EVALUATE THE PERFORMANCE OF UNIVERSITY TIMETABLING PROBLEM WITH VARIOUS ARTIFICIAL INTELLIGENCE TECHNIQUES

By

Charmaine Hooi Wai Yee

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONOURS)

Faculty of Information and Communication Technology

(Kampar Campus)

JUNE 2025

ACKNOWLEDGEMENTS

I would like to express my heartfelt gratitude to my supervisor, Ts. Dr. Ku Chin Soon, for giving me the incredible opportunity to work on this timetable scheduling project. Truly thankful for your guidance and support throughout this journey.

Lastly, I extend my deepest appreciation to my friends, parents and family. Your love, support, and constant encouragement have been the foundation of my strength throughout this entire course.

COPYRIGHT STATEMENT

This Final Year Project proposal is submitted in partial fulfillment of the requirements for the degree of Bachelor of Computer Science (Honours) at Universiti Tunku Abdul Rahman (UTAR). This Final Year Project proposal represents the work of the author, except where due acknowledgment has been made in the text. No part of this Final Year Project proposal may be reproduced, stored, or transmitted in any form or by any means, whether electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the author or UTAR, in accordance with UTAR's Intellectual Property Policy.

ABSTRACT

University timetabling is a complex and critical task in higher education institutions as it involves the assignment of courses, lecturers, and students to available timeslots and venues while satisfying various constraints. The project focuses on developing an automated university course timetable scheduling tool using Genetic Algorithm (GA). University course timetable scheduling (UCTTP) is a well-known optimization problem due to its NP-hard nature and the complexity of the problem increases exponentially with eh addition of constraints. Over time, numerous algorithms such as Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Simulated Annealing (SA), and other approaches have been introduced to address the challenges of optimizing class schedules. While each university or institution have its own unique constraints, this project aims to improve existing timetabling systems by introducing a new constraint, the 'Proximity and Travel Minimization Constraint' which optimizes class schedules to minimize travel distances between venues scheduled in adjacent time slots. By implementing this new constraint, the project addresses the gap in traditional timetabling methods, which often overlook the impact of travel distances on the efficiency and experience of both lecturers and students. Hence, through the application of GA, this project aims to develop an efficient university class timetabling tool that integrates the newly introduced constraint.

Area of Study (Maximum 2): Genetic Algorithm, Scheduling Algorithm

Keywords (Maximum 5): Genetic Algorithm, Optimization Problem, Artificial Intelligence Techniques, Timetabling Applications, Mobile Application

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

UCTTP	University Course Timetabling Problem
NP	Non-Polynomial
GA	Genetic Algorithm
PSO	Particle Swarm Optimization
SA	Simulated Annealing

CHAPTER 1

Introduction

In this chapter, we provide the motivation and background of our research, our research contributions, and the outline of the project.

1.1 Project Background

Creating a good timetable that satisfies both lecturers and students is a challenging task in higher education institutions. In universities, this complex process involves assigning a set of courses, lecturers, and students to available timeslots and classrooms with no conflicts while satisfying a range of constraints, both hard and soft [1]. Traditionally, a timetable is either scheduled manually or handled using basic software tools, which can be time-consuming, error-prone, and suboptimal. Thus, university course timetable tools are developed and utilized to solve this problem optimally while saving a significant amount of time [1], [2].

In the field of scheduling and optimization, university course timetable scheduling is a classic problem identified as an Non-Polynomial (NP-hard) optimization problem [3]. The optimization of hard problems falls under a category where solutions cannot be achieved using exact methods within polynomial time. Hence, resource assignment or allocation is classified as an NP-hard problem as it is an issue that is dependent on the resources of an organization to generate the optimal outcome and there are no one exact method to solve it [4].

By introducing optimization techniques, optimal solutions whether minimum optimal or maximum optimal can be achieved. The optimization techniques can be divided into heuristic and meta-heuristic methods, both of which are artificial intelligence techniques to obtain the best results. However, evolutionary algorithms like Genetic Algorithm (GA), Fuzzy logic (FL), Swarm Intelligence (SI) such as Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), and Artificial neural network (ANN) are primarily used for optimization problems [4]. In the case of university timetable scheduling, Genetic Algorithm (GA), a Population-Based Meta-Heuristic method, is the more common AI method applied to this problem [2].

As educational environments become more complex and diverse, the requirement for more sophisticated and dynamic systems that will meet the modern scheduling requirements is increasing. Genetic Algorithms' (GA) ability of searching through a massive search space to get near-optimal solutions have emerged it as a powerful method for tackling the university course timetabling problem. This project builds upon this foundation with the addition of a new constraint which resolves one particular problem in scheduling courses at a university.

1.2 Problem Statement

This project aims to implement a new constraint, the 'Proximity and Travel Minimization Constraint', which explores the optimization of class schedules to minimize travel distances between two different venues scheduled with adjacent timeslots. This constraint integrates the principles of accessibility and optimization, addressing a critical gap in traditional timetabling methods as it highlights key issues that affects a student's or lecturer's experience, the travel distance between classes and the time taken to travel in between those classes. Therefore, by introducing a 'Proximity and Travel Minimization Constraint', university course timetabling tools will be enhanced to create more efficient timetables.

One problem of current university timetabling tools is the **lack of features to generate timetables based on travel distance between venues** for both lecturers and students. Traditional timetabling methods often overlook the proximity of classrooms and the travel distances between them. While other constraints should be prioritized and satisfied, this problem is challenging because it often leads to schedules that requires excessive movement across the university campus within short timeframes, especially when classes are scheduled back-to-back in distant locations. Due to this, lecturers and students may experience tardiness, increase physical strain, and reduced academic efficiency. However, this issue is particularly problematic in large campuses where the distance between buildings can be larger. By implementing a 'Proximity and Travel Minimization Constraint' into the scheduling process, consecutive classes can be scheduled in nearby locations while classes that require significant movement across the campus can be allocated in non-consecutive timeslots.

1.3 Project Motivation

In order to improve the quality and usability of university course timetabling tools, this project introduces a new constraint: Proximity and Travel Minimization Constraint. The constraint aims to minimize traveling distances between classrooms for both students and lecturers when they have consecutive classes.

The motivation behind this constraint is to:

- Increase the accessibility of transitions between classes, particularly for big campuses.
- Improve punctuality and lessen fatigue that comes with frequent long-distance travelling between classes, enhancing overall academic experience for students and lecturers.

By adding this constraint to the Genetic Algorithm model, the proposed method aims to generate more realistic timetables that more accurately reflect real-world situations in a university setting.

1.4 Project Scope

This project focuses on developing a university course timetabling tool designed to address specific challenges in scheduling, particularly minimizing travel distances between consecutive classes. The project will be implemented for Universiti Tunku Abdul Rahman (UTAR) with the Faculty of Information and Communication Technology (FICT) acting as the primary collaborator for the validation and testing of the tool developed. However, the users of this university timetabling tool are not limited to only UTAR staff. Hence, the primary users of this tool will be any university's management team who is responsible for scheduling timetables at the start of each trimester.

The timetabling tool will be developed as a mobile application system and will primarily target automated timetable scheduling, also known as timetable generating. One of the primary features of the system is to **enable users to input key attributes such as courses, lecturers, venues, and student groups before the generating a**

conflict-free timetable based on these inputs. The system should also **efficiently handle hard and soft constraints, including the additional constraint, 'Proximity and Travel Minimization Constraint' using Genetic Algorithm** to ensure that consecutive classes are scheduled in nearby locations, minimizing travel time and improving accessibility for both lecturers and students. Additionally, users will be able to **customize scheduling preferences and ensure that the final timetable is tailored to the specific needs of the institution**. By resolving scheduling conflicts and offering customization options, the application will ensure that the generated timetables are both conflict-free and tailored to specific needs.

However, the application will not address non-scheduling-related features such as student grading, course, content management, or online and hybrid teaching modes. It will solely focus on **generating timetables based on the physical locations of classrooms and minimizing travel distances**. Besides, it will also aim to address major scheduling challenges and will not cover all possible constraints that might arise in complex timetabling scenarios.

Lastly, the hardware required for the development and deployment of this project will be a laptop with sufficient processing power. The mobile application itself will be designed to run on commonly used mobile devices such as smartphones and tablets. Software will include Python, MySQL, Flutter, and GitHub. Python will be used to implement the Genetic Algorithm (GA) logic. MySQL is used to develop the database. Flutter which uses the Dart programming language and provide cross-platform frameworks is used to develop mobile application. GitHub is used for version control.

1.5 Project Objectives

One of the objectives of this project is **to investigate the hard constraints and soft constraints for the university timetabling problem**. Hard constraints can be defined as requirements that must be satisfied when generating the timetable while soft constraints are conditions that are preferable but non-compulsory for the timetable [1]. This involves studying, identifying and categorizing the various constraints that impact the feasibility and quality of the timetables generated as they are requirements that

should be considered during timetable scheduling. By understanding the hard and soft constraints in detail, **a more efficient university course timetabling tool can be developed through this project**. In addition, an additional constraint (Proximity and Travel Minimization Constraint) will be included as a parameter in the project.

The next objective of this project is **to develop a university course timetabling tool that incorporates the 'Proximity and Travel Minimization Constraint' using Genetic Algorithm (GA)**. This tool aims to optimize class schedules by minimizing the travel distances between consecutive classes, targeting to improve accessibility and punctuality, as well as reduce physical strain on lecturers and students. Integrating this constraint into the Genetic Algorithm (GA) will address a common issue overlooked by traditional timetabling systems and ensure that the generated timetables are feasible, efficient and user-friendly. The success of this tool will be measured through the development and testing of a functional timetable generator that is capable of satisfying the identified constraints and improving the overall timetabling process.

Hence, this project aims to:

1. Develop an automated university timetabling tool that generates timetables while satisfying various constraints.
2. Introduce a new constraint, the 'Proximity and Travel Minimization Constraint', which minimizes travel distances between consecutive classes.
3. Create a more efficient, user-friendly timetabling system that improves scheduling efficiency and user experience.

1.6 Impact, Significance and Contribution

The main contribution of this project is the development of a university course timetabling mobile application designed to enhance the efficiency and effectiveness of the timetable scheduling process. This is because the application will automatically generate timetables instead of having the academic staff do the laborious work of manually inserting large amounts of data into the system. By using Genetic Algorithms and integrating the 'Proximity and Travel Minimization Constraint', the application will significantly reduce the workload of the academic staff involved in timetable scheduling. Hence, this streamlines the scheduling process, reducing the complexity

and time required to create timetables. Besides, students will experience improved class schedules that minimizes travel time between class, thereby reducing physical strain and improving punctuality. This improves their overall academic experience.

Within the academic community, the application will foster more efficient and effective scheduling practices by integrating the new constraint, 'Proximity and Travel Minimization Constraint', which contributes to a more accommodating scheduling system that better meets the needs of both lecturers and students.

1.7 Report Organization

In this report, the contents are organized as follows. **Chapter 1** introduces the project, covering the background, problem statement and motivation, objectives, scope, contributions, and an overview of the report. **Chapter 2** presents the literature review, discussing AI/optimization approaches for university timetabling, their strengths and limitations, the typical timetabling parameters, hard and soft constraints, evaluation metrics, and a comparison of related systems. **Chapter 3** describes the system methodology, including the overall architecture and data model, the GA formulation and constraint encoding, the development plan, and the verification/testing strategy. **Chapter 4** details the system implementation such as database schema, back-end services, and user-interface modules for creating, editing, archiving, and exporting timetables. **Chapter 5** reports the results and evaluation, including generated timetables, constraint-validation test cases, GA analytics, performance analysis, and key implementation issues and challenges. **Chapter 6** provides the discussion, evaluating the extent to which the objectives were achieved, outlining system limitations, and proposing future improvements (e.g., richer GA analytics and a student-facing mobile interface). Finally, **Chapter 7** concludes the report and summarizes the main findings and contributions.

CHAPTER 2

Literature Reviews

2.1 Overview of Past Research

University course timetable scheduling is the creation of a timetable that allocates time slots and resources such classrooms, lecturers and students for various courses while ensuring that there are no conflicts [2]. The timetable needs to accommodate different and numerous constraints, which are mainly categorized into hard constraints and soft constraints [1]. Some examples can be lecturer availability, room capacities, course requirements, student preferences, and university regulations. An efficient scheduling system ensures that all constraints are satisfied using the limited time and space available to get the optimal and best timetable.

UCTTP refers to the problem of creating a timetable that satisfies a set of constraints and preferences. Due to the numerous possible combinations, the problem is more complex and considered an NP-hard problem, whereby the objective is to find the optimal timetable out of the large number of potential combinations. Besides, there are no specific efficient solutions or algorithm to solve this problem [3], [2].

2.1.1 Types of AI Methods Used

Timetabling is categorized as a complex scheduling problem that requires the allocation of limited resources, such as lecturers and classrooms, to courses in an optimized way that satisfies a range of constraints. This problem is also known as the university course timetabling problem (UCTTP) [1]. Several Artificial Intelligence (AI) techniques are commonly used to solve this problem. However, each technique offers different strengths, weaknesses, and approaches. According to [1], some of the approaches used to solve the UCTTP can be categorized into Operational Research (OR) Based Techniques, Single Solution-Based Meta-Heuristics, Population-Based Meta-Heuristics, Hyper-Heuristics, and Hybrid Approaches. However, the most common approach is a Population-Based Meta-Heuristic method known as Genetic Algorithm (GA).

I) Genetic Algorithm (GA)

Genetic Algorithms (GAs), a Population-Based Meta-Heuristic method as mentioned above, are inspired by the process of natural selection as their concept works around generating a population of potential solutions and then producing new generations of solutions by iteratively selecting, crossing over, and mutating the population. GAs are particularly effective in solving complex problems with a large and irregular search space for possible solutions [3], [4], [5].

This effectiveness stems from the robustness of GAs, as they are designed to operate in complex environments with non-linear, discontinuous, or noisy objective functions [6], [7]. GAs avoid getting trap in local optima by evolving a diverse population of solutions, which is a challenge often faced by other optimization methods [8]. Instead, they comprehensively explore the global search space, guided by the fitness function that evaluates the quality of each solution [5].

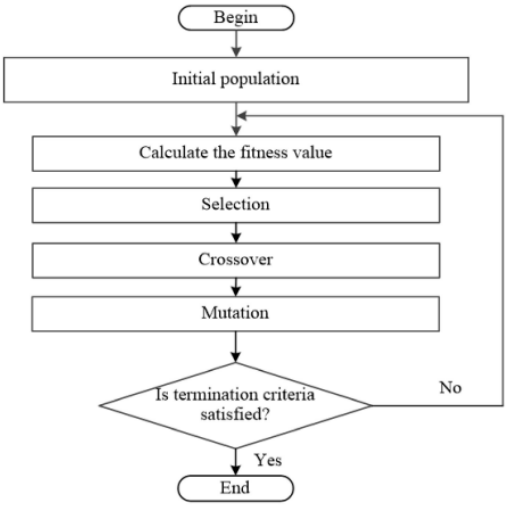

Figure 2.1.1.1 Process Flow of Genetic Algorithm [9]

GAs operate based on seven steps, which includes (1) initial population generation, (2) evaluation of the generated population using benchmark functions (calculation of fitness value), (3) selection of parents, (4) application of crossover operator, (5) application of mutation operator, (6) selection of parents and children to form new generation population, and (7) termination if condition is satisfied [1]. These steps are as shown in Figure 2.1.1.1 above [9]. In solving timetabling problems, a population of feasible and infeasible population of timetables are initialized. Through

a fitness-based process in which the fitness value of each chromosome is calculated and evaluated, only the fittest timetables will be selected as the new population for the next iteration [10], [11]. Examples of GAs being implemented in solving timetabling problems are as follows: [1], [2], [3], [10], [12], [13], [14], [15], [16], [17].

During the initialization step in GA, the chromosome representation of the UCTTP is a critical factor, as it directly impacts the algorithm's performance, such as how it explores and exploits the search space [4]. In this step, gene encoding occurs, whereby elements of a problem are represented within a chromosome [5], [18]. Most research papers represent a chromosome as a string or matrix where each gene encodes a course assignment, often specifying the time, room, lecturer, class, and students [17], [19], [20], [21]. For instance, each chromosome is encoded using a two-dimensional array where rows represent courses and columns contain details such as day, time, teacher, and venue [15], [18], [22]. Another way is that the chromosomes are represented as a group of elements, denoted by E, consists of three data types, namely subject (S), lecturer (L), and group (G) [23], [24]. This allows the GA to manipulate the structure easily during crossover and mutation, while maintaining the feasibility of schedules [10], [17], [21], [20].

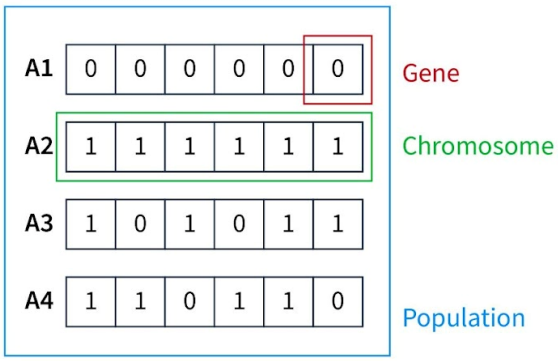

Figure 2.1.1.2 Chromosome Structure [25]

Based on Figure 2.1.1.2 above, the chromosome structure takes the form of a linear string, whereby each row A1 to A4 represents a chromosome [25]. Binary encoding is used as the genes are stored as binary digits [26]. In this case, a chromosome represents one feasible timetable. Each position, or gene, in this string corresponds to a specific component of the timetable, such as to a specific course. The value of each gene is the assignment data for that course, such as timeslot, room, and lecturer [11],

[27]. The length and organization of the chromosome depends on the number of courses, available timeslots, and venues within the university's scheduling constraints [26], [28]. Different studies employ various types of encoding schemes. For instance, some might use direct encoding where the genes explicitly store the assigned resource, while indirect encoding store priority or ordering information that is later decoded to generate the actual timetable. At the end of this process, a population of feasible solutions will be generated [11].

The next step is the evaluation of the solutions generated using a fitness function. The effectiveness of a GA for UCTTP solving highly relies on the design of the fitness function because the fitness value typically reflects the degree to which a suggested timetable satisfies the given constraints [5], [10]. Hard constraint violations usually result in penalties, making solutions that violate them less fit and less likely to be selected for reproduction. On the other hand, soft constraint violations result in smaller penalties, reflecting their desirable but not compulsory nature. A total fitness for a chromosome is then calculated as a function of these penalties, usually as a sum or weighted sum, in order to minimize constraint violation and maximize soft constraint satisfaction [23], [24], [29]. Chromosomes representing valid or near-valid timetables will be assigned higher fitness values. This guides the evolutionary process towards optimal or near-optimal solutions.

Next, the method of selection used determines which of the present population's chromosomes will be chosen as parents for producing children of the next generation [2]. The objective is to favor individuals with higher fitness scores, hence propagating potentially better solutions [11], [30]. Some of the most common selection techniques used in GA-based UCTTP include roulette wheel selection, tournament selection, rank-based selection, elitism selection, and stochastic remainder selection [4], [5], [7], [8], [14], [15], [17], [18], [19], [20], [22], [23], [27], [29], [31], [32]. The chosen selection technique ensures only the best solutions can move on to the next stage.

Following selection, the crossover operator is crucial in generating new offspring by combining the genetic material from two parent chromosomes. Its main goal is to explore new areas of the solution space by recombining high-quality traits from different parents. Common crossover operators applied to UCTTP include single-point crossover, two-point crossover, uniform crossover, shuffle crossover, and cycle

crossover [4], [5], [7], [14], [17], [30], [32], [33]. For example, in single-point crossover, a random point is chosen, and the segments of the parent chromosomes beyond this point are swapped to generate two new offspring chromosomes [10], [27]. Two-point crossover performs a similar operation but involves two crossover points, allowing the middle segment to be exchanged. The choice of crossover operator can significantly impact the algorithm's ability to discover better solutions.

After that, the mutation operator is applied to subject the offspring chromosomes with a certain mutation probability. This is done to maintain population diversity and prevent premature convergence towards local optima [11], [30], [34]. Mutation involves slight random changes in one or more individual genes within a chromosome so that new areas of the search space can be explored, which may not be reachable through the use of crossover alone [24], [30]. Common mutation techniques used in UCTTP include uniform mutation, bit mutation, single point mutation and swap mutation [9], [15], [17], [19], [23], [28]. The probability of mutation is typically set to a low value to ensure that changes are rare and do not disrupt good solutions excessively [8].

After crossover and mutation, the algorithm proceeds to generate the next generation by replacing the current population [4]. This selects which members of the current population are to be retained and replaced by offspring generated afresh. For example, only the two fittest chromosomes are selected from the population of two parents and two offspring. The fittest chromosome is directly moved to the next generation, and the second best is taken to replace the least fit chromosome in the current population [29]. The selection of which individuals to replace can influence the algorithm's convergence behavior, ensuring the new generation have better chromosomes than the preceding one [35].

The GA iteratively repeats steps 2 to 6 until a predefined termination condition is met. This condition could be a fixed number of generations, reaching a satisfactory fitness level, or observing no significant improvement in the population's fitness over a certain number of generations. To measure the performance of the proposed GA-based UCTTP method, several evaluation metrics are used. The primary metric is often the number of hard constraint violations in the final timetable. The number and severity

of soft constraint violations are also crucial. Other metrics include computational time, the convergence rate, and the scalability of the approach.

II) Particle Swarm Optimisation (PSO)

Similar to GA, Particle Swarm Optimisation (PSO) is also a Population-Based Meta-Heuristic method that is often utilized to solve UCTTP. This method is inspired by the behavior of animals flocking or schooling together, such as birds and fishes, operating based on social behavior instead of selection action [13]. The PSO model consists of particles which are comparable to potential solutions with an assigned fitness function. These particles move through the solution space and navigate by adjusting their positions based on their own experience and that of neighboring particles. Parameters such as the size of the swarm, inertia weight and acceleration coefficients affect the efficiency of the PSO model [2].

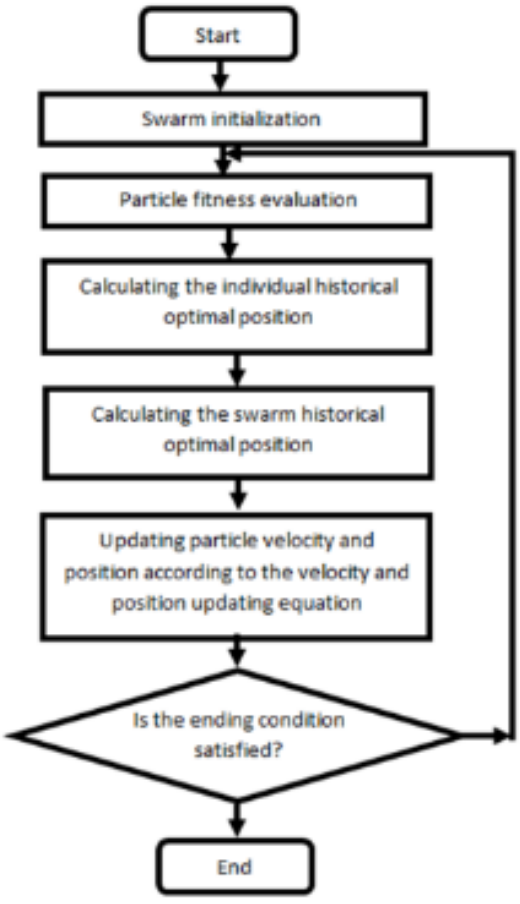

Figure 2.1.1.3 Process Flow of Particle Swarm Optimization [3]

Based on Figure 2.1.1.3 above which displays the PSO algorithm's flow, it is shown that just like GA, the algorithm starts with initialization. This is where a swarm of particles is randomly distributed across the solution space. With each particle representing a potential solution, the fitness of each particle is then determined using a problem-specific fitness function. After that, the velocity update step follows. This is where each particle updates their velocity based on three components, namely inertia, individual best position, and global best position. After updating the velocity, the particle's position and global best position are updated if needed. These operations are iterated for a fixed number of iterations or until the solution is satisfactory.

One of the advantages of PSO is its fast convergence rate, lesser parameters used and its ability to learn to handle dynamic changes in the problem environment, especially where timetabling constraints may evolve in real applications. Aside from improving solution quality and preventing premature convergence, the study also applied an interchange heuristic. This additional step allowed particles to travel to more varied areas of the solution space by interchanging components within a timetable [2].

Additionally, PSO's strength is that it is lightweight in nature due to having lesser parameters compared to GA or SA. Its velocity and position update processes are mathematically simple yet powerful. It is also easy to implement [34]. However, PSO by itself can still potentially converge to inferior solutions if the swarm is not diverse or if global best positions dominate too early. In addition, PSO does not converge as fast as other algorithms [3], [36].

These qualities make PSO an ideal choice for hybridizing or enhancing based on local search methods. For example, combining PSO with other techniques such as Genetic Algorithm (GA), Differential Evolution (DE), Simulated Annealing (SA) and more [34]. Such improvisation makes PSO robust and adaptable in solving UCTTPs as well as other hard constraint-based scheduling instances.

III) Simulated Annealing (SA)

Simulated Annealing (SA) is a Single Solution-Based Meta-Heuristic method inspired by the annealing process in metallurgy or the heating of solids in physic science. This method starts with a random initial solution by local search and iteratively

replaces the current solution with a random solution, whereby the replacing solution is possibly the optimal solution to the problem. This process repeats until a certain condition triggers termination [1]. As cited in [1] and [2], the best results and most satisfactory timetable achieved is obtained from the combination of simple search, swapping and simple swapping. Figure 2.1.1.4 below illustrates a very detailed flowchart of Simulated Annealing and its processes, which appears more complex compared to GA and PSO.

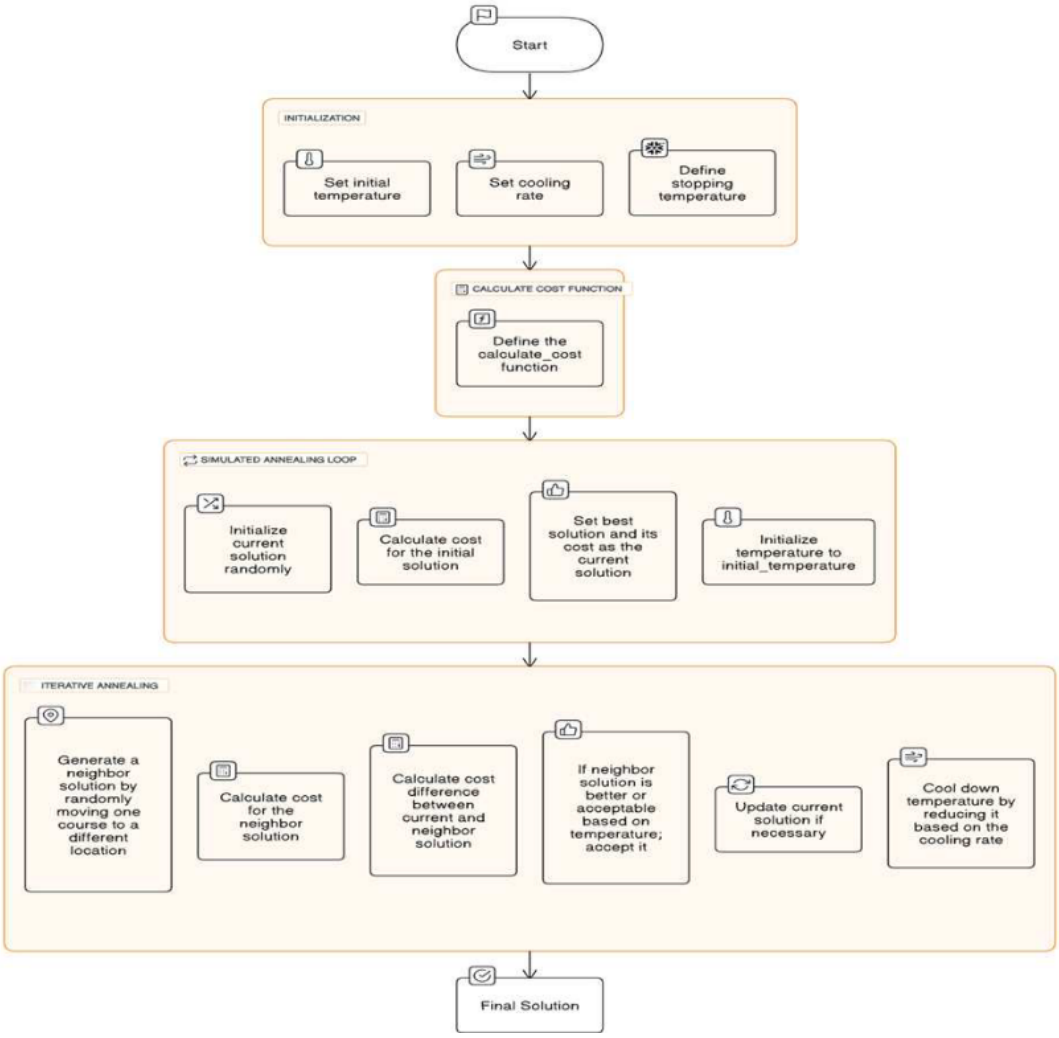

Figure 2.1.1.4 Process Flow of Simulated Annealing [6]

According to [14], SA is a neighborhood search method with a unique advantage. It occasionally accepts worse solutions in order to escape local optima and move in the direction of the global optimum. This is controlled by a temperature parameter, which is set high at first and then gradually reduced during the algorithm. When the temperature is high, the algorithm is more likely to accept worse solutions,

which allows wider search of the solution space. As the temperature decreases, the acceptance of worse solutions decreases, leading to a more focused search in better regions of the solution space. A helpful feature for real-life constraints like room availability, student distribution, and emergency evacuation planning is that SA has been able to create conflict-free timetables with a high location diversity in order to fulfill timetabling constraints [6].

Comparisons of performance with previous research have shown that SA can produce excellent solution quality that is often better than other approaches in terms of diversity and robustness. However, because it is an iterative process, it usually takes more computing time. Its success also depends on proper parameter tuning for the initial and final temperatures, cooling rate, and neighborhood structure. Despite the longer runtime, SA is a successful method for solving optimization problems with large, complex search areas, even though it takes longer to execute [6].

Table 2.1.1.1 Comparison of Genetic Algorithm, Particle Swarm Optimization and Simulated Annealing

	Genetic Algorithm	**Particle Swarm Optimization**	**Simulate Annealing**
Advantages	• Robust in complex spaces • Balanced exploration and exploitation • Flexible chromosome encoding	• Fast convergence • Easy implementation • Fewer parameters	• Strong escape from local optima • Very good solution quality • Suitable for complex constraints
Disadvantages	• Slower convergence • Requires careful parameter tuning	• May converge to inferior solutions if swarm diversity is low	• Requires careful tuning of parameters

	• Risk of premature convergence if diversity is low	• Global best may dominate too early • Slower convergence than some algorithms	(temperature, cooling rate) • Slower due to iterative nature • More complex process flow compared to GA and PSO
Parameter Sensitivity	Sensitive	Less sensitive	Sensitive
Ability to Escape Local Optima	Moderate, depends on mutation	Low, may get trapped in global best	High, due to probabilistic acceptance
Runtime Speed	Moderate	Fast	Slightly slower
Solution Quality	Good to Very Good	Good to Very Good	Very Good
Implementation Complexity	Moderate	Easy	Easy
Suitable for Timetabling	Yes	Yes	Yes

Based on Table 2.1.1.1, the comparison between Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and Simulated Annealing (SA) is displayed. It shows that each method has advantages and disadvantages when being applied to the UCTTP. GA works good in complex spaces and can balance exploration and exploitation, which makes it perform well in complex environments. It is also known for its flexible chromosome encoding, thus making it ideal for complex timetable constraints. On the other hand, while PSO is fast and simple to apply, it is sensitive to swarm diversity and can converge to inferior solutions if the global best is dominated too early. SA can escape local optima and generate high-quality solutions, but it has slower runtime due to its iterative nature. Besides, SA requires very careful parameter tuning.

In this project, GA is selected because it excels at handling complex timetabling constraints and is able to effectively balance exploration and exploitation. Although GA's rate of convergence is moderate and requires careful adjustment to avoid premature convergence, its ability to search for extensive solution spaces and learn the constraints of UCTTP makes it a stable and effective choice. GA's flexibility in chromosome encoding also allows it to encode different types of constraints such as student and teacher allocations, room allocation, and students' timetables so that it gets fittingly aligned with timetabling problem characteristics.

2.1.2 Parameters Used in University Course Timetabling

According to [1], some of the parameters used in scheduling a university course timetable include hard constraints and soft constraints such as:

- Session: Scheduled activities.
- Timeslot: A designated time interval in which sessions are scheduled, such as a weekly slot (every Monday) or a daily slot (8 a.m. – 10 a.m.).
- Resource: Items utilized by sessions, including equipment, venues, and timeslots.
- Constraint: A limitation in scheduling the sessions that can be categorized into hard and soft constraints, such as room capacity and availability of specific timeslots.
- People: Individuals involved in the sessions, such as lecturers and students.
- Conflict: Something that arises when two sessions are scheduled together and interfered with each other, such as assigning a lecturer having two classes in different venues at the same time.

2.1.3 Types of Hard Constraints and Soft Constraints

In the context of university course timetable scheduling, constraints are classified into two categories: **Hard Constraints** and **Soft Constraints.**

Hard constraints are compulsory requirements that must be strictly followed and satisfied for the timetable to be valid. Violating any of the hard constraints will result

in an invalid timetable. Table 2.1.3.1 below displays the list of hard constraints mentioned in previous works and past research papers.

Table 2.1.3.1 Hard Constraints Mentioned in Past Research Papers

Constraint No.	Hard Constraints	Ref. No.
1	A lecturer cannot teach two different classes in the same timeslot.	[1], [2], [3], [10], [12], [16], [17], [18], [22], [24], [32], [33], [37], [38], [39], [40]
2	Two courses (events) cannot be scheduled in the same venue at the same time.	[1], [2], [3], [12], [14], [17], [18], [22], [24], [32], [33], [37], [38], [39], [41]
3	A student cannot attend two courses (events) scheduled simultaneously.	[2], [3], [12], [17], [18], [22], [26], [31], [37], [38], [40], [41]
4	The venue is large enough to accommodate the number of students attending the class.	[1], [2], [3], [10], [14], [22], [26], [27], [29], [31], [32], [38]
5	Courses that require specific room types (e.g. labs, lecture halls) must be scheduled in appropriate facilities.	[2], [26]
6	Classes should only be scheduled during times when the lecturer is available, considering the workload, office hours and rest time of the lecturers.	[1], [27], [34]
7	Some courses are predefined and are scheduled to occur at specific timeslots.	[1], [2], [14], [31]
8	Venues cannot be assigned to the same timeslot more than once.	[2], [3], [10], [11], [22], [31]

9	The total hours available per day for the timetable is 8 hours (maximum).	[3], [26]
10	Each venue can only be assigned one lecturer, unless an assistant is assigned.	[10]
11	A student cannot be assigned to two different venues in the same timeslot.	[3], [10], [12], [22]
12	The timetable should be scheduled in accordance with the university calendar.	[3], [14], [37]
13	Lectures for courses within the same curriculum should be scheduled in consecutive time slots if they occur on the same day.	[2]
14	The facilities of the venue should meet the requirements of the course.	[2], [14], [31], [37], [41]
15	Certain classes must be scheduled in a specific chronological order.	[2], [12], [31]
16	There should be no classes missing in the generated timetable.	[33]
17	The number of courses and their scheduled hours are predetermined based on the teaching plan and training objectives.	[32]
18	No lectures should be scheduled during the weekends.	[37]

On the other hand, soft constraints are preferable or desirable conditions that are not strictly necessary. However, they are important for creating a more efficient and convenient timetable. Violating any of the soft constraints does not invalidate the timetable, but it may decrease the overall quality of the timetable. Table 2.1.3.2 displays a list of soft constraints found in past research papers.

Table 2.1.3.2 Soft Constraints Mentioned in Past Research Papers

Constraint No.	Soft Constraints	Ref. No.
1	Lecturers' preferences for teaching times can be accommodated.	[1], [12], [16], [17], [18], [32], [33], [37], [40]
2	Gaps between classes should be minimized, such as long breaks between consecutive classes.	[1], [12], [24]
3	Lecturers can request for their preferred venues.	[1], [12], [32], [33]
4	No classes should be scheduled during lunch break (12 p.m. – 2 p.m.).	[1], [2], [24], [37]
5	There should be no classes scheduled before 8 a.m. and after 8.30 p.m.	[1], [33]
6	The maximum hours for consecutive classes should be 4 hours.	[1]
7	There should be an allocation of break periods before other courses.	[3]
8	Each subject's second lecture session should ideally be scheduled on a different day.	[14], [17]
9	Students should not have classes in the last time slot of the day.	[31], [41]
10	Students should not have only one class in a day.	[31], [41]
11	There should be no more than 2 consecutive classes.	[22], [31], [41]
12	Timetables should ensure that as few classrooms as possible are used.	[18], [39]

13	Each course should be assigned to a classroom near the department building or the lecturer's office.	[33]

2.1.4 Types of Evaluation Metric Used to Measure Performance

In [2], some of the evaluation metrics used to measure the performance of a solution (timetable created) includes **computational speed**, **feasibility** and **quality**. If all specified hard constraints are satisfied, then the solution is considered feasible. The quality of the solution which is evaluated by a cost function is an evaluation metric that focuses on satisfying the soft constraints. Soft constraint violations determine the overall quality of the solution. The quality of the solution increases as the number of satisfied soft constraints increases [1]. In terms of computational speed, UCTTP is an NP-hard problem, and it cannot be solved in polynomial time. This is due the exponential growth of the problem and its complexity. Hence, heuristic methods are applied [1], [2].

2.2 Existing University Course Timetable Scheduling Systems

2.2.1: TimetableGeneratorApp by Huzaifa and Abdullah Saleem (GitHub) [42]

This timetable generator app was developed as a final project for the Analysis of Algorithm course by mHuzefa and Abdullah Saleem. It was published on GitHub in year 2020. The system utilizes Django and Python to automate the scheduling of activities. According to the team, it was initially planned to implement a Greedy algorithm to handle the scheduling, the project ended up using a Randomized Brute Force approach. This is due to the challenges of fulfilling the requirements with the Greedy method. Other than that, it was mentioned that the algorithm is connected with a backend database that saves the data which will be processed by the algorithm before generating the timetable.

In addition, this application features a range of user interfaces with a user-friendly GUI that facilitates the scheduling of timetables. The application includes interfaces to add courses, professors, classrooms, and sections (see Figure 2.2.1.1, Figure 2.2.1.2, Figure 2.2.1.3, Figure 2.2.1.4, Figure 2.2.1.5, Figure 2.2.1.6, Figure

2.2.1.7, Figure 2.2.2.11). It also supports additional CRUD (Create, Read, Update, Delete) functions such as viewing, updating, and deleting timetables (see Figure 2.2.1.8, Figure 2.2.1.9, Figure 2.2.2.10). Users can export the generated timetable in PDF file format (see Figure 2.2.2.10).

Strengths of TimetableGeneratorApp:

1. User-friendly Interface: The application provides an intuitive GUI that simplifies user experience when it comes to data input and timetable generation. Hence, users does not need tutorials on how to navigate the application.
2. Cross-Device Responsiveness: The application is responsive on both desktop and mobile devices. This enables usage across various devices.
3. PDF Export: Allows users to export and download timetables in PDF format.
4. CRUD Functions: Enables basic CRUD functions such as view, update and delete.

Weaknesses of TimetableGeneratorApp:

1. Algorithm Limitations: The transition from a Greedy algorithm to a Randomized Brute Force method may result in less optimized timetables.
2. Incomplete Database Integration: Due to the ongoing issue with connecting the algorithm to the database, the application's functionality and automation capabilities are limited.
3. Single Class Limitation: Support timetable generation for one class at a time which may be restrictive for larger institutions.
4. Limited Features and Functionalities: While a simple and intuitive GUI is provided, the application only offers basic functions and lacks additional features and functionalities to support administration use.
5. Constraints Input: Does not allow users to input or set soft constraints for the timetable.

