Project Report

SchedulerAI: TimeTable Assistant

May 14, 2024

—

Artificial Intelligence

—

Zarnain Maryam

Group Members:

Taha Hassan(K21-4680)
Shayan Haider (K21-3211)
Daniyal Haider (K21-3433)
Taha Ahmed(K21-4833)

Contents

1. INTRODUCTION

” SchedulerAI” aims to revolutionize timetable management at FAST University by leveraging Artificial Intelligence (AI) to automate timetable creation. By integrating advanced AI algorithms, the system will streamline the process of generating schedules, ensuring efficiency and accuracy in the allocation of classes and resources.

1.1. Background

Traditional timetable management at universities involves manual effort, often resulting in clashes and inefficiencies. To address these challenges, "SchedulerAI" is being developed. By leveraging Artificial Intelligence (AI), the project aims to automate timetable creation, ensuring efficiency and accuracy in class and resource allocation. The system will also integrate a timetable chatbot to assist users with schedule-related queries, further enhancing the user experience.

1.2. Purpose of Report:

- Provide a comprehensive overview of the "SchedulerAI" project, including its background, objectives, and features.

- Demonstrate the practical application of AI in educational settings, particularly in timetable management.

- Showcase how "SchedulerAI" will revolutionize timetable creation at FAST University, leading to increased efficiency and accuracy.

1.3. Intended Audience:

- University administrators responsible for timetable management.
- Faculty members involved in course scheduling and allocation.
- Students who will benefit from the automated timetable creation.
- AI researchers and developers interested in educational technology.

2. METHODOLOGY

A. Programming Design

- **Frontend Development**: The frontend will be developed using HTML, CSS, and JavaScript to create an intuitive and user-friendly interface for users to view schedules and interact with the system.

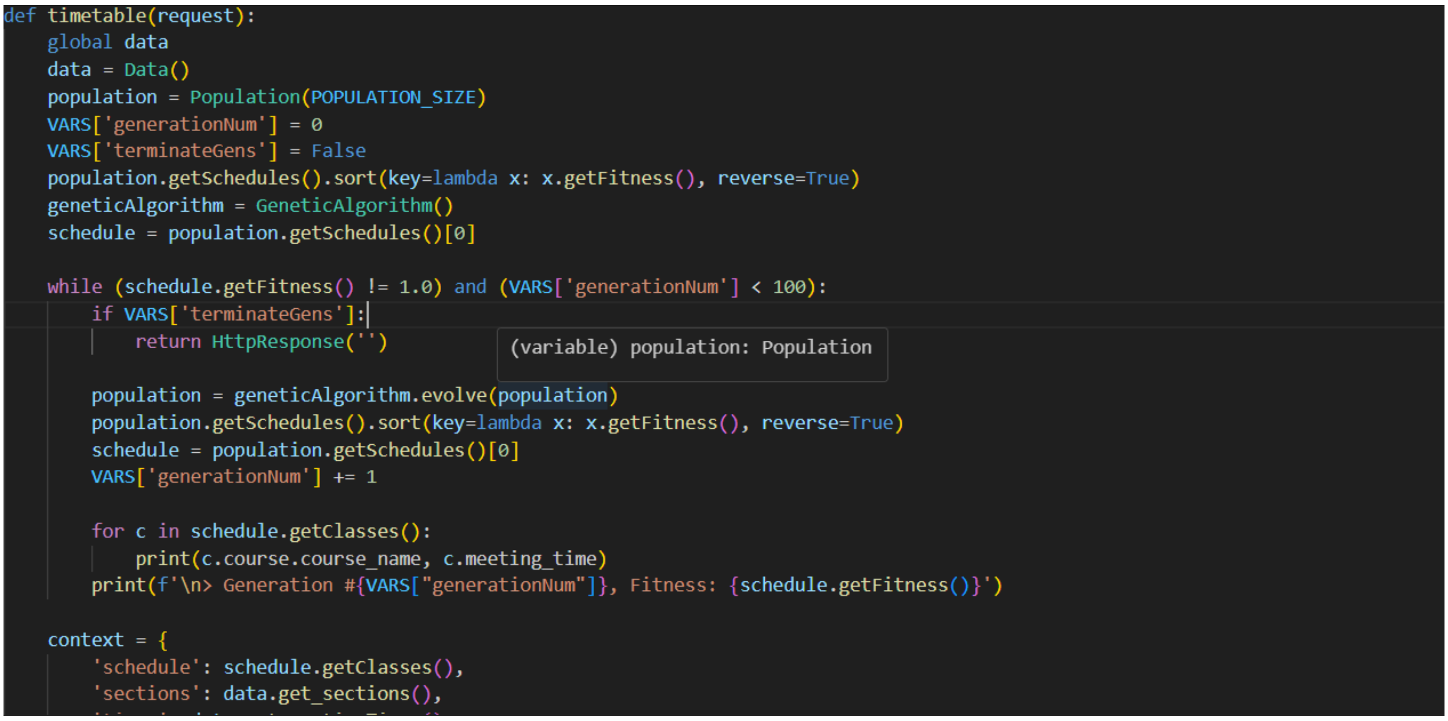
- **Backend Development**: Python with Django will be used for the backend to manage the business logic and interact with the SQLite3 database for storing user data, schedule information, and AI-generated timetables.

