UNIVERSITY TIMETABLE USING GENETIC ALGORITHM

LOO HEN SHEN

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

BORANG PENGESAHAN STATUS LAPORAN

JUDUL: UNIVERSITY TIMETABLE USING GENETIC ALGORITHM

SESI PENGAJIAN: 2022 / 2023

Saya: LOO HEN SHEN

mengaku membenarkan tesis Projek Sarjana Muda ini disimpan di Perpustakaan Universiti Teknikal Malaysia Melaka dengan syarat-syarat kegunaan seperti berikut:

1. Tesis dan projek adalah hakmilik Universiti Teknikal Malaysia Melaka.
2. Perpustakaan Fakulti Teknologi Maklumat dan Komunikasi dibenarkan membuat salinan unituk tujuan pengajian sahaja.
3. Perpustakaan Fakulti Teknologi Maklumat dan Komunikasi dibenarkan membuat salinan tesis ini sebagai bahan pertukaran antara institusi pengajian tinggi.
4. * Sila tandakan (✓)

____	SULIT	(Mengandungi maklumat yang berdarjah keselamatan atau kepentingan Malaysia seperti yang termaktub di dalam AKTA RAHSIA RASMI 1972)
____	TERHAD	(Mengandungi maklumat TERHAD yang telah ditentukan oleh organisasi / badan di mana penyelidikan dijalankan)
✓	TIDAK TERHAD	

LOO HEN SHEN	
(TANDATANGAN PELAJAR)	(TANDATANGAN PENYELIA)
Alamat tetap: 485, Jalan Bayan 11, Taman Desa Rasah, 70300 Seremban, Negeri Sembilan.	PROFESSOR MADYA TS DR ZERATUL IZZAH BINTI MOHD YUSOH Nama Penyelia
Tarikh: 7/9/2023	Tarikh: 20/09/2023

CATATAN: * Jika tesis ini SULIT atau TERHAD, sila lampirkan surat daripada pihak

UNIVERSITY TIMETABLE USING GENETIC ALGORITHM

LOO HEN SHEN

This report is submitted in partial fulfillment of the requirements for the Bachelor of Computer Science (Artificial Intelligence) with Honours.

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY
UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2023

DECLARATION

I hereby declare that this project report entitled

UNIVERSITY TIMETABLE USING GENETIC ALGORITHM

is written by me and is my own effort and that no part has been plagiarized without citations.

STUDENT : *LOO HEN SHEN* Date : 7/9/2023

(LOO HEN SHEN)

I hereby declare that I have read this project report and found this project report is sufficient in term of the scope and quality for the award of Bachelor of Computer Science (Artificial Intelligence) with Honours.

SUPERVISOR : ______________________ Date : 20/09/2023

(PROFESSOR MADYA TS. DR. ZERATUL IZZAH MOHD. YUSOH)

DEDICATION

To my beloved parents, thank you for everything you have done and continue to do. You are my inspiration, my guiding light, and my greatest blessing. I am honored to be your child, and I will forever cherish the love and memories we share.

ACKNOWLEDGEMENTS

I would like to thank Professor Madya Ts. Dr. Zeratul Izzah for providing invaluable assistance and guidance, which played a crucial role in the successful completion of this project. Your expertise and support have been instrumental in shaping its outcome.

I would also like to thank my beloved parents who have been giving me support and motivation throughout my project Your encouragement and belief in me have been the driving force behind my achievements. I am truly grateful for your love and encouragement.

ABSTRACT

This report presents the development and implementation of a university timetable using a genetic algorithm. The main problem addressed in this study is the manual and time-consuming process of creating timetables for university courses. The solution proposed involves the use of a genetic algorithm to optimize the allocation of classes, rooms, and resources, while satisfying various constraints and preferences. The research process begins with the collection of data on course requirements, room availability, and lecturer preferences. A genetic algorithm is then used to generate possible timetable solutions, which are evaluated based on their feasibility and efficiency. The algorithm is iteratively improved to find the most optimal timetable that meets all requirements. The results obtained from the implementation of the genetic algorithm show significant improvements in the efficiency of timetable generation, reducing the time and effort required by university administrators. Additionally, the generated timetables are more balanced and satisfying for both students and lecturers.

ABSTRAK

Laporan ini adalah berkenaan dengan pembangunan dan pelaksanaan penghasil jadual waktu universiti menggunakan algoritma genetik. Masalah utama yang dibincangkan dalam kajian ini adalah proses manual dan memakan masa dalam penyusunan jadual waktu kursus universiti. Cadangan penyelesaian melibatkan penggunaan algoritma genetik untuk mengoptimumkan pengagihan kelas, bilik, dan sumber, sambil memenuhi pelbagai kekangan dan keutamaan. Proses penyelidikan bermula dengan pengumpulan data mengenai keperluan kursus, ketersediaan bilik, dan keutamaan pensyarah. Algoritma genetik digunakan untuk menghasilkan pelbagai penyelesaian jadual waktu yang mungkin, yang dinilai berdasarkan kelayakan dan kecekapan mereka. Algoritma ini diperbaiki secara berulang untuk mencari jadual waktu yang paling optimum yang memenuhi semua keperluan. Hasil yang diperoleh daripada pelaksanaan algoritma genetik menunjukkan peningkatan yang signifikan dalam kecekapan penyusunan jadual waktu, mengurangkan masa dan usaha yang diperlukan oleh pentadbir universiti. Selain itu, jadual waktu yang dihasilkan lebih seimbang dan memuaskan untuk pelajar dan pensyarah.