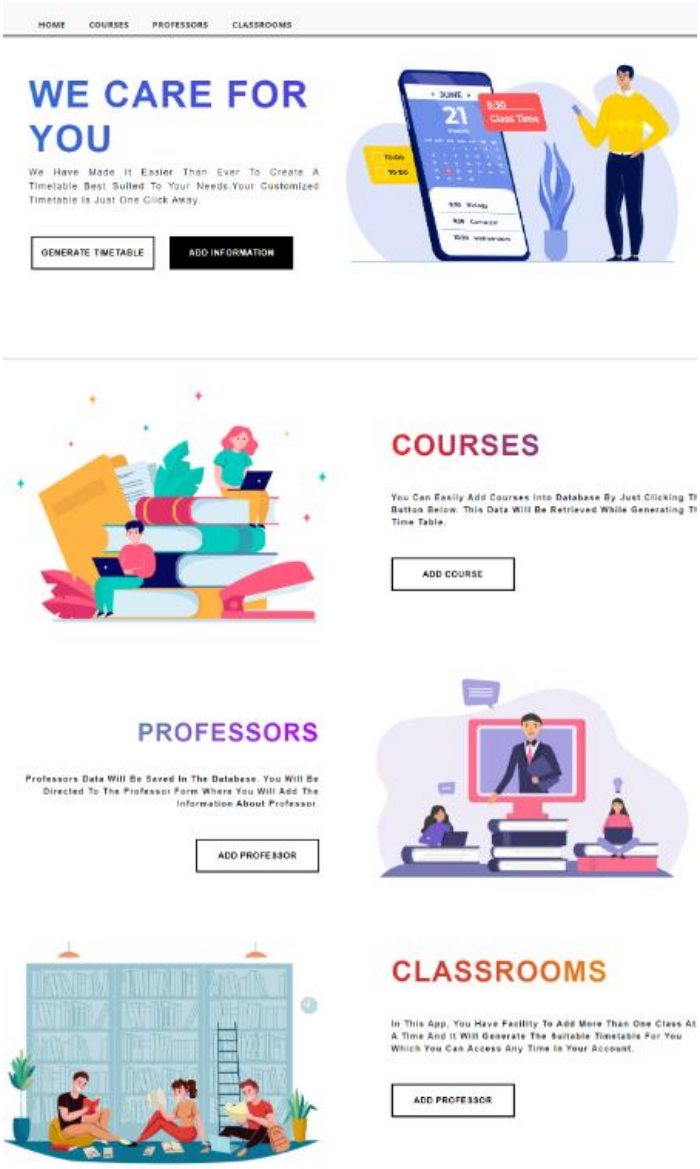

Figure 2.2.1.1 Main Page of TimetableGeneratorApp

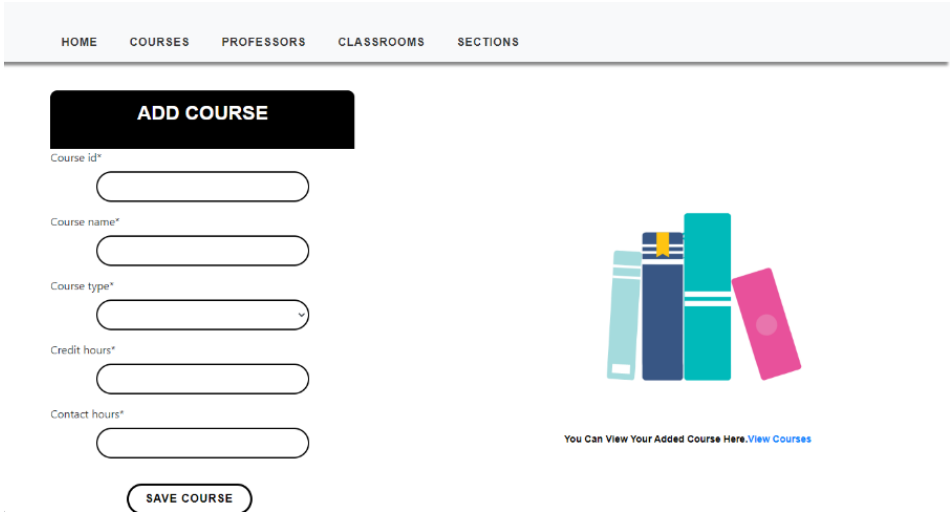

Figure 2.2.1.2 Add Course Page of TimetableGeneratorApp (Desktop)

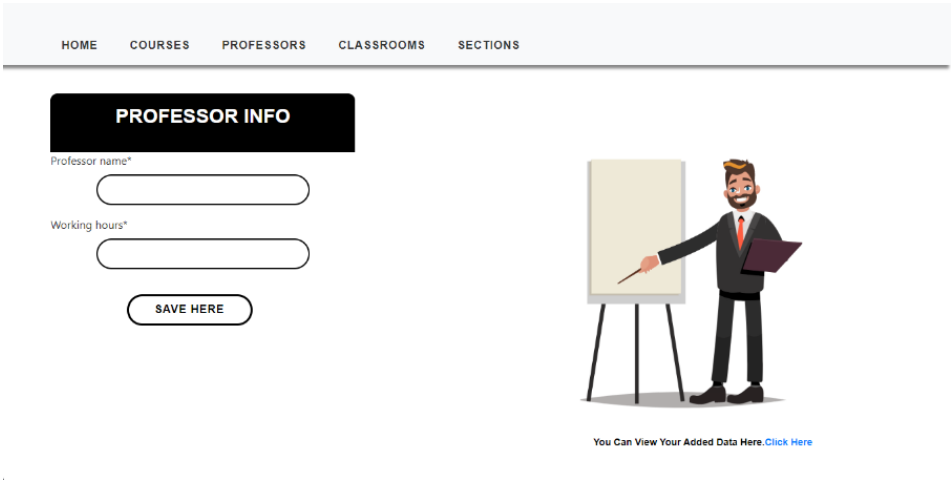

Figure 2.2.1.3 Add Professor Page of TimetableGeneratorApp (Desktop)

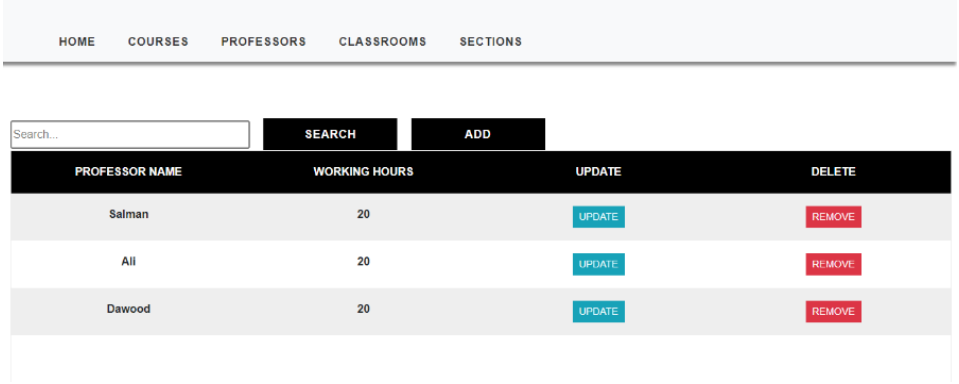

Figure 2.2.1.4 Professor Details Table of TimetableGeneratorApp (Desktop)

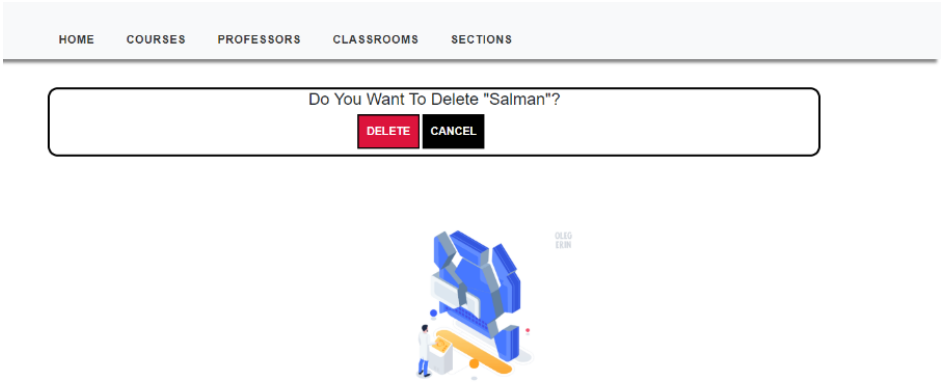

Figure 2.2.1.5 Delete Professor Confirmation Page of TimetableGeneratorApp (Desktop)

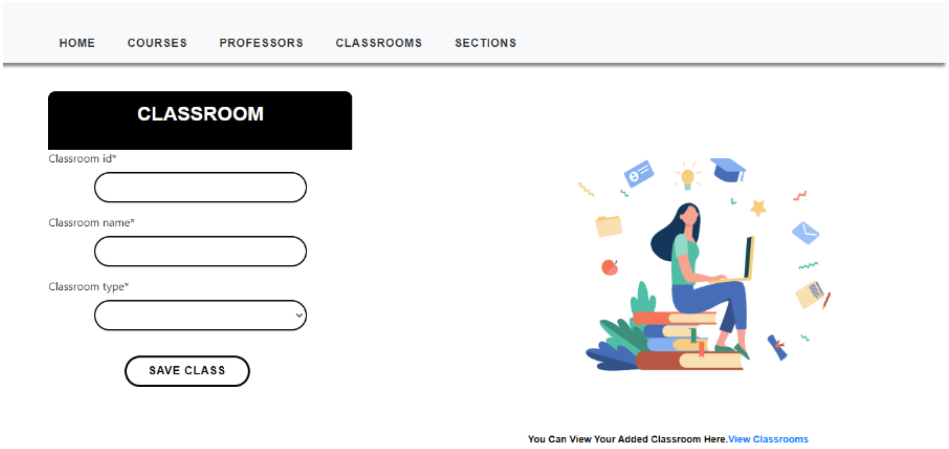

Figure 2.2.1.6 Add Classroom Page of TimetableGeneratorApp (Desktop)

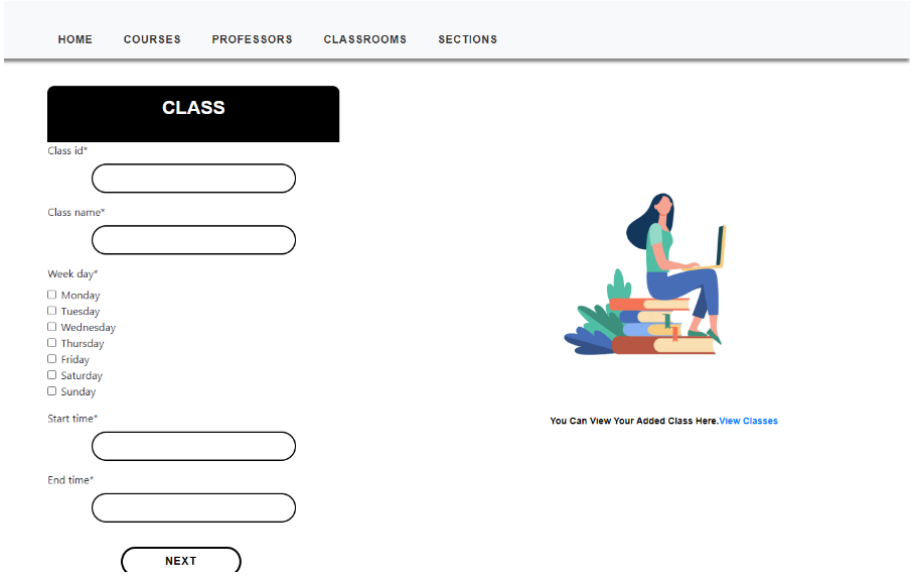

Figure 2.2.1.7 Add Classroom Details Page of TimetableGeneratorApp (Desktop)

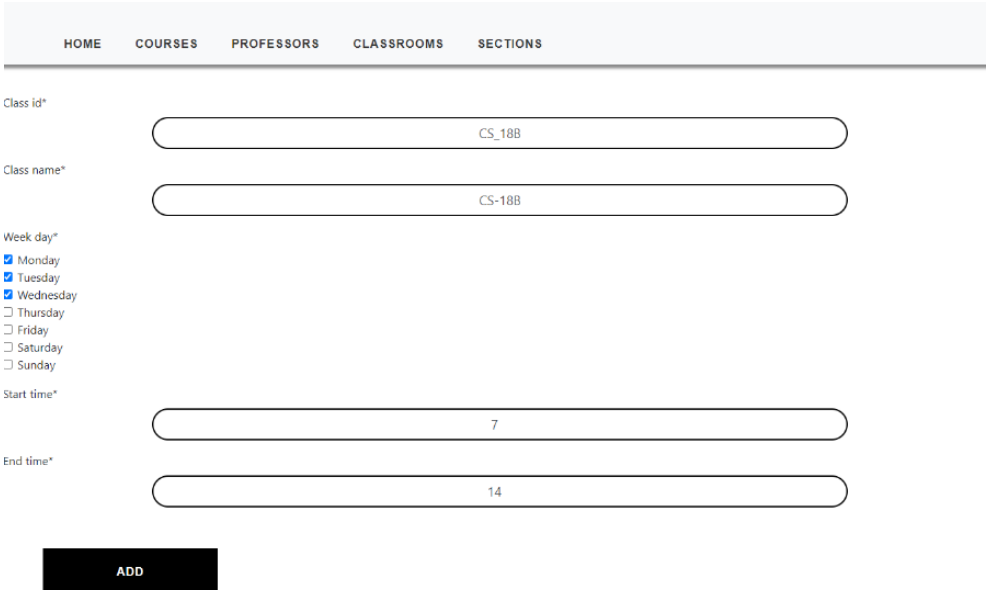

Figure 2.2.1.8 Update Page of TimetableGeneratorApp (Desktop)

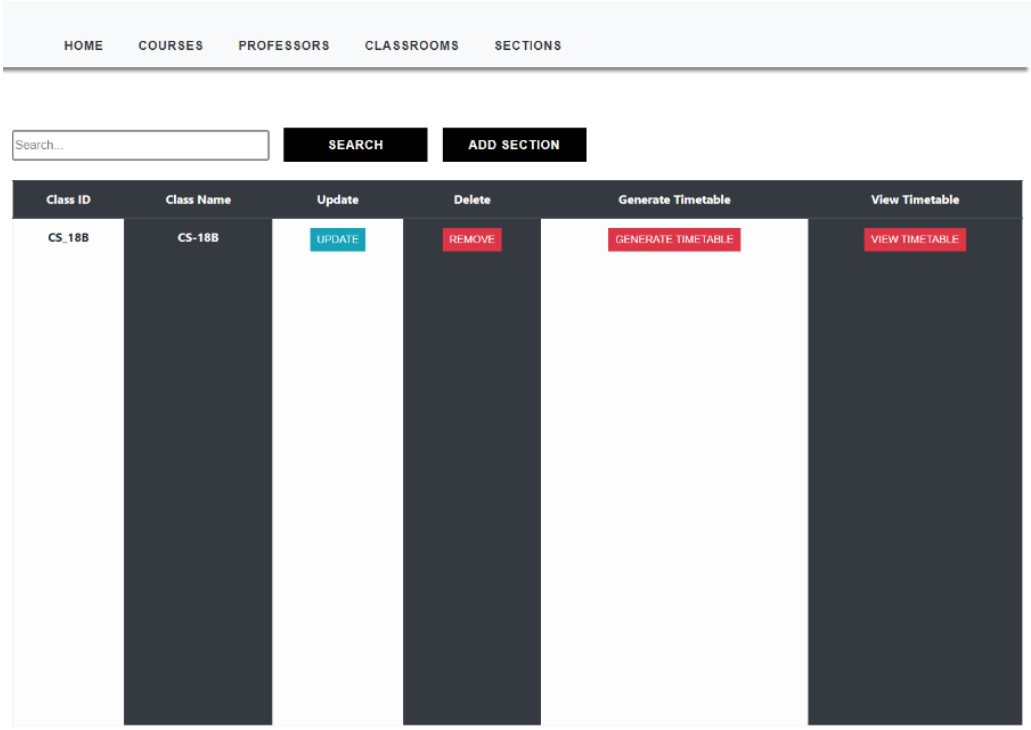

Figure 2.2.1.9 Class Table Page of TimetableGeneratorApp (Desktop)

Figure 2.2.1.10 Generated Timetable of TimetableGeneratorApp (Desktop)

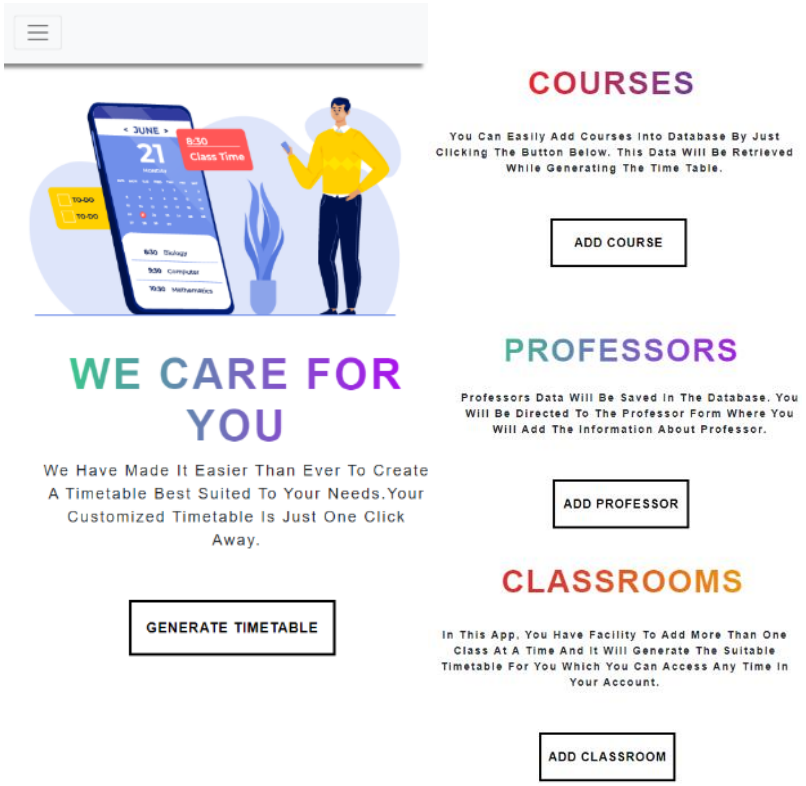

Figure 2.2.1.11 Mobile Layout of TimetableGeneratorApp

2.2.2: timetable-generator by Olayiwola Odunsi, Richman Clifford Jr and Bilal Rasool (GitHub) [43]

This web application was developed using Laravel PHP framework and jQuery by Olayiwola Odunsi, Richman Clifford Jr and Bilal Rasool. It was published on GitHub in year 2018. The main objective of this application is to facilitate timetable generation for a college, enabling users to input data. It utilizes Genetic Algorithm to generate the timetables and timetables are generated as background jobs within Laravel as users generate timetables.

The main page of this application features a comprehensive dashboard that is both user-friendly and intuitive. The dashboard enables users to navigate the application conveniently as it displays a sidebar menu with buttons leading to pages such as Rooms, Courses, Professors, Classes, Periods, User Account and Log Out (see Figure 2.2.2.1). It also shows the current number of lecture rooms, courses, professors and classes (see Figure 2.2.2.2, Figure 2.2.2.3). Besides, there is a table displaying a list of existing timetables and their details such as name, status, and the option to print it out. It supports CRUD functions such as create, update, delete and enables users to view the generated timetable via the print function. Through the print function, it prompts users to save it in PDF format beforehand (see Figure 2.2.2.4).

Strengths of timetable-generator:

1. User-Friendly Interface: The application features an intuitive dashboard that simplifies user navigation, making it accessible even for non-technical users.
2. PDF Export: Allows users to export and download timetables in PDF format.
3. Print Function: Allows users to print their generated timetables.
4. CRUD Functions: Enables basic CRUD functions such as view, update and delete.

Weaknesses of timetable-generator:

1. Tight Algorithm Coupling: The Genetic Algorithm currently tightly integrated with the Laravel framework limits its flexibility and reusability in other applications.
2. Limited Features and Functionalities: The application lacks more advanced features and functionalities to support administration use.

3. The Genetic Algorithm's implementation may not fully address all scheduling requirements. This may potentially lead to suboptimal timetables.
4. Constraints Input: Does not allow users to input or set soft constraints for the timetable.

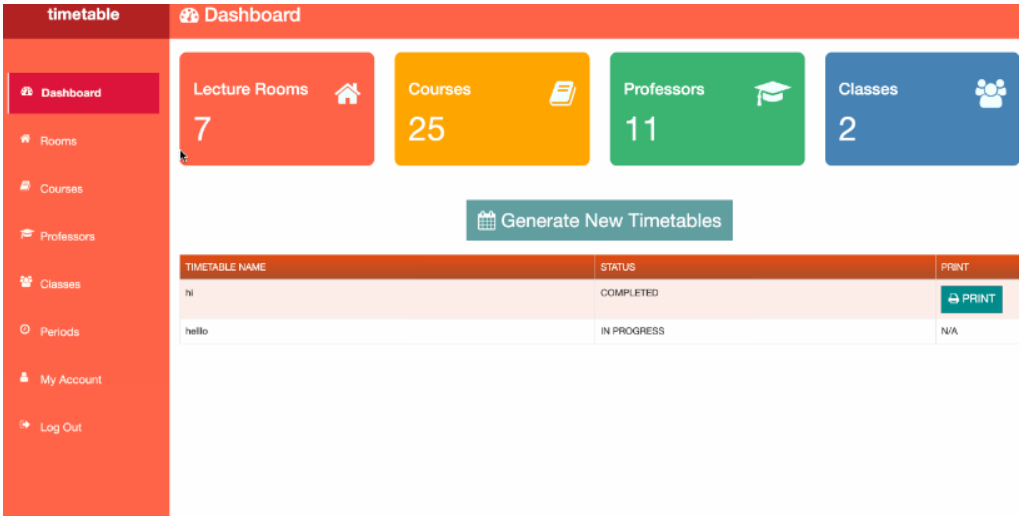

Figure 2.2.2.1 Main Page (Dashboard) of timetable-generator

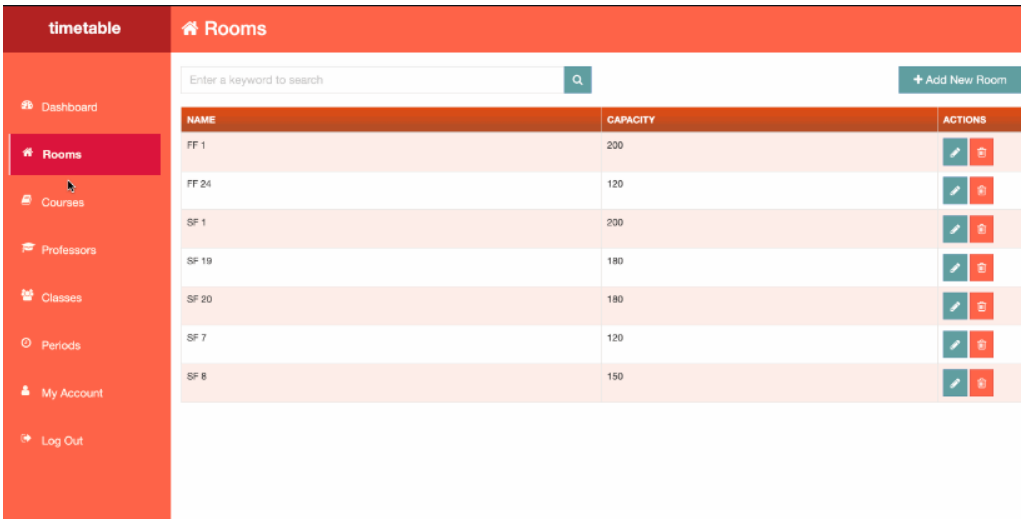

Figure 2.2.2.2 Rooms Page of timetable-generator

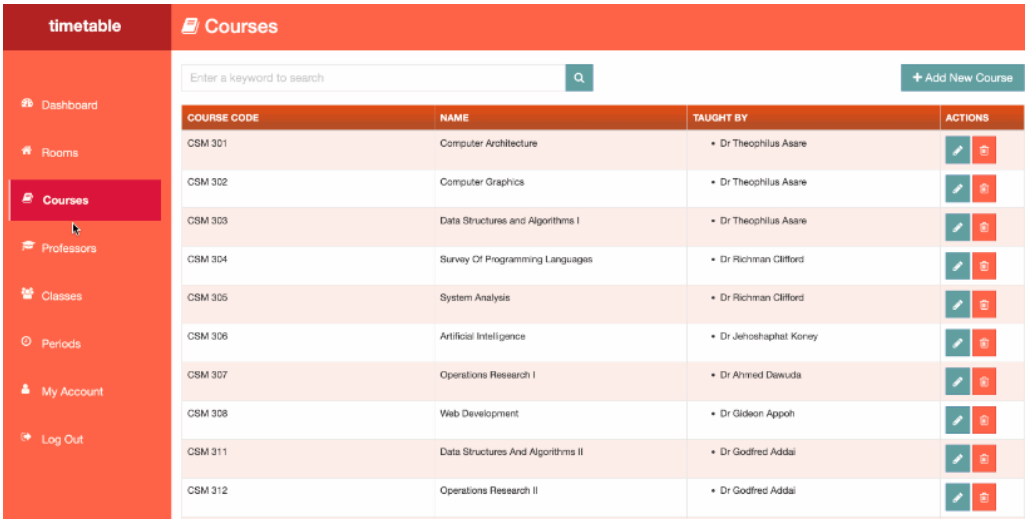

Figure 2.2.2.3 Courses Page of timetable-generator

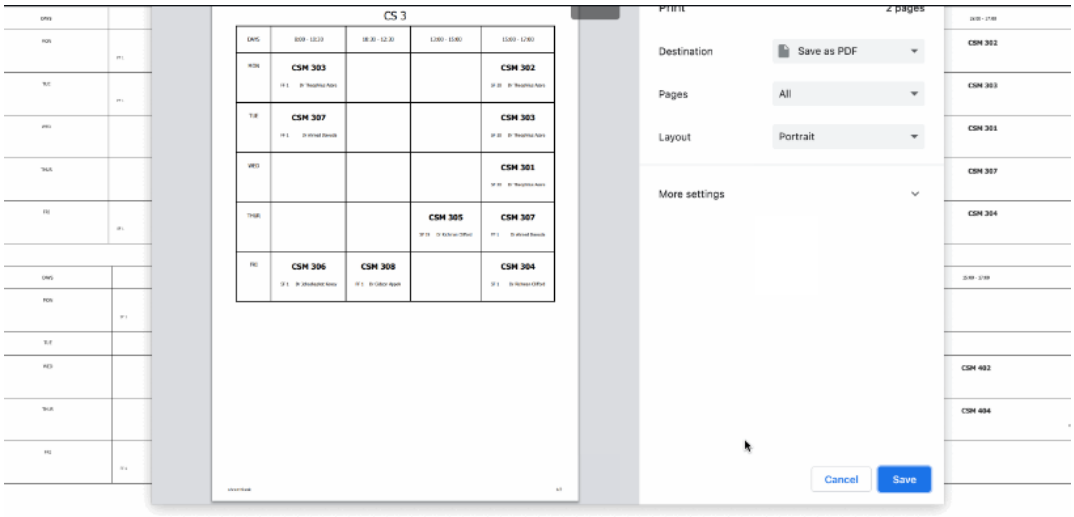

Figure 2.2.2.4 Save as PDF and Print Page of timetable-generator

2.2.3: UniTime by UniTime.org [44]

This educational scheduling system is an open-source system developed by UniTime.org. The system was originally developed through a collaborative effort between faculty, students, and staff from universities across North America and Europe, before becoming a sponsored initiative under the Apereo Foundation in March 2015. There are a few components integrated into the system, such as Course Timetabling & Management, Examination Timetabling, Event Management, and Student Scheduling. The Course Timetabling & Management component aims to assign each course a time slot that avoids conflicts for students, considering factors such as faculty availability,

rooms, and other constraints. UniTime minimizes student course conflicts by using actual student demands, curricula, and historical patterns. This is done while balancing faculty and room preferences. It also incorporates advanced algorithms and allows for different scheduling scenarios, supporting centralized, distributed and hybrid scheduling approaches.

Additionally, the system is a server-client with a web-based interface that using Java J2EE and SQL databases. It handles large-scale problems such as 9,000 classes and 39,000 students. It allows for manual or automated timetabling and offers extensibility for customization. The solver uses a constraint-based approach using a Constraint Solver Library which is a local search-based framework. This solver addresses various constraint satisfaction and optimization problems, identifying inconsistencies in input data and providing automated solutions. Some of the application's highlighted features include data entry, preferences and requirements setting, constraint setting, PDF export, CSV export, add courses, delete courses and many more (see Figure 2.2.3.1, Figure 2.2.3.2, Figure 2.2.3.3, Figure 2.2.3.4, Figure 2.2.3.5, Figure 2.2.3.6). Data entry can be done through a web-based interface and a series of steps. Overall, UniTime provides a comprehensive set of features and functionalities for the users. However, because of its wide range of features, users may find it too complex or difficult to navigate without prior training or tutorials.

Strengths of UniTime:

1. Comprehensive Application: UniTime offers a comprehensive and wide range of features for its users. It also covers a few scheduling needs, including course timetabling, examination timetabling, event management, and student scheduling.
2. Open Source and Extensible: The system is open-source and highly customizable, allowing institutions to modify it based on their specific requirements.
3. Multi-Component Integration: UniTime integrates multiple scheduling components such as course timetabling, examination timetabling, event management and student scheduling, which helps institutions manage all scheduling activities within a single platform.

4. CSV and PDF Export Feature: The generated timetable can be save and exported as a CSV or PDF file.

Weaknesses of UniTime:

1. Complexity: The system can be complex for users unfamiliar with it as it might require training to navigate its extensive features.
2. High Setup Effort: Setup steps such as data entry and customization can be time-consuming as many steps are required to get the results.
3. Limited Mobile Support: UniTime is primarily web-based. While it can be accessed on mobile browsers, it lacks a dedicated mobile application for easier use on phones and tablets.

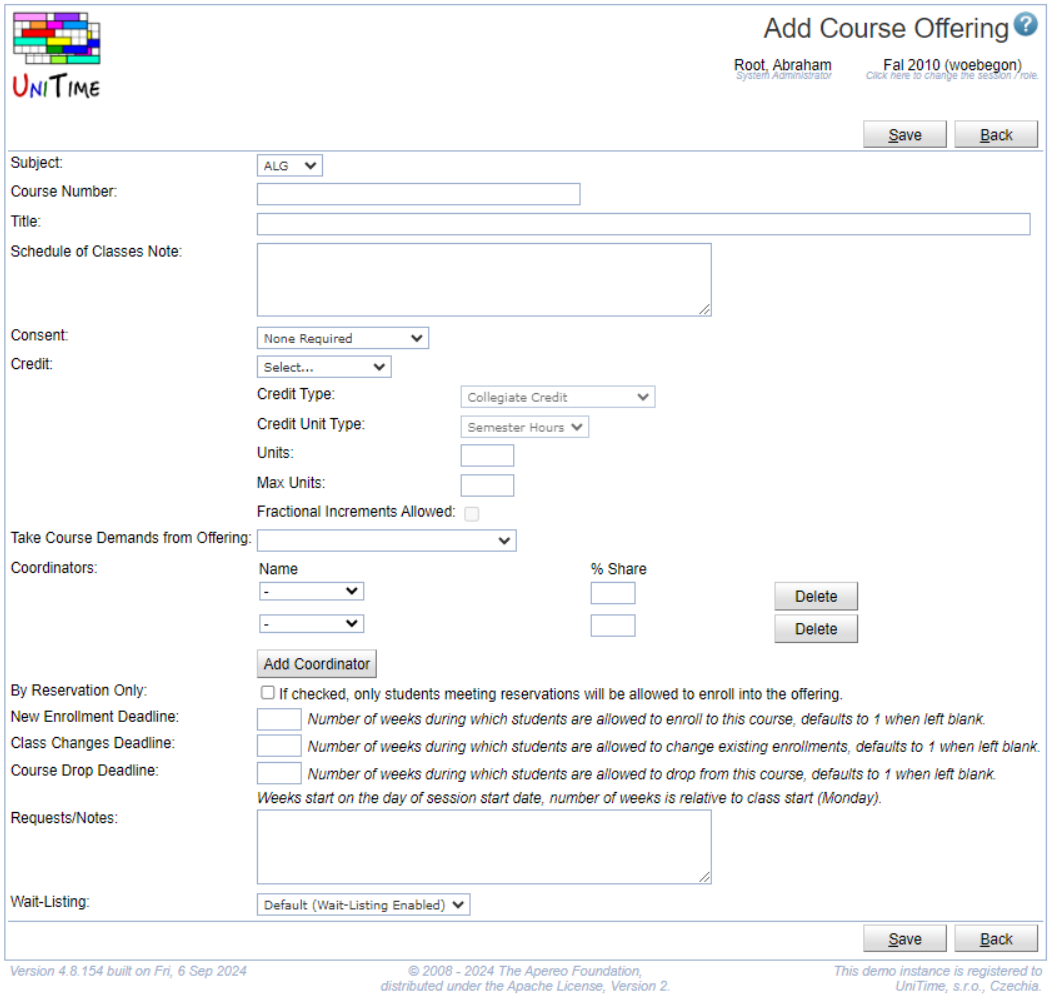

Figure 2.2.3.1 Data Entry for Adding Course

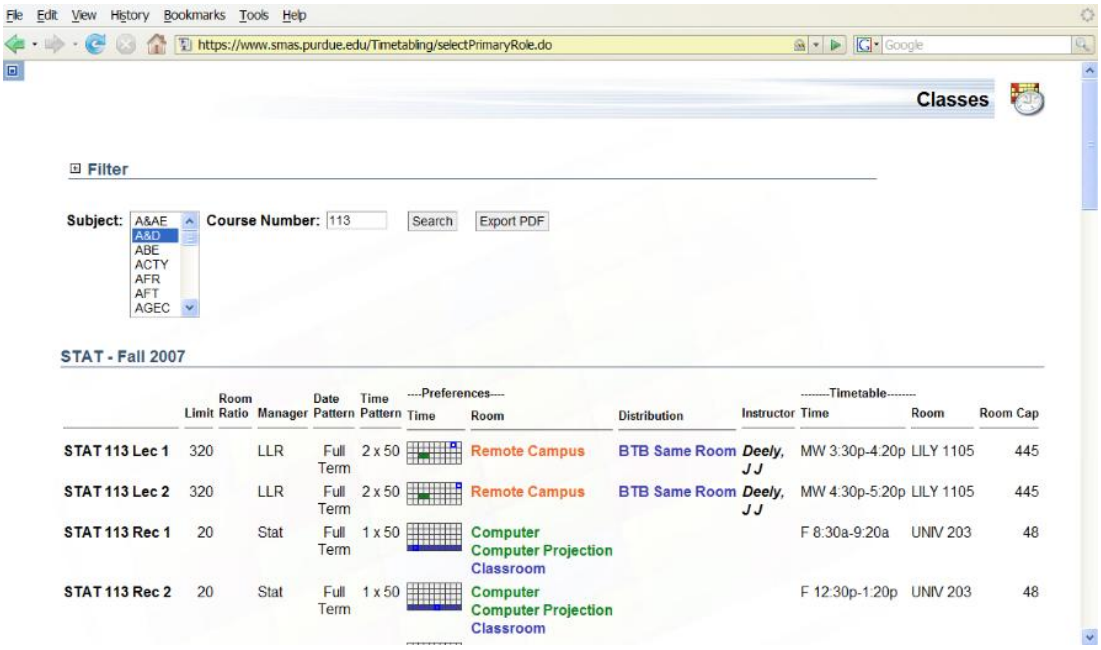

Figure 2.2.3.2 User Interface for Classes

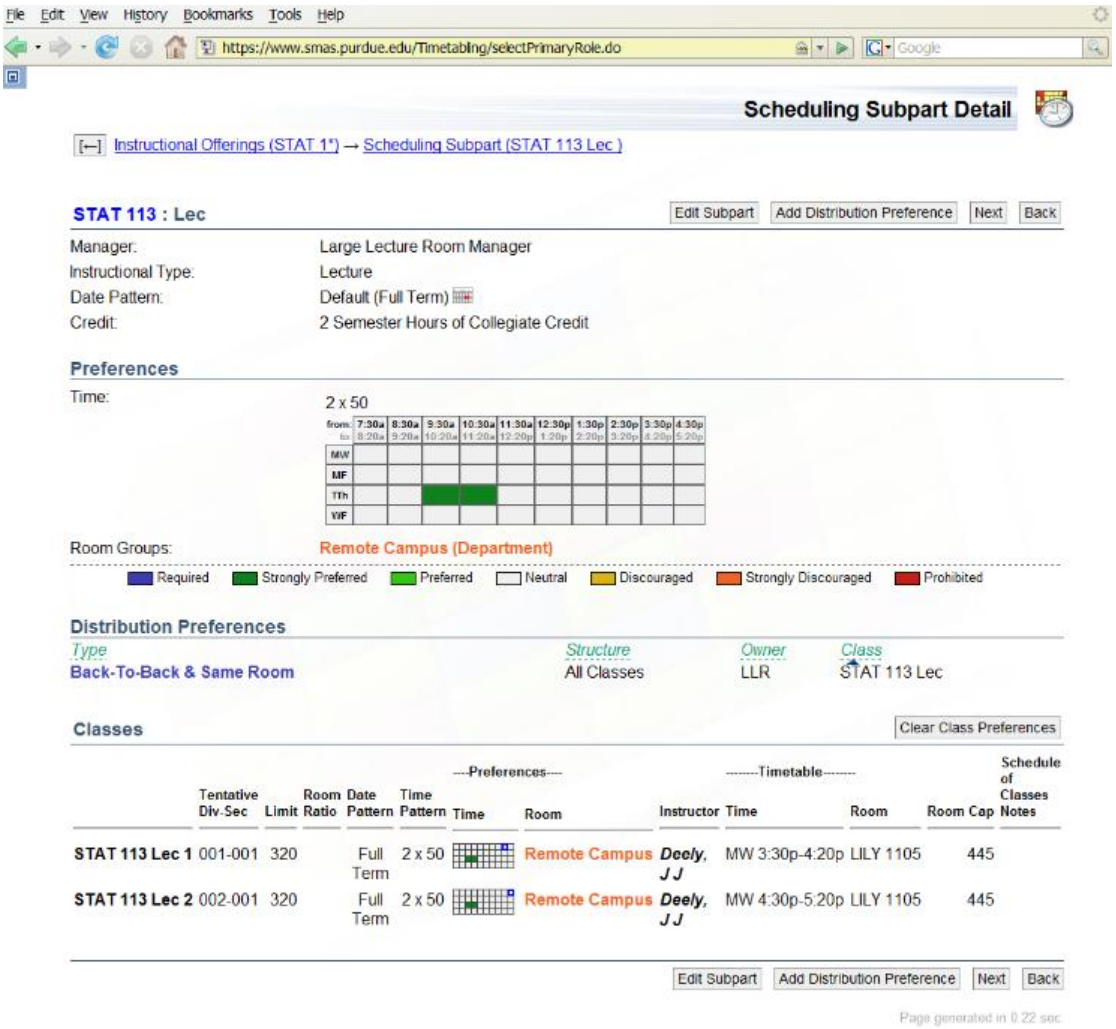

Figure 2.2.3.3 User Interface for Scheduling Subpart Detail

	Demand	Mins Per Week	Limit	Time Pattern	----Preferences---- Time	Room	Distribution	Instructor
MA 170 STAT 170	62		40					
Lecture		50	40	1 x 50		Classroom		
Laboratory		150	40	3 x 50		ENAD Dell 2.8 machines	BTB	
Lec 1		50	40	1 x 50		Classroom		S. Bell
Lab 1		150	20	3 x 50		ENAD Dell 2.8 machines	BTB	J. Beckley
Lab 2		150	20	3 x 50		ENAD Dell 2.8 machines	BTB	J. Beckley

Figure 2.2.3.4 Preferences

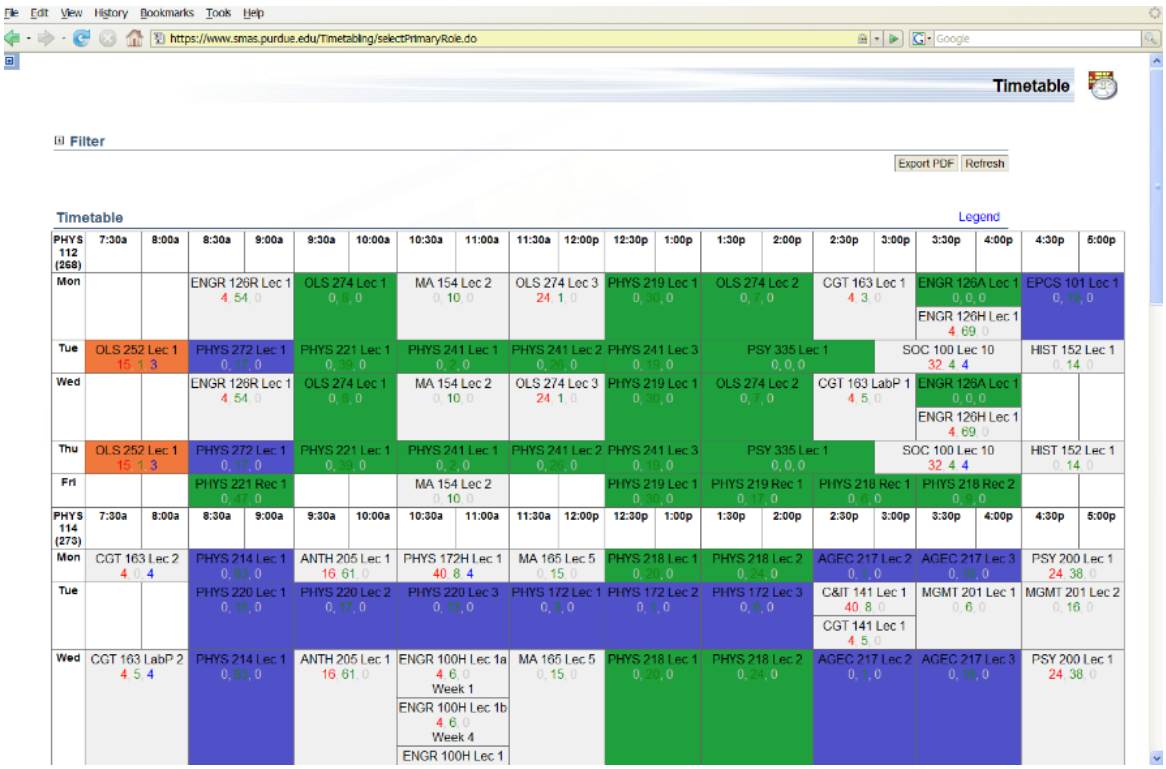

Figure 2.2.3.5 Generated Timetable

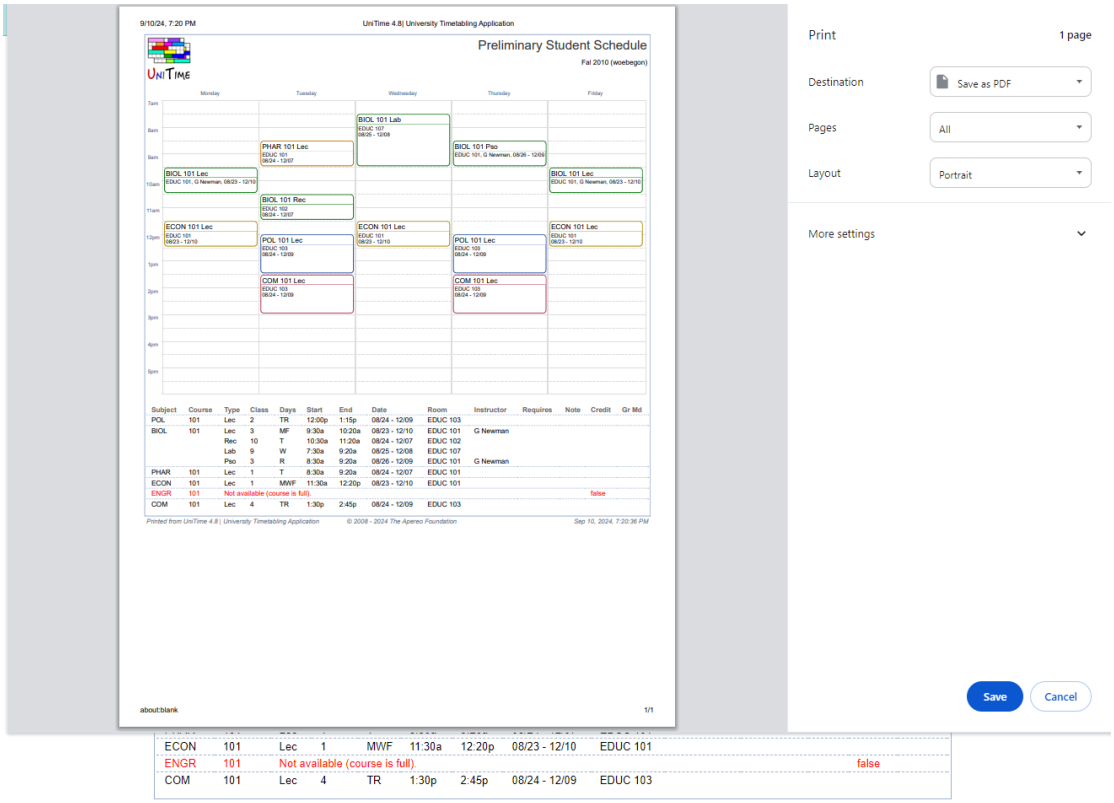

Figure 2.2.3.6 PDF Export

2.2.4 Comparison of Similar Existing Timetable Scheduling Systems

Table 2.2.4.1 Table of Comparison for The Systems

Features/Software	TimetableGeneratorApp	timetable-generator	UniTime
CRUD Features	Yes	Yes	Yes
Generate Timetable	Yes	Yes	Yes
PDF Export	Yes	Yes	Yes
CSV Export	No	No	Yes
Availability on Mobile	Yes	No	No
User-Friendly and Intuitive UI	Yes	Yes	No
Comprehensive Features	No	No	Yes

Based on Table 2.2.4.1 above, all three systems support CRUD (Create, Read, Update, Delete) features which are essential for managing timetable data and administrative work to schedule the timetables. Most importantly, all of the three systems have the main functionality to generate timetables. In addition, the systems also allow exporting the timetables in PDF format. This provides a convenient way for users to access and distribute the timetables. However, UniTime is the only system that supports exporting timetables in CSV format. This feature is useful for users who may want to export or analyze the data in spreadsheet applications. TimetableGeneratorApp and timetable-generator do not support this feature.

Other than that, TimetableGeneratorApp is the only one available on mobile platforms, making it more accessible for users who need to manage or view timetables. Neither timetable-generator nor UniTime offers mobile support. Both TimetableGeneratorApp and timetable-generator have user-friendly and intuitive interfaces. UniTime has a more complex interface that may require the users to go through prior training or tutorials to navigate it. However, while UniTime has a more complex interface, it is equipped with a comprehensive set of features and functionalities compared to TimetableGeneratorApp and timetable-generator whose user interfaces are simple but lacks additional functionalities.

2.2.5 Literature Review Findings

There have been numerous studies on university course timetabling that have examined multiple optimization techniques to satisfy hard and soft constraints. Over the years, several algorithms have been proposed to solve the problem, namely Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Simulated Annealing (SA), Tabu Search, hybrid approaches etc. [1], [12], [7], [30], [45]. However, no single algorithm can solve every timetabling issue optimally. This is mainly due to the diverse set of constraints, such as resource availability, room capacities, lecturer preferences, and class clashes, which can vary from one institution to another.

Additionally, the majority of research on timetable scheduling systems that apply optimization approaches focuses mostly on satisfying hard constraints like avoiding class clashes and ensuring room capacity. Soft constraints that have often been

dealt with include lecturer preferences and room allocation. However, the spatial aspect of timetabling, such as the minimization of travel distance between consecutive classes, has received limited attention in the literature. Hence, there is a research gap in the lack of attention to real-world constraints related to venue proximity and travel time between consecutive classes.

While there has been a recent study that introduced an recommendation system based on digital twins that incorporates traveling distances and vertical transitions into the timetabling process, the integration of spatial constraints remains relatively uncommon in modern timetabling systems [46]. Most classic models do not take into account the physical distances between classrooms, and thus the generated schedules may have students and lecturers travel significant distances between back-to-back classes.

Furthermore, the timetabling problem is already considered an NP-hard problem. Additional constraints might make it harder for one to predict the complexity of the problem. Hence, it is difficult to predict which algorithm would perform best, especially for constraints dealing with spatial and temporal dimensions. In such a situation, the selection of a suitable optimization algorithm is a crucial choice.

Genetic Algorithm has been widely utilized in the literature due its flexibility, population-based search procedure, and compatibility with multi-objective optimization. It is also frequently used as the foundation for hybrid algorithms, whereby it is combined with other algorithms to form hybrid algorithms to enhance performance [7], [30], [45]. For instance, GA has hybridized with PSO [34], Simulated Annealing [14], Tabu Search algorithm [39], Ant Colony Optimization [40], Guided and Local Search Strategies [21], [8], [37], [45] to compensate drawbacks of certain algorithms and improve the quality of solutions.

The main objective of this project is to include the ‘Proximity and Travel Minimization Constraint’. This restriction seeks to minimize unnecessary travelling across campus by having the locations of back-to-back classes close together. The Genetic Algorithm (GA), which is often used in the university course timetable scheduling problem is widely noted for its adaptability and robustness. It is chosen as the primary optimization strategy because of the unpredictability of this new constraint. Hence, because of these characteristics, GA is a good approach to the university course

timetabling problem, especially when faced with new restrictions such as the travel and proximity constraint.

2.2.6 Chapter 2 Summary

This chapter has reviewed some of the approaches used to solve the University Course Timetabling Problem (UCTTP). Genetic Algorithm (GA) was compared with other AI methods such as Particle Swarm Optimization and Simulated Annealing. However, there is more emphasis on GA because of their robustness, flexibility in handling complex constraints and large search spaces. This chapter also covered some of the main topics of GA, including chromosome representation, gene encoding, fitness computation, crossover, selection, and mutation techniques.

Most of the studies focuses on satisfying hard and soft constraints. However, there are few studies considering spatial constraints like travel distance from one location to another, leading to a research gap. In addition, present methods lack flexibility in allowing new constraints or optimizing closeness in locations. Thus, the project adopts GA to handle the complexity of the problem and introduces the 'Proximity and Travel Minimization Constraint' to solve an operational problem underserved by current systems.

In addition, a comparison of timetabling systems revealed that most systems do not accommodate custom constraints or are not mobile-capable, and they typically do not address the problems of travel distances. This further suggests the need for a more flexible, and user-friendly timetabling tool.