- **AI Integration**: AI algorithms, such as Genetic Algorithm, will be integrated to automate timetable creation, considering factors like free slots, clashes, and teacher availability.

- **Chatbot Integration**: A chatbot API will be integrated which will use ChatGPT4 to tackle user queries related to timetable by simply uploading timetable excel file to it.

- **Security Measures**: Data protection measures and secure authentication methods will be implemented to safeguard user data and prevent unauthorized access.

- **Maintenance and Updates**: Regular updates and maintenance will be planned to address issues and improve the system based on user feedback.



```
def timetable(request):
    global data
    data = Data()
    population = Population(POPULATION_SIZE)
    VARS['generationNum'] = 0
    VARS['terminateGens'] = False
    population.getSchedules().sort(key=lambda x: x.getFitness(), reverse=True)
    geneticAlgorithm = GeneticAlgorithm()
    schedule = population.getSchedules()[0]

    while (schedule.getFitness() != 1.0) and (VARS['generationNum'] < 100):
        if VARS['terminateGens']:
            return HttpResponse('')

        population = geneticAlgorithm.evolve(population)
        population.getSchedules().sort(key=lambda x: x.getFitness(), reverse=True)
        schedule = population.getSchedules()[0]
        VARS['generationNum'] += 1

        for c in schedule.getClasses():
            print(c.course.course_name, c.meeting_time)
        print(f'\n> Generation #{VARS["generationNum"]}, Fitness: {schedule.getFitness()}')

    context = {
        'schedule': schedule.getClasses(),
        'sections': data.get_sections(),
```