TABLE OF CONTENTS

PAGE

DECLARATION .... **II**

DEDICATION .... **III**

ACKNOWLEDGEMENTS .... **IV**

ABSTRACT .... **V**

ABSTRAK .... **VI**

TABLE OF CONTENTS .... **VII**

LIST OF TABLES .... **XII**

LIST OF FIGURES .... **XIII**

LIST OF ABBREVIATIONS .... **XV**

LIST OF ATTACHMENTS .... **XVI**

CHAPTER 1: INTRODUCTION **1**

1.1 Introduction 1

1.2 Problem statement(s) 1

1.3 Objective 2

1.4 Scope 2

1.4.1 Modules 3

1.4.2 Target User 4

1.5 Project Significance 4

1.6 Expected Output 4

1.7 Conclusion 4

CHAPTER 2: LITERATURE REVIEW AND PROJECT METHODOLOGY . 5

2.1 Introduction.. 5

2.2 Facts and findings 5

2.2.1 Domain 5

2.2.1.1 University Timetable Scheduling 5

2.2.2 Existing System 7

2.2.3 Technique 10

2.3 Project Methodology.. 12

2.4 Project Requirements... 14

2.4.1 Software Requirement 14

2.4.1.1 Python 14

2.4.1.2 Integrated Development Environment.. 14

2.4.1.3 Flask Web Development Framework 15

2.4.1.4 MySQL Workbench Database.. 15

2.4.2 Hardware Requirement.. 15

2.4.3 Other Requirements 15

2.5 Project Schedule and Milestones 16

2.6 Conclusion 16

CHAPTER 3: ANALYSIS.. 17

3.1 Introduction.. 17

3.2 Problem Analysis 17

3.3 Requirement Analysis.. 19

3.3.1 Data Requirement 19
3.3.2 Functional Requirement 19
3.3.3 Non-Functional Requirement 20
3.3.4 Other Requirement 21
3.3.4.1 Software Requirement 21
3.3.4.2 Hardware Requirement 22
3.4 Conclusion 22
CHAPTER 4: DESIGN 23
4.1 Introduction 23
4.2 High-Level Design 23
4.2.1 System Architecture 23
4.2.2 User Interface Design 24
4.2.2.1 Navigation Design 24
4.2.2.2 Input Design 25
4.2.2.3 Technical Design 29
4.2.2.4 Output Design 32
4.2.3 Database Design 32
4.2.3.1 Conceptual and Logical Database Design 32
4.3 Detailed Design 35
4.3.1 Software Design 35
4.3.1.1 Chromosome Representation 35
4.3.1.2 Initialization Method 36
4.3.1.3 Fitness Evaluation 36

4.3.1.4 Parent Selection 38

4.3.1.5 Crossover & Mutation 39

4.3.1.6 Survival Selection 40

4.3.1.7 Iteration of Generations and Termination 40

4.3.2 Physical Database Design 41

4.4 Conclusion 43

CHAPTER 5: IMPLEMENTATION 44

5.1 Introduction 44

5.2 Software Development Environment Setup 44

5.3 Software Configuration Management 45

5.3.1 Configuration environment setup 45

5.3.2 Version Control Procedure 46

5.4 Implementation Status 47

5.4.1 Website-based Information System 47

5.4.2 University Timetabling using Genetic Algorithm 48

5.5 Conclusion 49

CHAPTER 6: TESTING 51

6.1 Introduction 51

6.2 Test Plan 51

6.2.1 Test Organization 51

6.2.2 Test Environment 52

6.2.3 Test Schedule 52

6.3 Test Strategy 54

6.3.1 Classes of tests 54

6.4 Test Implementation 55

6.4.1 Experimental / Test Description 55

6.4.1.1 Graphical User Interface (GUI) Module Test Cases 55

6.4.1.2 Database Module Test Cases 56

6.4.1.3 AI (Genetic Algorithm) Module Test Cases 57

6.4.2 Test Data 58

6.5 Test Results and Analysis 58

6.5.1 Graphical Analysis... 58

6.5.2 Optimal University Timetable Generated...................................... 59

6.5.3 Algorithm Performance 59

6.6 Conclusion 60

CHAPTER 7: CONCLUSION.. 61

7.1 Observation on Weaknesses and Strengths.. 61

7.2 Propositions for Improvement 62

7.3 Project Contribution.. 62

7.4 Conclusion 63

REFERENCES 64

APPENDIX 66

LIST OF TABLES

PAGE

Table 2.1 Explanation of Existing System 7
Table 2.2 Minimum Requirement of Computing Device 15
Table 3.1 Hardware Specification 22
Table 5.1 Website-based Information System's Implementation......................... 47
Table 5.2 University Timetabling using Genetic Algorithm's Implementation . 48
Table 6.1 Test Schedule 53
Table 6.2 Minimum, Maximum, Average and Standard Deviation of 3 Runs Algorithm Performance.. 60
Table 7.1 Lecturer Information.. 72
Table 7.2 Room Information... 74
Table 7.3 Course Information... 76

LIST OF FIGURES

PAGE

Figure 2.1 Iterative Waterfall Model 13
Figure 2.2 Gantt Chart 16
Figure 3.1 Genetic Algorithm (GA) Process Diagram 18
Figure 3.2 Flowchart of the system 18
Figure 3.3 Use Case Diagram 20
Figure 4.1 Web-Server Architecture 24
Figure 4.2 Login Screen 25
Figure 4.3 Sign Up Screen 26
Figure 4.4 Home Screen 26
Figure 4.5 Lecturer Screen 27
Figure 4.6 Room Screen 27
Figure 4.7 Course Screen 28
Figure 4.8 Add Lecture/Lab Screen 28
Figure 4.9 Batch Screen 28
Figure 4.10 Pseudocode of Genetic Algorithm 29
Figure 4.11 Genes, Chromosomes and Population 30
Figure 4.12 Illustration of crossover 31
Figure 4.13 Entity Relationship Diagram 34
Figure 4.14 Chromosome Representation 35
Figure 4.15 DDL of Creating Lecturer Table 41
Figure 4.16 DDL of Creating Room Table 41
Figure 4.17 DDL of Creating Course Table 42
Figure 4.18 DDL of Creating Section & Group Table 42
Figure 4.19 DDL of Creating Lecture/Lab Table 42

Figure 4.20 DDL of Creating Lecturer_Preferences Table.................................. 42
Figure 4.21 DDL of Creating Lecturer_Qualifications Table.............................. 43
Figure 4.22 DDL of Creating User Table... 43
Figure 6.1 100 Generations with 550 Population Size 59
Figure 7.1 Population Initialization Function 66
Figure 7.2 Fitness Functions 68
Figure 7.3 Parent Selection Function 69
Figure 7.4 Crossover Functions 69
Figure 7.5 Mutation Function 70
Figure 7.6 Survival Selection Function 70
Figure 7.7 Iterations Function 71