In summary, the findings from the literature review concluded that GA will be used as the main optimization technique for this project due to its proven capability in solving UCTTP. Furthermore, with the introduction of the new 'Proximity and Travel Minimization Constraint', this project aims to bridge the research gap.

CHAPTER 3

System Methodology

The processes of the project were categorized into different phases in the development, which were project pre-development, data pre-processing, model training architecture building and data training, and prediction on test dataset.

3.1 Project Development

The first step in starting the project is the **planning phase**, where the project's scope, objectives, and requirements are defined, covering the aspects such as the dataset to be used, target users, and the development platform. The objectives are set to solve the identified problems. During this phase, the project's concept and initial draft for developing the timetabling system were created and the current challenges like the lack of certain constraints and functionality in the existing systems are addressed. On the other hand, algorithms and software were also identified to enable timetable generation.

In the next phase, the **analysis phase**, the selected algorithms and software were reviewed to assess their strengths and weaknesses, focusing on factors like time complexity, accuracy and performance. Additionally, existing timetabling systems are studied by having their system requirements and functionalities analysed. Analysing the functionalities of current system is used to identify certain weaknesses that can be further improved to propose a newer and better system. Besides, the structure of university timetables is examined through the documentation available on existing timetabling systems.

In the **design phase**, the system's structure is designed, and GA is selected, including the new constraint and the different stages of the algorithm such as chromosome structure, selection techniques, crossover techniques, mutation techniques, and replacement techniques. Diagrams like UML, use-case, and system architecture are also created, alongside designing the system database and selecting the hardware and software components.

The **implementation phase** is where the proposed system will be implemented or programmed. This phase involves developing the system as a mobile application using Flutter for the interface and Python for the algorithm handling. New requirements or constraints are constantly adjusted, and the changes will be implemented in the system during the development of the application.

In the **testing phase**, the developed system will be thoroughly evaluated to ensure that there are no errors in scheduling the timetable and that all constraints are adhered to. Each constraint will be tested using specific methods. For example, the system will verify that no student attends two classes at the same time slot, fulfilling the constraint that "A student cannot attend two courses (events) scheduled simultaneously".

Lastly, the **maintenance phase** involves carrying out ongoing updates to address any errors or bugs discovered in the system over time. This is done to ensure the system's continued reliability. Besides, updates to upgrade or add new functionality to the system will also be considered.

3.1.1 Project Gantt Chart

October 2024 Trimester (Proposal Writing)										
Tasks	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13
Decide on FYP Title										
Find suitable research papers										
Find similar existing systems										
Draft Chapter 2 (Literature Review)										
Draft Chapter 1 (Introduction)										
Draft Chapter 3 (Project Scope and Objectives)										
Draft Chapter 4 (Project Methodology)										
Prepare presentation slides										
Presentation										

Figure 3.1.1.1 Timeline for Proposal Writing Phase of the Project

February 2025 Trimester (FYP1)													
Tasks	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week13
Update Chapter 1													
Update Chapter 2													
Draft Chapter 3 (Proposed Method/Approach)													
Design Dataset													
Design Database													
Develop Algorithm													
Test Algorithm													
Draft Chapter 4 (Preliminary Work)													
Draft Chapter 5 (Conclusion)													
Prepare Plagiarism Check Result													
Design Poster													
Presentation													

Figure 3.1.1.2 Timeline for FYP1

June 2025 Trimester (FYP2)													
Tasks	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week13
Update Chapter 1													
Update Chapter 2													
Update Chapter 3													
Draft Chapter 4 (System Design)													
Design UI and Layout													
Develop UI and Layout													
Refine Algorithm													
Draft Chapter 5 (System Implementation)													
System Testing													
Draft Chapter 6 (System Evaluation & Discussion)													
Prepare Plagiarism Check Result													
Design Poster													
Presentation													

Figure 3.1.1.3 Timeline for FYP2

The figures above, Figure 3.1.1.1, Figure 3.1.1.2, and Figure 3.1.1.3, represents the timelines for the entire project, divided into three semesters' time. Each semester consists of 14 weeks, while the time given to complete the tasks is 12-13 weeks' time. The leftmost column lists the key tasks involved in the development of the project. The columns represent the timeline from each week. Each week is dedicated to completing various tasks, with some overlapping.

3.2 Data Collection for Timetabling Tool

In this project, data collection and analysis of the current timetable system of UTAR's Faculty of Information and Communication Technology (FICT) are conducted. Timetables of the various programmes conducted for the current trimester (February 2025 trimester) were gathered from UTAR FICT's official website.

The collected timetables were then examined to identify their underlying structure and main components. Important details included the list of courses offered to students from different courses and different trimesters, the lecturers responsible for each subject, the names and types of rooms used for teaching sessions, and whether any

special or fixed time slots existed in the schedule for the week. Figures 3.2.1, 3.2.2, and 3.2.3 below show some samples of the collected timetable data.

UNIVERSITI TUNKU ABDUL RAHMAN
FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY (KAMPAR CAMPUS)
BACHELOR OF COMPUTER SCIENCE (HONS)
10 February 2025 – 18 May 2025 (Year 1, Trimester 1)

Prepared On:
Revised On:

Day	8am-9am	9am-10am	10am-11am	11am-12pm	12pm-1pm	1pm-2pm	2pm-3pm	3pm-4pm	4pm-5pm	5pm-6pm	6pm-7pm	7pm-8pm
Monday	UCCM1153 (T1) N001	UCCM1153 (T2) N001			UCCN1004 (L) LDK3	UCCD1004 (L) LDK3	UCCD1143 (L) LDK3		UCCD2003 (L) LDK3			
Tuesday	UCCD1004 (P1) N008A		UCCM1153 (T4) N107 UCCD1143 (T1) N003 UCCD2003 (T2) N007	UCCD1143 (T2) N003 UCCD1004 (P4) N010B	UCCD2003 (T1) N106		UCCM1153 (L) LDK3	UCCD2003 (L) LDK3				
Wednesday			UCCN1004 (L) LDK3	UCCD1004 (L) LDK3					UCCM1153 (L) LDK3			
Thursday	UCCD1143 (L) LDK3			UCCN1004 (P4) N010B UCCD1004 (P3) N110A	UCCD1004 (P2) N008A	UCCD1143 (T3) N003 UCCD2003 (T4) N107 UCCN1004 (P1) N110A	UCCD1143 (T4) N003 UCCD2003 (T3) N107 UCCN1004 (P2) N009	UCCN1004 (P3) N110A				
Friday	UCCN1004 (P5) N112A		UCCD1004 (P5) N008A		FRIDAY PRAYER		UBMM1011 (L) LDK3					

CORE

UBMM1011	SUN ZI'S ART OF WAR AND BUSINESS STRATEGIES	2L	**Dr Tee Chee Wee**	L
UCCD1004	PROGRAMMING CONCEPTS AND PRACTICES	3L+2P	**Dr Tan Joi San**	L+P
			Ts Dr Phan Koo Yuen	L+P
			Dr Jasmina Khaw Yen Min	P
			Dr Kh'ng Xin Yi	P
			Prof Dr Leung Kar Hang	P
			Dr Ahmad Hakimi Bin Ahmad Sa'ahiry	P
UCCD1143	PROBABILITY AND STATISTICS FOR COMPUTING	3L+1T	**Ts Dr Lim Seng Poh**	L+T
			Dr Chai Tong Yuen	T
UCCD2003	OBJECT-ORIENTED SYSTEMS ANALYSIS AND DESIGN	3L+1T	**Ts Dr Ku Chin Soon**	L+T
			Ts Dr Mogana a/p Vadiveloo	L+T
			Dr Tahayna Bashar M. A.	T
			Cik Puteri Nursyawati Binti Azzuri	T
UCCM1153	INTRODUCTION TO CALCULUS AND APPLICATIONS	3L+1T	**Dr Nur Balqishanis Binti Zainal Abidin**	L+T
			Dr Lem Kong Hoong	L+T
UCCN1004	DATA COMMUNICATIONS AND NETWORKING	3L+2P	**Dr Aun Yichiet**	L+P
			Ms Tan Lyk Yin	P
			Ts Dr Ooi Chek Yee	P
			Dr Muhammad Syaiful Amri Bin Suhaimi	P
			Ts Dr Chang Jing Jing	P
			Dr Fityanul Akhyar	P
			Dr Abdulrahman Aminu Ghali	P
			Dr Rohani binti Bakar	P
			Dr Norliana Binti Muslim	P
			Dr Rahmad Sadli	P
			Dr Teoh Shen Khang	P
			Puan Nor 'Afifah Binti Sabri	P
			Dr Farina Saffa Binti Mohamad Samsampun	P

Figure 3.2.1 Timetable of Students in Computer Science Programme (Year 1 Sem 1)

UNIVERSITI TUNKU ABDUL RAHMAN
FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY (KAMPAR CAMPUS)
BACHELOR OF INFORMATION TECHNOLOGY (HONS) COMMUNICATIONS & NETWORKING
10 February 2025 – 18 May 2025 (Year 2, Trimester 1)

Prepared On:
Revised On:

Day	8am-9am	9am-10am	10am-11am	11am-12pm	12pm-1pm	1pm-2pm	2pm-3pm	3pm-4pm	4pm-5pm	5pm-6pm	6pm-7pm	7pm-8pm
Monday						UCCN2243 (L) LDK1		UCCD2103 (L) LDK1				
Tuesday				UCCN2243 (P) N108			UCCD2043 (T) N007		UAMG1043 (L) LDK3 UALE1083 (T) N001			
Wednesday	UCCD2043 (L) IDK7			UCCD2103 (T) N004	MPU34072 (L) N001 MPU34132 (L) N002 MPU34142 (L) N003			UCCD2044 (P) N104				
Thursday			MPU34102 (L) N001 MPU34152 (L) N002 MPU34162 (L) N003				UCCD2043 (L) LDK3	UCCD2044 (L) LDK2	UCCD2103 (L) LDK1			
Friday	UCCD2044 (L) LDK1			MPU34012 (L) N001 MPU34022 (L) N002 MPU34082 (L) N003	FRIDAY PRAYER		UAMG1043 (T) N003	UALE1083 (L) LDK1				

CORE

UCCD2043	INFORMATION TECHNOLOGY PROJECT MANAGEMENT	3L+1T	**Ts Yong Tien Fui**	L+T
			Dr Zurida Binti Ishak	T
UCCD2044	OBJECT-ORIENTED PROGRAMMING PRACTICES	3L+2P	**Dr Sayed Ahmad Zikri Bin Sayed Aluwee**	L+P
			Ts Dr Chai Meei Tyng	L+P
			Mr Luke Lee Chee Chien	P
			Ms Tseu Kwan Lee	P
			Dr Ng Hui Fuang	P
UCCD2103	OPERATING SYSTEMS	3L+1T	**Ts Wong Chee Siang**	L+T
			Mr Sor Kean Vee	L+T
			Encik Ahmad Zaffry Hadi Bin Mohd Juffry	T
UCCN2243	INTERNETWORKING PRINCIPLES AND PRACTICES	2L+2P	**Ts Dr Gan Ming Lee**	L+P
			Dr Adeb Ali Mohammed Ahmed Al-Samet	P
			Ms Oh Zi Xin	P
			Dr Farina Saffa Binti Mohamad Samsamnun	P
			Dr Abdulrahman Aminu Ghali	P
			Dr Nadeem Muhammad Waqas	P

UCCN-05 (Choose One)

UALE1083	BASIC PROFESSIONAL WRITING	2L+1T	**Puan Ayu Rita Binti Mohamad**	L+T
UAMG1043	INTERPERSONAL COMMUNICATION	2L+1T	**Ms Aruna Raj a/p Devarajoo**	L
			Ms Salomi a/p Simon	T
			Puan Liana binti Mat Nayan	T
			Puan Nor Ez-Zatul Hanani binti Mohamed Rosli	T

MPU34XX2 (Choose One)

MPU34012	SOCIAL ENTREPRENEURSHIP PROJECT	2L	**Ms Mah Siew Huei**	L
MPU34022	ARTS AND CULTURAL PERFORMANCE	2L	**Ms Alison Yoon**	L
MPU34072	ART, CRAFT, AND DESIGN	2L	**Ms Mah Siew Huei**	L
MPU34082	ORAL COMMUNICATION	2L	**Ms Erica Chua Ning Jia**	L
MPU34102	MANAGING PERSONAL FINANCE	2L	**Mr Fong Kah Hoong**	L
MPU34132	MANAGEMENT OF SPORTS ACTIVITY	2L	**Mr Fong Kah Hoong**	L
MPU34142	CRITICAL THINKING, CREATIVE THINKING AND PROBLEM SOLVING	2L	**Mr Kong Hoi Yoon**	L
MPU34152	LEADERSHIP AND TEAMBUILDING	2L	**Ms Erica Chua Ning Jia**	L
MPU34162	BUSINESS PLAN WRITING & PREPARATION	2L	**Mr Kong Hoi Yoon**	L
MPU34182	MASSIVE OPEN ONLINE COURSE (MOOC)	Online	**Cik Puteri Nursyawati Binti Azzuri**	

Figure 3.2.2 Timetable of Students in Communication & Networking Programme (Year 2 Sem 1)

UNIVERSITI TUNKU ABDUL RAHMAN
FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY (KAMPAR CAMPUS)
BACHELOR OF INFORMATION SYSTEMS (HONS) INFORMATION SYSTEMS ENGINEERING
10 February 2025 – 18 May 2025 (Year 3, Trimester 1)

Prepared On:
Revised On:

Day	8am-9am	9am-10am	10am-11am	11am-12pm	12pm-1pm	1pm-2pm	2pm-3pm	3pm-4pm	4pm-5pm	5pm-6pm	6pm-7pm	7pm-8pm
Monday	UCCD3243 (L) LDK5		UCCD3113 (L) N006	UCCD3343 (P2) N010B		UCCD3223 (P2) N110B UCCD3223 (P1) N110A			UBMM2013 (L) IDK1 UBMM1013 (L) IDK7 UBTM1013 (T) N002			
Tuesday	UCCB2333 (L) LDK1 UCCA2103 (L) N005	UCCA3053 (T) N005	UCCD3343 (L) N002	UCCD3113 (L) N006			UCCM1153 (L) LDK3	UCCD1213 (L) DDK1 UCCB1013 (L) IDK7	UAMG1043 (L) LDK3 UBMH1013 (L) LDK1 UBMM2023 (L) IDK4 UBTM1013 (L) IDK1			
Wednesday	UCCD1213 (L) EDK1 UCCB1013 (L) IDK5 UCCD2043 (L) IDK7		UCCD3013 (L) LDK5	UCCD3223 (L) LDK5			UCCD3053 (L) N003	UCCD2043 (T) N107 UCCA2103 (L) N003	UCCM1153 (L) LDK3 UCCB2333 (L) LDK1	UCCA3053 (L) N003		
Thursday	UCCD3053 (L) N001		UCCD3343 (P1) N011A UCCD3243 (P2) N009 UCCD3243 (P1) N009	UCCD3113 (T) N005	UCCA2103 (T) N107	UCCB2333 (T) N001 UCCD3013 (L) LDK2 UCCB1223 (P) N010A	UCCD2043 (L) LDK3	UCCD1213 (T) N003 UCCM1153 (T) N004	UALE1083 (T1) N002 UAMG1043 (T) N003 UBMM2013 (T) N005	UALE1083 (T2) N002 UBMM1013 (T) N004		
Friday	UCCD3053 (T1) N001	UCCD3053 (T2) N001	UCCB1223 (L) LDK1 UCCA3053 (L) N107 UCCD3013 (T) N007	UCCB1013 (T) N007	FRIDAY PRAYER			UALE1083 (L) LDK1 UBMH1013 (T) N004 UBMM2023 (T) N005				

CORE

UCCA3583	PROJECT I			
UCCD3053	INFORMATION TECHNOLOGY PROFESSIONAL ETHICS	3L+1T	Mr Lee Kim Hoe @ Farhan Lee Bin Abdullah	L+T
UCCD3223	MOBILE APPLICATIONS DEVELOPMENT	2L+2P	Mr Tan Chiang Kang	L+P
			Mr Tou Jing Yi	P
			Puan Syazwani Binti Yahya	P

UCIA-05 (Choose One)

UALE1083	BASIC PROFESSIONAL WRITING	2L+1T	Puan Ayu Rita Binti Mohamad	L+T
UAMG1043	INTERPERSONAL COMMUNICATION	2L+1T	Ms Aruna Raj a/p Devarajoo	L
			Ms Salomi a/p Simon	T
			Puan Liana binti Mat Nayan	T
			Puan Nor Ez-Zatul Hanani binti Mohamed Rosli	T
UBMH1013	ORGANISATION AND HUMAN RESOURCE	2L+1.5T	Ms Rajaletchumy a/p Mani	L+T
UBMM1013	MANAGEMENT PRINCIPLES	2L+1.5T	Dr Gopalan a/l Raman	L
			Dr Charles Ramendran a/l S PR Subramaniam	T
UBMM2013	OPERATIONS MANAGEMENT	2L+1T	Dr Chia Mei Si	L+T
UBMM2023	ORGANISATIONAL BEHAVIOUR	2L+1.5T	Cik Norhayati binti Md Isa	L+T
UBTM1013	PRINCIPLES OF MARKETING	2L+1.5T	Dr Loh Xiu Ming	L
			Mr K.Raja Kumar a/l K.Kathiravelu	T

UCIA-06 (Choose One)

UCCA2103	MANAGEMENT INFORMATION SYSTEMS	3L+1T	Puan Ayu Norafida Binti Ayob	L+T
UCCA3053	KNOWLEDGE MANAGEMENT	3L+1T	Ts Dr Lim Ean Heng	L+T
UCCB1013	FUNDAMENTALS OF BUSINESS SYSTEMS	3L+1T	Ts Tey Chee Chieh	L+T
UCCB1223	RAPID APPLICATION DEVELOPMENT	2L+2P	Dr Shakiroh Binti Khamis	L+P
			Ts Dr Wong Pei Voon	P
UCCB2333	SOCIAL MEDIA STRATEGIES FOR BUSINESS	3L+1T	Ts Ahmad Subhi Bin Zolkafly	L+T
UCCD1213	FUNDAMENTALS OF DIGITAL MEDIA TECHNOLOGY	3L+1T	Ts Dr Khor Siak Wang	L+T
			Ts Dr Lim Ean Heng	T
			Ms Kwang Wai Ching	T
UCCD2043	INFORMATION TECHNOLOGY PROJECT MANAGEMENT	3L+1T	Ts Yong Tien Fui	L+T
			Dr Zurida Binti Ishak	T
UCCD3013	ECOMMERCE PRACTICES	3L+1T	Mr Tan Min Khen	L+T
UCCM1153	INTRODUCTION TO CALCULUS AND APPLICATIONS	3L+1T	Dr Nur Balqishanis Binti Zainal Abidin	L+T
			Dr Lem Kong Hoong	L+T

UCIA-07 (Choose One)

UCCD3113	DISTRIBUTED COMPUTER SYSTEMS	3L+1T	Ts Dr Cheng Wai Khuen	L+T
			En Ahmad Zaffry Hadi Bin Mohd Juffry	T
UCCD3243	SERVER-SIDE WEB APPLICATIONS DEVELOPMENT	2L+2P	Cik Nur Athirah Nabila Binti Mohd Idros	L+P
			Ts Dr Mailasan a/l Jayakrishnan	P
UCCD3343	BIG DATA ANALYTICS	2L+2P	Dr Ramesh Kumar Ayyasamy	L+P

Figure 3.2.3 Timetable of Students in Information Systems Engineering Programme (Year 3 Sem 1)

Subsequently, additional academic information such as program structures for the students between different trimesters were carefully examined. This involved verifying course registration information provided by UTAR, such as the hours of

lectures, tutorials, and practices allocated to specific courses. Besides, information related to class sizes and the number of class sessions scheduled for specific courses were also taken into consideration. These facts are important in understanding how courses are structured, useful to generate a feasible and balanced timetable schedule. By studying these details, the system can ensure that course assignments are aligned with institutional requirements and resource constraints.

The following data in Figure 3.2.4, Figure 3.2.5, Figure 3.2.6 and Figure 3.2.7 is generated based on Figure 3.2.1, Figure 3.2.2, and Figure 3.2.3 above:

groupID	groupName	programName	studyName	studentNum	courseCode
25000	CSY1S1	Computer Science	1	200	UBMM1011, UCCD1004, UCCD1143, UCCD2003, UCCM1153, UCCN1004
24000	CNY2S1	Communications & Networking	2	200	UCCD2043, UCCD2044, UCCD2103,UCCN2243
23000	IAY3S1	Information Systems Engineering	3	200	UCCD3053, UCCD3223

Figure 3.2.4 Mock Data of Student Entity

lecID	lecName	courseCode
10001	Dr Tee Chee Wee	UBMM1011
10002	Dr Tan Joi San	UCCD1004
10003	Ts Dr Phan Koo Yuen	UCCD1004
10004	Dr Jasmina Khaw Yen Min	UCCD1004
10005	Dr Kh'ng Xin Yi	UCCD1004
10006	Prof Dr Leung Kar Hang	UCCD1004
10007	Dr Ahmad Hakimi Bin Ahman Sa'ahiry	UCCD1004
10008	Ts Dr Lim Seng Poh	UCCD1143
10009	Dr Chai Tong Yuen	UCCD1143
10010	Ts Dr Ku Chin Soon	UCCD2003
10011	Ts Dr Mogana a/p Vadiveloo	UCCD2003
10012	Dr Tahanya Bashar M. A.	UCCD2003
10013	Cik Puteri Nursyawati Binti Azzuri	UCCD2003
10014	Dr Nur Balqishanis Binti Zainal Abidin	UCCM1153
10015	Dr Lem Kong Hoong	UCCM1153
10016	Dr Aun Yichiet	UCCN1004
10017	Ms Tan Lyk Yin	UCCN1004
10018	Ts Dr Ooi Chek Yee	UCCN1004
10019	Dr Muhammad Syaiful Amri Bin Suhaimi	UCCN1004
10020	Ts Dr Chang Jing Jing	UCCN1004
10021	Dr Fityanul Akhyar	UCCN1004
10022	Dr Abdulrahman Aminu Ghali	UCCN1004, UCCN2243
10023	Dr Rohani binti Bakar	UCCN1004
10024	Dr Norliana Binti Muslim	UCCN1004
10025	Dr Rahman Sadli	UCCN1004
10026	Dr Teoh Shen Khang	UCCN1004
10027	Puan Nor 'Afifah Binti Sabri	UCCN1004
10028	Dr Farina Saffa Binti Mohamad Samsamnun	UCCN1004, UCCN2243
10029	Ts Yong Tien Fui	UCCD2043
10030	Dr Zurida Binti Ishak	UCCD2043
10031	Dr Sayed Admad Zikri Bin Sayed Aluwee	UCCD2044
10032	Ts Dr Chai Meei Tyng	UCCD2044
10033	Mr Luke Lee Chee Chien	UCCD2044
10034	Ms Tseu Kwan Lee	UCCD2044
10035	Dr Ng Hui Fuang	UCCD2044
10036	Ts Wong Chee Siang	UCCD2103
10037	Mr Sor Kean Vee	UCCD2103
10038	Encik Ahmad Zaffry Hadi Bin Mohd Juffry	UCCD2103
10039	Ts Dr Gan Ming Lee	UCCN2243
10040	Dr Adeb Alid Mohammed Ahmed Al-Samet	UCCN2243
10041	Ms Oh Zi Xin	UCCN2243
10042	Dr Nadeem Muhammad Waqas	UCCN2243
10043	Mr Lee Kim Hoe @ Farhan Lee Bin Abdullah	UCCD3053
10044	Mr Tan Chiang Kang	UCCD3223
10045	Mr Tou Jing Yi	UCCD3223
10046	Puan Syazwani Binti Yahya	UCCD3223

Figure 3.2.5 Mock Data of Lecturer Entity

courseCode	courseName	lecNum	tutorialNum	pracNum	creditHour
UCCD1004	PROGRAMMING CONCEPTS AND PRACTICES	2	0	7	4
UCCD1143	PROBABILITY AND STATISTICS FOR COMPUTING	2	5	0	3
UCCD2003	OBJECT-ORIENTED SYSTEMS ANALYSIS AND DESIGN	2	7	0	3
UCCM1153	INTRODUCTION TO CALCULUS AND APPLICATIONS	2	7	0	4
UCCN1004	DATA COMMUNICATIONS AND NETWORKING	2	8	0	3
UBMM1011	SUN ZI'S ART OF WAR AND BUSINESS STRATEGIES	1	0	0	1
UCCD2043	INFORMATION TECHNOLOGY PROJECT MANAGEMENT	2	6	0	4
UCCD2044	OBJECT-ORIENTED PROGRAMMING PRACTICES	2	0	10	4
UCCD2103	OPERATING SYSTEMS	2	5	0	4
UCCN2243	INTERNETWORKING PRINCIPLE AND PRACTICES	2	0	13	4
UCCD3053	INFORMATION TECHNOLOGY PROFESSIONAL ETHICS	2	4	0	3
UCCD3223	MOBILE APPPLICATIONS DEVELOPMENT	1	0	6	3

Figure 3.2.6 Mock Data of Course Entity

venueID	venueName	capacity	buildingName	venueType
100011	LDK1	200	Block L	L
100012	LDK2	200	Block L	L
100013	LDK3	300	Block L	L
100014	LDK4	200	Block L	L
100015	LDK5	200	Block L	L
100016	EDK1	300	Block E	L
100017	DDK1	300	Block D	L
100018	N001	30	Block N	T
100019	N002	30	Block N	T
100020	N003	30	Block N	T
100021	N004	30	Block N	T
100022	N005	60	Block N	T
100023	N006	60	Block N	T
100024	N007	25	Block N	T
100025	N008	30	Block N	P
100026	N009	30	Block N	P
100027	N010A	20	Block N	P
100028	N010B	20	Block N	P
100029	N101	25	Block N	T
100030	N102	25	Block N	T
100031	N103	25	Block N	T
100032	N104	25	Block N	T
100033	N105	25	Block N	T
100034	N106	25	Block N	T
100035	N107	20	Block N	T
100036	N108	30	Block N	P
100037	N109	30	Block N	P
100038	N110A	20	Block N	P
100039	N110B	20	Block N	P

Figure 3.2.7 Mock Data of Venue Entity

For this project, elective courses are partially considered. Core courses are mainly used to generate the mock dataset. The mock dataset may not contain information that is 100% accurate or true to real life.

3.3 Verification Plan

After the implementation phase, the testing phase will be conducted as system testing is carried out to evaluate the performance and accuracy of the system in generating timetables based on the users' input data. To confirm the correctness of the system's functions, verification testing is performed to validate that all features and functionalities are working as intended.

3.3.1 Test Plan for Hard Constraints

The hard constraints' test cases are listed as shown in Table 3.3.1.1 below, in order to validate the system's performance and ensure that the system generates the timetables as intended.

Table 3.3.1.1 Hard Constraints Test Plan

Test Case No.	Description	Test Data	Expected Result
1.	A lecturer can only teach one class at the same time.	Lecturers' timetables	In the lecturers' timetable generated, the lecturer is allocated to only one class in one timeslot.
2.	Two courses cannot be scheduled in the same venue at the same time.	Classrooms' timetables	In the classrooms' timetable generated, each timeslot should not have two courses in the same venue.
3.	A student can only attend one class at the same time.	Students'	In the students' timetable generated, the student is

		timetables	allocated to only one class in one timeslot.
4.	The number of students cannot exceed the seating capacity of the assigned venue.	Classrooms' timetables	In the classrooms' timetable generated, the class size that is larger than the classroom capacity cannot be allocated to that respective class.
5.	Courses that require specific room types should be scheduled in appropriate facilities.	Classrooms' timetables	In the classrooms' timetable generated, lecture classes should be located inside the class type of lecture. Practical classes should be located inside the class type of practical.
6.	Venues cannot be assigned to the same timeslot more than once.	Classrooms' timetables	In the classrooms' timetable generated, it cannot have the same venue allocated to the same timeslots more than one time.
7.	Total maximum hours for classes per day is 8 hours.	Classrooms' timetables	In the classrooms' timetable generated, classes can only be allocated within that 8-hour per day frame.
8.	A student cannot be assigned to two different venues in the same timeslot.	Students' timetables	In the students' timetable generated, a student can only be assigned to one classroom at a time.
9.	The travel distance between two venues of consecutive	Classrooms' timetables	In the classrooms' timetable generated, the travel distance

	classes should not be more than 500 meters.		between two consecutive classes should be lesser than 500 meters.

3.3.2 Test Plan for Soft Constraints

The test cases for the soft constraints are as shown in Table 3.3.2.1 below. This test plan is created so that all the soft constraints listed must be fulfilled in a way similar to the hard constraints.

Table 3.3.2.1 Soft Constraints Test Plan

Test Case No.	Description	Test Data	Expected Result
1.	Gaps between classes should be minimized, such as long breaks between consecutive classes.	Classrooms' timetables	In the classrooms' timetable generated, each timeslot should not have a break in between that is too long.
2.	No classes should be scheduled during lunch break (12 p.m. – 2 p.m.).	Classrooms' timetables	In the classrooms' timetable generated, there should be no classes from 12 p.m. to 2 p.m.
3.	There should be no classes scheduled before 8 a.m. and after 8.30 p.m.	Classrooms' timetables	In the classrooms' timetable generated, there should be no classes before 8 a.m. and after 8.30 p.m.
4.	The maximum hours for consecutive classes should be 4 hours.	Classrooms' timetables	In the classrooms' timetable generated, the total hours of consecutive hours cannot exceed 4 hours.

3.3.3 Test Plan for the Proximity and Travel Minimization Constraint

The test cases for the proximity and travel minimization constraint are as given in Table 3.3.3.1 below. The test plan here is that it ensures all the travel distance constraints to be satisfied in a way as close to hard constraints as possible, in order to maintain the practicability and efficiency of the timetable constructed.

Table 3.3.3.1 Proximity and Travel Minimization Constraint Test Plan

Test Case No.	Description	Test Data	Expected Result
1.	Two consecutive classes for the same lecturer should be in nearby venues.	Lecturers' timetables	In the lecturers' timetable generated, classes are scheduled in venues within 500 meters, satisfying the travel distance constraint.
2.	Two consecutive classes for the same student group must be within walkable distance.	Students' timetables	In the students' timetable generated, the distance between venues is under 500 meters, fulfilling the constraint.
3.	Same-building consecutive classes should be allowed without any restriction.	Classrooms' timetables	No travel distance issue since both classes are in the same building. Schedule is accepted.

3.3.4 Test Plan for Timetable Schedule Generation

The purpose of this test plan is to evaluate the effectiveness and efficiency of using GA to generate a timetable quickly. In this experiment, the design involves varying the number of class sessions to simulate timetabling under varying levels of complexity. Additionally, the number of user-selectable constraints is a manipulated variable which is used to analyse the impact that the constraints will have on the

generation time. These constraints may include lecturer preferences, maximum consecutive class hours, preferred teaching time slots, and travel distance limits.

By adjusting both the class sessions numbers and the selected constraints, this test aims to observe how the GA scales in terms of computational efficiency and scalability. The resources herein refer to the overall classroom timeslots, which are achieved by multiplying the number of time slots per week by the number of rooms. The result of this test will help in understanding the effect of input size and complexity of the constraints on the efficiency of the algorithm, and in optimizing the balance between quality and speed of the generated timetable. Table 3.3.4.1 below listed down the test plan for the timetable generation.

Table 3.3.4.1 Timetable Generation Test Plan

Test Case No.	Description
1.	Generate a timetable with a small number of classes (approximately 20% of the resources) and 2 selected constraints.
2.	Generate a timetable with a large number of classes (approximately 60% of the resources) and 2 selected constraints.
3.	Generate a timetable with a small number of classes (approximately 20% of the resources) and all selectable constraints.
4.	Generate a timetable with a large number of classes (approximately 60% of the resources) and all selectable constraints.
5.	Generate a timetable where the class events exceed the available resources, to evaluate how the algorithm handles over-constrained situations.

3.4 Main System Functionalities of Timetable Scheduling System

3.4.1 Steps and Actions Performed by Users in the Timetable System

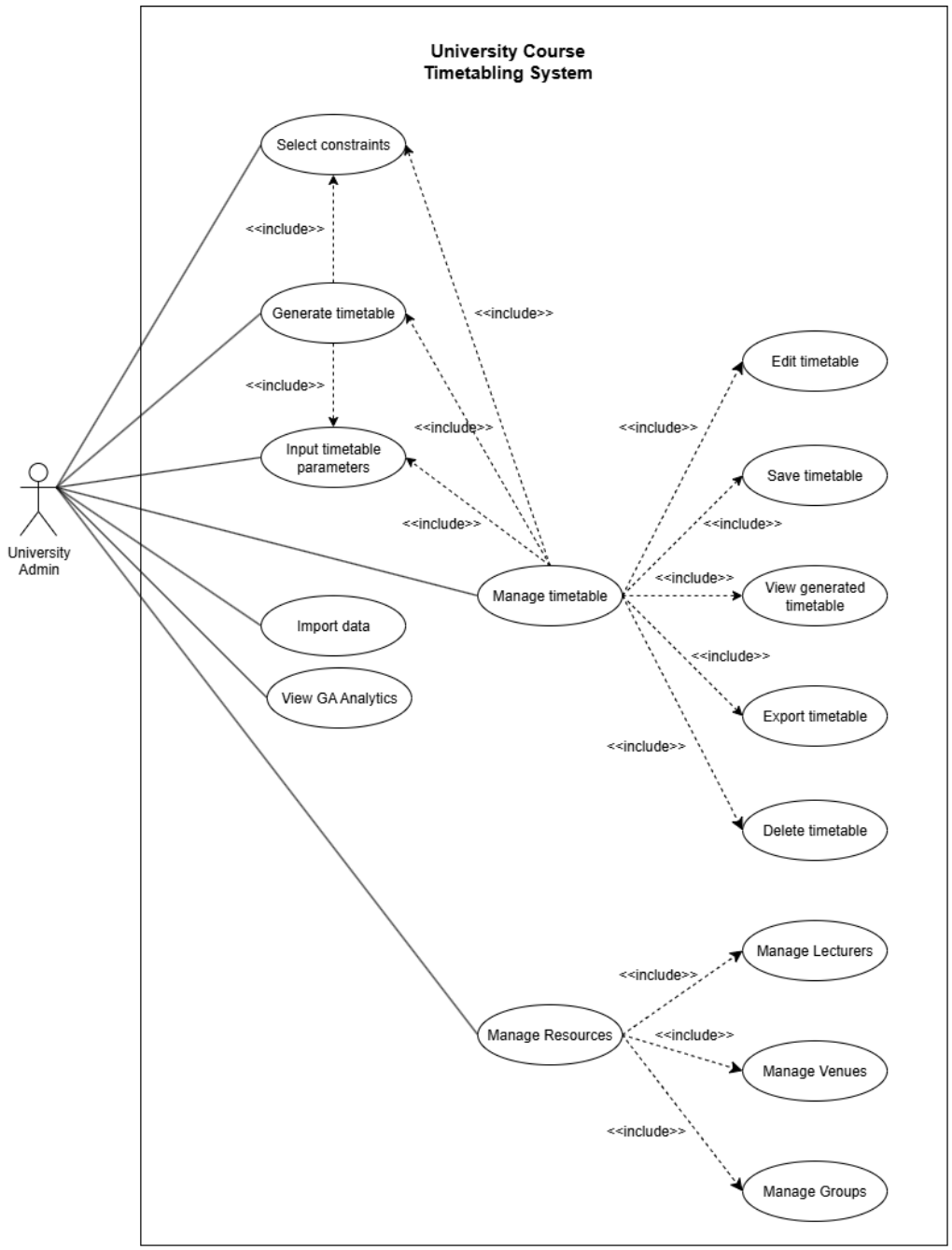

Figure 3.4.1.1 System Functionalities for the University Course Timetabling System

Figure 3.4.1.1 shows the interactions between the University Admin and the various functionalities of the system. Before the timetable is generated, the administrator can import the required data, and choose the relevant constraints. After it has been created, the administrator can manage the timetable by saving, viewing, exporting or deleting it.

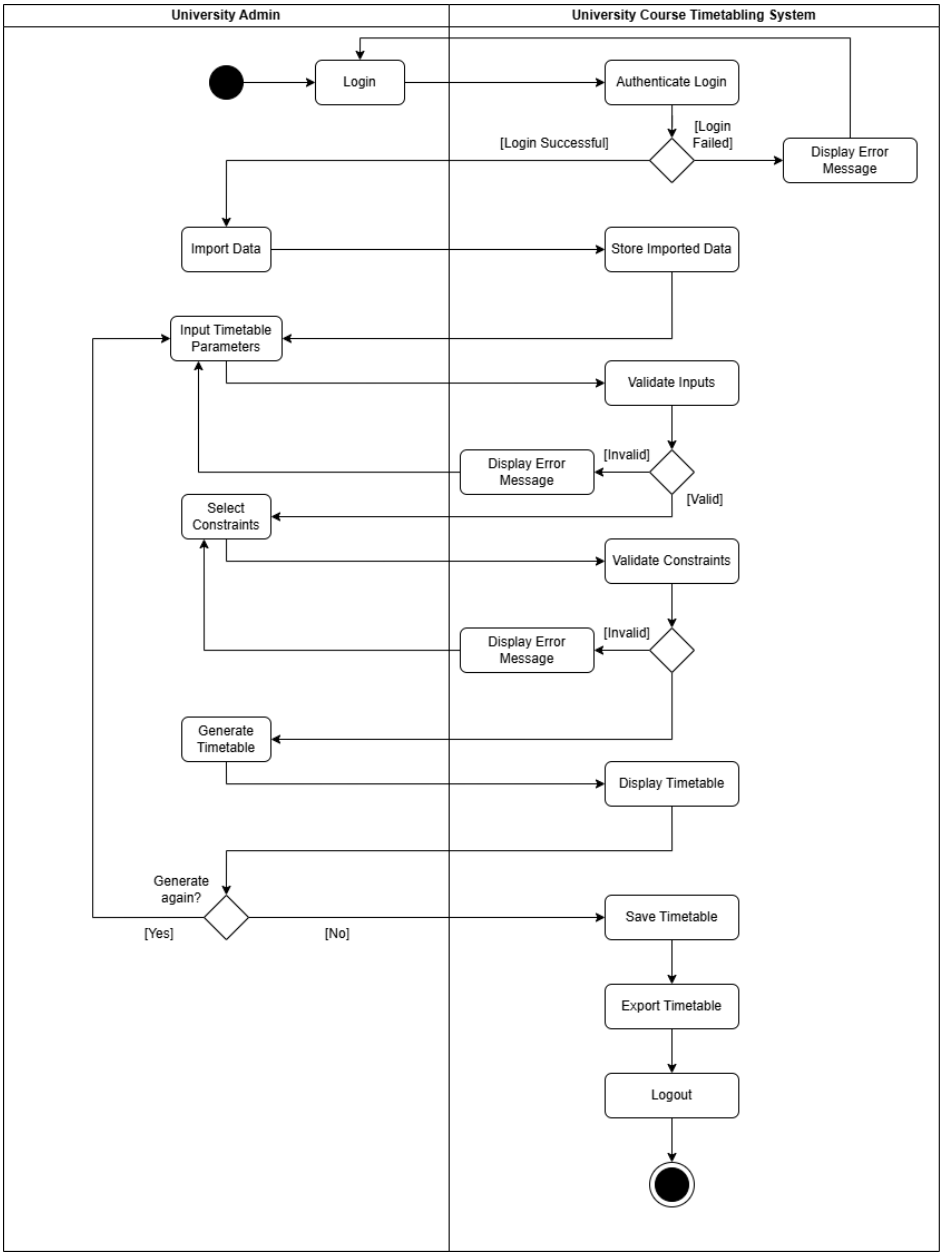

Figure 3.4.1.2 Process of University Course Timetabling System Creating a Timetable

Figure 3.4.1.2 above shows the process flow of creating a timetable. The first step is the login procedure, during which the system will authenticate user information and display an error message if necessary. The administrator can then import data, and the system will store it. Next, constraints and timetable parameters are selected, and the system validates the constraints and timetable parameters. Error messages are shown if the input is invalid, forcing the administrator to enter it again. The timetable generated is displayed after validation is successfully. Lastly, the administrator has the choice to

regenerate the timetable if needed. If the timetable generated is acceptable, the timetable may be exported, and the administrator may logout of the system.

3.4.2 Constraints Used in the University Course Timetabling System

3.4.2.1 Hard Constraints

Based on the literature review, most of the existing research on university course timetabling usually have a standard set of well-defined hard constraints. Avoiding class clashes for lecturers and students, room capacity and preventing overlapping use of venues are usually listed as hard constraints. These are the standard constraints that will be used in this project as part of the base model. Table 3.4.2.1.1 below has listed some of the hard constraints adopted from existing related work.

However, there is one constraint that is not highly common in the present literature, although it is highly relevant in real university settings, particularly in large campuses. It is the 'Proximity and Travel Minimization Constraint' with the purpose of reducing the physical distance between consecutive classes for lecturers and students, which helps reduce unnecessary movements between classes. Including this constraint is especially important in large university campuses, where distant venues can affect punctuality and academic efficiency.

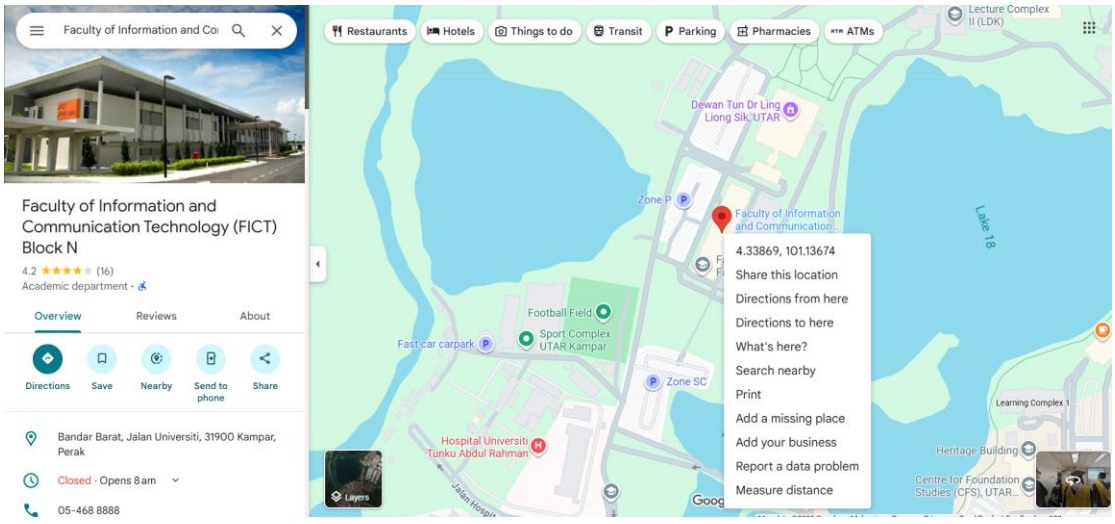

Figure 3.4.2.1.1 Collection of Coordinates via Google Maps

Venue coordinates were captured using Google Maps as the authoritative source, recorded in decimal degrees (WGS-84). For each teaching block, the operator located the building on the UTAR Kampar campus map, zoomed in, and placed a point at the primary pedestrian entrance. Coordinates were obtained via the Google Maps context menu (Right-click) which displays latitude and longitude at the top of the panel, as illustrated in Figure 3.4.2.1.1. Values were copied exactly in decimal format and recorded to six decimal places, providing sufficient precision for intra-campus distance checks.

To integrate these data, the coordinates were entered into the system's venues table as latitude and longitude fields in the database. During timetable generation and validation, the system calculates great-circle (straight-line) distances between the coordinates of venues in consecutive sessions using the Haversine formula. These distances are then compared against the project's Proximity and Travel Minimization constraint, which requires consecutive venues to be no more than 500 meters apart for any individual student or lecturer.

$$d = rhav^{-1}(h) = 2rsin^{-1}(\sqrt{h})$$

or

$$d = 2rsin^{-1}\left(\sqrt{sin^2\left(\frac{\Phi_2-\Phi_1}{2}\right) + cos(\Phi_1)cos(\Phi_2)sin^2\left(\frac{\lambda_2-\lambda_1}{2}\right)}\right)$$

Figure 3.4.2.1.2 Harvensine Formula [48]

For each day and for each participant, the system inspects consecutive sessions. It computes the geodesic distance between the two venues using the Haversine formula (Figure 3.4.2.1.2) based on the latitude/longitude stored for each venue. If the computed distance exceeds 500 meters, a penalty is applied to the chromosome's fitness, making schedules with excessive walking less likely to survive selection. The threshold is configurable but was fixed at 500 m for this evaluation.

Additionally, the course constraint whereby a fixed set of courses assigned to each student group should be scheduled without conflicts is added to the system. This is to ensure that the courses taken by a specific student group will not clash with each other. For example, class of course A clashing timeslots or venue with class of course B. While some may share the same timeslots, the system can assign them different venues or other available timeslots.

Table 3.4.2.1.3 Hard Constraints Used

Constraint No.	Hard Constraints
1.	A lecturer can only teach one class at the same time.
2.	Two courses cannot be scheduled in the same venue at the same time.
3.	A student can only attend one class at the same time.
4.	The number of students cannot exceed the seating capacity of the assigned venue.
5.	Courses that require specific room types should be scheduled in appropriate facilities.
6.	Total maximum hours for classes per day is 8 hours.
7.	A fixed set of courses assigned to each student group should be scheduled without conflicts.
8.	The travel distance between two venues of consecutive classes should not be more than 500 meters.

