critical.