LIST OF ABBREVIATIONS

FYP	**-**	**Final Year Project**
UTeM	**-**	**Universiti Teknikal Malaysia Melaka**
FTMK	**-**	**Faculty of Information Technology & Communication**
GA	**-**	**Genetic Algorithm**
SDLC	**-**	**Software Development Lifecycle**
IDE	**-**	**Integrated Development Environment**
GUI	**-**	**Graphical User Interface**

LIST OF ATTACHMENTS

		PAGE
Appendix A	**Sample Code**	**66**
Appendix B	**Data for University Timetable using Genetic Algorithm**	**72**
Appendix C	**Output**	**78**

CHAPTER 1: INTRODUCTION

1.1 Introduction

Timetabling, often known as scheduling, is the practice of allotting time for scheduled operations in an organized way to produce an outcome that is satisfying and unrestricted. Transportation, sports, the workforce, courses, and exam scheduling are a few examples. Two sorts of timetables, for instance, are used in educational institutions such as universities. The lecture and lab schedules are what they are.

Given the individual lecture rooms and labs best suited for each course and the total number of enrolled students, courses should be assigned to specified timeslots for five working days of the week. Thus, a practical timetable in a university outline how students and faculty go from a single lecture room to the next, as well as where the lecture rooms are located and when they are open. A university must deal with a number of constraints while constructing a timetable. These constraints can be categorized as either "hard" or "soft" constraints based on whether or not they are necessary or desirable.

1.2 Problem statement(s)

The first subproblem based on the first objective in Section 1.3 is the existing timetable of academic university can be improved further. When scheduling is not done effectively and systematically, it frequently results in timetables with conflicts for both students and lecturers, overlapping lectures, and underused resources.

The second subproblem is the existing technique of current academic universities' traditional timetable-making procedures frequently rely rule-based approaches, which can be inefficient and time-consuming. This technique might not take into account the countless alternative combinations and constraints involved in creating a timetable.

The third subproblem is that academic universities' current procedures for creating timetables frequently using manual or paper works and don't have an intuitive and effective approach. It may be difficult for faculty, administrators, and students to read the generated timetables, manage their schedules, and make the required adjustments. A user-friendly interface, dynamic visualization, and interaction with other university systems is lacking in the current system.

1.3 Objective

This project embarks on the following objectives:

- To formulate an optimal timetable in academic universities.
- To implement genetic algorithm in order to create an efficient timetable.
- To develop a website-based system to generate timetable.

1.4 Scope

The system's scope is purposefully constrained to three years, two programs which are BITI and BITS, and four batches which are S1G1, S1G2, S2G1, and S2G2 for a variety of reasons. First of all, broadening the system to include all years, programs, and batches would considerably raise the complexity and computational demands, which would have an impact on the effectiveness and performance of the genetic algorithm module. Secondly, concentrating on a certain scope enables a more targeted and controllable design of the user interface and genetic algorithm modules. It makes it possible for the system to meet the current scheduling requirements and limitations of the chosen years, programs, and batches. As a result, within the boundaries of the established scope, efficiency and effectiveness are given priority in the system.

1.4.1 Modules

i. AI Module

The main part of the system that creates the university timetables is the genetic algorithm module. In this module, the timetable-generating process is iteratively improved and evolved using a genetic algorithm, a heuristic search technique inspired by the process of natural selection. The required genetic operators, such as population initialization, fitness evaluation, parent and offspring selection, crossover, and mutation, must be defined. These operators are used by the genetic algorithm module to develop and improve prospective timetables while accounting for constraints like lecturer room and lab availability as well as other pertinent criteria. The goal is to create timetables that are optimized to reduce disputes and maximize resource use.

ii. User Interface Module

The front-end of website-based system is the user interface module. It offers a user-friendly interface that enables users to access different features and modules and manage their data. Users should be able to enter their data, view generated timetables, and make changes as necessary through an interface that is simple to use. The primary goal of this module is to ensure a positive user experience by displaying the university's data and timetables in a visually appealing and dynamic way.

iii. Database Module

The system's foundation is the database module, which was created using MySQL Workbench. In order to handle the storage and retrieval of user's data, timetables, and other pertinent information. It offers a reliable and effective database administration system. To guarantee data integrity and quick access, this module is in charge of building and maintaining the relevant tables, relationships, and queries. It keeps records of user information, course availability, lecturer availability, lecturer room and lab availability, and other details that are essential for the scheduling to produce the best optimized timetables.

1.4.2 Target User

The main target users for the university timetable using a genetic algorithm are faculty members and university personnel who are in charge of scheduling lecturers and rooms.

1.5 Project Significance

The significance of this project is to illustrate how effectively genetic algorithms can be used to find the most effective approaches for scheduling timetables in general. Despite the abundance of commercially available software for scheduling, its lack of generality makes it difficult for it to satisfy the needs of varied universities. The main challenge to be conquered is the demand for particular coding as per the distinct universities.

1.6 Expected Output

The expected output is the university staff members who are in charge of timetabling lectures and labs are anticipating receiving a thorough and intelligent university timetable from the genetic algorithm. Improved levels of satisfaction among students and lecturers will arise from the system's provision of an optimized timetable that takes into consideration a variety of factors, including lecturer preferences, lecture room and lab availability, as well as other pertinent constraints.

1.7 Conclusion

The purpose of this research project is to provide an explanation and experimentally validate a genetic algorithm that can be used to solve scheduling issues at universities. Additionally, we hope to use this approach to solve a real-world scheduling issue for UTeM. The goal of this study was to resolve the scheduling issue and come up with a fix that would be embraced by all users. The methodology of the project and the literature review of pertinent publications will then be covered in the following chapter.

CHAPTER 2: LITERATURE REVIEW AND PROJECT METHODOLOGY

2.1 Introduction

Early on in the project's development, a literature analysis was done to investigate the approaches and methods used to address project-related issues. This is crucial because it offers information on subject-matter expertise and potential problem-solving strategies. The explanation of the domain, the methodology, and the project needs are the important points of this chapter.

2.2 Facts and findings

The facts and findings in this section is related to the project domain and the problem domain.

2.2.1 Domain

2.2.1.1 University Timetable Scheduling

The FTMK’s class schedules are successfully managed and organized by the Committee at UTeM. The major goal is to efficiently utilize the classrooms, lecturers, and student of sections and groups that are already available by allocating resources and time slots in an ideal way.