3.4.2.2 Soft Constraints

Most of the existing literature on university course timetabling from the literature review also includes some of the most common soft constraints to provide quality and improve usability for the timetable generated. These soft constraints are not necessary but fulfilling them improves the overall experience for the students and the lecturers. A few of the standard soft constraints include accommodating the lecturers' preferences, minimizing the gaps between classes and not assigning unwanted class times such as lunchtimes, early mornings and late evenings. These soft constraints are listed in Table 3.4.2.2.1 below and will be adopted in this project.

Table 3.4.2.2.1 Soft Constraints Used

Constraint No.	Soft Constraints
1.	Gaps between classes should be minimized, such as long breaks between consecutive classes.
2.	No classes should be scheduled during lunch break (12 p.m. – 2 p.m.).
3.	There should be no classes scheduled before 8 a.m. and after 6 p.m.
4.	The maximum hours for consecutive classes should be 4 hours.

3.4.3 Design of Proposed Genetic Algorithm for the University Course Timetabling System

The step of the genetic algorithm involves initialization, fitness evaluation, selection, crossover, and mutation. Hence, the process and flow to generate an optimized timetable will be explained and illustrated using Figure 3.4.3.1 below.

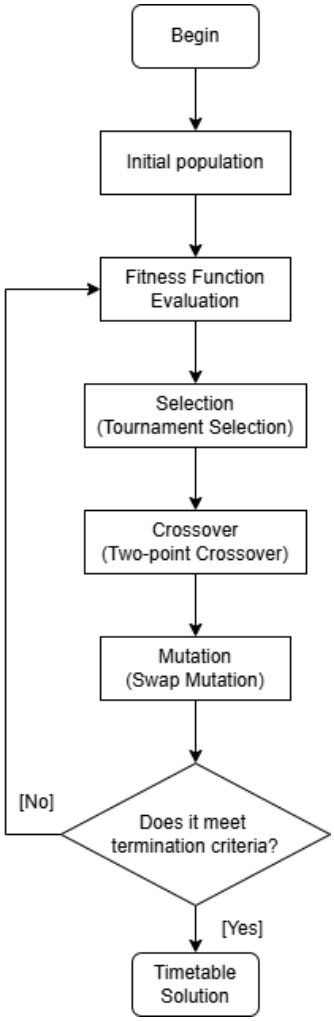

Figure 3.4.3.1 Flow of Proposed Genetic Algorithm for University Course Timetabling System

The initialization phase of the scheduling process starts with the random generation of an initial population of schedules, or chromosomes. These initial timetables are likely to be infeasible and might violate multiple constraints. Each chromosome represents a possible solution which is a full timetable made up of scheduled class sessions.

After initialization, every chromosome is evaluated using a fitness function. This function measures how well a timetable satisfies all the hard and soft constraints, such as lecturer availability, student conflicts, classroom capacity, room types, and travel distance between classes. The objective is to reduce violations of constraints and increase overall schedule efficiency.

After that, the algorithm moves on to the selection phase. To select the best chromosomes to become the parents of the following generation, Tournament Selection is used in this project. In this scenario, the fittest chromosome out of the group is selected as a parent for each tournament, and the process is carried out by randomly picking a sample of chromosomes from the population. This approach preserves genetic diversity while promoting the selection of stronger individuals.

Next, Two-Point Crossover is used to generate new offspring after the parent chromosomes have been chosen. In this process, two chromosomal crossover locations are chosen at random. Two child chromosomes are produced when the genes (timetable components) between these sites are switched between the two parent chromosomes. Compared to Single-Point Crossover, this approach enables more varied gene exchange, expanding the solution space's exploration and potentially improving solution quality.

After crossover, the new offspring undergo mutation in order to introduce diversity into the population and prevent premature convergence. Swap Mutation technique is applied, where two randomly chosen class sessions are switched within the chromosome. This improves the algorithm's capacity to identify optimal or nearly ideal solutions and aids in the exploration of new areas of the solution space.

The algorithm then checks whether the solution meets the termination criteria - reaching the maximum number of generations or finally having a solution with an acceptable fitness level. If it does not satisfy the termination criteria, the offspring of the new generation becomes the current population, and it goes back to the fitness function evaluation so that the process initiates the next generation. When the termination conditions are reached, then the algorithm stops and the chromosome with the best fitness found so far is presented as the timetable solution.

Class sessions are created when users enter data into the timetable scheduling tool, such as student information, lecturer availability, classroom details, and course requirements. Chromosomes then encode these details. This input is processed by the genetic algorithm through a series of evolutionary cycles until optimized and feasible timetable is generated.

Constraints which include room capacity, lecturer and student availability, session overlap avoidance, travel distance between classes, and preferred room types are continuously evaluated during the process. This is to ensure quality and feasibility of the solutions is maintained. By including these limitations, the system can generate useful schedules that satisfy user requirements and university regulations.

3.4.3.1 Initialization

The initialization starts with the generation of an initial population of chromosomes, each of which is a complete candidate timetable. This process starts with the conversion of the raw session data, such as courses, lecturers, classrooms, and student groups, into a list of schedulable sessions. Each session represents a class session which are then used to populate the available timeslots in different classrooms. The arrangement of these sessions across all available rooms and timeslots forms the chromosomes (timetables), which forms the initial population.

To generate a timetable, the system considers the following resources:

- **Days per week:** The number of academic days available for classes (For example, Monday to Friday, avoiding weekend classes)
- **Slots per day:** The number of available class periods within a single day (The total hours available per day for the timetable is 10 hours)
- **Available classrooms and their capacities:** The physical rooms where sessions will be held. Each classroom has a specific type (lecture, tutorial, practical), capacity, and location.

Each classroom has a consistent number of available slots per week, which is calculated as:

Slots per day x Days per week

All available slots across all classrooms are represented as genes in a chromosome.

Each course may consist of multiple session types. Each event contains the following information:

- Event type: L(Lecture), T(Tutorial), P(Practical)
- Number of students

- Lecturer assigned
- Course code
- Student group
- Session ID (Unique identifier)
- Group ID (To group related events, such as multiple tutorials for the same course)

In addition to session information, each classroom in the system is defined with its own properties, including:

- Venue ID
- Room type
- Capacity
- Location

Before assigning a session to a classroom-timeslot pair, the system checks:

- Whether the classroom type matches the session type.
- Whether the room capacity can accommodate the session's number of students.

Table 3.4.3.1.1 Example of Session Representation in a Timetable

Session	Session Type	No. of Students	Lecturer	Course Code	Student Group	Session ID	Group ID
1	L	100	Dr. Lim	UCCD1143	CSY1S1	1	1
2	P	20	Dr. Tan	UCCD1004	CSY1S2	2	7
3	P	20	Dr. Tan	UCCD1004	CSY1S2	3	7
4	T	25	Dr. Ku	UCCD2003	CSY1S3	4	8
5	T	30	Dr. Mogana	UCCD2003	CSY1S2	5	2
6	L	100	Dr. Ku	UCCD2003	CSY1S3	6	5

Based on Table 3.4.3.1.1 above, the session is split into different sessions with a shared Group ID in the case that the number of students is greater than the classroom's capacity. In this process, every group of students receives enough hours for every

subject and is subjected to classroom capacity regulations. Group ID allows sessions to be categorized such that associated sessions can be determined and handled as a single session at scheduling and optimization.

Chromosome Structure and Encoding:

In this project, the non-binary representation is used to construct the chromosome of the GA. Each chromosome in the population is a one-dimensional array that encodes the full timetable. This direct encoding using a non-binary integer is a direct encoding technique where the structure of the chromosome is designed such that each segment corresponds to the schedule of a particular classroom across the week. If the slot is unused, the gene is set to 0. The system maintains a consistent order of classrooms so that gene positions are easily mapped to actual rooms and time slots (classroom-timeslot pair).

Each session is assigned to a specific classroom timeslot during chromosome generation. The total number of available genes in a chromosome is calculated as:

Number of classrooms x Slots per day x Days per week

The total number of available slots for sessions that can be scheduled throughout the schedule is defined by this value. Each gene in the chromosome represents one timeslot in one classroom. Each gene may be populated with a specific Session ID, signifying that a session is planned for that time slot, or it may be empty (value 0).

For example, there are 5 days per week and 5 slots per day for lectures (assuming 2 hours per slot), and lastly 3 lecture classrooms. Each classroom has **5 x 5 = 25** timeslots. Hence, total chromosome length is: **3 x 5 x 5 = 75 genes.** Table 3.4.3.1.1 illustrates it.

Table 3.4.3.1.2 Example of Chromosome Length

Gene Index	Classroom	Day	Slot	Value (Session ID)
0	LDK4	Mon	1	0
1	LDK4	Mon	2	1
2	LDK4	Mon	3	0

…	…	…	…	…
25	LDK5	Mon	1	5

Table 3.4.3.1.3 Example of Chromosome Structure

Chromosome A	0	1	0	3	0	2	4	5
Chromosome B	2	1	0	0	0	0	8	15
Chromosome C	17	0	4	15	5	0	3	0
Chromosome D	25	0	0	6	0	0	0	0

Here in Table 3.4.3.1.2, each row is a chromosome (complete timetable), and each cell represents a gene (timeslot). As mentioned above, the Session ID is filled in if a class is scheduled in that slot and 0 means the slot is empty. Constraints are handled by condition checking at initialization and are also maintained by the fitness function throughout evolution. The initialization process continues until the number of chromosomes reaches the defined population size (for example, 100 in this system). The genetic algorithm will later perform operations such as crossover, mutation, and fitness evaluation based on these chromosomes.

3.4.3.2 Fitness Evaluation

After the initial population of chromosomes is generated, the next step is to determine the fitness value of each chromosome, which indicates how good or possible the generated schedule is. The greater the fitness value, the better the schedule and the closer it is to satisfying the given constraints, while the lower the fitness value, the higher the number of violations. In this system, the fitness score is calculated by the number of hard constraints violated. For each violation, the fitness score goes down by 1. The ideal fitness score is 0, meaning that there are no violations in the timetable. Common constraint violations are placing a lecturer into more than one class at a time,

putting a student group into more than one class at the same time, or placing two sessions to the same room at the same timeslot. These issues decrease the fitness score and indicate that the schedule is infeasible.

3.4.3.3 Selection

In this project, Tournament Selection is applied to select chromosomes for the next process of crossover. In Tournament Selection, a small group of chromosomes is selected randomly from the population and their fitness values are compared. Among the selected group, the chromosome with the best fitness value wins and will proceed to the crossover process. This is repeated until the required number of parent chromosomes is selected.

For example, let the population consist of the following chromosomes and their respective fitness values:

Table 3.4.3.3.1 Example of Chromosome with their Fitness Values

Chromosome	Fitness Value
A	-5
B	-1
C	-3
D	-2

Assuming the size of the tournament is 2, the algorithm can select chromosome C and D randomly in the first tournament. Out of the two, chromosome D is selected as it has a higher fitness value (-2 > -3). In another tournament, chromosomes A and B can be selected where B is selected because it has a higher fitness (-1 > -5). This cycle is repeated again and again until some number of parents to be selected for the crossover operation is chosen. Compared to other methods like Roulette Wheel Selection, Tournament Selection offers greater control over selection pressure and ensures that the fittest chromosomes get a higher probability of survival while being fair enough to give lower-performing ones an opportunity to contribute towards maintaining genetic diversity.

3.4.3.4 Crossover

Following the selection phase, Two-Point Crossover technique is used to generate new offspring from selected parent chromosomes. Two-Point Crossover chooses two points of crossing places on the chromosome, while Single-Point Crossover splits and exchanges at a single point. The section in between these two points is exchanged between a pair of parents, and the generated offspring inherit traits from parents, resulting in offspring that inherit traits from both parents more diversely. This method brings in diversity and enables more exploration of the solution space. Figure 3.4.3.1 illustrates the Two-Point Crossover model, whereby two lines represents the two crossover points in the chromosomes.

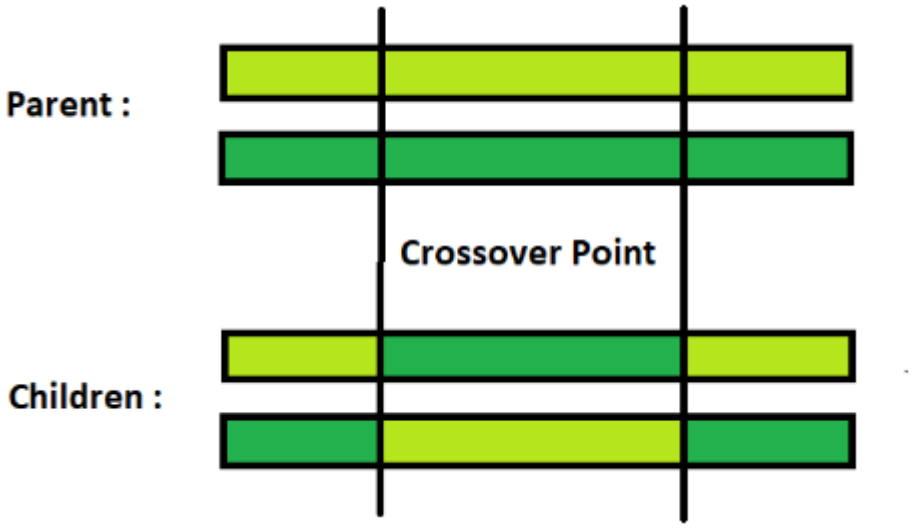

Figure 3.4.3.4.1 Two-Point Crossover Model [47]

For instance, look at the following two parent chromosomes:

Table 3.4.3.4.1 Example of Two Parent Chromosomes

Chromosome	Gene Sequence							
A	2	0	3	1	4	5	6	7
B	8	9	0	2	1	3	5	4

Suppose the two crossover positions are between 2 and 5. The gene segment from position 2 to 5(inclusive) will be swapped between the two parents.

Table 3.4.3.4.2 Example of Two Children Chromosomes (After Crossover)

Chromosome	Gene Sequence							
A	2	0	0	2	1	3	6	7
B	8	9	3	1	4	5	5	4

Here, the coloured font areas are the crossed-over areas. The crossover couples are picked randomly from the population in every iteration, and offspring produced have evolved for mutation and fitness testing.

3.4.3.5 Mutation

In this phase, the mutation operator used is Swap Mutation, and it swaps at timeslot level in the timetable. Each timeslot in a chromosome has a small chance of being selected for mutation, which is set at 5% in this example. After having picked out one timeslot, the algorithm would randomly pick another timeslot from the same chromosome and then swap their positions. This mutation operation allows the system to introduce new variations to the population, preventing local optima and increasing the diversity of candidate solutions. The mutation rate may be optimized during testing to find the best value.

For example, assume that a timeslot that contains an session of ID 7 is selected for mutation. A random timeslot is next selected, possibly with the session ID 12. The two sessions' positions are then exchanged within the chromosome. This is as illustrated in Figure 3.4.3.5.1 below.

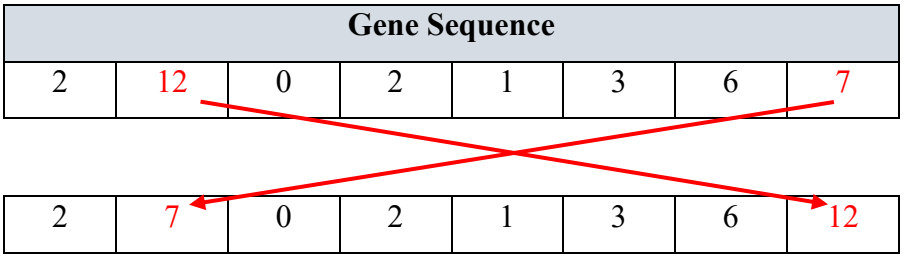

Figure 3.4.3.5.1 Example of Swap Mutation

CHAPTER 4

System Design

4.1 System Architecture Design

Figure 4.1.1 Architecture Design of the System

In this project, the system architecture design is as shown above in Figure 3.6.1, which follows a three-tier architecture model. There are three parts: the presentation tier, the logic tier, and the data tier. The presentation tier represents the user device whereby the standalone web application is developed using Eclipse Java. This tier captures input from the user and sends it to the logic tier. The logic tier is implemented using Python and Flask API. The logic tier receives the input and runs the Genetic Algorithm (GA) for scheduling timetables. The logic tier interacts with the data tier, as it stores and retrieves data from the data tier. The data tier is a MySQL database that is mainly responsible for data storage and retrieval.

Based on Figure 3.6.1, mysql.connector acts as the link between the logic tier and the data tier. It makes it possible for the Python and Flask backend to access the MySQL database in a way that the system can perform various database operations such as inserting, updating, reading, and deleting data. Through this connection, the logic tier can utilize the timetable data generated by the GA and ensure that timetables of classes are being stored in the database correctly.

4.2 Data Storage Design

In this project, the data is stored in a MySQL database, which is a type of Relational Database Management System (RDBMS). The reason MySQL was chosen is because it is a well-known and open-source RDBMS that is simple to integrate with Python applications through the usage of modules like the MySQL Connector. To generate the timetable, important details would be stored in the database. These include venue details, courses, lecturers, and students. XAMPP would also be used to host the MySQL database locally throughout the development period. This is for easy access and management. The data storage structure of data in the MySQL database is shown in the Figure 3.6.1 below.

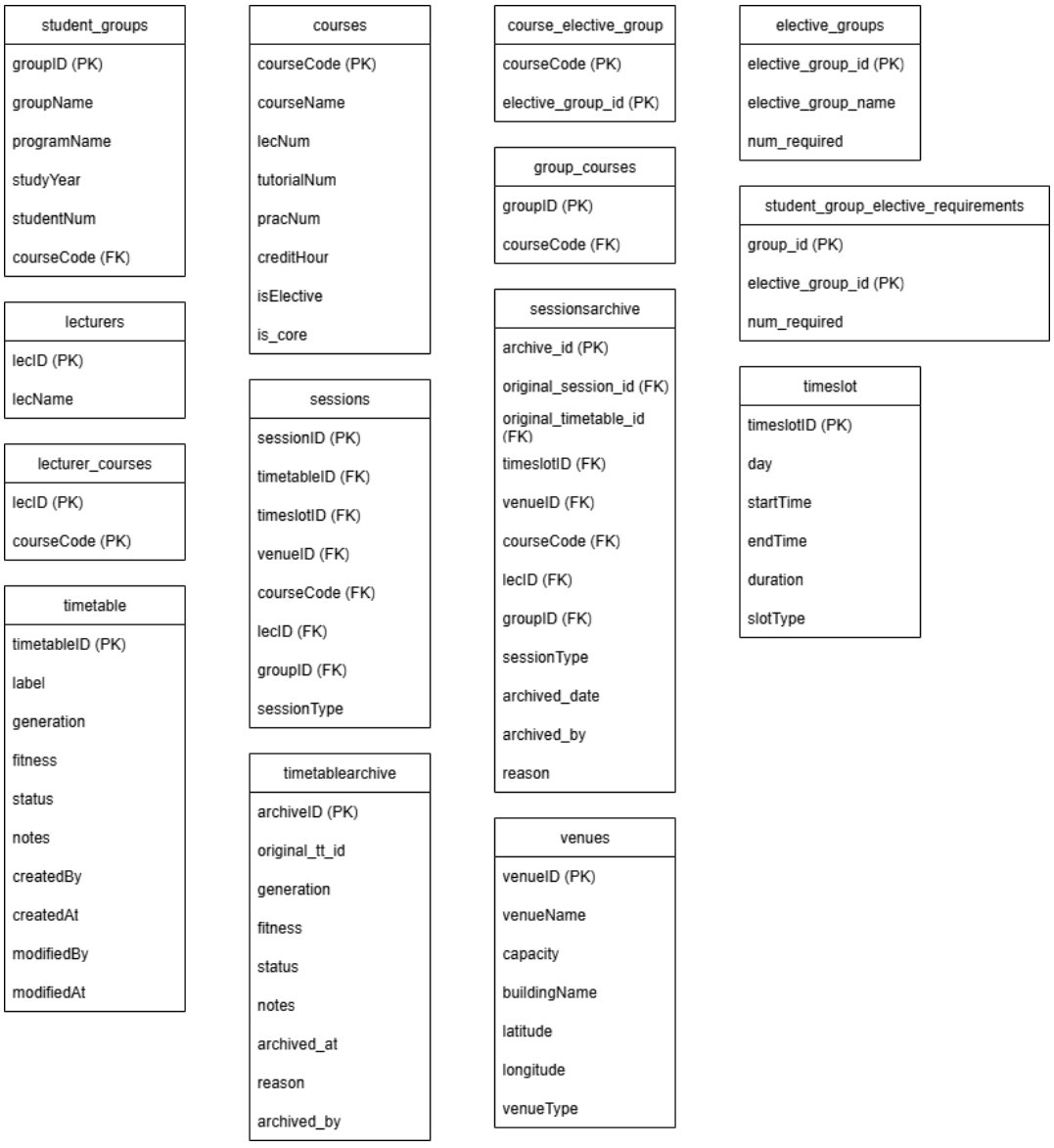

Figure 4.2.1 Database Structure Design of the Timetabling Tool

The MySQL database contains main entities required for timetabling generation of courses, such as classroom information, course information, lecturer allocations, student group registrations, and timetables generated. Each entity is tabled in a relational format and represented with foreign keys (FK) and primary keys (PK) to minimize redundancy and support data integrity.

Instead of being binary objects, JSON-serialized data is placed in LONGTEXT fields for timeline and Genetic Algorithm configuration outcome. This approach improves transparency since the stored data can be viewed easily, edited, or exported without resorting to de-serialization libraries. Compared to other alternatives such as storing in text files or XML files, using a MySQL database is more scalable, data-consistent, and provides easier querying alternatives, which are essential in handling complex timetable generation processes in a university environment.

Table 4.2.1, Table 4.2.2, Table 4.2.3, Table 4.2.4, Table 4.2.5, Table 4.2.6, Table 4.2.7, Table 4.2.8, and Table 4.2.9 below displays the data dictionary of the entities in this system.

Table 4.2.1 Data Structure of Student Group Entity

Field Name	**Data Type**	**Description**
groupID (PK)	INT(6)	Unique identifier for student group
groupName	VARCHAR(255)	Name of the student group
programName	VARCHAR(60)	Name of the program
studyYear	INT(6)	Year of study
studentNum	INT(6)	Number of students in a student group

Table 4.2.2 Data Structure of Lecturer Entity

Field Name	Data Type	Description
lecID (PK)	INT(6)	Unique identifier for each lecturer
lecName	VARCHAR(255)	Name of the lecturer

Table 4.2.3 Data Structure of Course Entity

Field Name	Data Type	Description
courseCode (PK)	VARCHAR(20)	Code of the course
courseName	VARCHAR(255)	Full name of the course
lecNum	INT(2)	Number of lecture classes in the course
tutorialNum	INT(2)	Number of tutorial classes in the course
pracNum	INT(2)	Number of practical classes in the course
creditHour	INT(2)	Number of credit hours for the course

Table 4.2.4 Data Structure of Venue Entity

Field Name	Data Type	Description
venueID (PK)	INT(6)	Unique identifier for the venue

venueName	VARCHAR(50)	Name of the venue
capacity	INT(4)	Maximum number of students the room can accommodate
buildingName	VARCHAR(50)	Building name where the classroom is located
venueType	VARCHAR(50)	Type of room

Table 4.2.5 Data Structure of Timetable Entity

Field Name	**Data Type**	**Description**
timetableID (PK)	INT(6)	Unique identifier for the timetable
timetableData	LONGTEXT	Serialized text storing generated timetable data
gaParameters	LONGTEXT	Serialized text storing GA (Genetic Algorithm) parameters

Table 4.2.6 Data Structure of Student_Course Entity (Join Table)

Field Name	**Data Type**	**Description**
groupID (PK, FK)	INT(6)	Linked to Student Group (groupID)
courseCode (PK, FK)	VARCHAR (20)	Linked to courses (courseCode)

Table 4.2.7 Data Structure of Lecturer_Course Entity (Join Table)

Field Name	Data Type	Description
lecID (PK, FK)	INT(6)	Linked to Lecturer (lecID)
courseCode (PK, FK)	VARCHAR (20)	Linked to courses (courseCode)

Table 4.2.8 Data Structure of Timeslot Entity

Field Name	Data Type	Description
timeslotID (PK)	INT(6)	Linked to Lecturer (lecID)
day	VARCHAR (20)	Linked to courses (courseCode)
startTime	TIME	Start time of the timeslot
endTime	TIME	End time of the timeslot

Table 4.2.9 Data Structure of Session Entity

Field Name	Data Type	Description
sessionID (PK)	INT(6)	Unique identifier for the session
timetableID (FK)	INT(6)	Linked to timetable (timetableID)
timeslotID (FK)	INT(6)	Linked to timeslots (timeslotID)

venueID (FK)	INT(6)	Linked to venues (venueID)
courseCode (FK)	VARCHAR(20)	Linked to courses (courseCode)
groupID (FK)	INT(6)	Linked to student_groups (groupID)
lecID (FK)	INT(6)	Linked to lecturers (lecID)
sessionType	VARCHAR(10)	Type of session ('Lecture', 'Tutorial', 'Practical')

4.3 Graphical User Interface Design

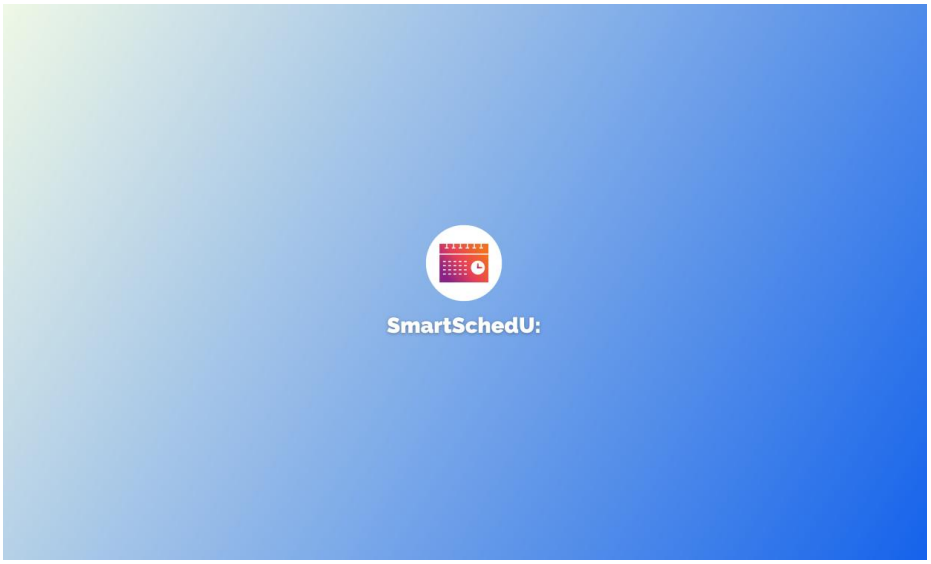

Figure 4.3.1 Start Screen of the Standalone Web Interface

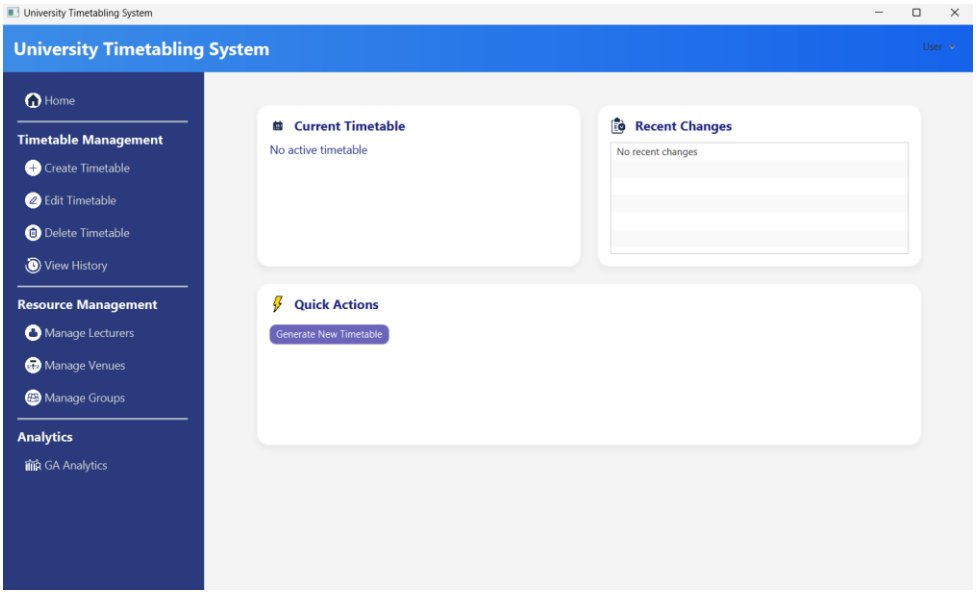

Figure 4.3.2 Home Screen

Figure 4.3.1 shows the start screen being displayed when the application is run. After 1 second, the home screen in Figure 4.3.2 will appear. The home screen consists of the side left bar which enables users or university administrators to manage the timetables and resources such as lecturers, venues, and student groups. This management of resources is directly linked to the database. This means that the CRUD functionalities are being applied to the backend database as the user interacts with the interface here.

Under timetable management, the user may create a timetable, edit the timetables, delete a timetable or view history. Under resource management, the user may manage lecturers, manage venues, or manage groups. Lastly, the GA Analytics is used to display the generation of the timetable, its parameters and the evolution of the solutions generated.

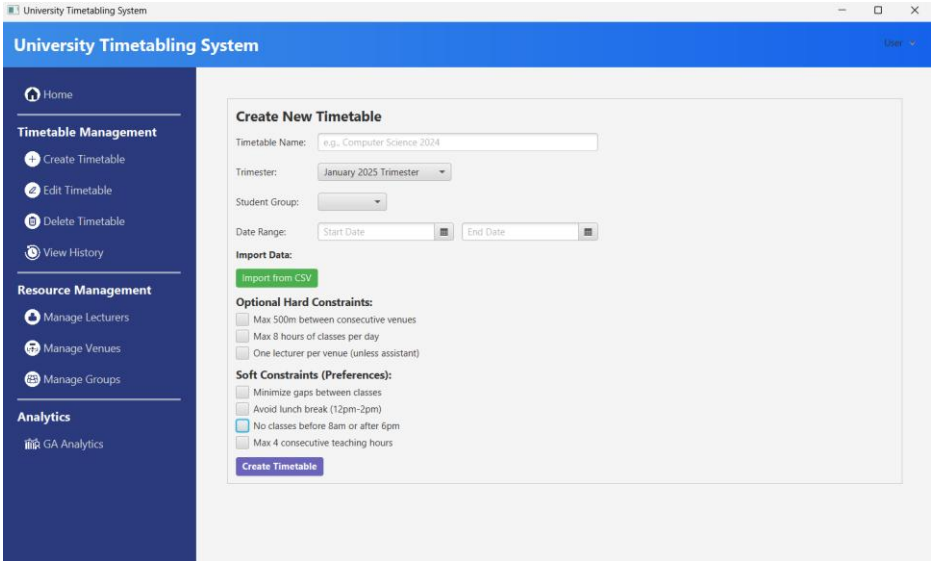

Figure 4.3.3 Create New Timetable Screen

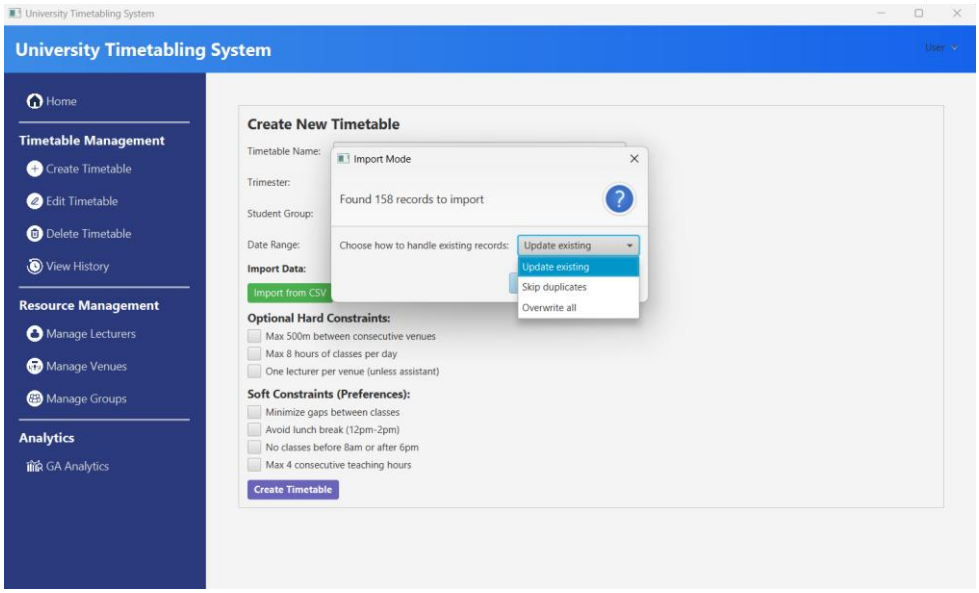

Figure 4.3.4 Import Data Options

In Figure 4.3.3, the user can create a new timetable by filling in the following details. The user can fill in the timetable name field, select which trimester, which student group and the date range of the semester. The import data feature is for users to import their own data in a CSV format. Referring to Figure 4.3.4, there will be three options in the dropdown list when importing data, namely ‘Update Existing’, ‘Skip Duplicates’, and ‘Overwrite All’. This is for data handling purposes for when the database contains existing data.

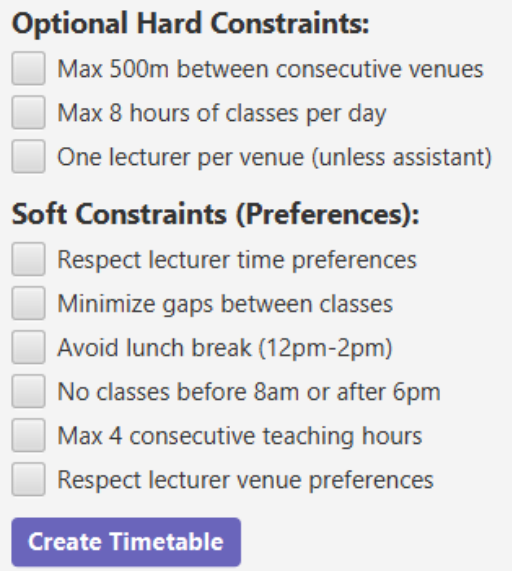

Figure 4.3.5 Hard and Soft Constraints Selection

In figure 4.3.5, users may choose which hard or soft constraints they want to include when generating their timetable.

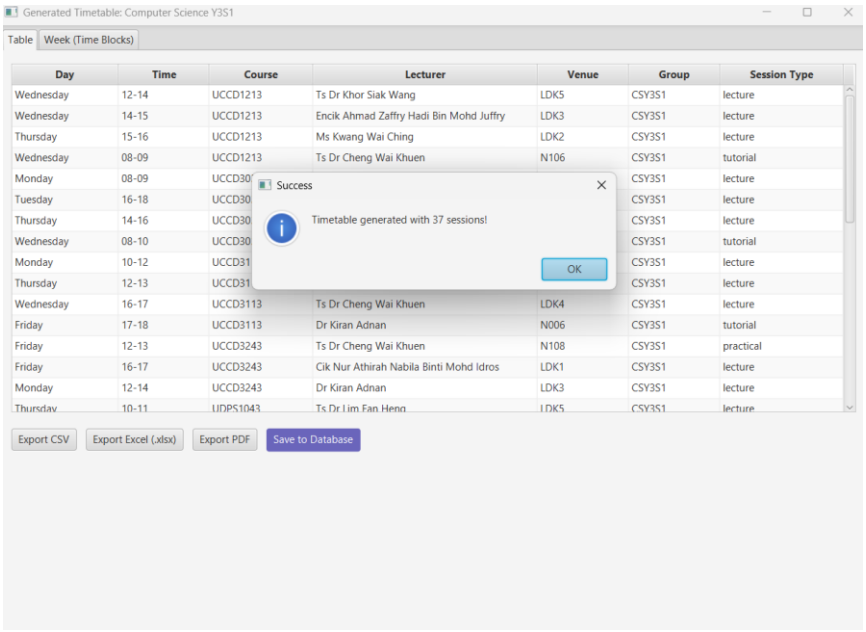

Figure 4.3.6 Timetable Generated in Table

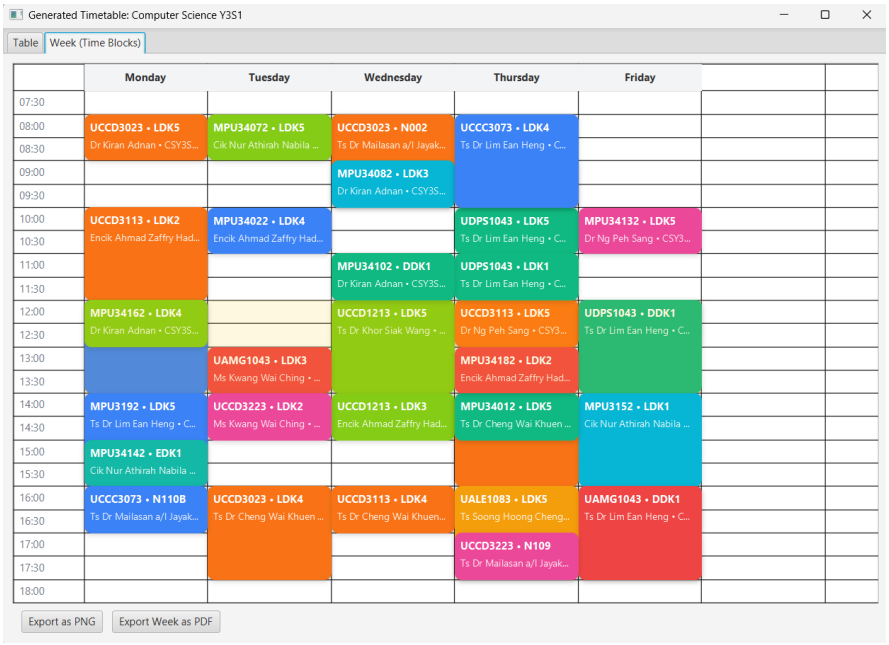

Figure 4.3.7 Timetable Generated Timeslots

The developed timetable generator system provides an automated solution for creating class schedules using a genetic algorithm. It offers two main forms of output, which is a structured tabular view and a visual weekly timetable. The tabular view lists detailed session information such as course code, lecturer, venue, student group, and session type, and it supports exporting to CSV, Excel, or PDF, as well as direct saving into the database. The visual timetable presents the same information in a calendar-style grid, where sessions are displayed as color-coded blocks arranged by day and time, making it easy for students and lecturers to understand their schedules at a glance. To enhance usability, the system also includes features such as clash detection, export options, and confirmation messages, ensuring that the generated timetable is valid, user-friendly, and suitable for both administrative and academic use.

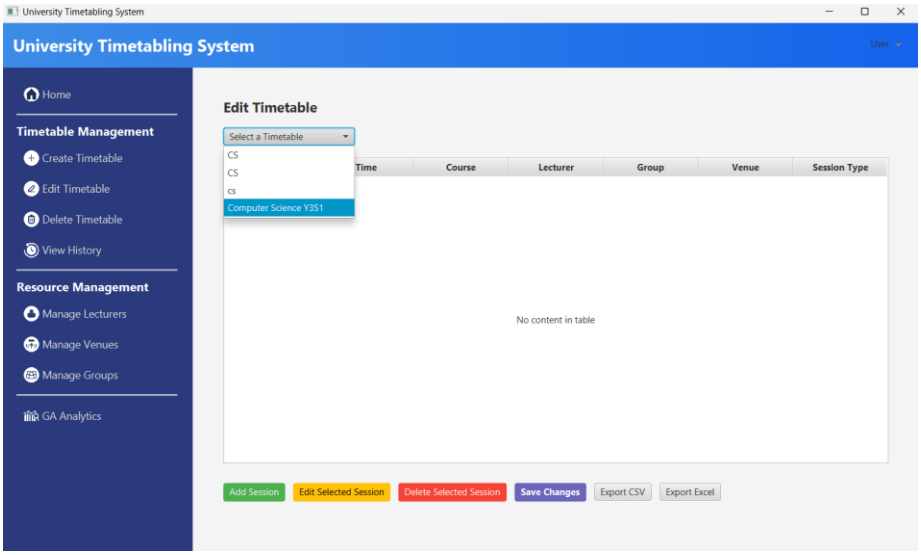

Figure 4.3.8 Edit Timetable Selection

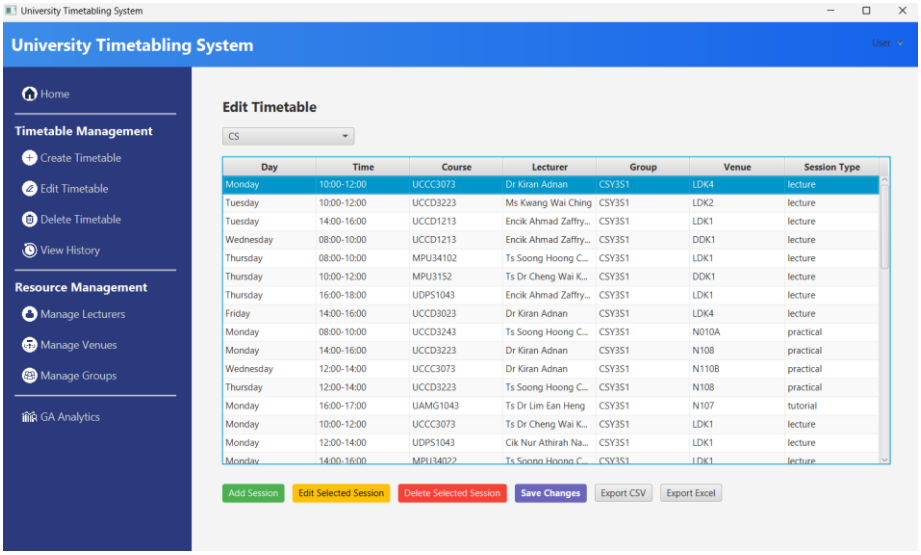

Figure 4.3.9 Edit Timetable Display Screen

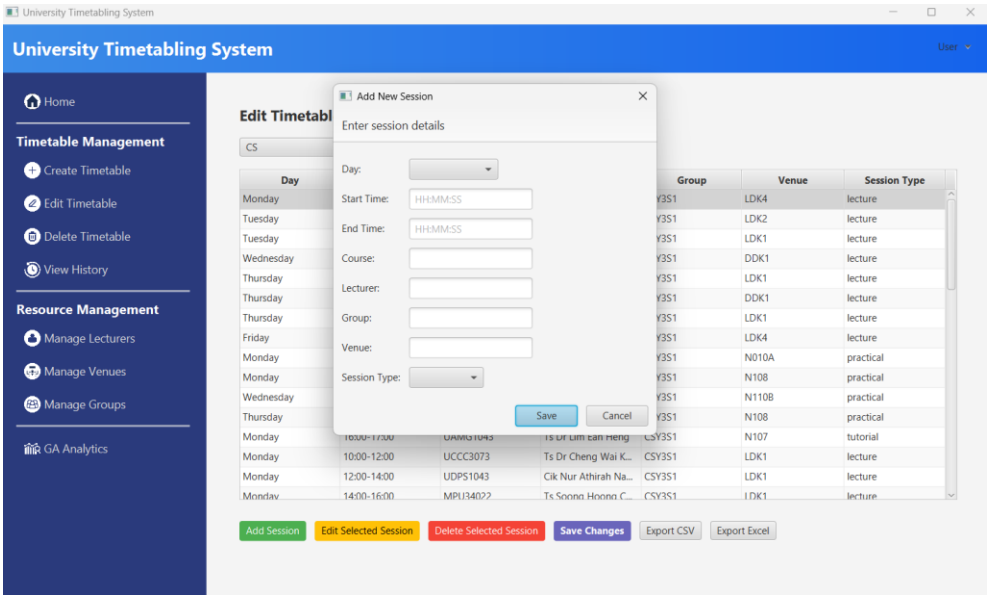

Figure 4.3.10 Edit Timetable Add Session

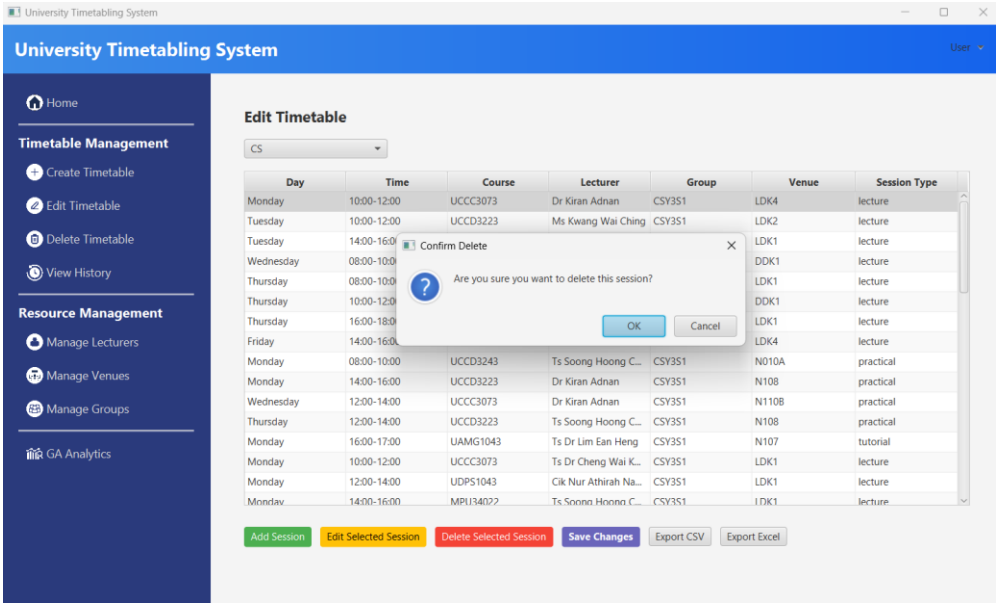

Figure 4.3.11 Edit Timetable Delete Session

The Edit Timetable function allows users to view, modify, and manage existing timetables stored in the system. When a user selects this feature, a dropdown menu is provided to choose from available timetables (Figure 4.3.8). Once selected, the system retrieves and displays the sessions associated with the chosen timetable in a tabular format, showing details such as day, time, course, lecturer, group, venue, and session

type (Figure 4.3.9). This ensures that users can easily locate and manage specific sessions.