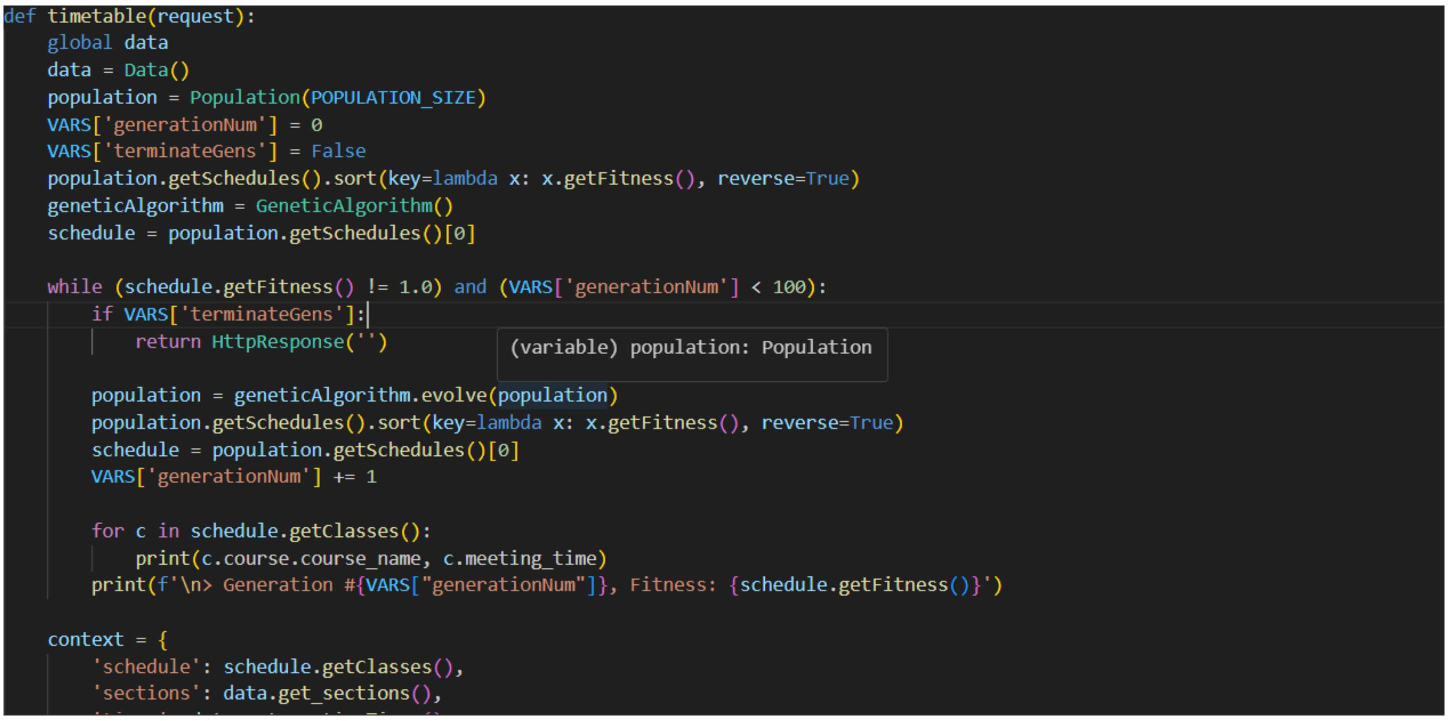
3. Experimental Setup

A. Input Data Preparation:

Input data preparation for the "SchedulerAI" project involves collecting timetable data by user from UI, formatting it to be compatible with AI algorithms, integrating it with other relevant information, validating its accuracy, transforming it into a suitable format, and augmenting it with additional features to enhance algorithm performance. This process ensures that the timetable data is ready for processing and scheduling by the AI algorithms in the system.

B. Algorithm Implementation:

The algorithm implementation revolves around using a genetic algorithm (GA) to automate timetable creation. This involves representing potential timetable solutions as chromosomes, with each chromosome containing genes that represent classes or time slots. An initial population of chromosomes is generated randomly or using heuristics. A fitness function is then defined to evaluate each chromosome based on factors such as clashes, free slots, teacher availability, and course credit hours fulfillment. Selection, crossover, mutation, and replacement operations are iteratively applied to evolve the population, with higher fitness chromosomes being more likely to be selected for reproduction. This process continues for a certain number of generations or until a termination condition is met, resulting in the generation of high-quality timetables that meet the scheduling requirements of universities.



```
def timetable(request):
    global data
    data = Data()
    population = Population(POPULATION_SIZE)
    VARS['generationNum'] = 0
    VARS['terminateGens'] = False
    population.getSchedules().sort(key=lambda x: x.getFitness(), reverse=True)
    geneticAlgorithm = GeneticAlgorithm()
    schedule = population.getSchedules()[0]

    while (schedule.getFitness() != 1.0) and (VARS['generationNum'] < 100):
        if VARS['terminateGens']:
            return HttpResponse('')        (variable) population: Population

        population = geneticAlgorithm.evolve(population)
        population.getSchedules().sort(key=lambda x: x.getFitness(), reverse=True)
        schedule = population.getSchedules()[0]
        VARS['generationNum'] += 1

        for c in schedule.getClasses():
            print(c.course.course_name, c.meeting_time)
        print(f'\n> Generation #{VARS["generationNum"]}, Fitness: {schedule.getFitness()}')

    context = {
        'schedule': schedule.getClasses(),
        'sections': data.get_sections(),
```

C. Algorithm Analysis

In the context of the "SchedulerAI" project, algorithm analysis involves evaluating the efficiency of the genetic algorithm (GA) used for timetable creation. This includes assessing the GA's time and space complexity to ensure it can efficiently handle the scheduling requirements of FAST University as the timetable size or complexity increases.

D. Performance Analysis

It evaluates how well a system, program, or algorithm performs in terms of speed, efficiency, resource usage, and other relevant metrics. It aims to assess the effectiveness of the system or algorithm in meeting its objectives and to identify areas for improvement. Performance analysis often involves measuring and comparing different aspects of performance, such as response time, throughput, scalability, and reliability, to determine the overall performance of the system or algorithm under various conditions.

4. CONCLUSION

The conclusion of the "SchedulerAI" project would likely highlight the successful implementation of the AI-based timetable management system at FAST University. It would emphasize how the project has revolutionized timetable creation by automating the process, reducing errors, and improving efficiency. The conclusion would also mention the integration of a chatbot to assist users with schedule-related queries, enhancing the user experience. Additionally, the conclusion may discuss the practical applications of AI in educational settings and the potential for further enhancements and improvements to the system in the future.

5. REFERENCES

- (PDF) Automated Timetable Generation using Genetic Algorithm (researchgate.net)
- https://www.atlantis-press.com/article/125993049.pdf

6. CODE REPOSITORY

The codebase developed for the SchedulerAI, is available in the following GitHub repository: SchedularAI: TimeTable Assistant