There are various key entities in the domain. First, FTMK offers a variety of courses, each of which has its own course code, name, and number of credits. These classes fall under several disciplines or fields of study, including artificial intelligence, software development, computer networking, and others. In order to guarantee that courses are properly scheduled, prerequisites or corequisite requirements should be taken into account while constructing schedules.

FTMK features a number of classrooms with various seating arrangements and essential facilities. Each classroom is recognized by a certain room number or code, and it could contain particular facilities or specifications, such as computer laboratories, projectors, or specialized equipment, that are necessary for a given course. A group of knowledgeable lecturers working for the faculty is in charge of delivering the courses. Each professor possesses unique qualifications, areas of specialization, and scheduling hard and soft constraints. They might have preferences or limitations when it comes to teaching certain lectures or at particular times.

In FTMK, students are divided into various groups and sections according to their systems of study and academic levels. There are a certain number of students in each group. These student groups have specific academic needs that must be met within appropriate time windows. The academic week is divided into time slots with predetermined lengths, usually between one and four hours, as part of the scheduling process. An organized class schedule is produced by allocating certain days and times during the week to each time slot.

Several hard constraints and soft constraints are taken into account when making the timetable. Conflicts in the timetable, such as lecturers' or students' classes running concurrently, must be avoided. To make sure that courses are arranged in a logical sequence, prerequisites and corequisites must be taken into consideration. There should be consideration for the unique needs of lectures or labs for particular courses. With consideration for their knowledge and availability, the workload among lecturers should be handled equally. The best way to use resources is to make the most of each class period in the lecture and avoid unnecessary pauses. Additionally, if applicable, lecturers' preferences or qualifications should be taken into account. Finally, it is preferable to reduce the amount of time lecturers and students spend travelling between lectures.

A thorough timetable scheduling system for the UTeM FTMK’s can be created by taking these key entities, constraints, and factors into account. This will enable the successful administration of class schedules and foster an environment that is favorable to learning for both students and lecturers.

2.2.2 Existing System

Table 2.1 Explanation of Existing System

Existing System	**Brief Description**	**Advantages**	**Disadvantages**
UniTime	UniTime is a comprehensive educational scheduling system that enables the creation of course and examination timetables as well as the management of modifications.	- Makes use of constraint-based optimization strategy. - Can take care of the scheduling requirements for big academic institutions. - Allows for flexible configuration of scheduling preferences and policies to match the institution's specific needs. - Offers administrations with tools for reporting and analytics.	- Configuration and setup are challenging. - Time and resource-consuming. - Not delivering synchronize data updates

Optaplanner	OptaPlanner is the leading Open-Source Java™ AI constraint solver to optimize maintenance scheduling, adhering to skill, capacity, SLAs and other constraints.	- Can solve near-optimal timetable within a reasonable of time. - Can employ incremental scoring and score calculation caching. - Enables scenarios for real-time planning and rescheduling. - Compatible with multi-objective optimization - Can swiftly recalculate optimal solutions in response to changes in the issue domain.	- Require high computational resource. - Designed primarily to address static optimization timetable scheduling. - Bad dealing with soft constraints. - Needs fine-tuning.
TimeTabler	TimeTabler is a quick and user-friendly computer program that was carefully created based	- Can automatically finds and eliminates conflicts that might occur.	- Expensive for implementing and maintaining system

	on years of timetabling experience to assist user in precisely and swiftly scheduling timetable.	- Can produce thorough information on timetables. - Can track and monitor the progress of timetable generation. - Can be used to find areas that need improvement. - Provides training for first time user.	
Prime Timetable	Prime Timetable is a school scheduling program designed for both automatic and manual timetabling on any device.	- Can share timetables and update with the appropriate stakeholders. - Can browse and review historical timetables. - Support for data import and export from outside sources.	- Need payments for subscriptions. - Simple constraint-based scheduling

		- Can create paper timetables and online publication	
Coursicle	Coursicle is a website and app that designed for college students to create a timetable and keep track of their schedules.	- Students can rate and evaluate courses and lecturers - Giving customized course suggestions based on students' academic preferences. - Interacts with well-known calendar applications like Outlook, Apple Calendar, and Google Calendar.	

2.2.3 Technique

There are other methods available besides the genetic algorithm that can be used to handle the issue of creating a workable and optimized timetable. These strategies are all examples of meta-heuristics. Metaheuristics are more frequently employed for addressing NP-hard issues than when solving other types of problems. The most commonly used technique for scheduling timetables is genetic algorithms. Among these well-liked strategies are, but are not limited to:

i. Hyper-heuristics

It can be challenging and time-consuming to put hyper-heuristic algorithms into practice for creating university timetable. A hyper-heuristic algorithm demands extensive domain knowledge, algorithm design experience, and substantial computer resources to create and optimize timetables. This may be impractical for university with lack of time, money, or optimization algorithm competence and can increase the development effort.

ii. Multi-objective optimization

The goal of multi-objective optimization is to finding solutions along the Pareto front. It is a collection of trade-off options where no one answer predominates the others. The Pareto front can depict a variety of realistic schedules with various trade-offs between competing goals in the context of university timetables. Nevertheless, there couldn’t be a single "optimum" answer that appeases all interested parties. The Pareto front's interpretation and solution selection become arbitrary and necessitate extra manual intervention and judgement, thus negating the automated advantages of applying optimization approaches.

iii. Tabu Search Algorithm

The Tabu Search Algorithm functions as a black-box optimization technique, like many other metaheuristic algorithms. Although it can develop optimized timetables, it cannot be transparent or easy to understand how the solutions are created or how various constraints and preferences are met. In contrast, other systems such as rule-based or constraint-based, may provide more transparency by directly embedding domain knowledge into the timetabling process, facilitating easier comprehension and validation of the generated timetables by stakeholders.

iv. Integer Programming

The Integer Programming model would require updating according additional attributes such as courses, lecturers, or scheduling constraints when added. It might be difficult to maintain and update an Integer Programming model for university timetable, especially when university requirements constant over time. The model may need continuing work and knowledge to ensure its accuracy and relevance, making it less useful for university with changing timetables requirements.

v. Elite Immune Ant Colony Optimization Algorithm

Elite Immune Ant Colony Optimization Algorithm might not be as flexible or customizable comparing to genetic algorithm. This is due to the reason that university timetable frequently entails certain constraints, preferences, and goals that differ between university. It may be difficult to modify the algorithm to satisfy particular needs or to include domain-specific knowledge.