As shown in Figure 4.3.10, the module supports multiple editing operations. Users can add new sessions by entering relevant details such as day, time range, course, lecturer, group, venue, and session type through a form interface. Existing sessions can be modified by selecting them from the table and updating the required information. Additionally, sessions may be deleted, with the system providing a confirmation dialog to prevent accidental removals (Figure 4.3.11).

At the bottom of the interface, functional buttons allow users to perform key operations: Add Session, Edit Selected Session, Delete Selected Session, and Save Changes. For data portability and reporting, the system also includes options to export the timetable into CSV or Excel format.

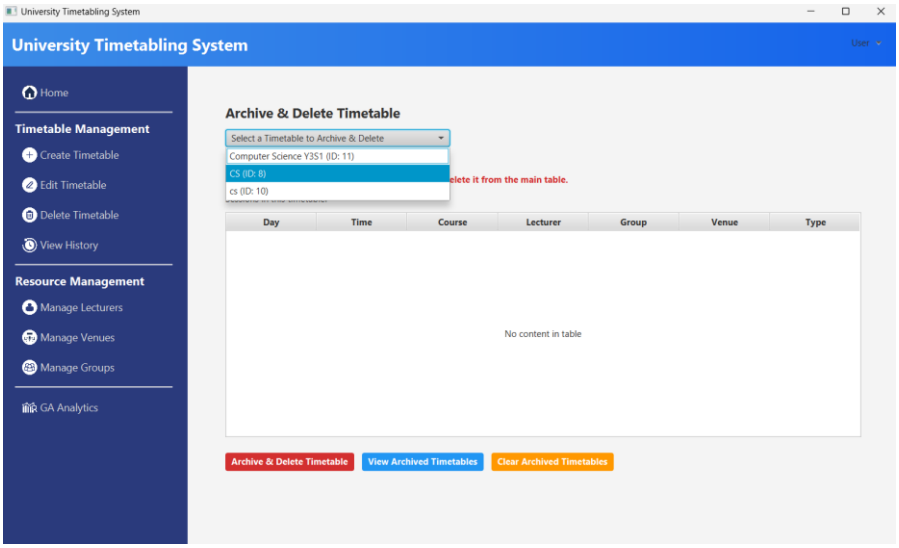

Figure 4.3.12 Delete Timetable Selection

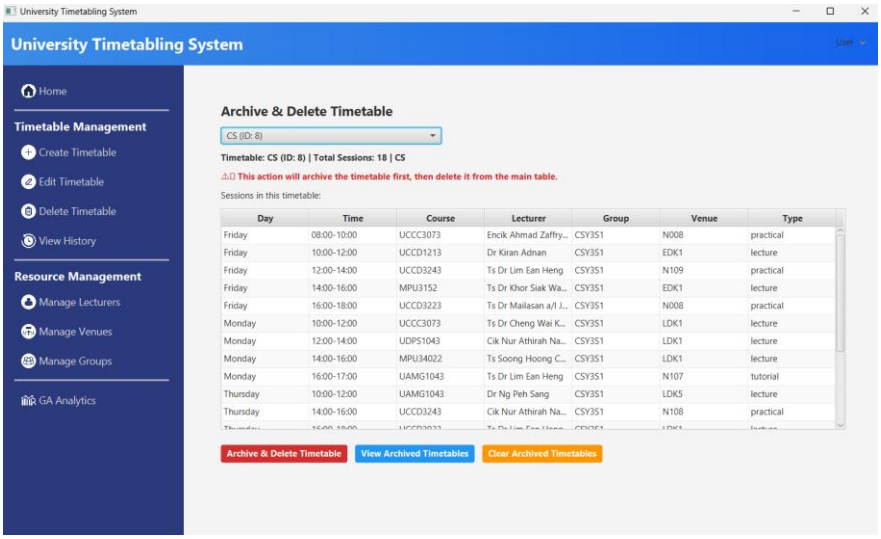

Figure 4.3.13 Delete Timetable Screen

The Delete Timetable function provides administrators with the ability to remove outdated or redundant timetables from the system while maintaining data integrity. In Figure 4.3.12, users can select a timetable from the dropdown list, where each entry is identified by its name and unique ID. Once selected, the system retrieves and displays the full list of sessions within the timetable, including details such as day, time, course, lecturer, group, venue, and session type as shown in Figure 4.3.13.

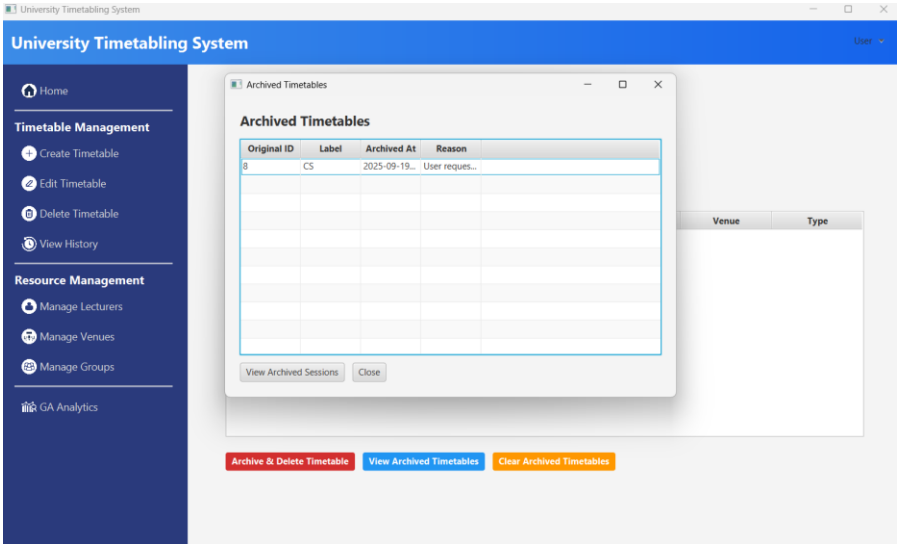

Figure 4.3.14 View Archived Timetables

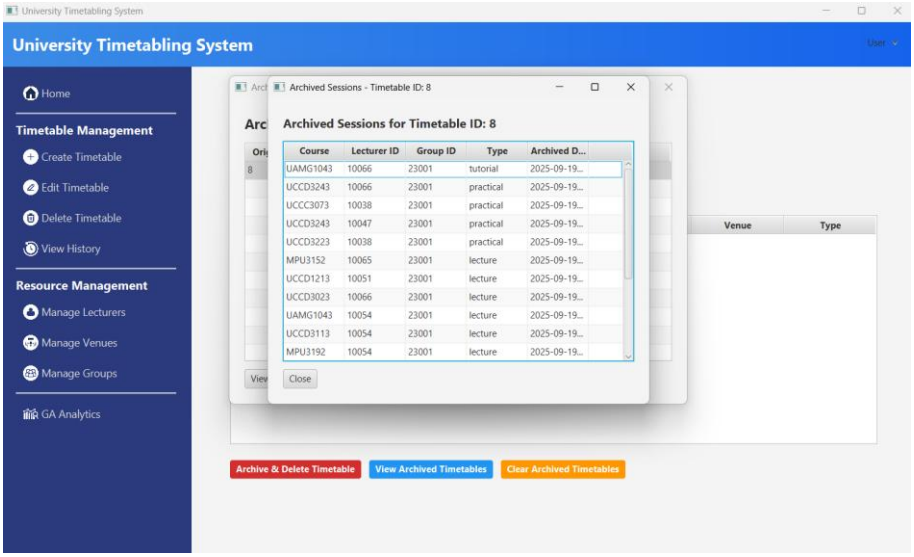

Figure 4.3.15 View Archived Sessions of Selected Timetables

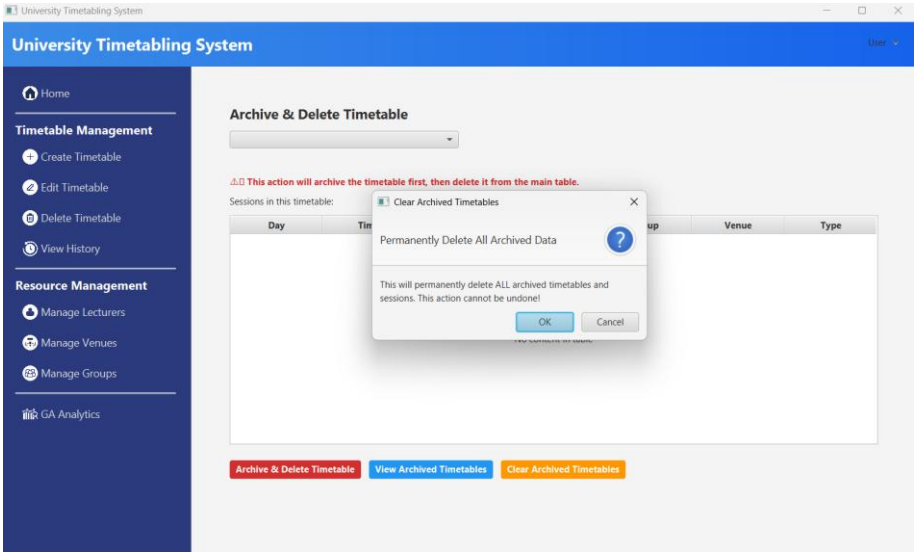

Figure 4.3.16 Clear Archived Timetables

Before deletion, the system applies an archive-first approach, meaning the timetable is first archived for record-keeping and then deleted from the active timetable list. This prevents accidental permanent loss of data and ensures that historical records remain available if needed. The red warning text reinforces the importance of the action, reminding users that this process removes the timetable from the main table.

The Archived Timetables feature provides a safe way to retain old timetables after they are deleted from the active list. When a timetable is archived, it is stored in a separate section along with metadata such as its original ID, label, archive date, and reason for archiving. Users can open the archive window to review all previously

archived timetables and select any timetable to view its detailed sessions (Figure 4.3.14, Figure 4.315). The detailed session view lists the archived courses together with lecturer IDs, group IDs, session types, and archive dates, ensuring that historical scheduling information remains accessible for reference or auditing purposes.

In addition to viewing, the system also provides a Clear Archived Timetables function as shown in Figure 4.3.16, which permanently deletes all archived data. A confirmation dialog is displayed before the action is performed, clearly warning that the deletion is irreversible. This safeguard ensures users are fully aware of the consequences before permanently removing historical timetable records.

Overall, this module strengthens data management by maintaining an archive of past timetables for accountability and traceability, while still allowing administrators to permanently clear data when it is no longer required.

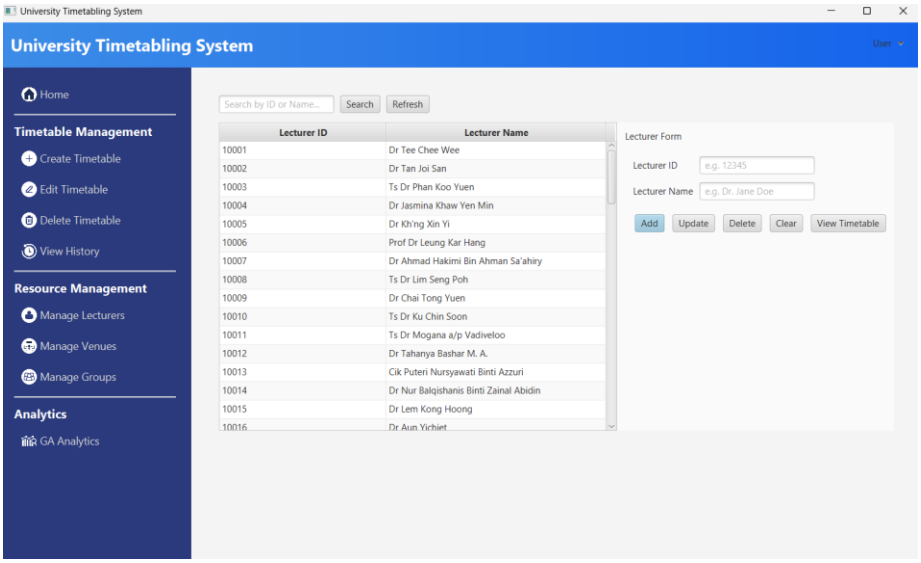

Figure 4.3.17 Manage Lecturers

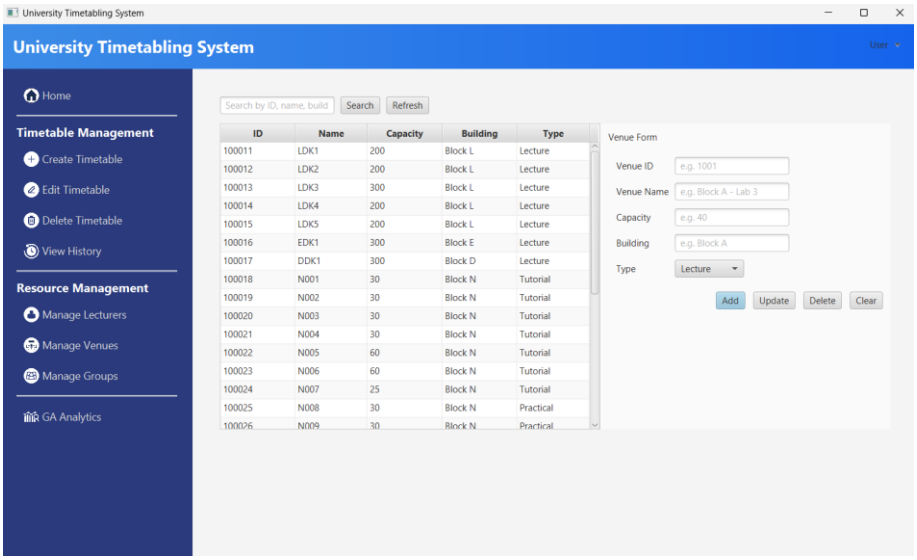

Figure 4.3.18 Manage Venues

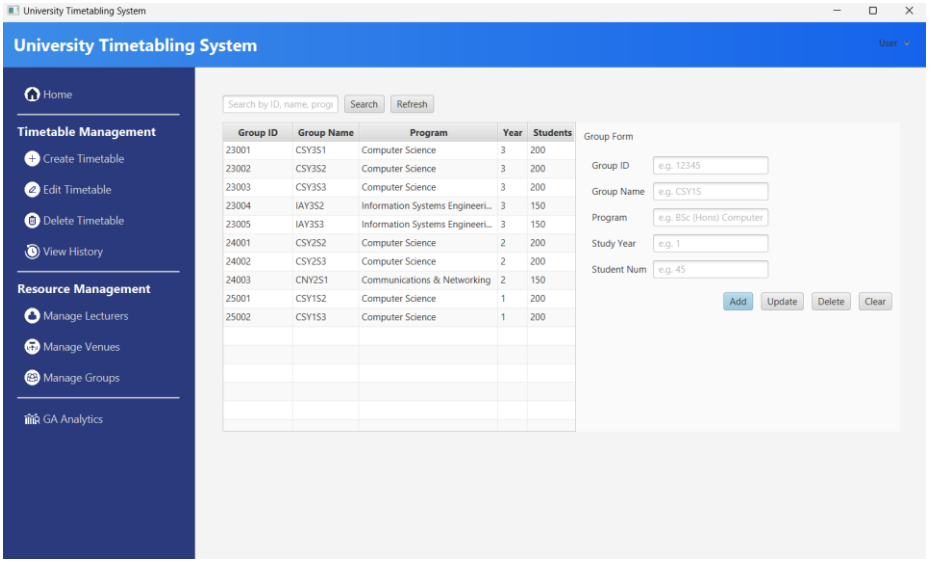

Figure 4.3.19 Manage Groups

The Resource Management section of the University Timetabling System allows administrators to manage the core resources required for timetable generation: lecturers, venues, and student groups. Each module provides CRUD (Create, Read, Update, Delete) functionality with a straightforward form interface and tabular data view.

The Manage Lecturers module in Figure 4.3.17 maintains a list of lecturers along with their unique IDs and names. Administrators can add new lecturers, update existing records, delete lecturers who are no longer active, or clear the form for new

input. A search function is available to quickly locate lecturers by ID or name, making the process efficient and scalable.

The Manage Venues module in Figure 4.3.18 handles information about available classrooms, labs, and lecture halls. Each venue record stores attributes such as venue ID, name, capacity, building, and type (lecture, tutorial, or practical). This ensures that room assignments during timetable generation respect constraints such as seating capacity and room type suitability. The module includes functions to add, update, delete, and search for venues, ensuring the database remains accurate and up to date.

The Manage Groups module in Figure 4.3.19 organizes student groups by ID, group name, program, year of study, and number of students. This information is crucial for scheduling as it ensures that the system allocates the correct sessions to the right groups, considering both academic program and cohort size. Like the other modules, it supports adding, updating, and deleting group records, as well as searching based on group details.

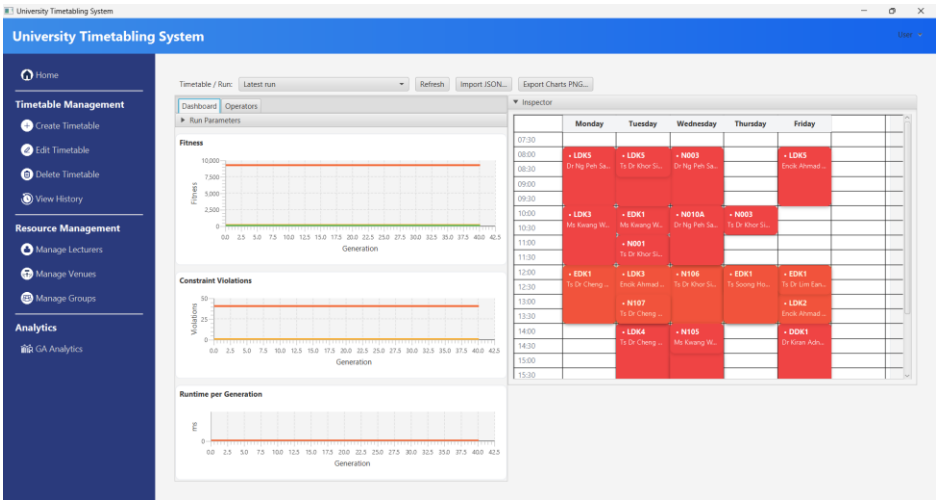

Figure 4.3.20 GA Analytics

The GA Analytics module in Figure 4.3.20 provides insights into the genetic algorithm's timetable generation process by displaying key parameters, operations, and fitness evaluation results. At the top of the interface, the system shows the run parameters used for the current execution, such as population size, mutation rate, crossover rate, and constraints. This makes it easier for users to understand the configuration that influenced the timetable generation.

On the left side of the interface, the system tracks the evolutionary operations across generations, including selection, crossover, and mutation events. This log enables users to follow how the genetic algorithm progresses over time, refining solutions until a feasible timetable is reached.

The right-hand section includes a visual inspector, where the generated timetable is displayed in a weekly grid format. Each session is placed in its respective day and timeslot, showing course codes, lecturers, and venues. The color-coded visualization makes it easy to identify allocated sessions and spot potential constraint violations.

By combining algorithmic details with a clear timetable visualization, the GA Analytics module bridges the gap between backend optimization and user understanding, ensuring that administrators can both validate the output and analyze the effectiveness of the genetic algorithm.

CHAPTER 5

System Implementation

5.1 Hardware Setup

The hardware used in this project includes an android device for interface testing and a laptop computer that can handle the development and test phases of the timetabling system. Python will be used to implement the genetic algorithm, MySQL to manage databases, and Flutter using Android Studio to build the smartphone interface. The high-end hardware ensures maximum processing, quick compilation, and smooth system execution, especially on schedule generation problems with many class events and constraints. Although the algorithm relies less on GPU acceleration, the overall performance and responsiveness of the system are enhanced by the high-end specifications. Table 3.4.1.1 below shows the detailed specifications of the laptop used.

Table 5.1.1 Specifications of laptop

Description	Specifications
Model	MSI Katana A15 AI B8VE
Processor	AMD Ryzen 7 8845HS
Operating System	Windows 11 64-bit
Graphic	NVIDIA® GeForce® RTX 4050
Memory	16.0GB RAM
Storage	512GB SSD

5.2 Software Setup

Table 5.2.1 below summarizes the software to be used in the project.

Table 5.2.1 Software Used

Description	Specifications
Operating System	Windows 11 Home Single Language
Programming Language	Python – Visual Studio Code
Database Management System	MySQL
Standalone Web Interface	Eclipse IDE for Java Developers – 2025-06
API	Flask
Version Control System	GitHub

5.3 Setting and Configuration

Visual Studio Code Setup:

Figure 5.3.1 and Figure 5.3.2 below shows the basic setup for Visual Studio Code.

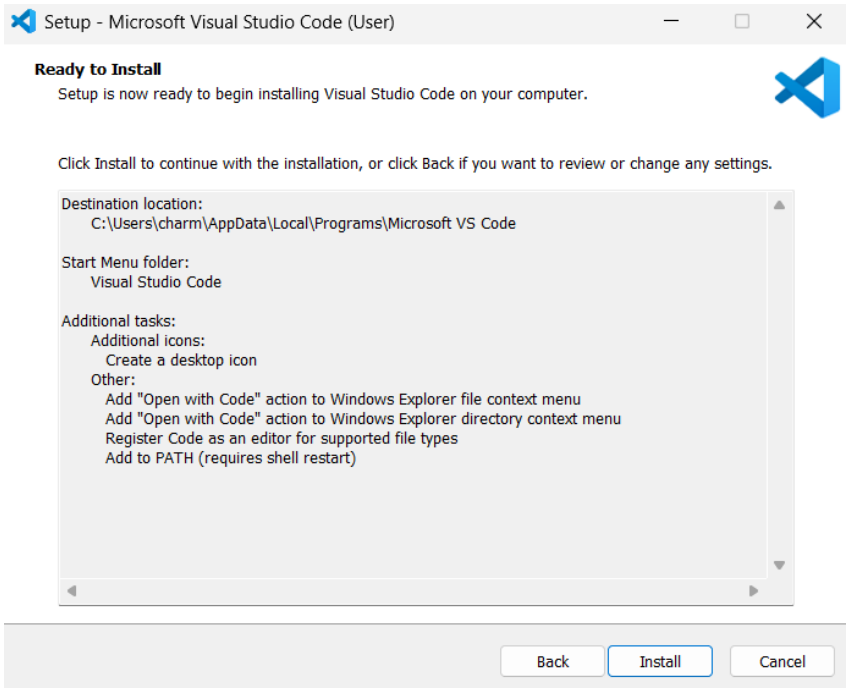

Figure 5.3.1 Visual Studio Code Installation

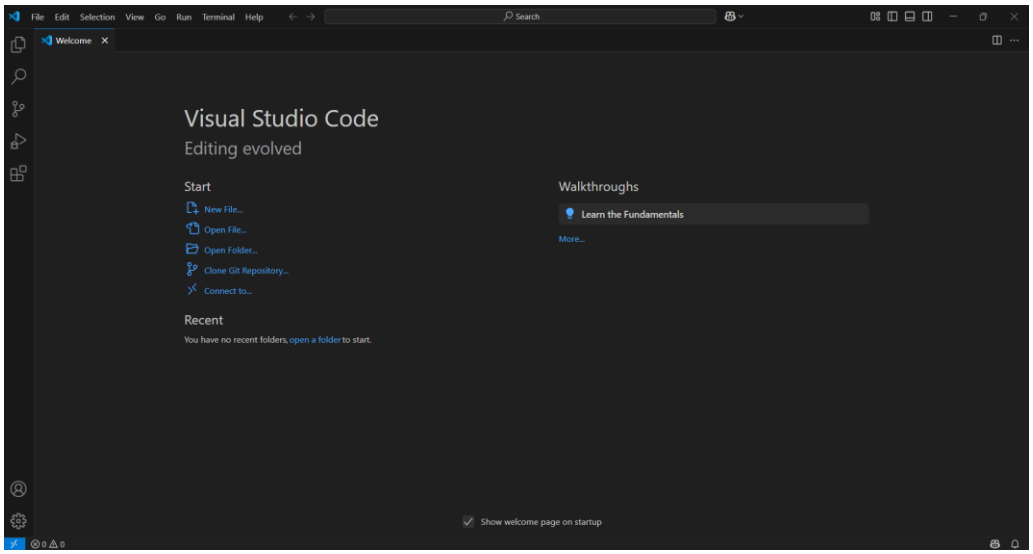

Figure 5.3.2 Visual Studio Code Default Home Screen

Python:

Figure 5.3.3 displays the installation of Python 3.13.3 for Windows locally on the laptop. This package is downloaded from the official website: https://www.python.org/

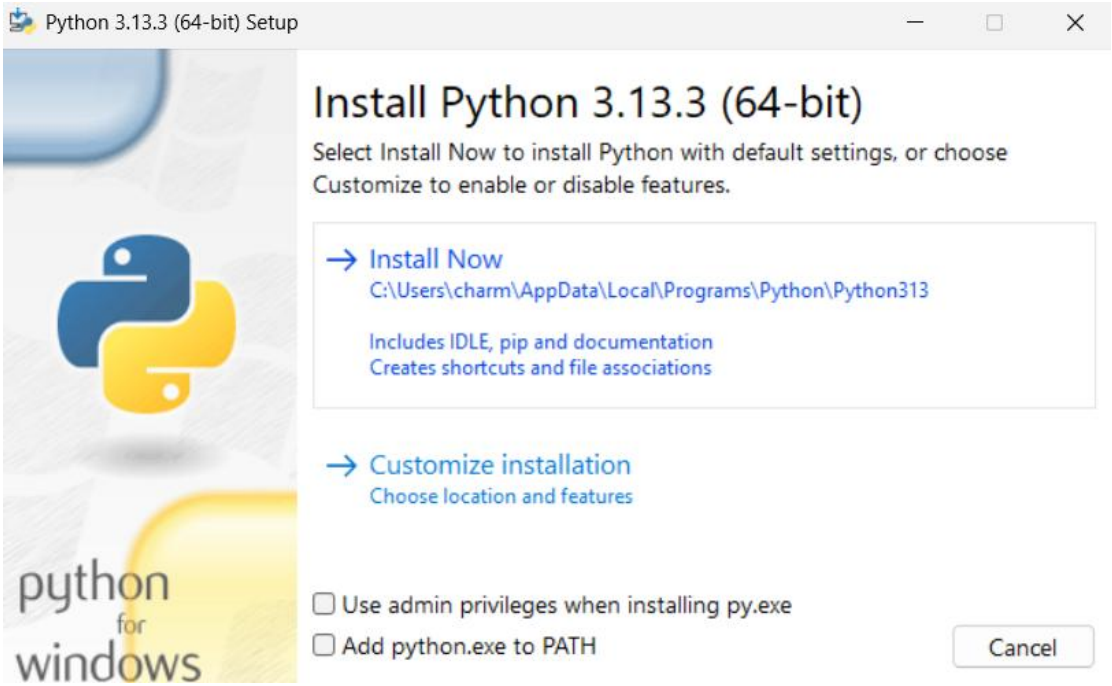

Figure 5.3.3 Python 3.13.3 Installation

Figure 5.3.4 below shows the installation of the Python Extension in Visual Studio Code.

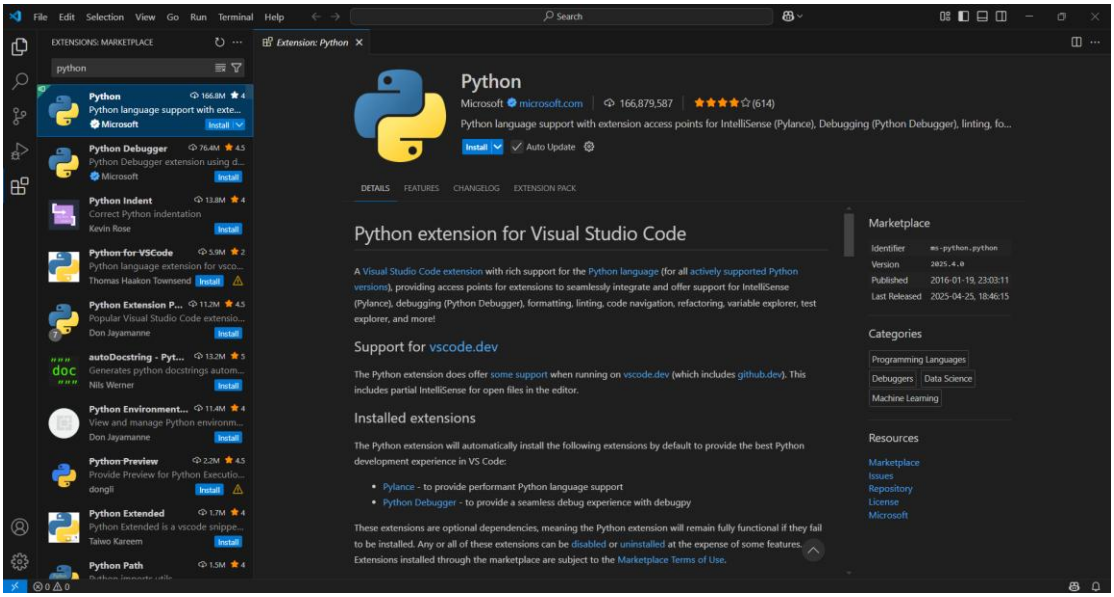

Figure 5.3.4 Visual Studio Code Python Extension Installation

Figure 5.3.5 below shows the installation of the Code Runner in Visual Studio Code.

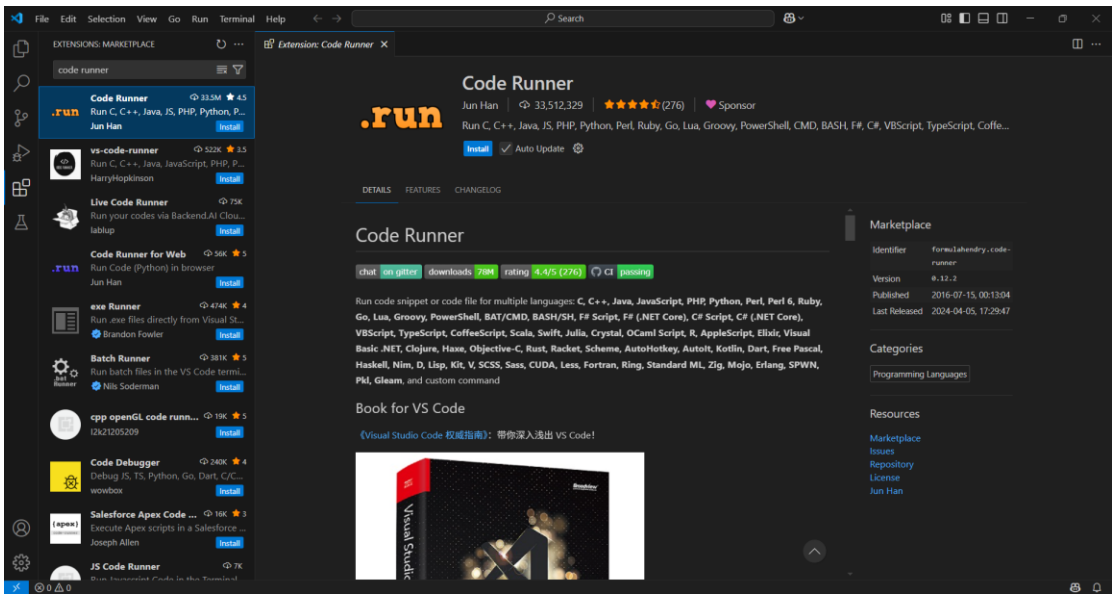

Figure 5.3.5 Visual Studio Code Runner Extension Installation

Figure 5.3.6 and Figure 5.3.7 below shows the configuration of the Code Runner extension in Visual Studio Code. This step enables Visual Studio Code to run and debug the Python code easily as it enables the built-in Visual Studio Code Terminal to be activated.

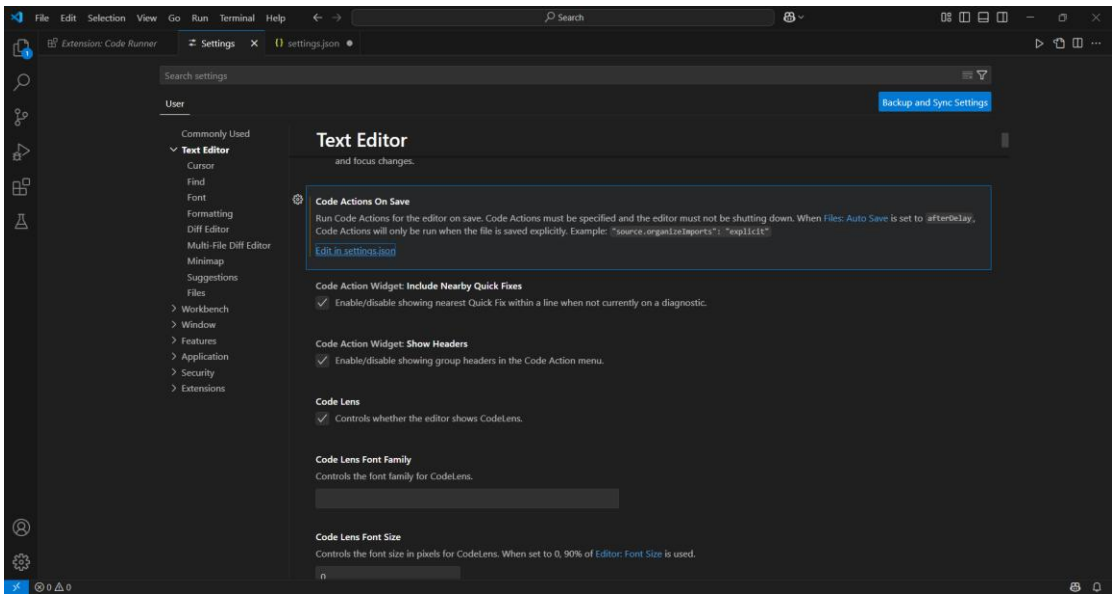

Figure 5.3.6 Visual Studio Code Runner Extension Configuration

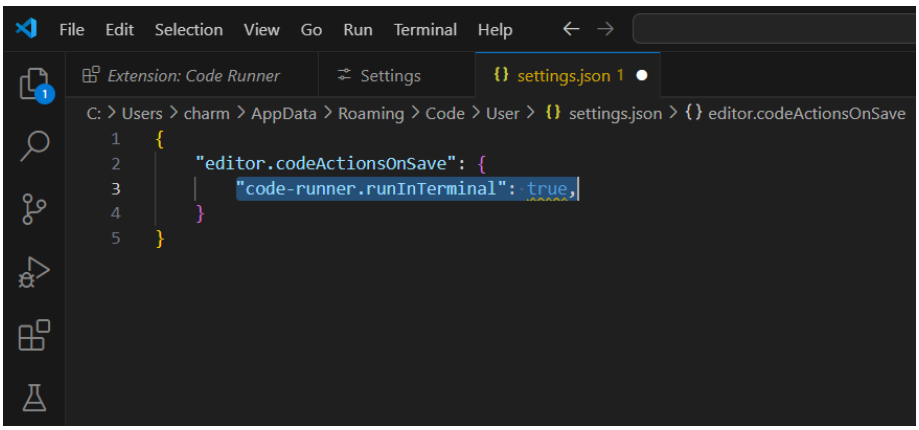

Figure 5.3.7 Visual Studio Code Runner Extension Setup

Figure 5.3.8 and Figure 5.3.9 below shows the configuration of the path for Python to be defined.

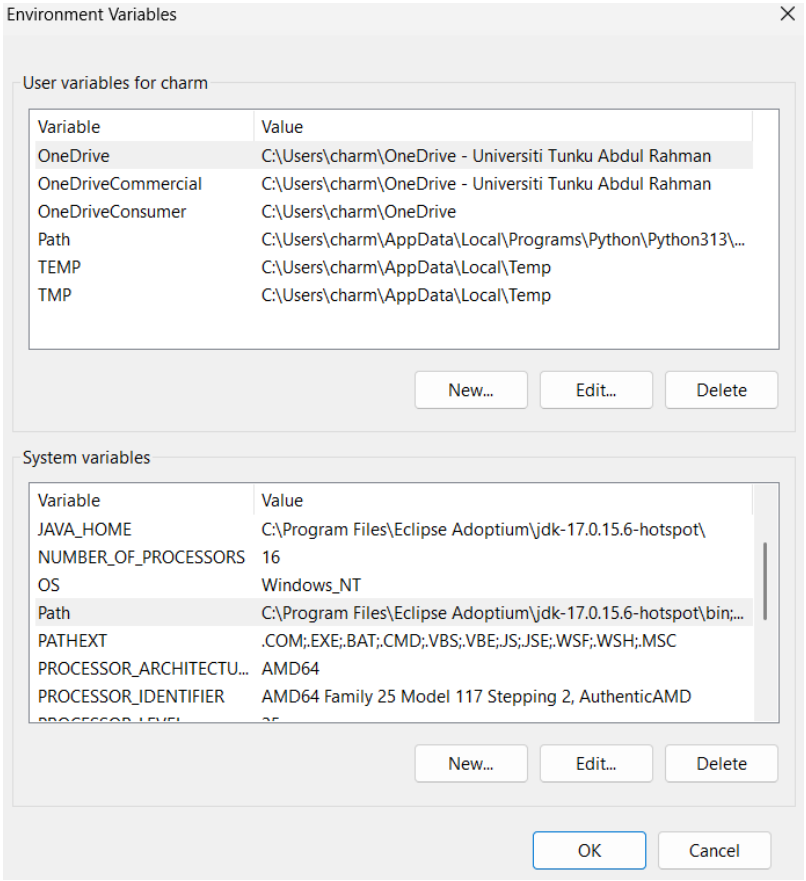

Figure 5.3.8 Python Path Configuration in Environment Variables

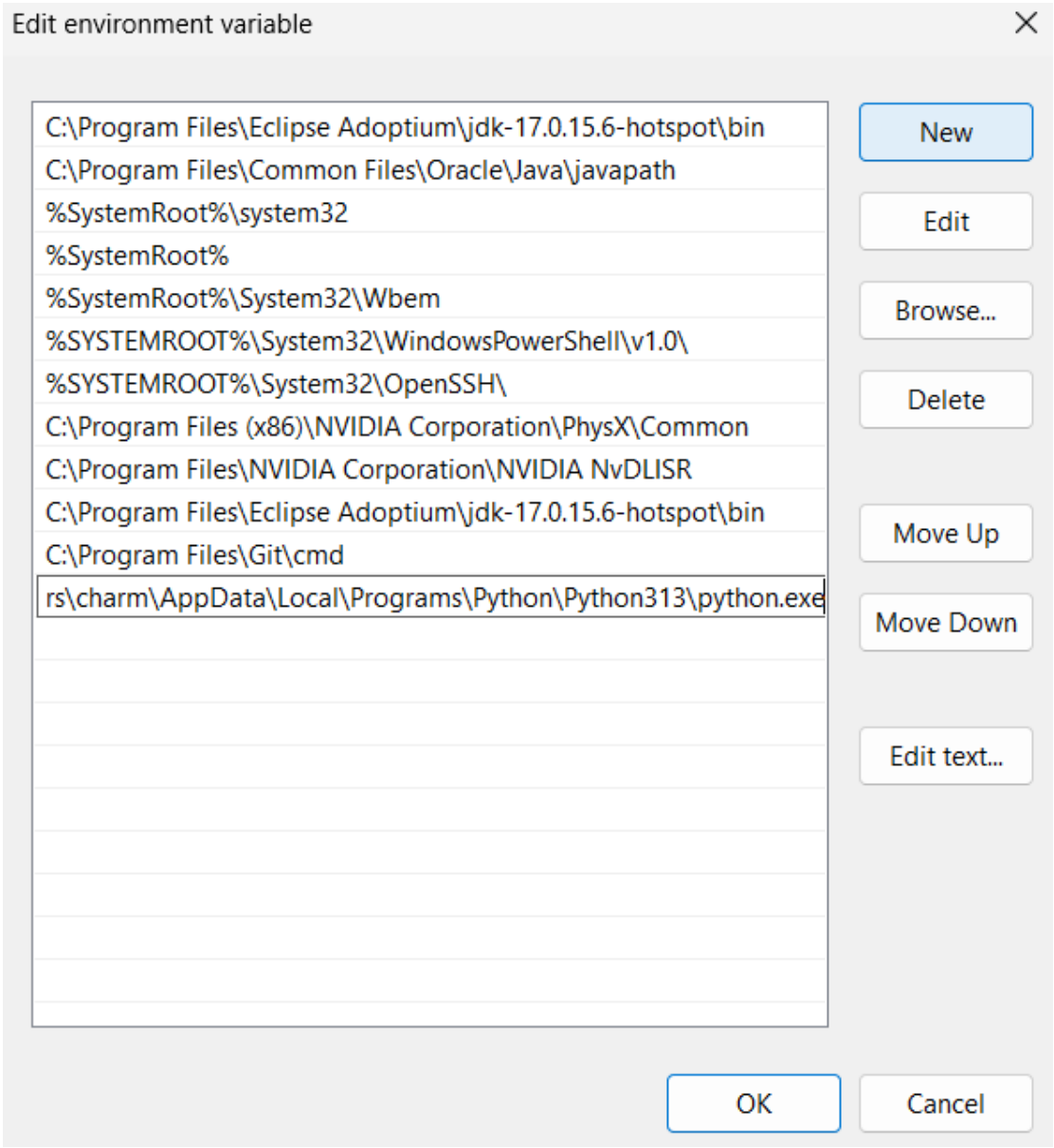

Figure 5.3.9 Python New Path Configuration in Environment Variables

Flask:

Figure 5.3.10 and Figure 5.3.11 below shows the installation steps for Flask in Visual Studio Code.