2.3 Project Methodology

The traditional waterfall model is challenging to employ in a real-world software development project. Therefore, it is possible to think of the iterative waterfall model as adding the essential modifications to the traditional waterfall model to make it applicable to real-world software development projects. With a few adjustments to boost the effectiveness of software development, it is nearly identical to the traditional waterfall model. The biggest difference between the iterative waterfall model and the traditional waterfall model is the provision of feedback channels from each step to its predecessor phases. The figure 2.1 depicts the feedback pathways that the iterative waterfall model.

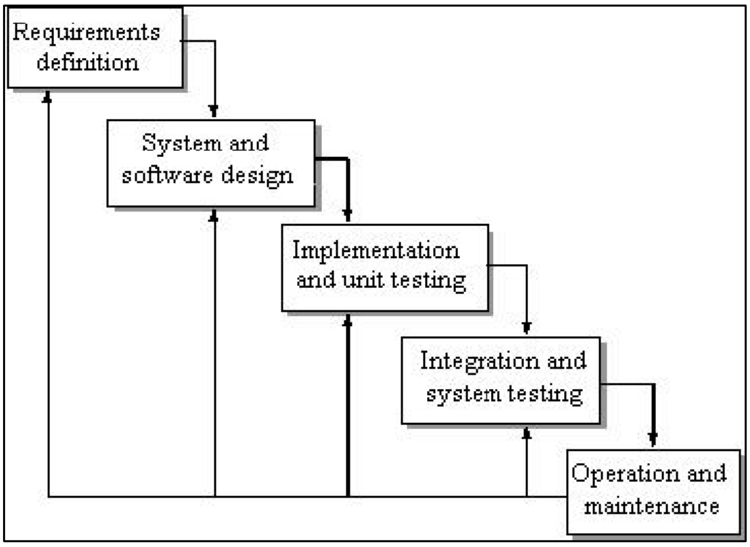

Figure 2.1 Iterative Waterfall Model

i. Requirements Definition

In the first stage of iterative waterfall model is to determine and record the prerequisites for the FYP. In order to do this, it must comprehend the problem statements and do research papers. For example, meeting with FYP's supervisor regularly with no less than 7 times. Other than that, performing literature studies, and generating a thorough requirements document that details the objectives of FYP are also part of this stage.

ii. System and Software Design

Designing the system and structure of the FYP during this stage using the requirements determined during the previous stage. The development of a plan for the implementation stage is the main objective of this stage. The system design step for FYP will entail tasks like drawing a diagram of the system architecture, developing the user interface, specifying the database structure, and establishing the general structure of the project's coding.

iii. Implementation and Unit Testing

In this stage, it begins with constructing FYP in accordance with the design stage. In relation to FYP, the implementation stage will entail tasks like writing the system of university timetable scheduling's code, creating the user interface, coming up with genetic algorithms, incorporating outside libraries or frameworks, and performing routine testing to make sure the functionality is implemented properly.

iv. Integration and System Testing

During integration and system testing, FYP is tested to make sure it adheres to the requirements and functions as expected. The testing and evaluation phase entails tasks including developing test cases, running tests to confirm functioning, documenting and reporting any problems or bugs discovered, and doing acceptability testing to make sure the project hit to desired expectations.

v. Operation and Maintenance

In the final stage of iterative waterfall model, FYP will be released at this point to the evaluator. Preparing the project for presentation and demonstration, documenting the deployment process and possibly even continuing maintenance and support after the project has been deployed.

2.4 Project Requirements

2.4.1 Software Requirement

2.4.1.1 Python

Python is a popular programming language for computers that is used to create software and websites, automate processes, and analyze data.

2.4.1.2 Integrated Development Environment

Programmers can create software code quickly and effectively with the aid of an IDE in the implementation stage of this project. By incorporating features like software editing, building, testing, and packaging into a simple-to-use programme, it boosts developer productivity.

2.4.1.3 Flask Web Development Framework

With the help of helpful tools and capabilities, Flask is a compact and lightweight Python web framework that facilitates the development of online applications. Since you can quickly create a web application using just one Python file, it allows developers flexibility and is a more approachable framework for novice developers.

2.4.1.4 MySQL Workbench Database

Database architects, developers, and DBAs can all use MySQL Workbench as a single visual tool. In-depth administrative tools for server configuration, user management, backup, and many other tasks are provided by MySQL Workbench, along with data modelling and SQL creation.

2.4.2 Hardware Requirement

This system must be developed on a computing device. To create the system for this project, a laptop or desktop can be used. The minimum requirement of computing device is shown as table 2.2.

Table 2.2 Minimum Requirement of Computing Device

Operating System	Microsoft Windows 10
RAM Space	8GB
Disk Space	500GB SSD
Processor Card	Intel® Core™ i5
Graphics Card	NVIDIA GeForce GTX 1080

2.4.3 Other Requirements

Utilizing the timetable data from the FTMK at UTeM is one of these essential requirements. The FTMK scheduling data and dataset provide a plethora of knowledge essential to the project's efficient operation and completion. The FTMK timetable data enhances the project's framework or system by combining real-time data and current information. It offers priceless information on scheduling, number of course, lecture assignments, and lecturer availability, among other crucial aspects.

2.5 Project Schedule and Milestones

The Gantt chart in figure 2.2 below shows the project schedule and milestones.

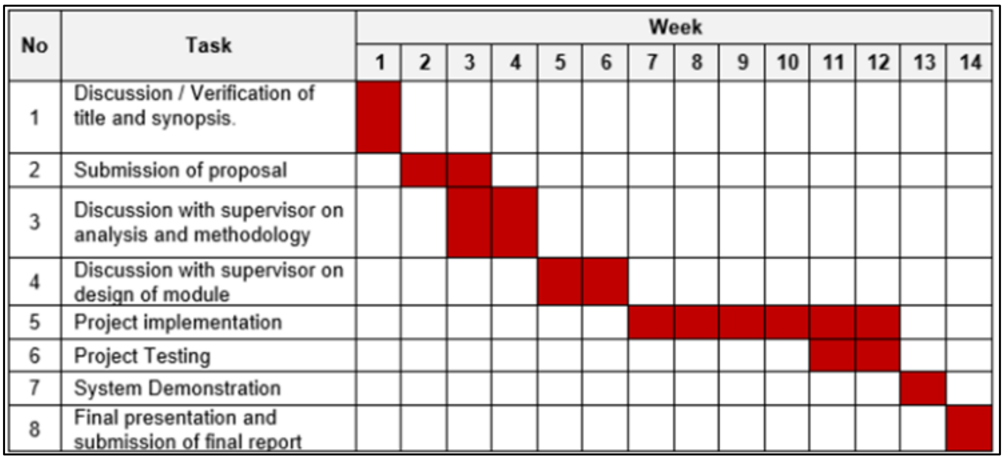

Figure 2.2 Gantt Chart

2.6 Conclusion

In conclusion, conducting a literature study is crucial and should be done at the beginning of a project because it offers information and insight into certain problems. Reviewing the literature can teach us a lot about the advantages and disadvantages of various problem-solving techniques. As a result, we may improve the quality of our project development by further building on the expertise and experiences of others. The study of the projects' requirements and problems is further covered in detail in the following chapter.

CHAPTER 3: ANALYSIS

3.1 Introduction

This chapter discusses and defines a complete study of the genetic algorithm problem applied to the university timetable. For better representation of the issue, the problem's requirement analysis is divided into a number of sections.

3.2 Problem Analysis

The problem statements emphasize the requirement for the timetabling procedure to handle scheduling conflicts and constraints. To prevent scheduling conflicts, the system should make sure that hard constraints are carefully followed. By resolving these issues, the goal of the system is to determine whether the university timetable using GA is generating a optimized timetables to satisfy lecturers and students satisfaction. It entails comprehending how a timetable is planned out and how FTMK students, courses, lecturers, rooms, and time slots are all scheduled and arranged. The analysis also emphasizes discovering the constraints that have an impact on the timeline's effectiveness. Thereby a timetable is efficiently and effectively developed, which will be advantageous to students, lectures, rooms, courses, and the smooth operation of the university as a whole.

To identify the optimum timetables, the system requires a dataset containing university’s data and GA model training. After this step, the GA model's structure is created. The process of utilizing GA for generating optimized timetables is depicted in Figure 3.1.