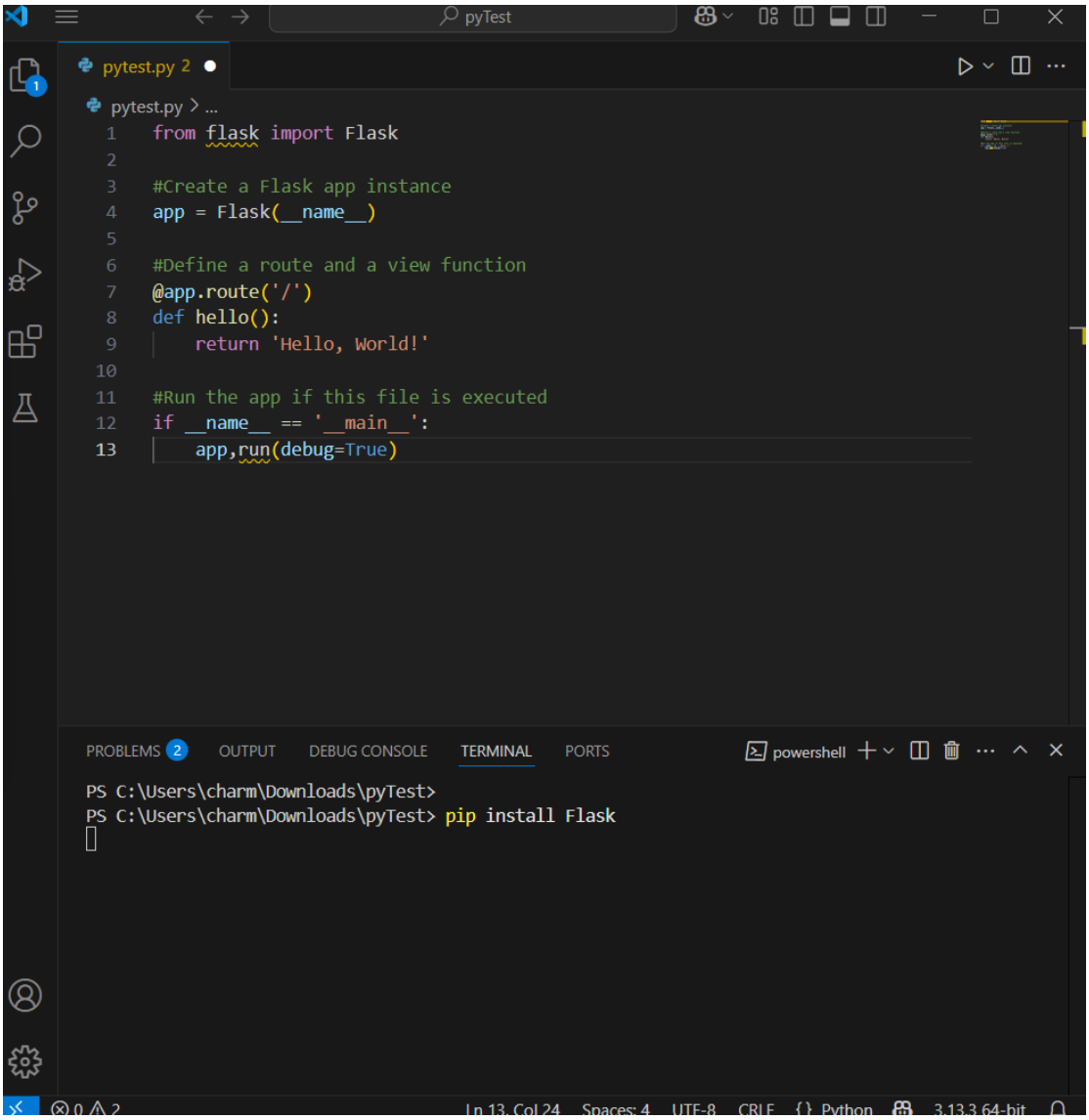

Figure 5.3.10 Flask API in Visual Studio Code Installation

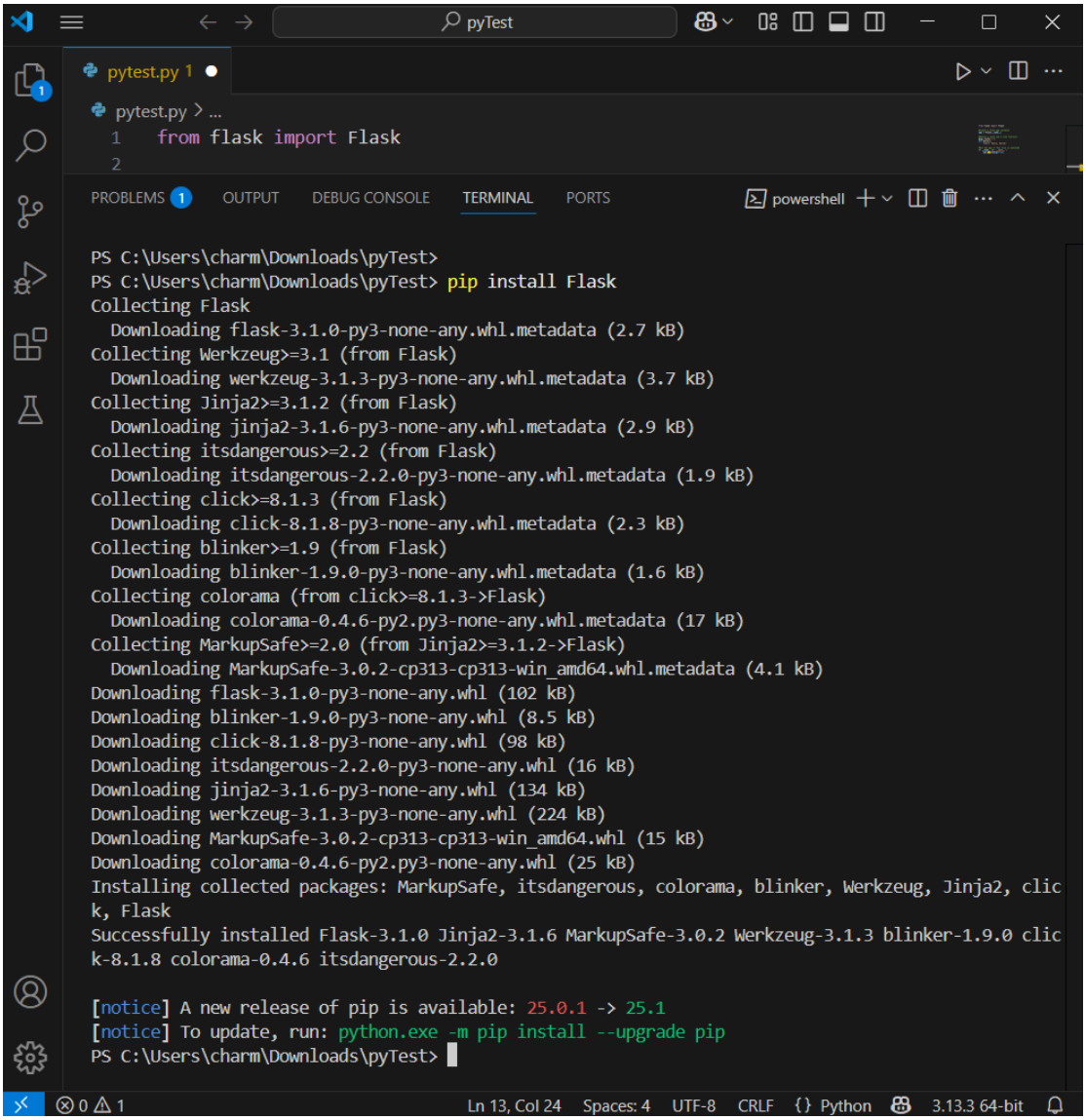

Figure 5.3.11 Flask API in Visual Studio Code Installation Success

Some basic code and configuration commands are used to test the API connection.

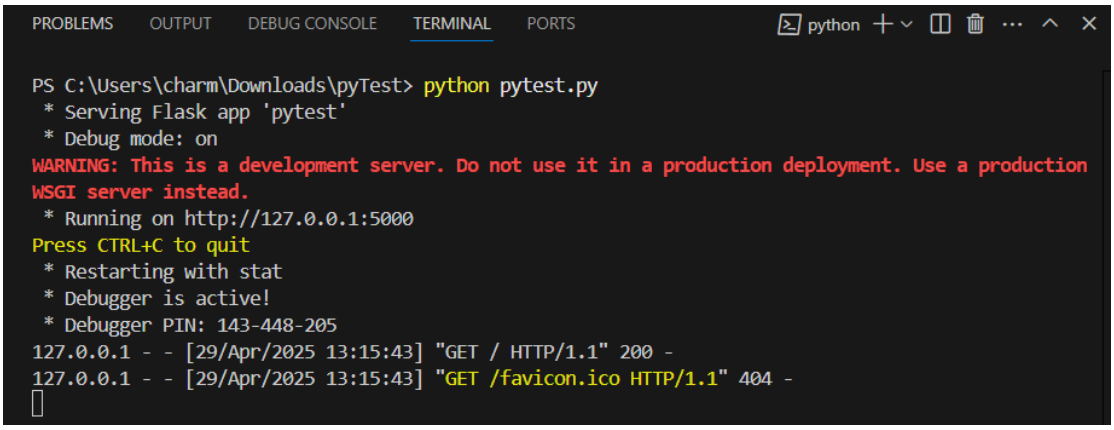

Figure 5.3.12 Flask API in Visual Studio Code Testing

MySQL:

Figure 5.2.13 below shows the version of MySQL that was installed.

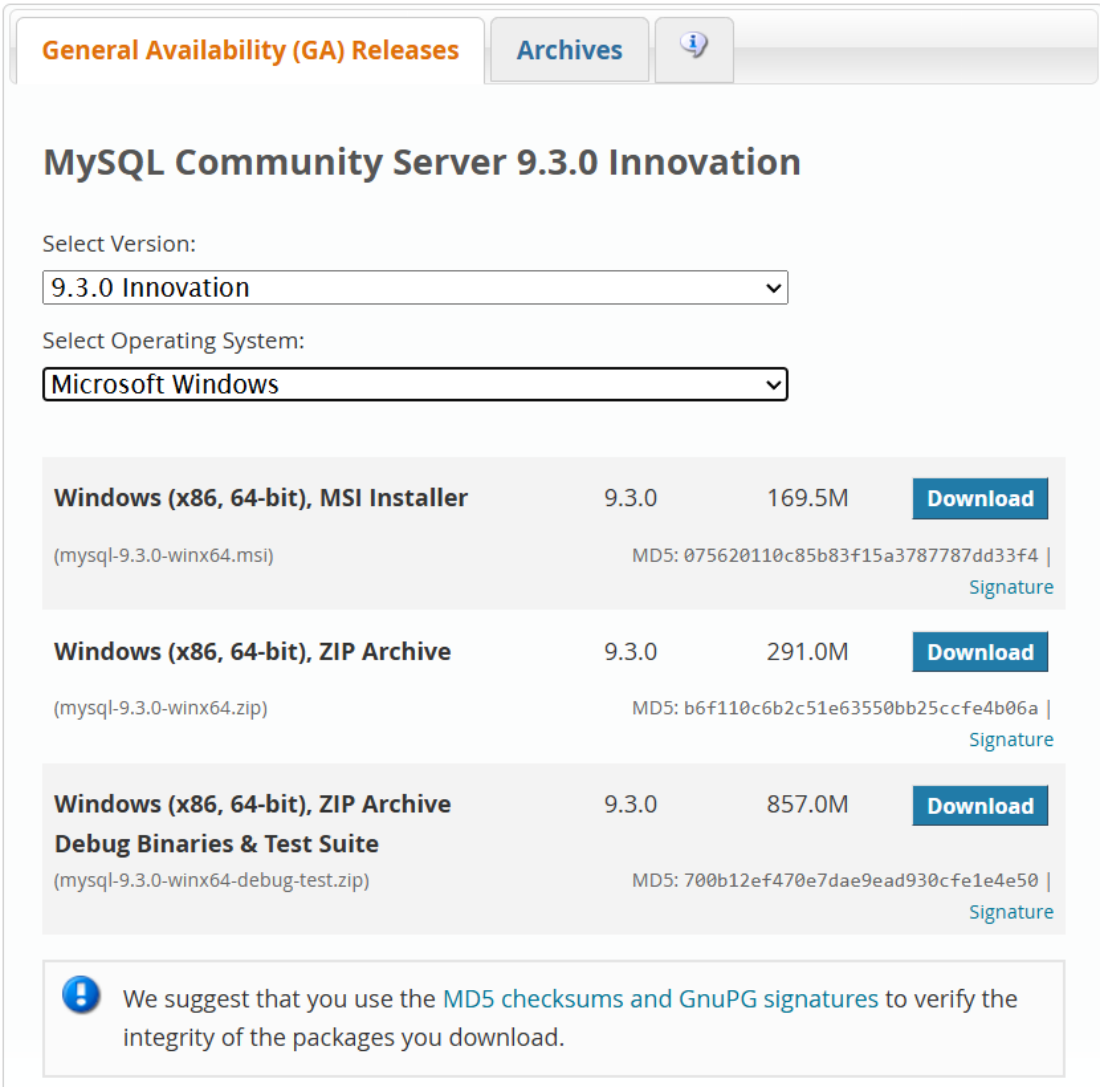

Figure 5.3.13 MySQL Installation

Figure 5.3.14 below shows the version of MySQL that was installed.

Figure 5.3.14 MySQL Setup 1

Figure 5.3.15, Figure 5.3.16, Figure 5.3.17, Figure 5.3.18, Figure 5.3.19, Figure 5.3.20, Figure 5.3.21, Figure 5.3.22, Figure 5.3.23, Figure 5.3.24 and Figure 5.3.25 below shows the configuration steps for installing MySQL.

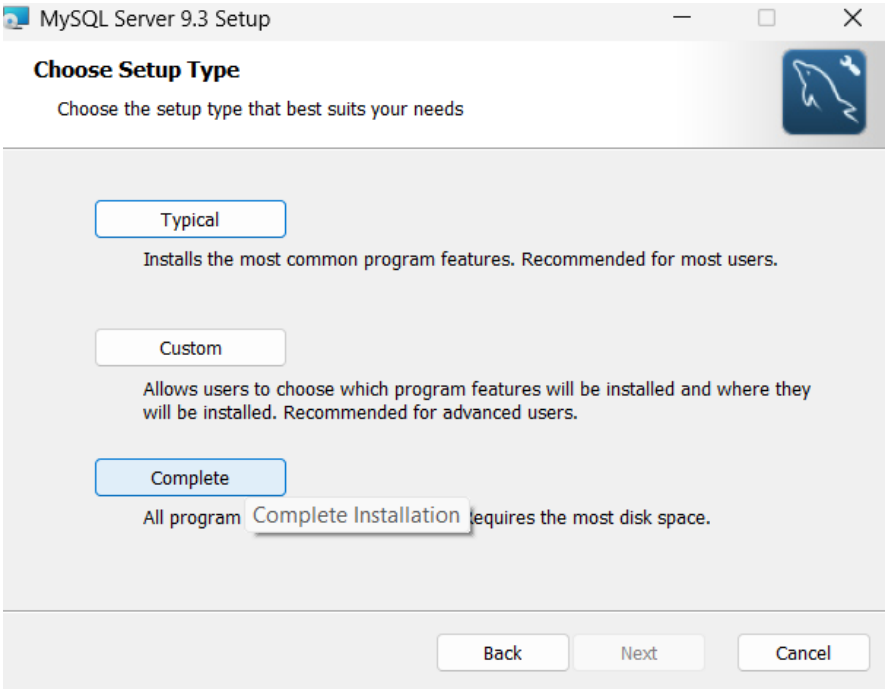

Figure 5.3.15 MySQL Setup 2

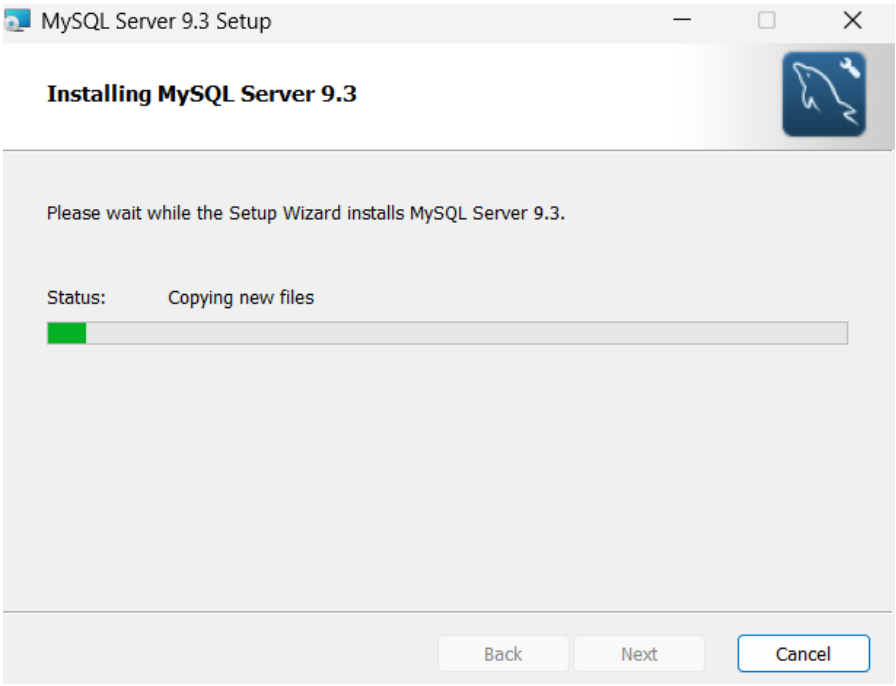

Figure 5.3.16 MySQL Setup 3

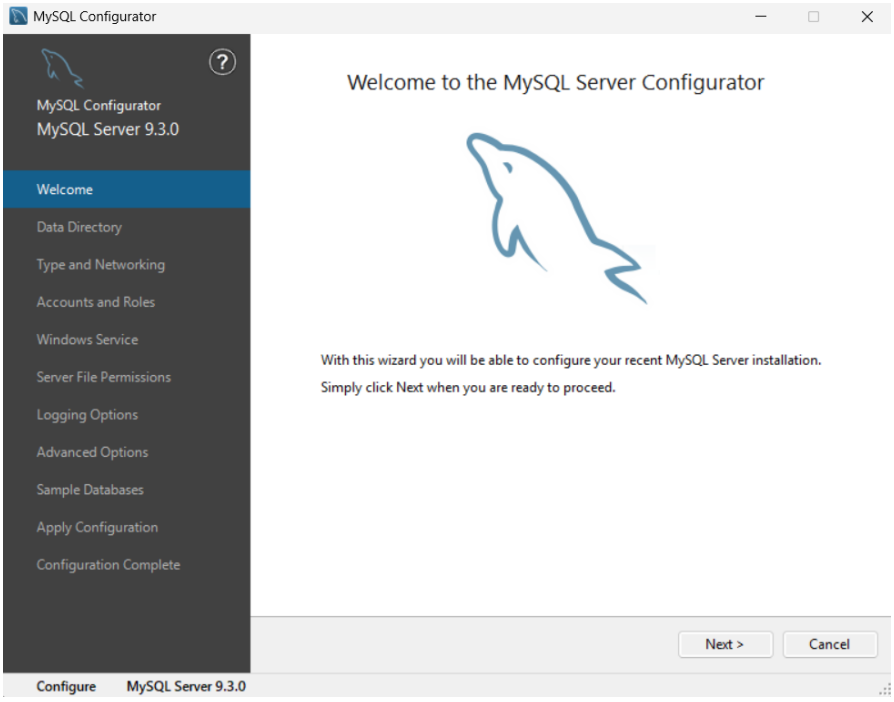

Figure 5.3.17 MySQL Setup 4

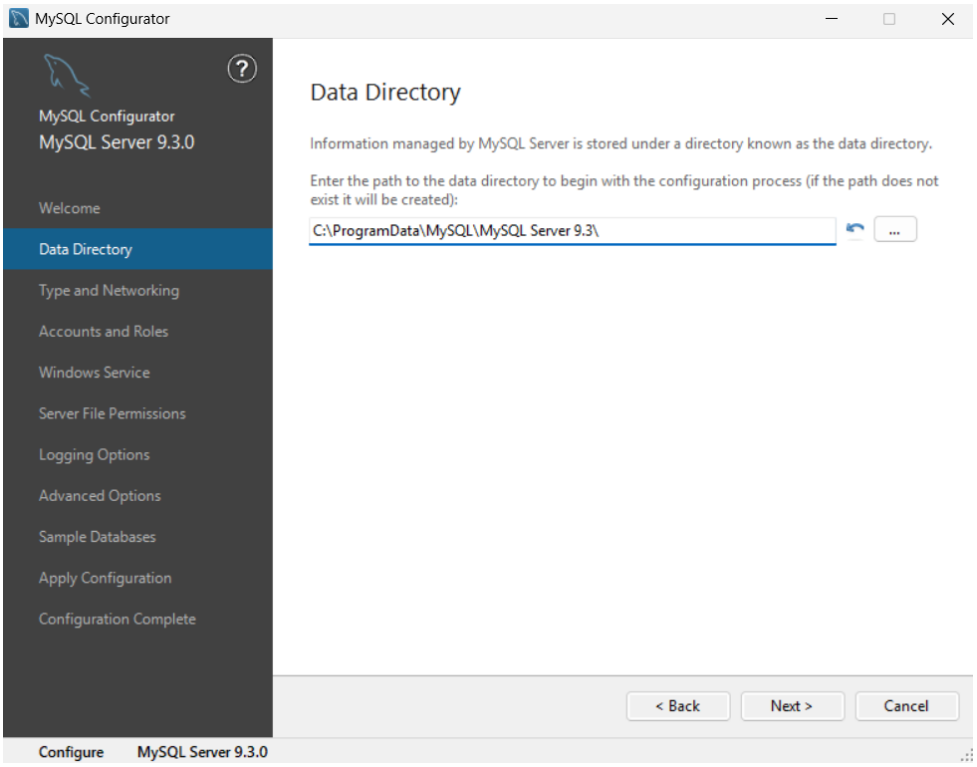

Figure 5.3.18 MySQL Setup 5

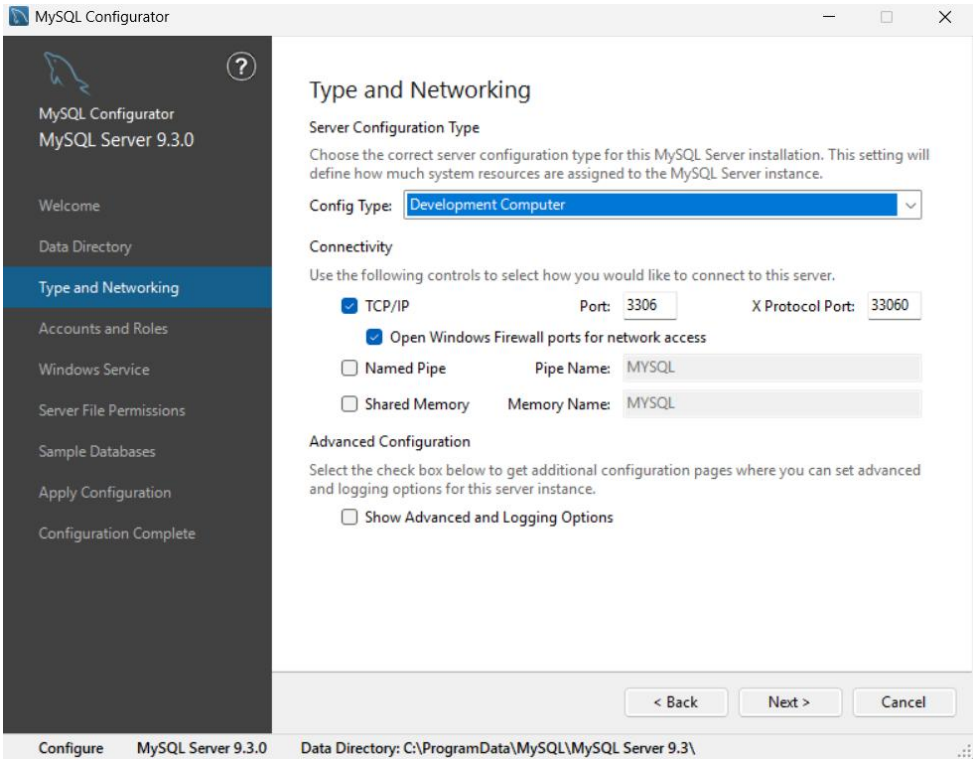

Figure 5.3.19 MySQL Setup 6

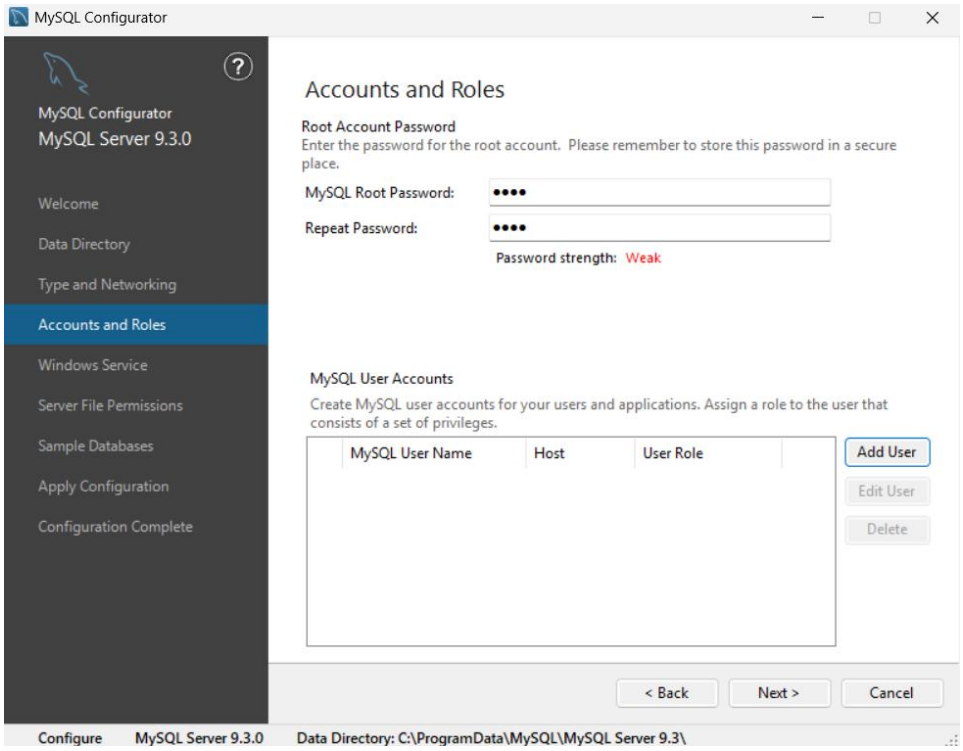

Figure 5.3.20 MySQL Setup 7

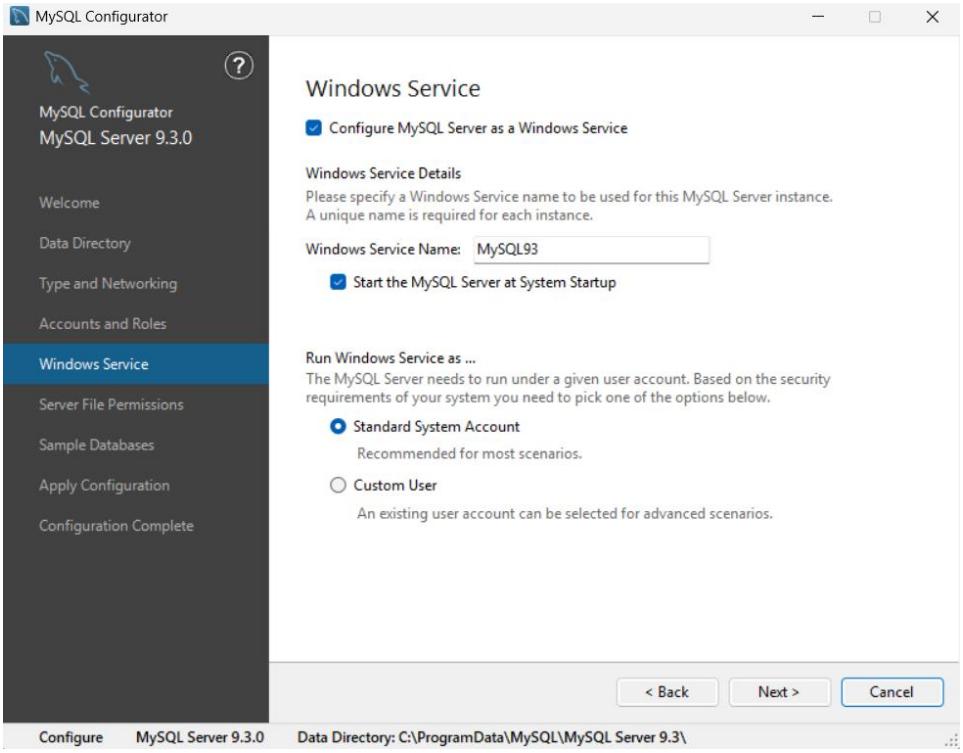

Figure 5.3.21 MySQL Setup 8

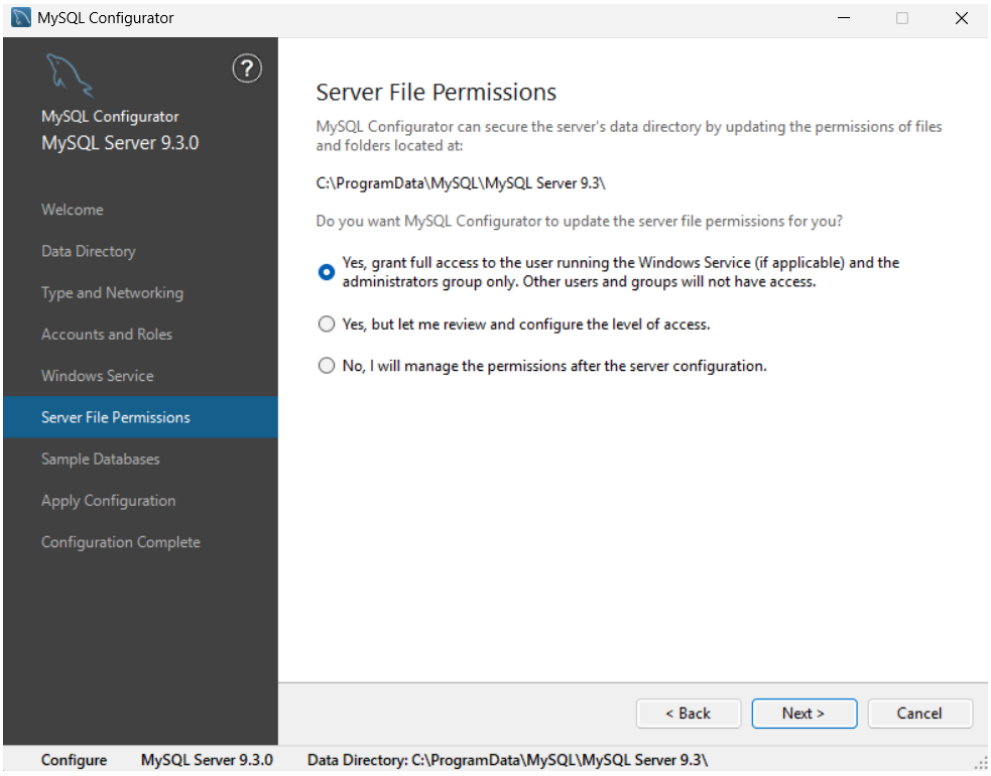

Figure 5.3.22 MySQL Setup 9

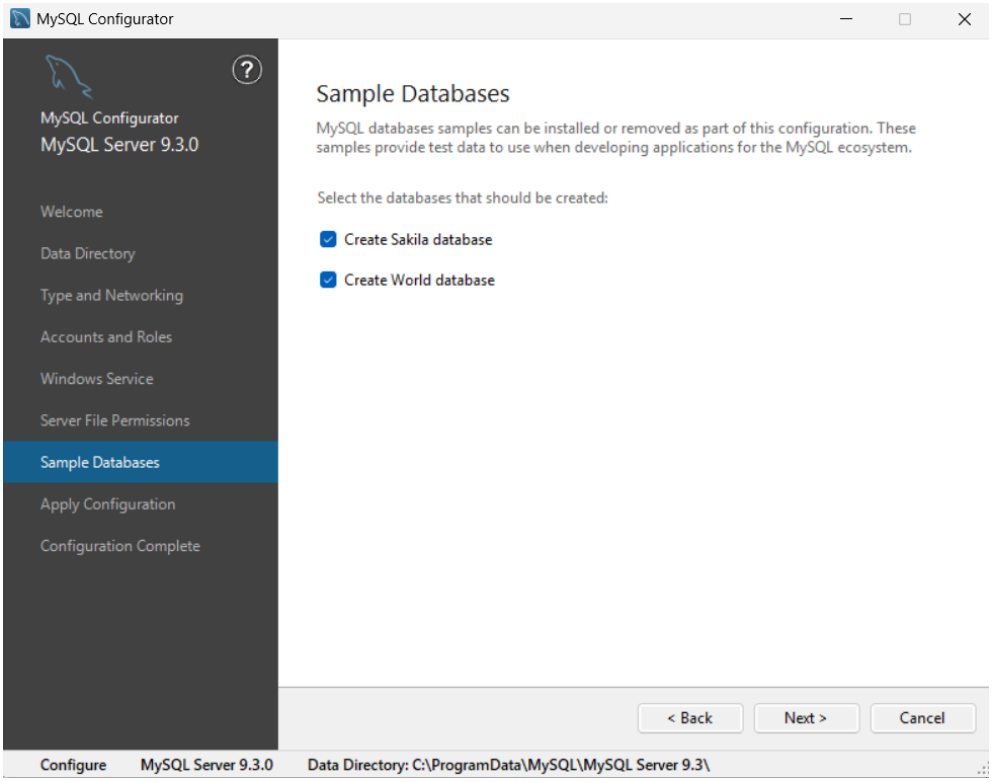

Figure 5.3.23 MySQL Setup 10

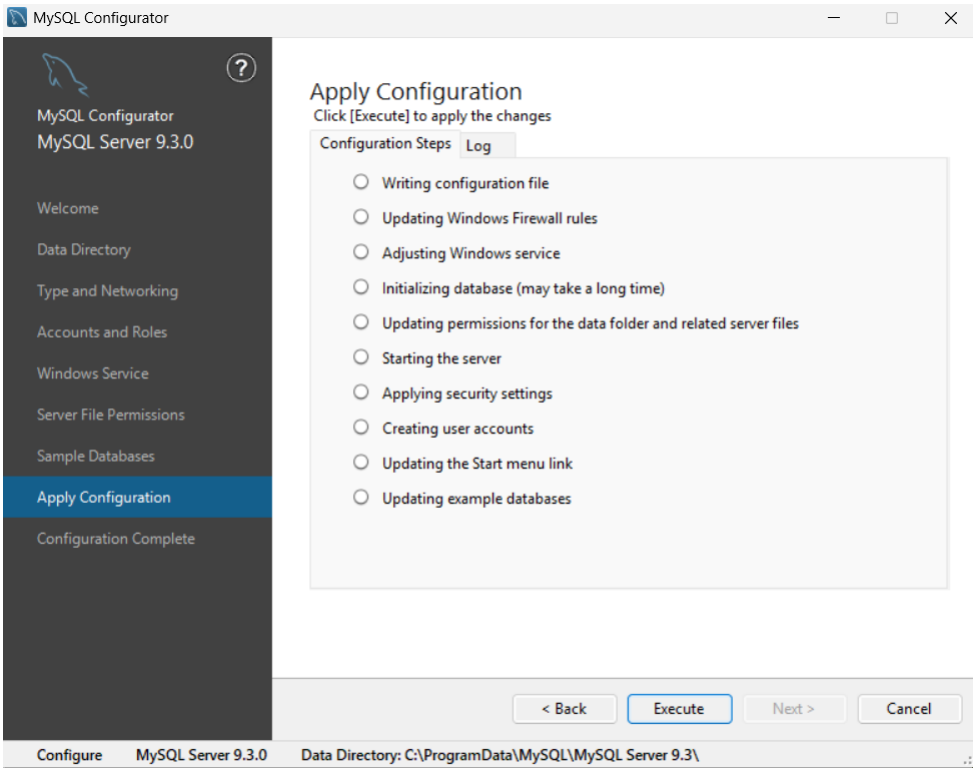

Figure 5.3.24 MySQL Setup 11

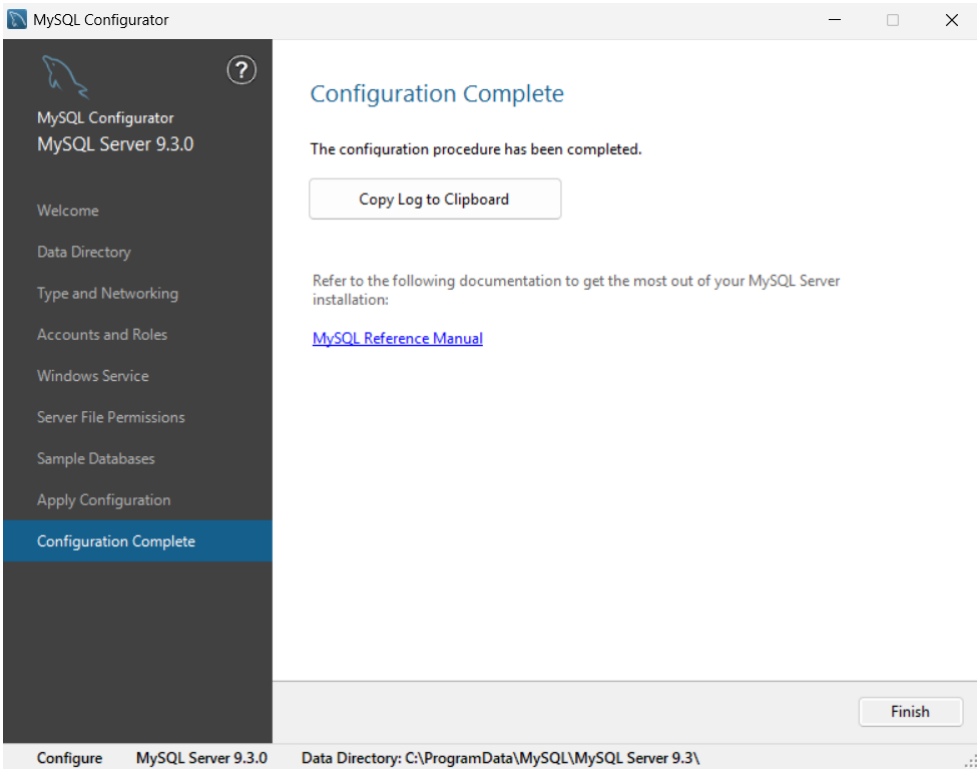

Figure 5.3.25 MySQL Configuration Completed

Figure 5.3.26 shows the installation of SQLTools extension in Visual Studio Code which enables for SQL queries to be used in Visual Studio Code.

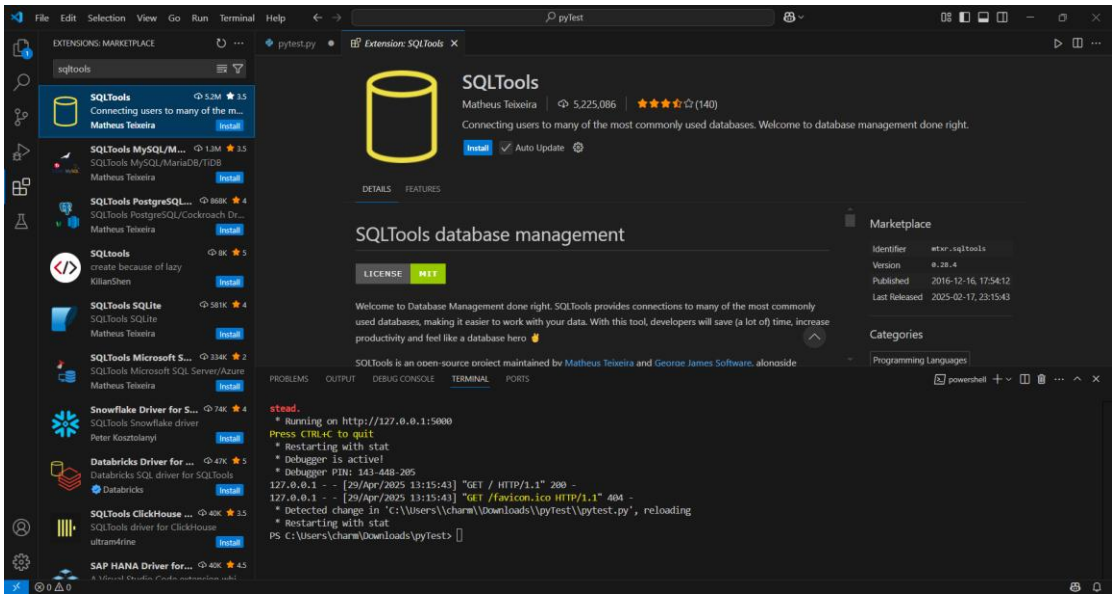

Figure 5.3.26 Visual Studio Code SQLTools Extension Installation

For SQLTools to be able to work, a driver is required. Figure 5.3.27 displays the installation of SQLTools MySQL/MariaDB.TiDB extension in Visual Studio Code.

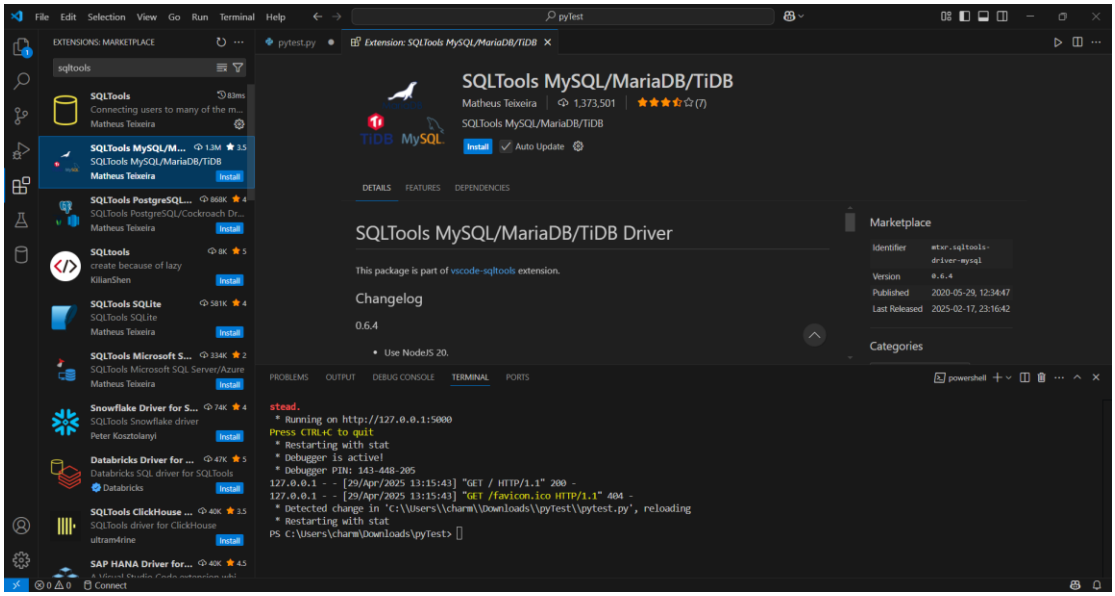

Figure 5.3.27 Visual Studio Code SQLTools MySQL/MariaDB/TiDB Extension Installation

Figure 5.3.28, Figure 5.3.29, Figure 5.3.30 and Figure 5.3.31 continues with the steps for configuring SQLTools in Visual Studio Code until the connection with the database is successful.

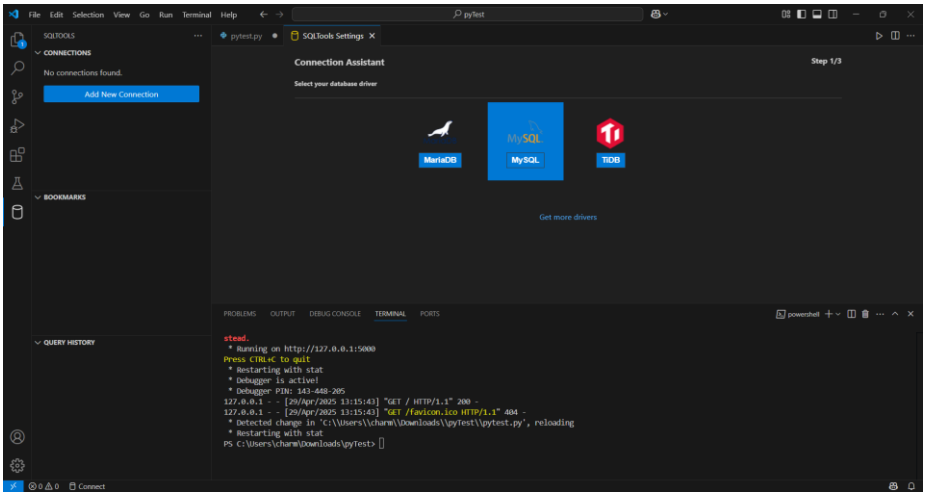

Figure 5.3.28 Visual Studio Code SQLTools Settings for MySQL

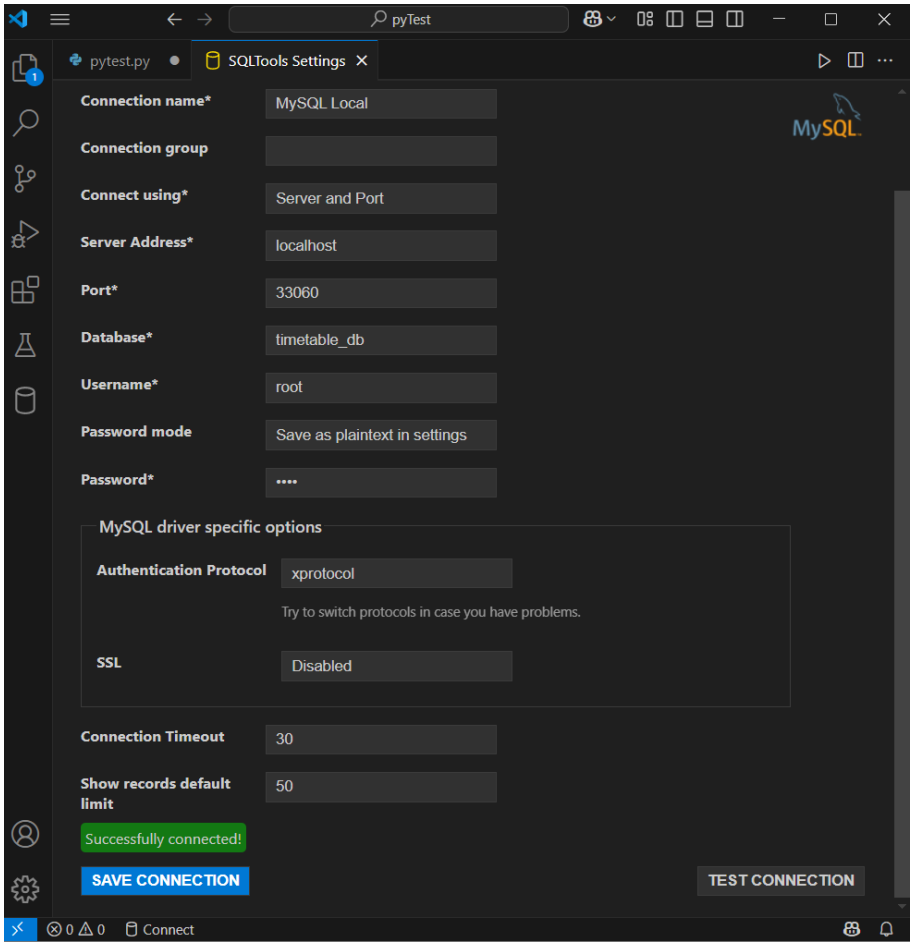

Figure 5.3.29 Visual Studio Code SQLTools Configuration

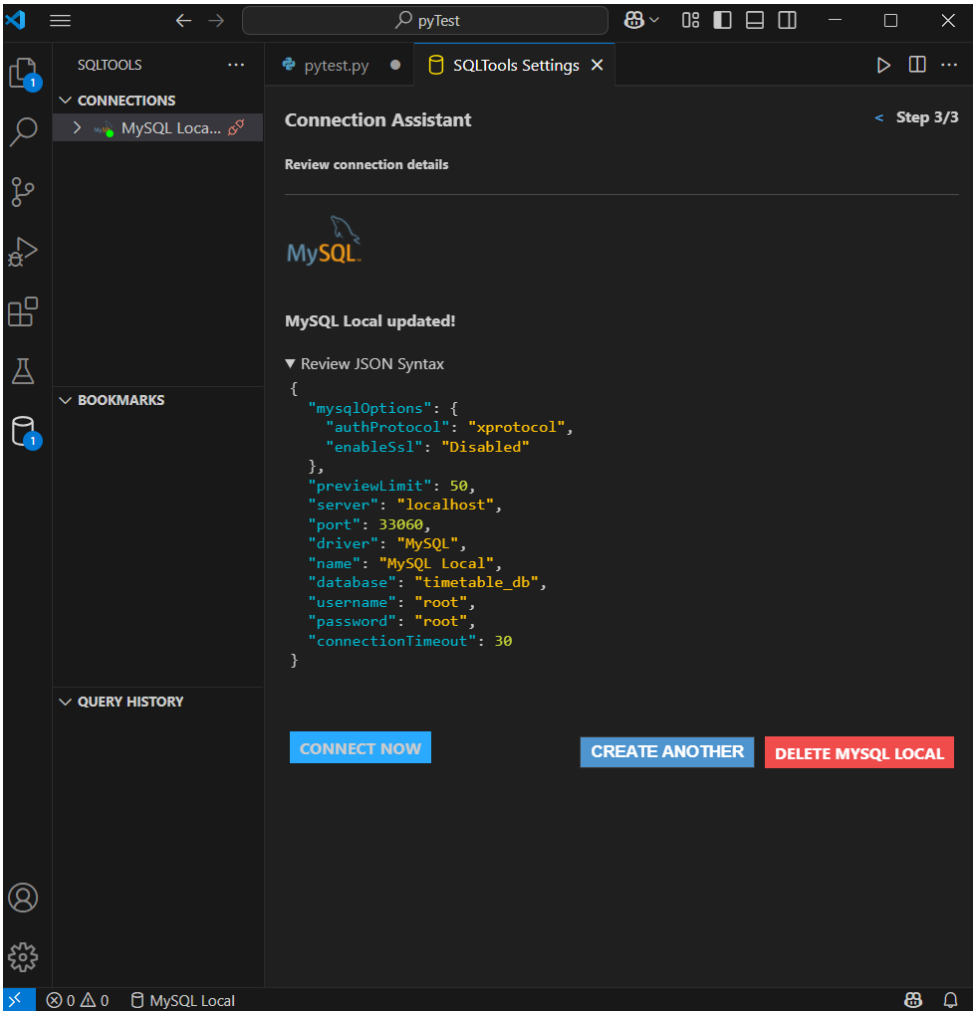

Figure 5.3.30 Visual Studio Code SQLTools Connection Completed

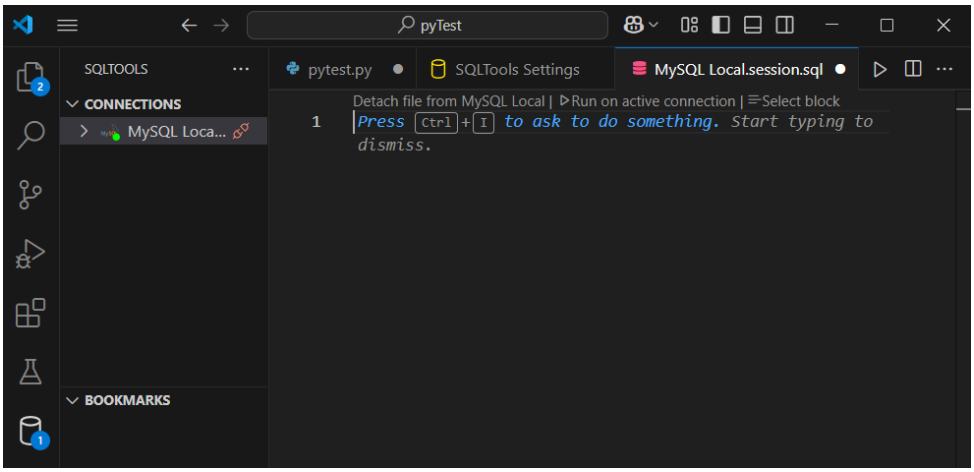

Figure 5.3.31 Visual Studio Code MySQL Successful Connection

To connect to the database in Python, the MySQL Connector/Python is required. Figure 5.3.32 shows the installation of the mysql-connector-python in Command Prompt (Run as Administrator).

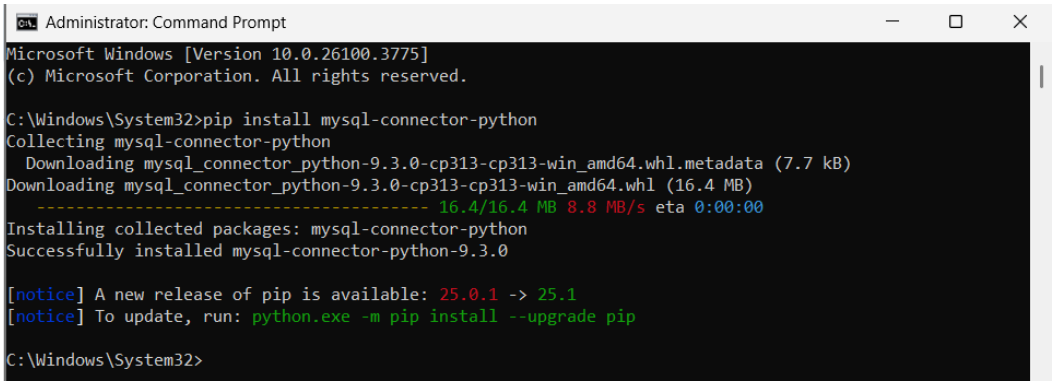

Figure 5.3.32 Visual Studio Code MySQL Connector/Python Installation

Figure 5.3.33 shows that it has successfully connected to the database.

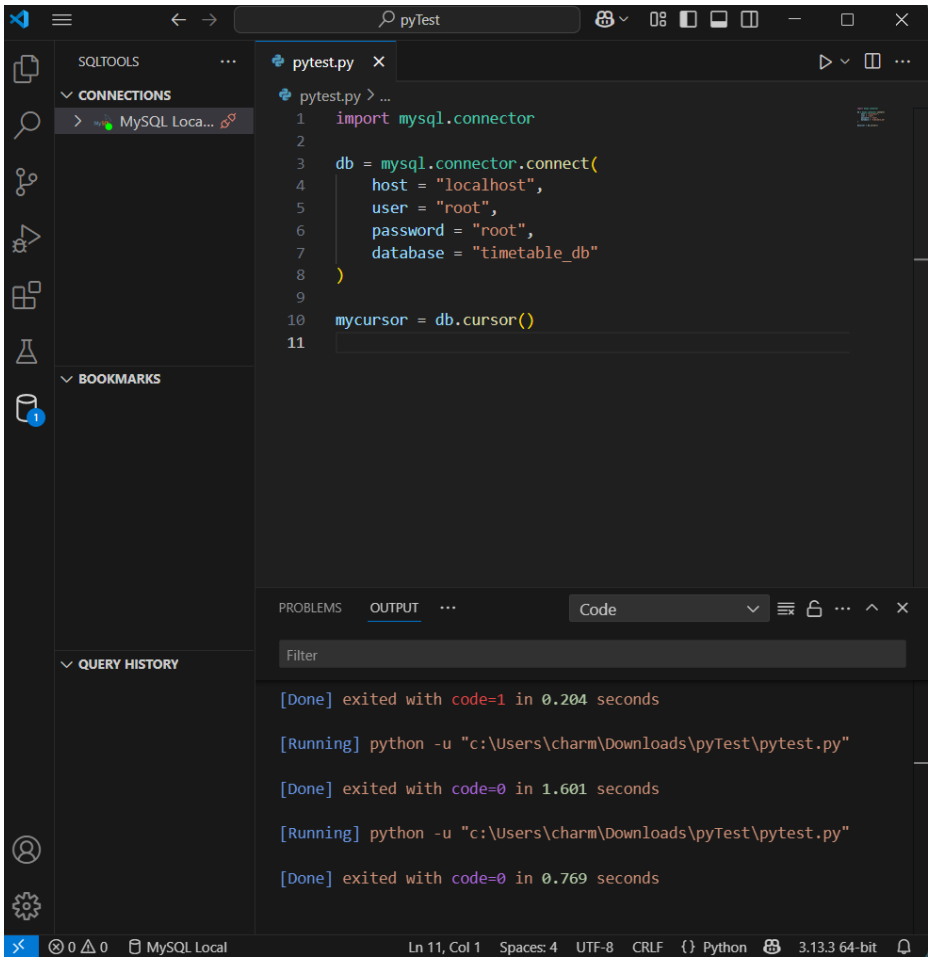

Figure 5.3.33 Visual Studio Code MySQL Connector/Python Successful Connection

CHAPTER 6

System Evaluation And Discussion

6.1 System Testing and Performance Metrics

In this section, system testing will be performed to evaluate the performance and correctness of the system by referring to the test cases defined in Chapter 3.

6.1.1 System Testing Setup

type	groupID	groupName	programName	studyYear	studentNum	courseCodes
student_group	25001	CSY1S2	Computer Science	1	200	UBMM1011,UCCD1004,UCCD1143,UCCD2003,UCCM1153,UCCN1004
student_group	25002	CSY1S3	Computer Science	1	200	UCCD1024,UCCD1203,UCCD2003,UCCM1353,UCCM1363,MPU3152/MPU3192
student_group	24001	CSY2S2	Computer Science	2	200	UCCD2044,UCCD2083,UCCD2103,UCCD2303,UCCN2243,MPU34012/MPU34022/MPU34072/MPU34082/MPU34102/MPU34132/MPU34142/MPU34152/MPU34162/MPU34182
student_group	24002	CSY2S3	Computer Science	2	200	UCCD2003,UCCD2103,UCCD2213,UCCD2303,UCCN2243
student_group	23001	CSY3S1	Computer Science	3	200	UCCD1213,UCCD3023,UCCD3113,UCCD3243,UDPS1043,UALE1083/UAMG1043,UCCC3073/UCCD3223
student_group	23002	CSY3S2	Computer Science	3	200	MPU33013,UCCD1213,UCCD3023,UCCD3113,UCCD3243,UDPS1043,UCCC3073/UCCD3223
student_group	23003	CSY3S3	Computer Science	3	200	UCCD1213,UCCD3023,UCCD3113,UCCD3243,UDPS1043,UALE1083/UAMG1043,UCCC3073/UCCD3223
student_group	24003	CNY2S1	Communications & Networking	2	150	UCCD2043,UCCD2044,UCCD2103,UCCN2243
student_group	23004	IAY3S2	Information Systems Engineering	3	150	UCCD3053,UCCD3223
student_group	23005	IAY3S3	Information Systems Engineering	3	150	UCCD3053

Figure 6.1.1.1 Test Plan Data for Student Groups

type	lecID	lecName	courseCode
lecturer	10001	Dr Tee Chee Wee	UBMM1011
lecturer	10002	Dr Tan Joi San	UCCD1004
lecturer	10003	Ts Dr Phan Koo Yuen	UCCD1004
lecturer	10004	Dr Jasmina Khaw Yen Min	UCCD1004
lecturer	10005	Dr Kh'ng Xin Yi	UCCD1004
lecturer	10006	Prof Dr Leung Kar Hang	UCCD1004
lecturer	10007	Dr Ahmad Hakimi Bin Ahman Sa'ahiry	UCCD1004
lecturer	10008	Ts Dr Lim Seng Poh	UCCD1143
lecturer	10009	Dr Chai Tong Yuen	UCCD1143
lecturer	10010	Ts Dr Ku Chin Soon	UCCD2003
lecturer	10011	Ts Dr Mogana a/p Vadiveloo	UCCD2003
lecturer	10012	Dr Tahanya Bashar M. A.	UCCD2003
lecturer	10013	Cik Puteri Nursyawati Binti Azzuri	UCCD2003,MPU34182
lecturer	10014	Dr Nur Balqishanis Binti Zainal Abidin	UCCM1153
lecturer	10015	Dr Lem Kong Hoong	UCCM1153
lecturer	10016	Dr Aun Yichiet	UCCN1004
lecturer	10017	Ms Tan Lyk Yin	UCCN1004
lecturer	10018	Ts Dr Ooi Chek Yee	UCCN1004
lecturer	10019	Dr Muhammad Syaiful Amri Bin Suhaimi	UCCN1004
lecturer	10020	Ts Dr Chang Jing Jing	UCCN1004
lecturer	10021	Dr Fityanul Akhyar	UCCN1004
lecturer	10022	Dr Abdulrahman Aminu Ghali	UCCN1004,UCCN2243
lecturer	10023	Dr Rohani binti Bakar	UCCN1004
lecturer	10024	Dr Norliana Binti Muslim	UCCN1004
lecturer	10025	Dr Rahman Sadli	UCCN1004
lecturer	10026	Dr Teoh Shen Khang	UCCN1004
lecturer	10027	Puan Nor 'Afifah Binti Sabri	UCCN1004
lecturer	10028	Dr Farina Saffa Binti Mohamad Samsamnun	UCCN1004,UCCN2243
lecturer	10029	Ts Yong Tien Fui	UCCD2043
lecturer	10030	Dr Zurida Binti Ishak	UCCD2043
lecturer	10031	Dr Sayed Admad Zikri Bin Sayed Aluwee	UCCD2044
lecturer	10032	Ts Dr Chai Meei Tyng	UCCD2044
lecturer	10033	Mr Luke Lee Chee Chien	UCCD2044
lecturer	10034	Ms Tseu Kwan Lee	UCCD2044
lecturer	10035	Dr Ng Hui Fuang	UCCD2044
lecturer	10036	Ts Wong Chee Siang	UCCD2103
lecturer	10037	Mr Sor Kean Vee	UCCD2103
lecturer	10038	Encik Ahmad Zaffry Hadi Bin Mohd Juffry	UCCD2083,UCCD3113
lecturer	10039	Ts Dr Gan Ming Lee	UCCN2243
lecturer	10040	Dr Adeb Alid Mohammed Ahmed Al-Samet	UCCN2243
lecturer	10041	Ms Oh Zi Xin	UCCN2243
lecturer	10042	Dr Nadeem Muhammad Waqas	UCCN2243
lecturer	10043	Mr Lee Kim Hoe @ Farhan Lee Bin Abdullah	UCCD3053
lecturer	10044	Mr Tan Chiang Kang	UCCD3223
lecturer	10045	Mr Tou Jing Yi	UCCD3223
lecturer	10046	Puan Syazwani Binti Yahya	UCCD3223
lecturer	10047	Cik Nur Athirah Nabila Binti Mohd Idros	UCCD3023
lecturer	10048	Ts Dr Mailasan a/l Jayakrishnan	UCCD3023
lecturer	10049	Dr Yiew Thian Hee	MPU33013
lecturer	10050	Mr Thurai Murugan a/l Nathan	MPU33013
lecturer	10051	Dr Kiran Adnan	UCCD3023
lecturer	10052	Ts Soong Hoong Cheng	UCCD3113
lecturer	10053	Ts Dr Cheng Wai Khuen	UCCD3113
lecturer	10054	Dr Ng Peh Sang	UDPS1043
lecturer	10055	Ms Aruna Raj a/p Devarajoo	UAMG1043
lecturer	10056	Ms Salomi a/p Simon	UAMG1043
lecturer	10057	Puan Liana binti Mat Nayan	UAMG1043
lecturer	10058	Puan Nor Ez-Zatul Hanani binti Mohamed Rosil	UAMG1043
lecturer	10059	Ts Dr Tong Dong Ling	UCCC3073
lecturer	10060	Mr Tan Chiang Kang	UCCD3223
lecturer	10062	Puan Syazwani Binti Yahya	UCCD3223
lecturer	10063	Puan Ayu Rita Binti Mohamad	UALE1083
lecturer	10064	Ms Jenifer Ann a/p Felix Leo	UCCD2213
lecturer	10065	Ts Dr Khor Siak Wang	UCCD1213
lecturer	10066	Ts Dr Lim Ean Heng	UCCD1213
lecturer	10067	Ms Kwang Wai Ching	UCCD1213
lecturer	10068	Ts Tan Teik Boon	UCCD3064
lecturer	10069	Ts Dr Tan Hung Khoon	UCCD3074
lecturer	10070	Ts Dr Saw Seow Hui	UCCD3084
lecturer	10071	Ms Mah Siew Huei	MPU34012,MPU34072
lecturer	10072	Ms Alison Yoon	MPU34022
lecturer	10073	Ms Erica Chua Ning Jia	MPU34082,MPU34162
lecturer	10074	Mr Fong Kah Hoong	MPU34102,MPU34132
lecturer	10075	Mr Kong Hoi Yoon	MPU34142,MPU34162
lecturer	10076	Puan Sarah Binti Shamshul Anwar	MPU3152
lecturer	10077	Ms Yuvashini a/p Salvamani	MPU3192

Figure 6.1.1.2 Test Plan Data for Lecturers

type	courseCode	courseName	lecNum	tutorialNur	pracNum	creditHour	isElective	isCore
course	UCCD1004	PROGRAMMING CONCEPTS AND PRACTICES	2	0	7	4	0	1
course	UCCD1143	PROBABILITY AND STATISTICS FOR COMPUTING	2	5	0	3	0	1
course	UCCD2003	OBJECT-ORIENTED SYSTEMS ANALYSIS AND DESIGN	2	7	0	3	0	1
course	UCCM1153	INTRODUCTION TO CALCULUS AND APPLICATIONS	2	7	0	4	0	1
course	UCCD2303	DATABASE TECHNOLOGY	2	0	6	4	0	1
course	UCCN1004	DATA COMMUNICATIONS AND NETWORKING	2	8	0	3	0	1
course	UBMM1011	SUN ZI'S ART OF WAR AND BUSINESS STRATEGIES	1	0	0	1	0	1
course	UCCM1353	BASIC ALGEBRA	2	10	0	4	0	1
course	UCCM1363	DISCRETE MATHEMATICS	2	10	0	4	0	1
course	UCCD2043	INFORMATION TECHNOLOGY PROJECT MANAGEMENT	2	6	0	4	0	1
course	UCCD2044	OBJECT-ORIENTED PROGRAMMING PRACTICES	2	0	10	4	0	1
course	UCCD1203	DATABSE DEVELOPMENT AND APPLICATIONS	1	0	9	3	0	1
course	UCCD1024	DATA STRUCTURE AND ALGORITHMIC PROBLEM SOLVING	2	0	8	4	0	1
course	UCCD2103	OPERATING SYSTEMS	2	5	0	4	0	1
course	UCCN2243	INTERNETWORKING PRINCIPLE AND PRACTICES	2	0	13	4	0	1
course	UCCD2213	SOFTWARE ENGINEERING PRINCIPLES	1	8	0	3	0	0
course	UCCD3053	INFORMATION TECHNOLOGY PROFESSIONAL ETHICS	2	4	0	3	0	1
course	UCCD3223	MOBILE APPLICATIONS DEVELOPMENT	1	0	6	3	1	0
course	UCCD1213	FUNDAMENTALS OF DIGITAL MEDIA TECHNOLOGY	3	1	0	3	1	0
course	UCCD3023	DIGITAL ENTREPRENEURSHIP	3	1	0	3	1	0
course	UCCD3113	DISTRIBUTED COMPUTER SYSTEMS	3	1	0	3	1	0
course	UCCD3243	SERVER-SIDE WEB APPLICATIONS DEVELOPMENT	2	0	2	3	1	0
course	UDPS1043	INTRODUCTION TO OPERATIONS RESEARCH	3	1	0	3	1	0
course	UALE1083	BASIC PROFESSIONAL WRITING	2	1	0	3	1	0
course	UAMG1043	INTERPERSONAL COMMUNICATION	2	1	0	3	1	0
course	UCCC3073	DATA SCIENCE	2	0	2	3	1	0
course	MPU33013	MALAYSIAN ECONOMY	2	1	0	3	1	0
course	UCCD3064	SOFTWARE TESTING	3	0	2	4	1	0
course	UCCD3074	DEEP LEARNING FOR DATA SCIENCE	3	0	2	4	1	0
course	UCCD3084	GRAPHICS PROGRAMMING FOR EXTENDED REALITY	3	0	2	4	1	0
course	UCCD2083	CLOUD COMPUTING AND SERVICES	1	3	0	3	1	0
course	MPU34012	SOCIAL ENTREPRENEURSHIP PROJECT	1	0	0	2	1	0
course	MPU34022	ARTS AND CULTURAL PERFORMANCE	1	0	0	2	1	0
course	MPU34072	ART, CRAFT, AND DESIGN	1	0	0	2	1	0
course	MPU34082	ORAL COMMUNICATION	1	0	0	2	1	0
course	MPU34102	MANAGING PERSONAL FINANCE	1	0	0	2	1	0
course	MPU34132	MANAGEMENT OF SPORTS ACTIVITY	1	0	0	2	1	0
course	MPU34142	CRITICAL THINKING, CREATIVE THINKING AND PROBLEM SOLVING	1	0	0	2	1	0
course	MPU34152	LEADERSHIP AND TEAMBUILDING	1	0	0	2	1	0
course	MPU34162	BUSINESS PLAN WRITING & PREPARATION	1	0	0	2	1	0
course	MPU34182	MASSIVE OPEN ONLINE COURSE (MOOC)	1	0	0	2	1	0
course	MPU3152	PENGHAYATAN ETIKA DAN PERADABAN (FOR LOCAL STUDENTS)	3	0	0	2	1	0
course	MPU3192	PHILOSOPHY AND CURRENT ISSUES (FOR INTERNATIONAL STUDENT	1	0	0	2	1	0

Figure 6.1.1.3 Test Plan Data for Courses

type	venueID	venueName	capacity	buildingName	latitude	longitude	venueType
venue	100011	LDK1	200	Block L	4.34172886	101.14005	L
venue	100012	LDK2	200	Block L	4.34172886	101.14005	L
venue	100013	LDK3	300	Block L	4.34172886	101.14005	L
venue	100014	LDK4	200	Block L	4.34172886	101.14005	L
venue	100015	LDK5	200	Block L	4.34172886	101.14005	L
venue	100016	EDK1	300	Block E	4.3386569	101.143654	L
venue	100017	DDK1	300	Block D	4.33799076	101.143837	L
venue	100018	N001	30	Block N	4.33846232	101.136894	T
venue	100019	N002	30	Block N	4.33846232	101.136894	T
venue	100020	N003	30	Block N	4.33846232	101.136894	T
venue	100021	N004	30	Block N	4.33846232	101.136894	T
venue	100022	N005	60	Block N	4.33846232	101.136894	T
venue	100023	N006	60	Block N	4.33846232	101.136894	T
venue	100024	N007	25	Block N	4.33846232	101.136894	T
venue	100025	N008	30	Block N	4.33846232	101.136894	P
venue	100026	N009	30	Block N	4.33846232	101.136894	P
venue	100027	N010A	20	Block N	4.33846232	101.136894	P
venue	100028	N010B	20	Block N	4.33846232	101.136894	P
venue	100029	N101	25	Block N	4.33846232	101.136894	T
venue	100030	N102	25	Block N	4.33846232	101.136894	T
venue	100031	N103	25	Block N	4.33846232	101.136894	T
venue	100032	N104	25	Block N	4.33846232	101.136894	T
venue	100033	N105	25	Block N	4.33846232	101.136894	T
venue	100034	N106	25	Block N	4.33846232	101.136894	T
venue	100035	N107	20	Block N	4.33846232	101.136894	T
venue	100036	N108	30	Block N	4.33846232	101.136894	P
venue	100037	N109	30	Block N	4.33846232	101.136894	P
venue	100038	N110A	20	Block N	4.33846232	101.136894	P
venue	100039	N110B	20	Block N	4.33846232	101.136894	P

Figure 6.1.1.4 Test Plan Data for Venues

Input data: Figure 6.1.1.1, Figure 6.1.1.2, Figure 6.1.1.3, and Figure 6.1.1.4. The figures above show the data that will be stored in the database.

6.1.2 System Testing Process

Once the data has been stored in the database, it will be retrieved by the system during execution. After the user specifies the required constraints, the system proceeds to generate the timetable. The effectiveness of the test plans is then evaluated by examining the timetable produced.

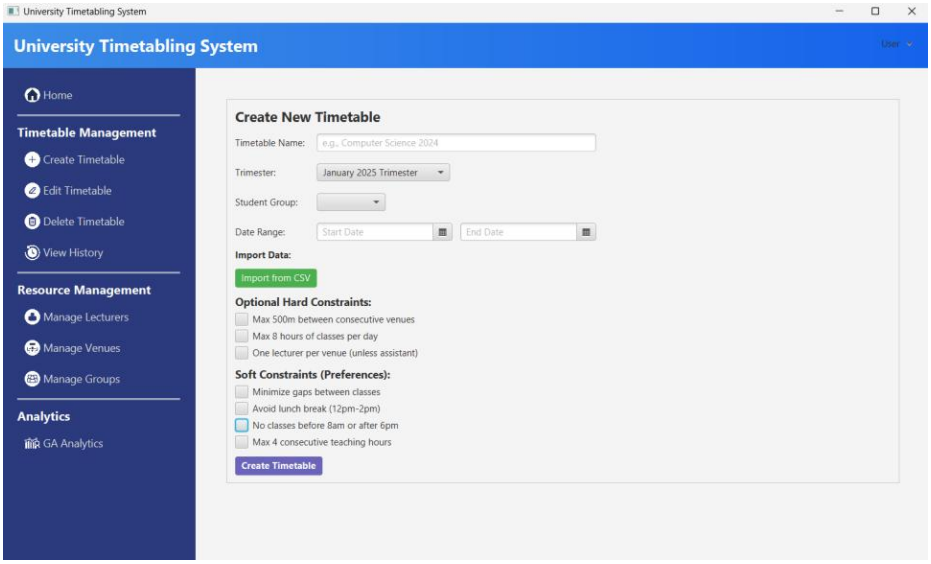

Figure 6.1.2.1 Hard and Soft Constraint Selection

The user may check all or some of the constraints to be included in the timetable. Then click on 'Create Timetable'. The timetable generated can be seen in table format or timeslot format. It can also be exported in CSV file format.

6.2 System Testing for Hard Constraints Defined

6.2.1 Lecturers can only teach one class at the same time

In the lecturers' timetable generated, the lecturer is allocated to only one class in one timeslot.

Table 6.2.1.1 Test Case of 'Lecturers can only teach one class at the same time'

Input Data	Description	Expected Result	Actual Result	Status
Dataset 1	This test case ensures that each lecturer can only teach one class at the same time.	In the lecturers' timetable generated, the lecturer is allocated to only one class in one timeslot.	The lecturers are allocated to teach one class in one timeslot.	Pass

Day	Time	Course	Lecturer	Venue	Group	Session Type
Tuesday	16:00-17:00	UCCD1213	Ms Kwang Wai Ching	N103	CSY3S1	tutorial
Monday	08:00-10:00	UCCD3243	Ts Dr Cheng Wai Khuen	LDK3	CSY3S1	lecture
Monday	14:00-16:00	MPU3152	Ts Dr Lim Ean Heng	LDK2	CSY3S1	lecture
Tuesday	12:00-14:00	UDPS1043	Ms Kwang Wai Ching	LDK5	CSY3S1	lecture
Wednesday	08:00-10:00	MPU3152	Cik Nur Athirah Nabila Binti Mohd Idros	LDK5	CSY3S1	lecture
Wednesday	12:00-14:00	UCCD3023	Ms Kwang Wai Ching	DDK1	CSY3S1	lecture
Wednesday	14:00-16:00	UCCD1213	Dr Ng Peh Sang	DDK1	CSY3S1	lecture
Thursday	08:00-10:00	UCCD3113	Ts Dr Lim Ean Heng	LDK3	CSY3S1	lecture
Thursday	10:00-12:00	UCCD3023	Ts Dr Khor Siak Wang	EDK1	CSY3S1	lecture
Thursday	14:00-16:00	MPU34132	Ts Dr Cheng Wai Khuen	DDK1	CSY3S1	lecture
Thursday	16:00-18:00	UDPS1043	Ts Soong Hoong Cheng	LDK1	CSY3S1	lecture
Friday	08:00-10:00	UCCD3113	Ts Dr Mailasan a/l Jayakrishnan	LDK3	CSY3S1	lecture
Friday	12:00-14:00	MPU34102	Ts Dr Mailasan a/l Jayakrishnan	DDK1	CSY3S1	lecture
Friday	14:00-16:00	MPU34182	Dr Kiran Adnan	LDK5	CSY3S1	lecture
Monday	12:00-14:00	UCCD3223	Encik Ahmad Zaffry Hadi Bin Mohd Juffry	N108	CSY3S1	practical
Monday	16:00-18:00	UCCC3073	Dr Kiran Adnan	N108	CSY3S1	practical
Tuesday	10:00-12:00	UCCD3243	Dr Ng Peh Sang	N010A	CSY3S1	practical
Wednesday	16:00-18:00	UCCD3243	Ts Dr Mailasan a/l Jayakrishnan	N109	CSY3S1	practical
Friday	10:00-12:00	UCCD3223	Encik Ahmad Zaffry Hadi Bin Mohd Juffry	N110A	CSY3S1	practical

Figure 6.2.1.1 Proof of 'Lecturers can only teach one class at the same time'

From Figure 6.2.1.1 above, lecturers are restricted to teaching only one class at a given time. The timetable shows that no lecturer has been assigned to multiple sessions in the same timeslot, which satisfies the teaching constraint. This guarantees that a lecturer is not double-booked and is only required to deliver one class at a time. With both student and lecturer constraints fulfilled, the test case is considered successful.

6.2.2 Two courses cannot be scheduled in the same venue at the same time

Table 6.2.2.1 Test Case of 'Two courses cannot be scheduled in the same venue at the same time'

Input Data	Description	Expected Result	Actual Result	Status
Dataset 1	This test case ensures that two courses cannot be scheduled in the same venue at the same time.	In the timetable generated, the courses cannot be scheduled in the same venue at the same time.	The courses were not scheduled in the same venue at the same time.	Pass

Day	Time	Course	Lecturer	Venue	Group	Session Type
Tuesday	16:00-17:00	UCCD1213	Ms Kwang Wai Ching	N103	CSY3S1	tutorial
Monday	08:00-10:00	UCCD3243	Ts Dr Cheng Wai Khuen	LDK3	CSY3S1	lecture
Monday	14:00-16:00	MPU3152	Ts Dr Lim Ean Heng	LDK2	CSY3S1	lecture
Tuesday	12:00-14:00	UDPS1043	Ms Kwang Wai Ching	LDK5	CSY3S1	lecture
Wednesday	08:00-10:00	MPU3152	Cik Nur Athirah Nabila Binti Mohd Idros	LDK5	CSY3S1	lecture
Wednesday	12:00-14:00	UCCD3023	Ms Kwang Wai Ching	DDK1	CSY3S1	lecture
Wednesday	14:00-16:00	UCCD1213	Dr Ng Peh Sang	DDK1	CSY3S1	lecture
Thursday	08:00-10:00	UCCD3113	Ts Dr Lim Ean Heng	LDK3	CSY3S1	lecture
Thursday	10:00-12:00	UCCD3023	Ts Dr Khor Siak Wang	EDK1	CSY3S1	lecture
Thursday	14:00-16:00	MPU34132	Ts Dr Cheng Wai Khuen	DDK1	CSY3S1	lecture
Thursday	16:00-18:00	UDPS1043	Ts Soong Hoong Cheng	LDK1	CSY3S1	lecture
Friday	08:00-10:00	UCCD3113	Ts Dr Mailasan a/l Jayakrishnan	LDK3	CSY3S1	lecture
Friday	12:00-14:00	MPU34102	Ts Dr Mailasan a/l Jayakrishnan	DDK1	CSY3S1	lecture
Friday	14:00-16:00	MPU34182	Dr Kiran Adnan	LDK5	CSY3S1	lecture
Monday	12:00-14:00	UCCD3223	Encik Ahmad Zaffry Hadi Bin Mohd Juffry	N108	CSY3S1	practical
Monday	16:00-18:00	UCCC3073	Dr Kiran Adnan	N108	CSY3S1	practical
Tuesday	10:00-12:00	UCCD3243	Dr Ng Peh Sang	N010A	CSY3S1	practical
Wednesday	16:00-18:00	UCCD3243	Ts Dr Mailasan a/l Jayakrishnan	N109	CSY3S1	practical
Friday	10:00-12:00	UCCD3223	Encik Ahmad Zaffry Hadi Bin Mohd Juffry	N110A	CSY3S1	practical

Figure 6.2.2.1 Proof of 'Two courses cannot be scheduled in the same venue at the same time'

From Figure 6.2.2.1 above, the timetable displays all scheduled sessions for the student group, including information such as course code, lecturer, venue, and session type. The focus of this test case is to ensure that two different courses are not allocated

to the same venue at the same timeslot. After observing the timetable, it can be confirmed that every venue is assigned to only one course at any given time.

6.2.3 A student can only attend one class at the same time

Table 6.2.3.1 Test Case of 'A student can only attend one class at the same time'

Input Data	Description	Expected Result	Actual Result	Status
Dataset 1	This test case ensures that each student can only attend one class at the same time.	In the timetable generated, A student can only attend one class at the same time.	A student can only attend one class at the same time	Pass

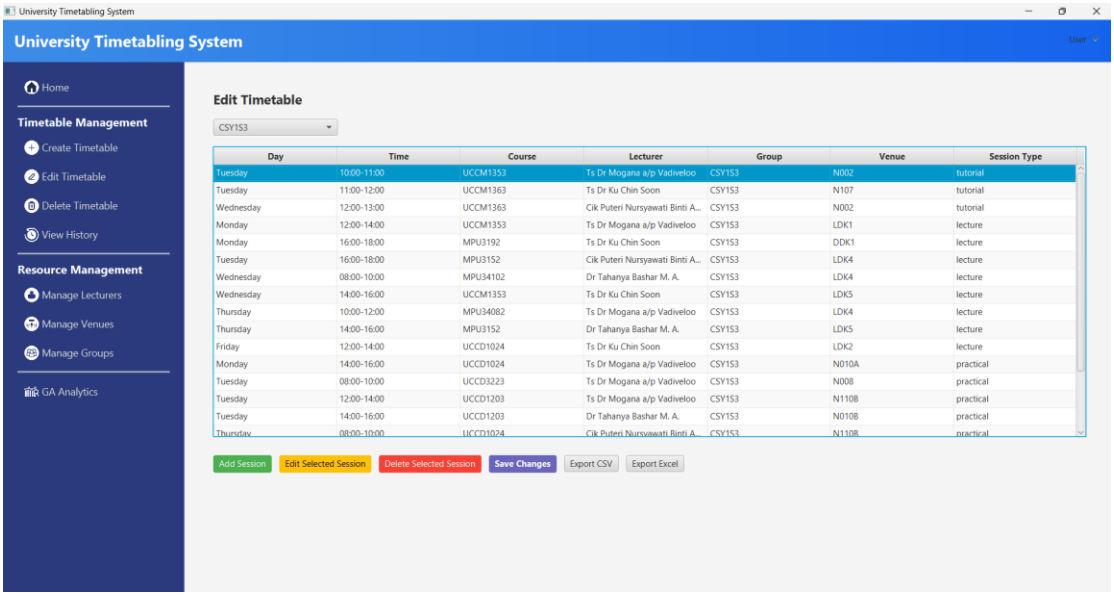

Figure 6.2.3.1 Proof of 'A student can only attend one class at the same time'

From Figure 6.2.2.1 above, the timetable displayed belongs to the predefined student group CSY1S3. Since the timetable is generated for a specific group, the system ensures that no two classes overlap in the same timeslot. This means that the student group is only required to attend one session at a time. When observing the timetable, all lecture, tutorial, and practical sessions are allocated at separate times, and there are no clashes between courses.

This confirms that the rule "a student can only attend one class at the same time" has been satisfied. The system successfully prevents scheduling conflicts within the same group, and the test case is therefore considered fulfilled.

6.2.4 The number of students cannot exceed the seating capacity of the assigned venue

Table 6.2.4.1 Test Case of 'The number of students cannot exceed the seating capacity of the assigned venue'

Input Data	Description	Expected Result	Actual Result	Status
Dataset 1	This test case ensures that the number of students cannot exceed the seating capacity of the assigned venue.	In the timetable generated, the number of students cannot exceed the seating capacity of the assigned venue.	The number of students did not exceed the seating capacity of the assigned venue	Pass

Day	Time	Course	Lecturer	Venue	Group	Session Type
Tuesday	16:00-17:00	UCCD1213	Ms Kwang Wai Ching	N103	CSY3S1	tutorial
Monday	08:00-10:00	UCCD3243	Ts Dr Cheng Wai Khuen	LDK3	CSY3S1	lecture
Monday	14:00-16:00	MPU3152	Ts Dr Lim Ean Heng	LDK2	CSY3S1	lecture
Tuesday	12:00-14:00	UDPS1043	Ms Kwang Wai Ching	LDK5	CSY3S1	lecture
Wednesday	08:00-10:00	MPU3152	Cik Nur Athirah Nabila Binti Mohd Idros	LDK5	CSY3S1	lecture
Wednesday	12:00-14:00	UCCD3023	Ms Kwang Wai Ching	DDK1	CSY3S1	lecture
Wednesday	14:00-16:00	UCCD1213	Dr Ng Peh Sang	DDK1	CSY3S1	lecture
Thursday	08:00-10:00	UCCD3113	Ts Dr Lim Ean Heng	LDK3	CSY3S1	lecture
Thursday	10:00-12:00	UCCD3023	Ts Dr Khor Siak Wang	EDK1	CSY3S1	lecture
Thursday	14:00-16:00	MPU34132	Ts Dr Cheng Wai Khuen	DDK1	CSY3S1	lecture
Thursday	16:00-18:00	UDPS1043	Ts Soong Hoong Cheng	LDK1	CSY3S1	lecture
Friday	08:00-10:00	UCCD3113	Ts Dr Mailasan a/l Jayakrishnan	LDK3	CSY3S1	lecture
Friday	12:00-14:00	MPU34102	Ts Dr Mailasan a/l Jayakrishnan	DDK1	CSY3S1	lecture
Friday	14:00-16:00	MPU34182	Dr Kiran Adnan	LDK5	CSY3S1	lecture
Monday	12:00-14:00	UCCD3223	Encik Ahmad Zaffry Hadi Bin Mohd Juffry	N108	CSY3S1	practical
Monday	16:00-18:00	UCCC3073	Dr Kiran Adnan	N108	CSY3S1	practical
Tuesday	10:00-12:00	UCCD3243	Dr Ng Peh Sang	N010A	CSY3S1	practical
Wednesday	16:00-18:00	UCCD3243	Ts Dr Mailasan a/l Jayakrishnan	N109	CSY3S1	practical
Friday	10:00-12:00	UCCD3223	Encik Ahmad Zaffry Hadi Bin Mohd Juffry	N110A	CSY3S1	practical

Figure 6.2.4.1 Proof of 'The number of students cannot exceed the seating capacity of the assigned venue'

From Figure 6.2.4.1, the timetable assigns sessions for the student group CSY3S1, which has an estimated batch size of approximately 200 students. The venues allocated for this group, such as LDK halls and larger classrooms, all have seating capacities that meet or exceed this requirement. For example, lecture halls like LDK3, LDK4, and LDK5 are designed to accommodate around 200 students or more, making them suitable for lectures involving the entire batch.

By ensuring that the number of students does not exceed the seating capacity of the assigned venue, the system successfully enforces the venue capacity constraint. This validation confirms that all scheduled sessions for CSY3S1 can be conducted without exceeding the available room size, and the test case is therefore considered successful.

6.2.5 Courses that require specific room types should be scheduled in appropriate facilities

Table 6.2.5.1 Test Case of 'Courses that require specific room types should be scheduled in appropriate facilities'

Input Data	Description	Expected Result	Actual Result	Status

Dataset 1	This test case ensures that courses that require specific room types are scheduled in appropriate facilities.	In the timetable generated, courses that require specific room types should be scheduled in appropriate facilities.	Courses that require specific room types are scheduled in appropriate facilities.	Pass

Day	Time	Course	Lecturer	Venue	Group	Session Type
Tuesday	16:00-17:00	UCCD1213	Ms Kwang Wai Ching	N103	CSY3S1	tutorial
Monday	08:00-10:00	UCCD3243	Ts Dr Cheng Wai Khuen	LDK3	CSY3S1	lecture
Monday	14:00-16:00	MPU3152	Ts Dr Lim Ean Heng	LDK2	CSY3S1	lecture
Tuesday	12:00-14:00	UDPS1043	Ms Kwang Wai Ching	LDK5	CSY3S1	lecture
Wednesday	08:00-10:00	MPU3152	Cik Nur Athirah Nabila Binti Mohd Idros	LDK5	CSY3S1	lecture
Wednesday	12:00-14:00	UCCD3023	Ms Kwang Wai Ching	DDK1	CSY3S1	lecture
Wednesday	14:00-16:00	UCCD1213	Dr Ng Peh Sang	DDK1	CSY3S1	lecture
Thursday	08:00-10:00	UCCD3113	Ts Dr Lim Ean Heng	LDK3	CSY3S1	lecture
Thursday	10:00-12:00	UCCD3023	Ts Dr Khor Siak Wang	EDK1	CSY3S1	lecture
Thursday	14:00-16:00	MPU34132	Ts Dr Cheng Wai Khuen	DDK1	CSY3S1	lecture
Thursday	16:00-18:00	UDPS1043	Ts Soong Hoong Cheng	LDK1	CSY3S1	lecture
Friday	08:00-10:00	UCCD3113	Ts Dr Mailasan a/l Jayakrishnan	LDK3	CSY3S1	lecture
Friday	12:00-14:00	MPU34102	Ts Dr Mailasan a/l Jayakrishnan	DDK1	CSY3S1	lecture
Friday	14:00-16:00	MPU34182	Dr Kiran Adnan	LDK5	CSY3S1	lecture
Monday	12:00-14:00	UCCD3223	Encik Ahmad Zaffry Hadi Bin Mohd Juffry	N108	CSY3S1	practical
Monday	16:00-18:00	UCCC3073	Dr Kiran Adnan	N108	CSY3S1	practical
Tuesday	10:00-12:00	UCCD3243	Dr Ng Peh Sang	N010A	CSY3S1	practical
Wednesday	16:00-18:00	UCCD3243	Ts Dr Mailasan a/l Jayakrishnan	N109	CSY3S1	practical
Friday	10:00-12:00	UCCD3223	Encik Ahmad Zaffry Hadi Bin Mohd Juffry	N110A	CSY3S1	practical

Figure 6.2.5.1 Proof of 'Courses that require specific room types should be scheduled in appropriate facilities'

From Figure 6.2.5.1, the timetable demonstrates that courses are scheduled in facilities that match their specific room type requirements. For example, lecture sessions are allocated to lecture halls such as LDK1, LDK3, and LDK5, which are designed to accommodate large student groups for lectures. Similarly, practical classes are assigned to laboratory rooms such as N108, N109, and N110A/B, which provide the necessary facilities and equipment to support hands-on sessions. Tutorials, on the other hand, are placed in smaller tutorial rooms such as N103, which is suitable for smaller group-based learning.

6.2.6 Venues cannot be assigned to the same timeslot more than once

Table 6.2.6.1 Test Case of 'Venues cannot be assigned to the same timeslot more than once'

Input Data	Description	Expected Result	Actual Result	Status
Dataset 1	This test case ensures that venues are not assigned to the same timeslot more than once.	In the timetable generated, venues cannot be assigned to the same timeslot more than once.	Venues are not assigned to the same timeslot more than once.	Pass

Day	Time	Course	Lecturer	Venue	Group	Session Type
Tuesday	16:00-17:00	UCCD1213	Ms Kwang Wai Ching	N103	CSY3S1	tutorial
Monday	08:00-10:00	UCCD3243	Ts Dr Cheng Wai Khuen	LDK3	CSY3S1	lecture
Monday	14:00-16:00	MPU3152	Ts Dr Lim Ean Heng	LDK2	CSY3S1	lecture
Tuesday	12:00-14:00	UDPS1043	Ms Kwang Wai Ching	LDK5	CSY3S1	lecture
Wednesday	08:00-10:00	MPU3152	Cik Nur Athirah Nabila Binti Mohd Idros	LDK5	CSY3S1	lecture
Wednesday	12:00-14:00	UCCD3023	Ms Kwang Wai Ching	DDK1	CSY3S1	lecture
Wednesday	14:00-16:00	UCCD1213	Dr Ng Peh Sang	DDK1	CSY3S1	lecture
Thursday	08:00-10:00	UCCD3113	Ts Dr Lim Ean Heng	LDK3	CSY3S1	lecture
Thursday	10:00-12:00	UCCD3023	Ts Dr Khor Siak Wang	EDK1	CSY3S1	lecture
Thursday	14:00-16:00	MPU34132	Ts Dr Cheng Wai Khuen	DDK1	CSY3S1	lecture
Thursday	16:00-18:00	UDPS1043	Ts Soong Hoong Cheng	LDK1	CSY3S1	lecture
Friday	08:00-10:00	UCCD3113	Ts Dr Mailasan a/l Jayakrishnan	LDK3	CSY3S1	lecture
Friday	12:00-14:00	MPU34102	Ts Dr Mailasan a/l Jayakrishnan	DDK1	CSY3S1	lecture
Friday	14:00-16:00	MPU34182	Dr Kiran Adnan	LDK5	CSY3S1	lecture
Monday	12:00-14:00	UCCD3223	Encik Ahmad Zaffry Hadi Bin Mohd Juffry	N108	CSY3S1	practical
Monday	16:00-18:00	UCCC3073	Dr Kiran Adnan	N108	CSY3S1	practical
Tuesday	10:00-12:00	UCCD3243	Dr Ng Peh Sang	N010A	CSY3S1	practical
Wednesday	16:00-18:00	UCCD3243	Ts Dr Mailasan a/l Jayakrishnan	N109	CSY3S1	practical
Friday	10:00-12:00	UCCD3223	Encik Ahmad Zaffry Hadi Bin Mohd Juffry	N110A	CSY3S1	practical

Figure 6.2.6.1 Proof of 'Venues cannot be assigned to the same timeslot more than once'

From Figure 6.2.6.1, the timetable shows that each venue is only assigned to one course per timeslot. This means no two sessions share the same room at the same time. For example, lecture halls such as LDK3, LDK4, and LDK5 are allocated to different sessions across the week, but at no point are two courses scheduled in the same

hall during the same timeslot. Similarly, tutorial and practical rooms such as N002, N108, and N110A/B are also used exclusively within each timeslot without overlapping assignments.