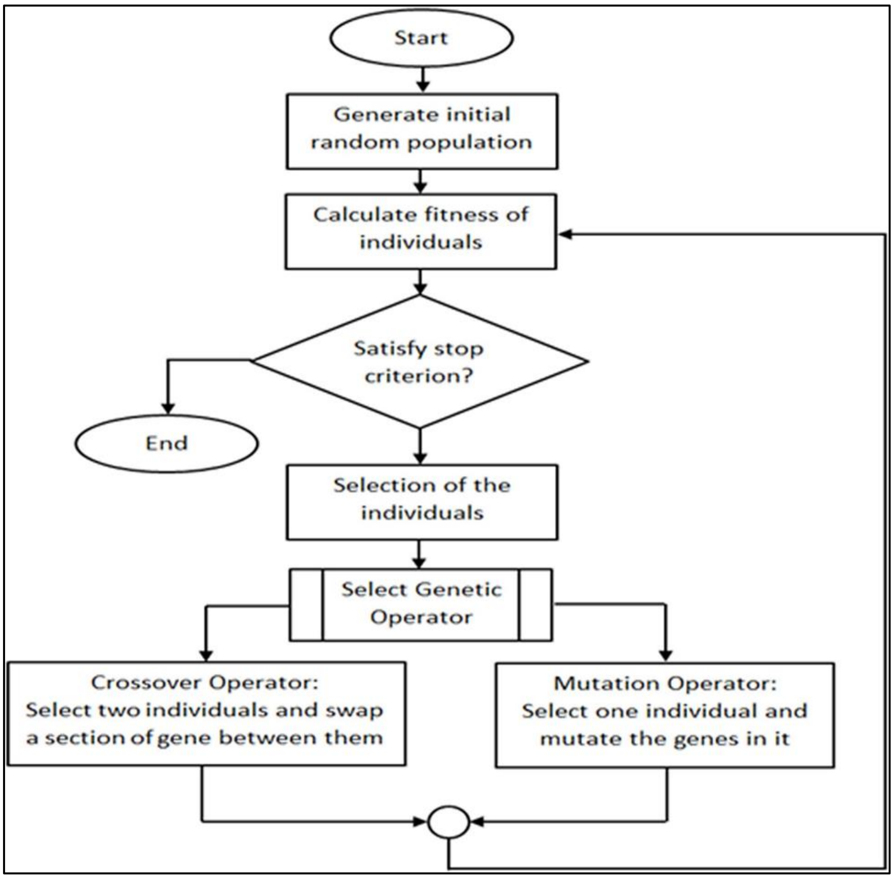

Figure 3.1 Genetic Algorithm (GA) Process Diagram

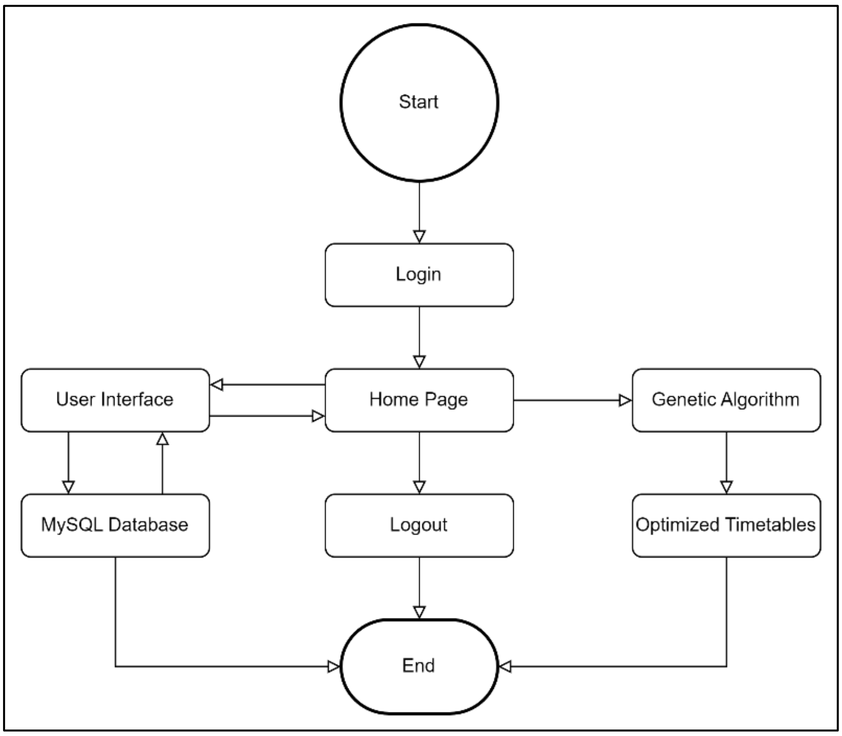

Figure 3.2 Flowchart of the system

The GA model that was developed will be installed in the website. Diagrammatically depicted in figure 3.2 is the model's implementation procedure.

3.3 Requirement Analysis

3.3.1 Data Requirement

This project requires information about the university's timetable data such as courses that provided by the university, lecturers information, batches information, room availability and constraints. These are supplied by UTeM's FTMK. To gain a grasp of their timetable, hard restrictions, and soft constraints through observation, a review of one of UTeM FTMK's has been done on-site. In addition, information about the facilities and how each facility is managed is included in the collected data. To assess the fitness value of each function, information about lecturer preferences and qualifications is also required.

3.3.2 Functional Requirement

This project has two functionals, namely the university timetable scheduling function and the data storing function. The Genetic Algorithm, also known as GA, is the primary model that is applied in the university timetable scheduling function. In the data storing function, the data information that mentioned in Section 3.3.1 Data Requirement will be stored into MySQL database. Therefore, all of the information is incorporated into GA.

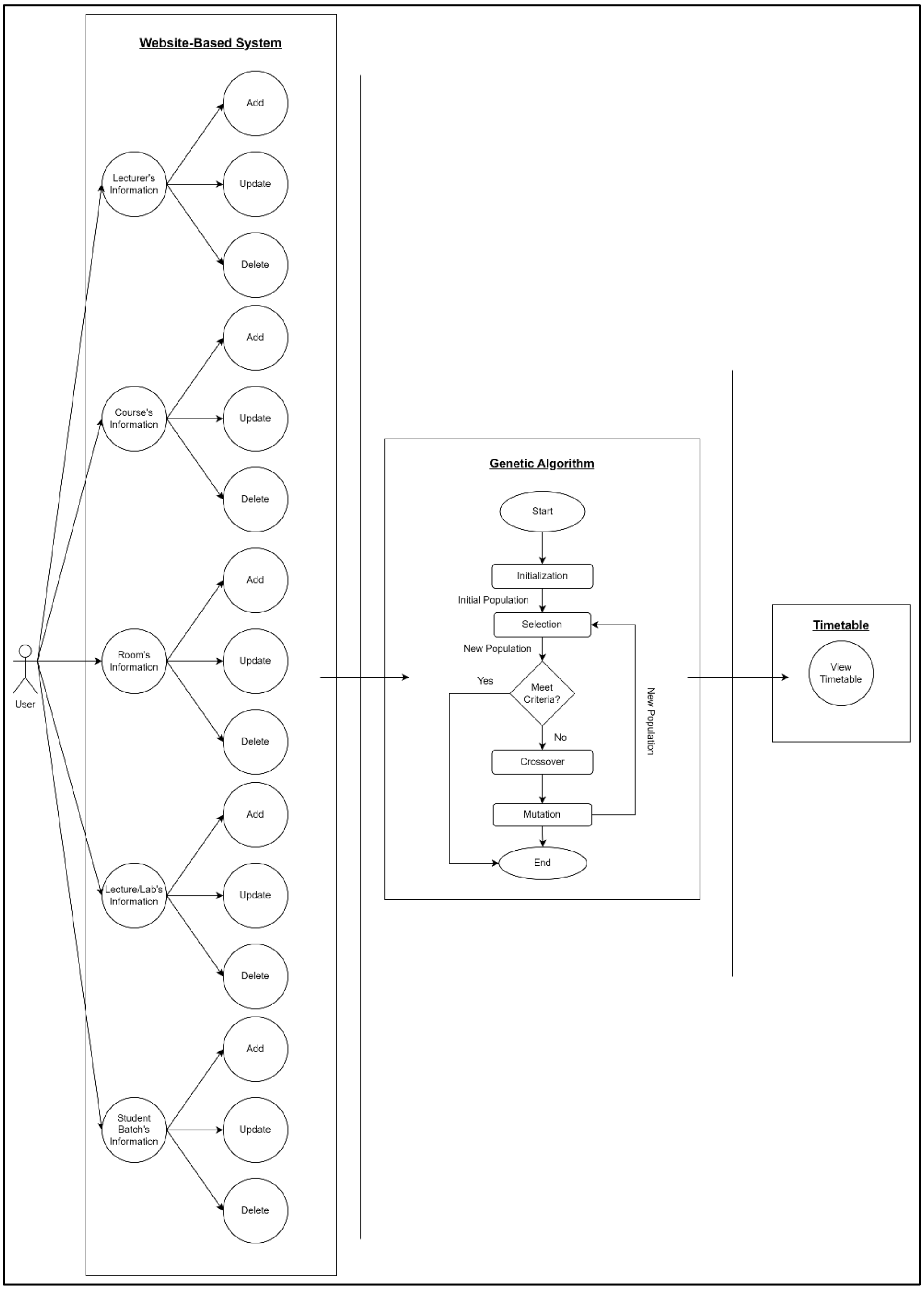

Figure 3.3 Use Case Diagram

3.3.3 Non-Functional Requirement

The non-functional components must be included in the system. Since constructing a system more efficient while also increasing its level of safety. This system cannot be used or utilized until the user inputs or key in the requirement information such as lecturer, room, course, lecture/lab and batch of student to generate timetable. The university timetables should be appropriately evaluated by the fitness function based on the specified constraints. In addition to that, this system is able to provide a timely response to the user whenever the user initiates the process of starting the system.