This confirms that the system successfully enforces the rule that venues cannot be double-booked. By preventing multiple sessions from being scheduled in the same venue at once, the system ensures that teaching facilities are used efficiently and that logistical conflicts are avoided. Therefore, the test case is considered successful.

6.2.7 Total maximum hours for classes per day is 8 hours

Table 6.2.7.1 Test Case of 'Total maximum hours for classes per day is 8 hours'

Input Data	Description	Expected Result	Actual Result	Status
Dataset 1	This test case ensures that the total maximum hours for classes per day is 8 hours.	In the timetable generated, the total maximum hours for classes per day is 8 hours.	Total maximum hours for classes per day is 8 hours.	Pass

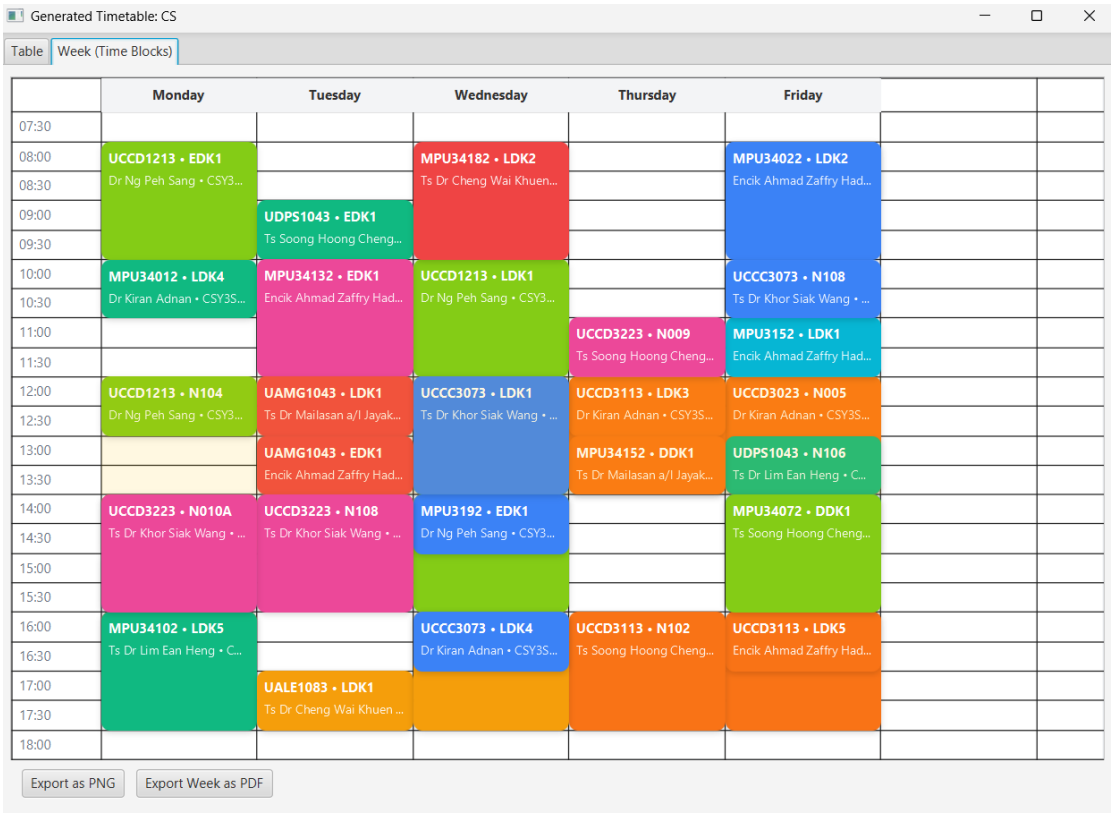

Figure 6.2.7.1 Proof of 'Total maximum hours for classes per day is 8 hours'

From Figure 6.2.7.1, the timetable includes both core and elective courses allocated across different days. Since elective courses are only taken by a subset of students, not every student will attend all of the sessions shown. This means that while the timetable visually appears to contain a large number of sessions in a single day, the actual load for each student is lower because elective classes vary by individual choice.

When calculating the maximum hours per day, the system ensures that even with electives included, no student exceeds the total daily limit of eight hours of scheduled classes. For example, a student who selects certain electives may have a longer day compared to others, but the system guarantees that the total number of class hours does not go beyond the allowed maximum.

6.2.8 A student cannot be assigned to two different venues in the same timeslot

Table 6.2.8.1 Test Case of 'A student cannot be assigned to two different venues in the same timeslot'

Input Data	Description	Expected Result	Actual Result	Status
Dataset 1	This test case ensures that students are not assigned to two different venues in the same timeslot.	In the timetable generated, a student cannot be assigned to two different venues in the same timeslot.	A student is not assigned to two different venues in the same timeslot	Pass

Day	Time	Course	Lecturer	Venue	Group	Session Type
Tuesday	16:00-17:00	UCCD1213	Ms Kwang Wai Ching	N103	CSY3S1	tutorial
Monday	08:00-10:00	UCCD3243	Ts Dr Cheng Wai Khuen	LDK3	CSY3S1	lecture
Monday	14:00-16:00	MPU3152	Ts Dr Lim Ean Heng	LDK2	CSY3S1	lecture
Tuesday	12:00-14:00	UDPS1043	Ms Kwang Wai Ching	LDK5	CSY3S1	lecture
Wednesday	08:00-10:00	MPU3152	Cik Nur Athirah Nabila Binti Mohd Idros	LDK5	CSY3S1	lecture
Wednesday	12:00-14:00	UCCD3023	Ms Kwang Wai Ching	DDK1	CSY3S1	lecture
Wednesday	14:00-16:00	UCCD1213	Dr Ng Peh Sang	DDK1	CSY3S1	lecture
Thursday	08:00-10:00	UCCD3113	Ts Dr Lim Ean Heng	LDK3	CSY3S1	lecture
Thursday	10:00-12:00	UCCD3023	Ts Dr Khor Siak Wang	EDK1	CSY3S1	lecture
Thursday	14:00-16:00	MPU34132	Ts Dr Cheng Wai Khuen	DDK1	CSY3S1	lecture
Thursday	16:00-18:00	UDPS1043	Ts Soong Hoong Cheng	LDK1	CSY3S1	lecture
Friday	08:00-10:00	UCCD3113	Ts Dr Mailasan a/l Jayakrishnan	LDK3	CSY3S1	lecture
Friday	12:00-14:00	MPU34102	Ts Dr Mailasan a/l Jayakrishnan	DDK1	CSY3S1	lecture
Friday	14:00-16:00	MPU34182	Dr Kiran Adnan	LDK5	CSY3S1	lecture
Monday	12:00-14:00	UCCD3223	Encik Ahmad Zaffry Hadi Bin Mohd Juffry	N108	CSY3S1	practical
Monday	16:00-18:00	UCCC3073	Dr Kiran Adnan	N108	CSY3S1	practical
Tuesday	10:00-12:00	UCCD3243	Dr Ng Peh Sang	N010A	CSY3S1	practical
Wednesday	16:00-18:00	UCCD3243	Ts Dr Mailasan a/l Jayakrishnan	N109	CSY3S1	practical
Friday	10:00-12:00	UCCD3223	Encik Ahmad Zaffry Hadi Bin Mohd Juffry	N110A	CSY3S1	practical

Figure 6.2.8.1 Proof of 'Venues cannot be assigned to the same timeslot more than once'

From Figure 6.2.8.1, the timetable was checked to ensure that no student is required to be in two different venues at the same timeslot. Since the timetable is generated for a predefined student group, all sessions are allocated sequentially without overlap. For example, when a class is assigned to LDK3 at 08:00–10:00, no other session for the same group is scheduled at that exact time in a different venue.

6.2.9 The travel distance between two venues of consecutive classes should not be more than 500 meters

Table 6.2.9.1 Test Case of 'The travel distance between two venues of consecutive classes should not be more than 500 meters'

Input Data	Description	Expected Result	Actual Result	Status
Dataset 1	This test case ensures that the travel distance between two venues of consecutive classes should not be more than 500 meters.	In the timetable generated, the travel distance between two venues of consecutive classes should not be more than 500 meters.	The travel distance between two venues of consecutive classes is not more than 500 meters.	Pass

This requirement ensures that no participant (student or lecturer), is scheduled for back-to-back classes that would require walking more than 500 meters between venues. The constraint is enforced as a hard rule in the genetic algorithm's fitness function.

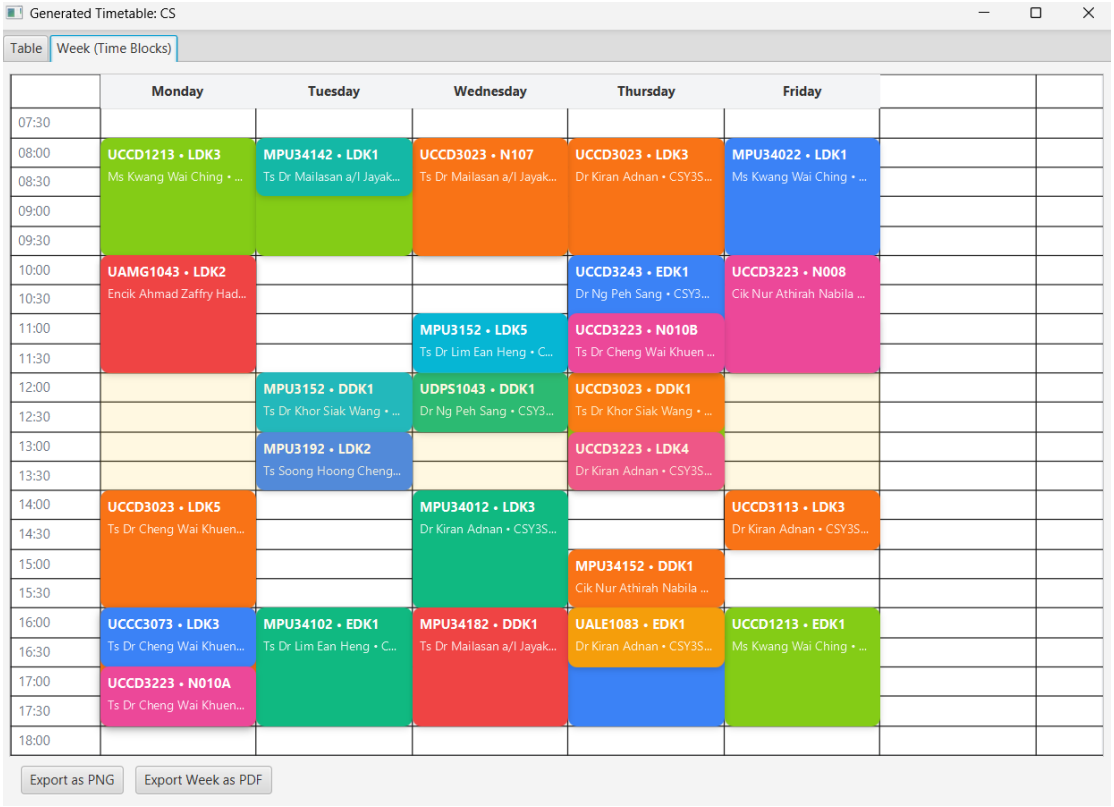

Figure 6.2.9.2 Proof of 'The travel distance between two venues of consecutive classes should not be more than 500 meters'

Figure 6.2.9.1 illustrates representative consecutive sessions where the computed Haversine distances between venues are annotated (for example, 310 m or 480 m), all within the 500 m limit. The timetable appears to show certain sessions scheduled consecutively in venues that may be farther than 500 meters apart. However, these cases occur because of elective courses, where different groups of students choose different subjects. This means that even though two sessions are displayed side by side in the same timeslot or consecutive slots, no single student is required to attend both.

6.3 System Testing for Soft Constraints Defined

6.3.1 Gaps between classes should be minimized, such as long breaks between consecutive classes

Table 6.3.2.1 Test Case of 'Gaps between classes should be minimized, such as long breaks between consecutive classes'

Input Data	Description	Expected Result	Actual Result	Status
Dataset 1	This test case ensures that gaps between classes should be minimized, such as long breaks between consecutive classes.	In the timetable generated, gaps between classes should be minimized, such as long breaks between consecutive classes.	Gaps between classes are minimized, such as long breaks between consecutive classes.	Pass

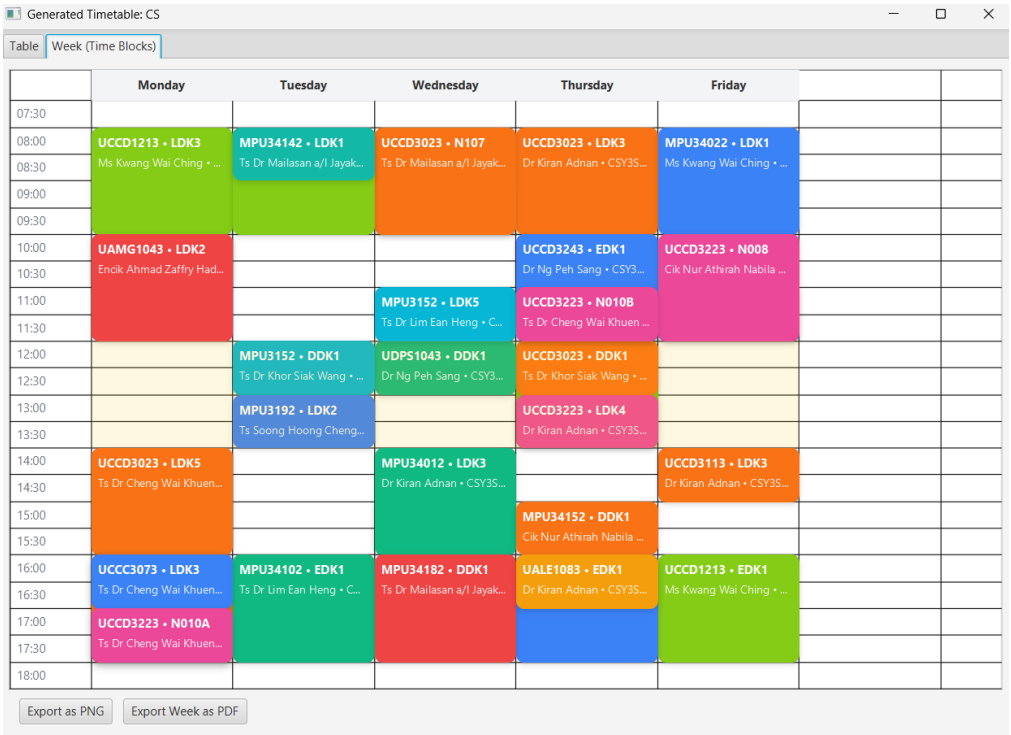

Figure 6.3.1.1 Proof of 'Gaps between classes should be minimized, such as long breaks between consecutive classes'

From Figure 6.3.1.1, the timetable was examined to check whether long gaps between consecutive classes were avoided. The system aims to minimize idle time during the day so that students do not experience unnecessary breaks between sessions.

When reviewing the schedule, it can be observed that most classes are arranged in a continuous flow, with only short gaps between sessions.

6.3.2 No classes should be scheduled during lunch break (12 p.m. – 2 p.m.)

Table 6.3.2.1 Test Case of 'No classes should be scheduled during lunch break (12 p.m. – 2 p.m.)'

Input Data	Description	Expected Result	Actual Result	Status
Dataset 1	This test case ensures that no classes should be scheduled during lunch break (12 p.m. – 2 p.m.).	In the timetable generated, no classes should be scheduled during lunch break (12 p.m. – 2 p.m.).	Some classes are scheduled during lunch break (12 p.m. – 2 p.m.).	Partial

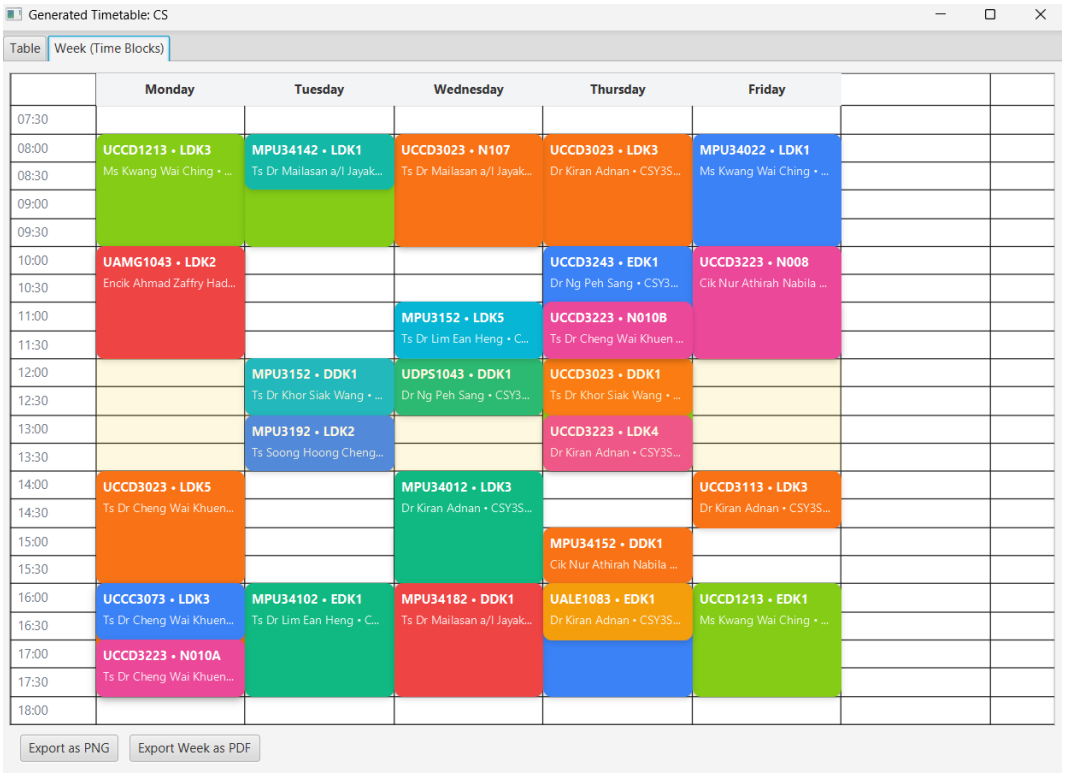

Figure 6.3.2.1 Proof of 'No classes should be scheduled during lunch break (12 p.m. – 2 p.m.)'

From Figure 6.3.2.1, it can be observed that the timetable still contains classes scheduled during the typical lunch period, around 12:00 PM to 2:00 PM. This indicates that the soft constraint of minimizing or avoiding classes during lunch hours was not strongly enforced by the system. While the timetable generator successfully handles hard constraints such as avoiding class clashes, ensuring correct venue assignments, and limiting maximum daily hours, the lunch break consideration was given lower priority.

As a result, some students may still have sessions that overlap with lunchtime, leading to less convenience in their daily schedules. However, since this is a soft constraint, the timetable can still be considered valid if all hard constraints are satisfied. In this case, the test outcome suggests that the system prioritizes fulfilling hard constraints over strictly enforcing break times, meaning this soft constraint is only partially achieved.

6.3.3 There should be no classes scheduled before 8 a.m. and after 8.30 p.m.

Table 6.3.3.1 Test Case of 'There should be no classes scheduled before 8 a.m. and after 8.30 p.m.'

Input Data	Description	Expected Result	Actual Result	Status
Dataset 1	This test case ensures that there should be no classes scheduled before 8 a.m. and after 8.30 p.m.	In the timetable generated, there should be no classes scheduled before 8 a.m. and after 8.30 p.m.	There are no classes scheduled before 8 a.m. and after 8.30 p.m.	Pass

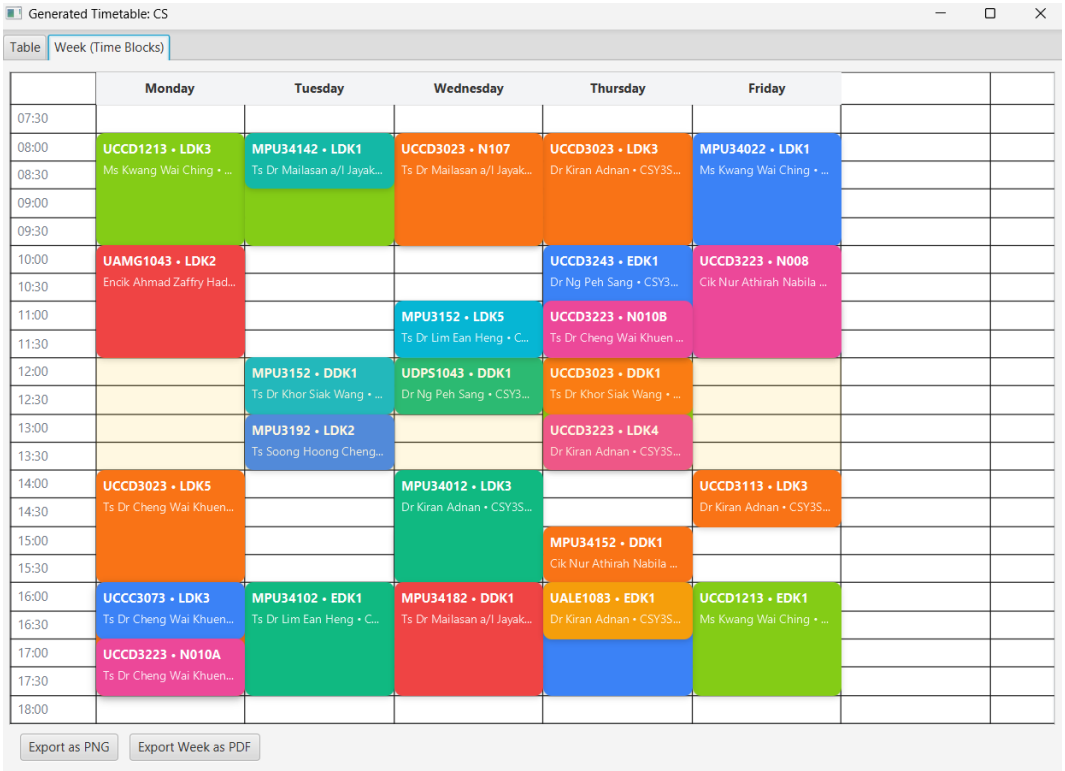

Figure 6.3.3.1 Proof of 'There should be no classes scheduled before 8 a.m. and after 8.30 p.m.'

From Figure 6.3.3.1, the timetable was reviewed to ensure that no sessions are scheduled outside the permitted time window, which is between 8:00 a.m. and 8:30

p.m. The earliest class in the timetable begins at 8:00 a.m., and the latest class ends by 6:00 p.m. This confirms that all scheduled sessions fall within the acceptable range.

6.3.4 The maximum hours for consecutive classes should be 4 hours

Table 6.3.4.1 Test Case of 'The maximum hours for consecutive classes should be 4 hours'

Input Data	Description	Expected Result	Actual Result	Status
Dataset 1	This test case ensures that the maximum hours for consecutive classes should be 4 hours.	In the timetable generated, the maximum hours for consecutive classes should be 4 hours	The total maximum hours for consecutive classes is 4 hours	Pass

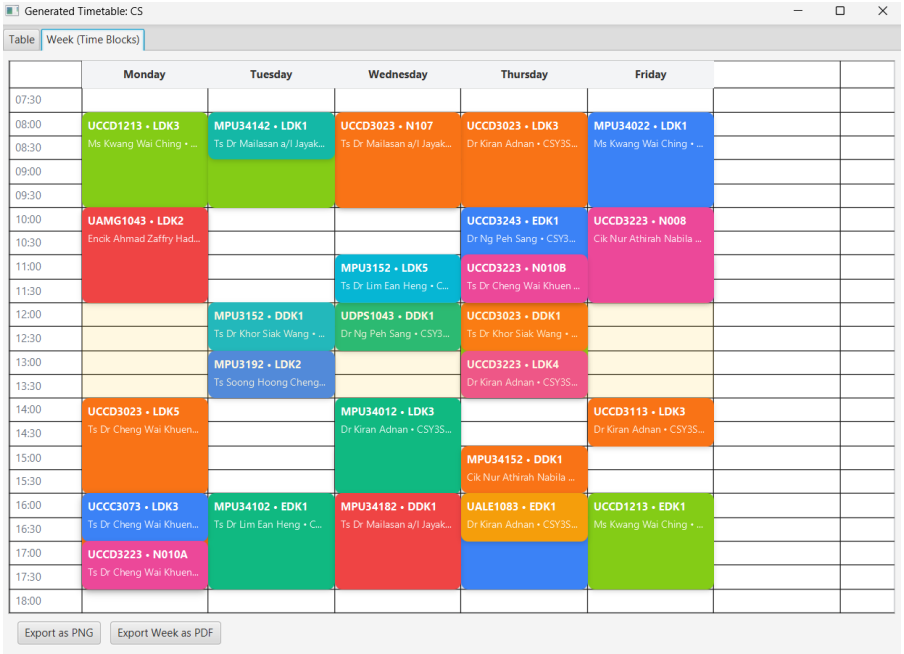

Figure 6.3.4.1 Proof of 'The maximum hours for consecutive classes should be 4 hours'

From Figure 6.3.3.1, the timetable was examined to ensure that no student is required to attend more than four consecutive hours of classes without a break. Consecutive hours refer to back-to-back sessions scheduled in the same day, where students would have to remain in classes continuously. The timetable shows that most class sequences are limited to either two-hour or three-hour blocks, and in some cases up to four hours. However, none of the schedules exceed the maximum of four consecutive hours. For example, on Monday, classes run from 8:00 a.m. to 12:00 p.m., which is exactly four hours, after which a break is given before the next session. This demonstrates that the system respects this constraint.

6.4 Project Challenges

The development of the University Timetabling System faced several key challenges. One major difficulty was managing the many constraints in timetable generation. While hard constraints such as avoiding clashes and preventing venue double-bookings were strictly enforced, balancing soft constraints like minimizing gaps or avoiding lunch-hour classes was harder, as improving one often weakened another.

Designing and integrating the database was also complex. The system had to handle different types of sessions (lectures, tutorials, practicals) and ensure accurate information for lecturers, student groups, and venues. This required careful structuring to support both the genetic algorithm and manual edits in the interface.

Configuring the genetic algorithm posed another challenge. Parameters such as population size and mutation rate required extensive testing, and while the GA produced valid timetables, optimizing soft constraints consistently was more difficult. Preventing the algorithm from getting stuck in local optima was also an ongoing issue.

Finally, time and resource limitations made testing across large-scale, real-world datasets difficult. While the system performs well under controlled conditions, broader validation with more complex scenarios remains a future improvement area.

Despite these challenges, the project achieved a functioning system that generates clash-free timetables and enforces the most critical scheduling rules.

6.5 Objectives Evaluation

The first objective of this project was to investigate and categorize the hard and soft constraints relevant to the university timetabling problem. This has been successfully achieved. Hard constraints such as preventing student and lecturer clashes, ensuring venues are not double-booked, and restricting classes within defined hours were implemented and validated through testing. Soft constraints, such as minimizing gaps between classes and avoiding classes during lunch, were also considered, although they were not given as much weight in the optimization process. While the soft constraints were only partially satisfied, their identification and integration into the system laid the groundwork for further improvements.

The second objective was to introduce the Proximity and Travel Minimization Constraint into the scheduling process using a Genetic Algorithm. This has been integrated into the system, where consecutive classes for the same student group or lecturer are allocated to venues within a maximum distance of 500 meters. Test cases confirmed that this constraint was largely satisfied, though some exceptions appeared in the full timetable due to elective courses, where students only attend one of the overlapping sessions. This demonstrates that the proximity rule functions correctly on an individual student basis.

The third objective was to create a more efficient, user-friendly timetabling system that improves scheduling efficiency and user experience. This was achieved through the development of resource management modules for lecturers, venues, and groups, as well as editing, archiving, and exporting features for timetables. The GA Analytics module further adds transparency by allowing users to view the algorithm's operations and outputs. Although some soft constraints remain less emphasized, the system overall delivers valid and practical timetables while providing administrators with tools to manage and maintain scheduling data effectively.

In summary, the project objectives were successfully met. The system can generate clash-free timetables under strict hard constraints, incorporates the new proximity and travel minimization rule, and provides a platform that is functional and user-friendly. The partial fulfilment of soft constraints highlights an area for future refinement but does not undermine the overall success of the project.

CHAPTER 7

Conclusion

7.1 System Limitations

This project has successfully developed an automated University Timetabling System that applies a Genetic Algorithm to generate valid, clash-free schedules. The system achieves its primary objectives by enforcing key hard constraints such as preventing class overlaps, ensuring venues are not double-booked, and respecting daily scheduling limits. Additional features, including resource management, editing, archiving, and exporting timetables, enhance its practicality and usability. The introduction of the Proximity and Travel Minimization Constraint further improves the realism of the generated schedules, ensuring that students and lecturers are not required to travel excessively between consecutive classes.

Despite these achievements, the system still presents certain limitations. Soft constraints, such as minimizing long gaps between classes or avoiding sessions during lunch hours, were not heavily emphasized in the optimization process. This means that while the timetables are feasible and valid, they may not always represent the most convenient arrangements for students. Furthermore, the current GA Analytics provides only basic insights into algorithm performance, and the system is limited to a desktop-based interface designed primarily for administrators.

7.2 Future Improvement

For future improvements, the GA Analytics module can be expanded to provide more detailed visualizations and performance metrics, allowing administrators to better evaluate and fine-tune the optimization process. In addition, developing a dedicated mobile interface for the student side would make the system more accessible, enabling students to view their personalized timetables directly on their devices. These enhancements would not only strengthen the system's functionality but also improve overall user experience for both administrators and students.

Overall, the project demonstrates that Genetic Algorithms are an effective approach for tackling the complex university timetabling problem. While there remain areas for further refinement, the system provides a strong foundation for future work and has the potential to evolve into a complete, institution-ready solution.

7.3 Concluding Remarks

In conclusion, the development of the University Timetabling System has demonstrated how genetic algorithms can be applied effectively to solve complex scheduling problems. The system integrates multiple modules, including timetable creation, editing, deletion, archiving, and resource management, while also providing analytics to evaluate the performance of the algorithm. Despite the challenges encountered during implementation, such as handling multiple constraints, optimizing performance, and ensuring data consistency, the final system is able to generate valid and practical timetables that meet academic requirements.

The project not only highlights the technical feasibility of combining Python, JavaFX, and MySQL into a unified solution but also emphasizes the importance of usability through features such as export options, archive management, and an intuitive interface. By addressing both functional and user-oriented needs, the system provides a balanced solution for administrators and students alike. Overall, this project has achieved its objectives and contributes a valuable approach to automated timetable generation. While there remains room for further refinement and enhancement, the system serves as a solid foundation that can be extended to support larger datasets, additional constraints, and broader institutional needs in the future.

REFERENCES

[1] H. Babaei, J. Karimpour, and A. Hadidi, "A survey of approaches for university course timetabling problem," *Computers & Industrial Engineering*, vol. 86, pp. 43–59, Nov. 2014, doi: 10.1016/j.cie.2014.11.010.

[2] M. C. Chen, S. N. Sze, S. L. Goh, N. R. Sabar, and G. Kendall, "A Survey of University Course Timetabling Problem: Perspectives, Trends and opportunities," *IEEE Access*, vol. 9, pp. 106515–106529, Jan. 2021, doi: 10.1109/access.2021.3100613.

[3] H. Alghamdi, T. Alsubait, H. Alhakami, and A. Baz, "A review of Optimization Algorithms for university timetable scheduling," *Engineering Technology & Applied Science Research*, vol. 10, no. 6, pp. 6410–6417, Dec. 2020, doi: 10.48084/etasr.3832.

[4] A. H. Khan and T. Imtiaz, "A Novel Genetic Algorithm Based Timetable Generator for Optimized University Timetable Solution," in *2024 International Conference on Engineering & Computing Technologies (ICECT)*, Islamabad, Pakistan: IEEE, May 2024, pp. 1–6. doi: 10.1109/ICECT61618.2024.10581296.

[5] H. M and M. M, "Solving Timetabling problems using Genetic Algorithm Technique," *International Journal of Computer Applications*, vol. 134, no. 15, pp. 33–38, Jan. 2016, doi: 10.5120/ijca2016907960.

[6] I. Lulu, H. Alowais, A. Turky, S. Harous, and A. Hussain, "Efficient Solution for Exam Timetabling Problem: A Case Study of the University of Sharjah," in *2024 17th International Conference on Development in eSystem Engineering (DeSE)*,

Khorfakkan, United Arab Emirates: IEEE, Nov. 2024, pp. 101–106. doi: 10.1109/DeSE63988.2024.10911922.

[7] M. S. Kohshori, D. Zeynolabedini, M. S. Liri, and L. Jadidi, "Multi population hybrid genetic Algorithms for university course timetabling problem," *International Journal of Information Technology and Computer Science*, vol. 4, no. 6, pp. 1–11, Jun. 2012, doi: 10.5815/ijitcs.2012.06.01.

[8] Yang and S. N. Jat, "Genetic algorithms with guided and local search strategies for university course timetabling," *IEEE Transactions on Systems Man and Cybernetics Part C (Applications and Reviews)*, vol. 41, no. 1, pp. 93–106, Jun. 2010, doi: 10.1109/tsmcc.2010.2049200.

[9] M. A. Albadr, S. Tiun, M. Ayob, and F. Al-Dhief, "Genetic algorithm based on natural selection theory for optimization problems," *Symmetry*, vol. 12, no. 11, p. 1758, Oct. 2020, doi: 10.3390/sym12111758.

[10] E. A. Abdelhalim and G. A. El Khayat, "A Utilization-based Genetic Algorithm for Solving the University Timetabling Problem (UGA)," *Alexandria Engineering Journal*, vol. 55, no. 2, pp. 1395–1409, Mar. 2016, doi: 10.1016/j.aej.2016.02.017.

[11] S. K. Jha, "EXAM TIMETABLING PROBLEM USING GENETIC ALGORITHM" *International Journal of Research in Engineering and Technology*, vol. 03, no. 05, pp. 649–654, May 2014, doi: 10.15623/ijret.2014.0305120.

[12] J. S. Tan, S. L. Goh, G. Kendall, and N. R. Sabar, "A survey of the state-of-the-art of optimisation methodologies in school timetabling problems," *Expert Systems*

With Applications, vol. 165, p. 113943, Sep. 2020, doi: 10.1016/j.eswa.2020.113943.

[13] S. Cheuk and S. Y. L. Chai, "Auto Timetable Management Mobile Application," *Trends in Undergraduate Research*, vol. 6, no. 2, pp. c1-7, Dec. 2023, doi: 10.33736/tur.5730.2023.

[14] A. M. Hambali, Y. A. Olasupo, and M. Dalhatu, "Automated university lecture timetable using Heuristic Approach," *Nigerian Journal of Technology*, vol. 39, no. 1, pp. 1–14, Apr. 2020, doi: 10.4314/njt.v39i1.1.

[15] S. Kazarlis, V. Petridis, and P. Fragkou, "Solving University Timetabling Problems Using Advanced Genetic Algorithms," 2005, [Online]. Available: https://api.semanticscholar.org/CorpusID:17910307

[16] S. Abdennadher and M. Marte, "University course timetabling using constraint handling rules," *Applied Artificial Intelligence*, vol. 14, no. 4, pp. 311–325, Apr. 2000, doi: 10.1080/088395100117016.

[17] S. Ghaemi, M. Taghi Vakili, and A. Aghagolzadeh, "Using a genetic algorithm optimizer tool to solve University timetable scheduling problem," in *2007 9th International Symposium on Signal Processing and Its Applications*, Sharjah, United Arab Emirates: IEEE, Feb. 2007, pp. 1–4. doi: 10.1109/ISSPA.2007.4555397.

[18] P. Pongcharoen, W. Promtet, P. Yenradee, and C. Hicks, "Stochastic Optimisation Timetabling Tool for university course scheduling," *International Journal of*

Production Economics, vol. 112, no. 2, pp. 903–918, Sep. 2007, doi: 10.1016/j.ijpe.2007.07.009.

[19] Y. Sun, X. Luo, and X. Liu, "Optimization of a university timetable considering building energy efficiency: An approach based on the building controls virtual test bed platform using a genetic algorithm," *Journal of Building Engineering*, vol. 35, p. 102095, Dec. 2020, doi: 10.1016/j.jobe.2020.102095.

[20] S. Abdullah and H. Turabieh, "On the use of multi neighbourhood structures within a Tabu-based memetic approach to university timetabling problems," *Information Sciences*, vol. 191, pp. 146–168, Jan. 2012, doi: 10.1016/j.ins.2011.12.018.

[21] S. Abdipoor, R. Yaakob, S. L. Goh, and S. Abdullah, "Meta-heuristic approaches for the University Course Timetabling Problem," *Intelligent Systems With Applications*, vol. 19, p. 200253, Jun. 2023, doi: 10.1016/j.iswa.2023.200253.

[22] J. Frausto-Solis, J. Mora-Vargas, M. Larre, and J. L. Gomez-Ramos, "Genetic Algorithm with Forced Diversity for the University TimeTabling Problem," *Proceedings of the 10th WSEAS International Conference on SYSTEMS*, pp. 553–558, Jul. 2006

[23] P. Khonggamnerd and S. Innet, "On Improvement of Effectiveness in Automatic University Timetabling Arrangement with Applied Genetic Algorithm," in *2009 Fourth International Conference on Computer Sciences and Convergence Information Technology*, Seoul, Korea: IEEE, 2009, pp. 1266–1270. doi: 10.1109/ICCIT.2009.202.

[24] N. Nuntasen and S. Innet, "A Novel Approach of Genetic Algorithm for Solving University Timetabling Problems: a case study of Thai Universities" *7th WSEAS International Conference on APPLIED COMPUTER SCIENCE*, pp. 246–252, Nov. 2007, [Online]. Available: https://dl.acm.org/citation.cfm?id=1348171.1348214

[25] Team Applied AI, "Genetic algorithm in machine learning," Applied AI Course. [Online]. Available: https://www.appliedaicourse.com/blog/genetic-algorithm-in-machine-learning/

[26] I. Balan, "A New Genetic Approach for Course Timetabling Problem," *Journal of Applied Computer Science & Mathematics/Journal of Applied Computer Science*, vol. 15, no. 1, pp. 9–14, Jan. 2021, doi: 10.4316/jacsm.202101001.

[27] P. Boonyopakorn and P. Meesad, "A Hybrid Immune Genetic Algorithm to Solve University Time Table Problems*" *Walailak Journal of Science and Technology (WJST)*, vol. 14, no. 10, pp. 825–835, Jun. 2017, [Online]. Available: https://103.58.148.28/index.php/wjst/article/view/4170

[28] M. Hosny and M. Al-Olayan, "A mutation-based genetic algorithm for room and proctor assignment in examination scheduling," in *2014 Science and Information Conference*, London, UK: IEEE, Aug. 2014, pp. 260–268. doi: 10.1109/SAI.2014.6918199.

[29] A. Jula and N. K. Naseri, "Using CMAC to Obtain Dynamic Mutation Rate in a Metaheuristic Memetic Algorithm to Solve University Timetabling Problem" *European Journal of Scientific Research*, vol.63, no.2 (2011), pp.172-181, Sep. 2021 [Online]. Available: https://www.academia.edu/54080602/Using_CMAC_to_Obtain_Dyna

mic_Mutation_Rate_in_a_Metaheuristic_Memetic_Algorithm_to_Solve_University_Timetabling_Problem

[30] S. Tkatek, S. Bahti, and J. Abouchabaka, "Artificial Intelligence for Improving the Optimization of NP-Hard Problems: A Review," *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 9, no. 5, pp. 7411–7420, Oct. 2020, doi: 10.30534/ijatcse/2020/73952020.

[31] S. N. Jat and S. Yang, "A hybrid genetic algorithm and tabu search approach for post enrolment course timetabling," *Journal of Scheduling*, vol. 14, no. 6, pp. 617–637, Nov. 2010, doi: 10.1007/s10951-010-0202-0.

[32] W. Wen-jing, "Improved Adaptive Genetic Algorithm for Course Scheduling in Colleges and Universities," *International Journal of Emerging Technologies in Learning (iJET)*, vol. 13, no. 06, p. 29, May 2018, doi: 10.3991/ijet.v13i06.8442.

[33] J. Nourmohammadi-Khiarak, Y. Zamani-Harghalani, and M.-R. Feizi-Derakhshi, "Combined Multi-Agent Method to Control Inter-Department Common Events Collision for University Courses Timetabling," *Journal of Intelligent Systems*, vol. 29, no. 1, pp. 110–126, Dec. 2017, doi: 10.1515/jisys-2017-0249.

[34] E. H. Houssein, A. G. Gad, K. Hussain, and P. N. Suganthan, "Major Advances in Particle Swarm Optimization: Theory, Analysis, and Application," *Swarm and Evolutionary Computation*, vol. 63, p. 100868, Mar. 2021, doi: 10.1016/j.swevo.2021.100868.

[35] A. I. Diveev and O. V. Bobr, "NP-Hard Task Schedules and Methods of Its Decision," in *2017 IEEE 11th International Conference on Application of*

Information and Communication Technologies (AICT), Moscow, Russia: IEEE, Sep. 2017, pp. 1–5. doi: 10.1109/ICAICT.2017.8686990.

[36] F. Marini and B. Walczak, "Particle swarm optimization (PSO). A tutorial," *Chemometrics and Intelligent Laboratory Systems*, vol. 149, pp. 153–165, Sep. 2015, doi: 10.1016/j.chemolab.2015.08.020.

[37] O. Dulger, "SOLVING WEEKLY COURSE TIMETABLING PROBLEM WITH GENETIC ALGORITHM AND LOCAL SEARCH," *3rd International Symposium on Computing in Science & Engineering*, Oct. 2013.

[38] L. Wu, "The application of Coarse-Grained Parallel Genetic Algorithm with Hadoop in University Intelligent Course-Timetabling System," *International Journal of Emerging Technologies in Learning (iJET)*, vol. 10, no. 8, p. 11, Dec. 2015, doi: 10.3991/ijet.v10i8.5206.

[39] J. Arias-Osorio and A. Mora-Esquivel, "A solution to the university course timetabling problem using a hybrid method based on genetic algorithms," *DYNA*, vol. 87, no. 215, pp. 47–56, Nov. 2020, doi: 10.15446/dyna.v87n215.85933.

[40] Al-Mahmud and M. A. H. Akhand, "ACO with GA operators for solving University Class Scheduling Problem with flexible preferences," in *2014 International Conference on Informatics, Electronics & Vision (ICIEV)*, Dhaka, Bangladesh: IEEE, May 2014, pp. 1–6. doi: 10.1109/ICIEV.2014.6850742.

[41] R. Lewis and B. Paechter, "Finding Feasible Timetables Using Group-Based Operators," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 3, pp. 397–413, Jun. 2007, doi: 10.1109/tevc.2006.885162.

[42] Huzaifa and A. Saleem, *TimetableGeneratorApp*. (Aug. 16, 2020). Python, HTML, CSS, JavaScript. [Online]. Available: https://github.com/mHuzefa/TimetableGeneratorApp

[43] O. Odunsi, R. Clifford Jr, and B. Rasool, *timetable-generator*. (Jan. 07, 2018). [Online]. Available: https://github.com/olaysco/timetable-generator

[44] UniTime.org, *UniTime | University Timetabling*. (Oct. 2023). [Online]. Available: https://www.unitime.org/

[45] Tarek A. El-Mihoub et al., "Hybrid Genetic Algorithms : A Review," *Engineering Letters*, vol. 13, no. 2, pp. 124-137, Aug. 2006, [Online]. Available: https://www.researchgate.net/publication/26623711_Hybrid_Genetic_Algorithms_A_Review

[46] K. Wu, X. Ye, S. Jamonnak, and X. Feng, "A Digital Twin-Driven Recommendation System for Adaptive Campus Course Timetabling," Mar. 08, 2025, *arXiv*: arXiv:2503.06109. doi: 10.48550/arXiv.2503.06109.

[47] "Crossover in Genetic Algorithm," GeeksForGeeks. Accessed: Apr. 21, 2025. [Online]. Available: https://www.geeksforgeeks.org/crossover-in-genetic-algorithm/

[48] GeeksforGeeks, "Haversine formula to find distance between two points on a sphere," GeeksforGeeks, Sep. 05, 2022. Available: https://www.geeksforgeeks.org/dsa/haversine-formula-to-find-distance-between-two-points-on-a-sphere/

APPENDIX

Poster