3.3.4 Other Requirement

3.3.4.1 Software Requirement

(a) Python Programming Language

Python is frequently used to support software developers in a variety of capacities, including build control and management, testing, and many others. SCons is used to manage builds. For automated continuous testing and compilation, use Buildbot and Apache Gump. To organise projects and monitor bugs, use Roundup or Trac.

(b) Visual Studio Code

The entire development process can be completed in a single location using Visual Studio Code. It is a comprehensive IDE that can be used to write, modify, debug, and build code prior to deploying the project.

(c) Flask Web Development Framework

Python is frequently used to assist software developers in numerous capacities, including build control and administration, testing, and many others. SCons is used for build management. Use Buildbot and Apache Gump for continuous automated testing and compilation. Use Roundup or Trac to organize projects and track issues.

(d) MySQL Workbench Database

MySQL databases are created, managed, and administered using MySQL Workbench, a visual database design and modelling tool. It offers a user-friendly interface that enables developers, database administrators, and data analysts to effectively design, build, and administer MySQL databases.

3.3.4.2 Hardware Requirement

This system must be developed on a computing device. The specification that is utilized and advised is listed in table 3.1. This specification can act as a foundational standard for references.

Table 3.1 Hardware Specification

Operating System	Microsoft Windows 11
RAM Space	16GB
Disk Space	500GB SSD + 1TB HDD
Processor Card	Intel® Core™ i5-10300H
Graphics Card	NVIDIA GeForce RTX 2060

3.4 Conclusion

As a summary, the problem has been further clarified for a more thorough explanation in this chapter's investigation of the issue. In order to properly prepare for the design phase of the development process, requirements have been identified. This serves as a foundation for comprehension and lays the groundwork for growth. In the following chapter, it will be covered how a genetic algorithm can be used to create a university timetable.

CHAPTER 4: DESIGN

4.1 Introduction

The outcome of both high-level and detailed design is covered in this chapter. This chapter also features system architecture and algorithm design.

4.2 High-Level Design

4.2.1 System Architecture

A web browser serves as the client for the university's timetable scheduling system, which uses a client-server architecture for hosting its data and logic. The Python web framework Flask is used to implement the system's server-side as figure 4.1. There are many different parts to the system, such as HTML templates, CSS stylesheets, JavaScript code, and Python files. The system's functionality and user interface are delivered by these parts working together.

The server-side code, which is contained in Python files, forms the system's foundation. The logic and routes used to process HTTP requests and provide responses are contained in the auth.py file. For route definition, form data retrieval, SQL query execution, and HTML template rendering, it makes use of the Flask framework. The base.html, lecturer.html, and so on are HTML templates specify the organization and design of the web pages. They are HTML documents with Jinja2 template language code integrated. The templates establish a base template (base.html), which other templates expand to inherit standard layout components using inheritance. Additionally, they contain blocks, which are dynamic content placeholders that each page's unique data fills in.

To specify the visual appearance and styling of the web pages, CSS stylesheets are employed. They have control over the user interface's layout, colors, and other elements, making it uniform and aesthetically pleasing. To make the pages more interactive and dynamic, JavaScript code is incorporated into the HTML templates.

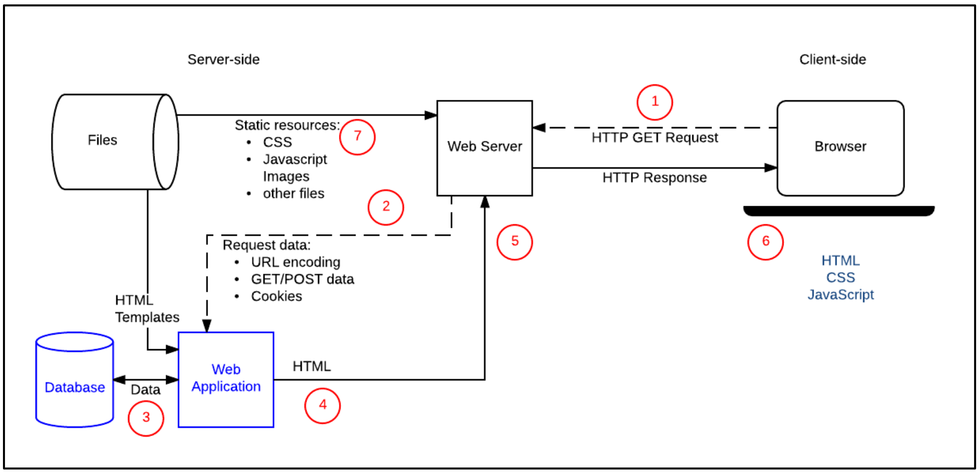

Figure 4.1 Web-Server Architecture

4.2.2 User Interface Design

Using HTML templates, the general design of the user interface is organized, with the common elements defined in a basic template and expanded by unique templates for each functionality. The elements in the templates are responsively styled using Bootstrap classes.

Users would use a browser to view the web application and engage with the system. They can access several pages relevant to the functionalities by navigating. They can add new entries to such pages by completing the forms and sending them in. For simple reference, existing entries are displayed in tables, and corresponding buttons allow users to do actions like delete entries.

4.2.2.1 Navigation Design

A smooth user experience and simple access to the system's various areas and operations are both made possible thanks in large part to the navigation design. The navigation design describes the layout and arrangement of the system's navigational elements, such as menus, links, and buttons. This system appears to include numerous pages or perspectives, including "Lecturer," "Course," and many others. This system probably uses a menu or navigation bar that is present on each page to allow users to switch between various sites. Users can now quickly transition between the system's many sections without losing their place in the overall scheme of things.

Each page may include additional navigational components in addition to the primary navigation. For instance, the "Lecture/Lab" page may feature a form for adding class groups; after the form is submitted, visitors may be taken back to the original page with a success or error message. By including buttons like "Add" or "Submit" on the form, you can make this form-oriented workflow easier for users to complete. This gives them a simple and straightforward way to engage with the system.

4.2.2.2 Input Design

The input design makes sure users fill out the appropriate forms with the necessary data and choose appropriate options. It offers an intuitive interface that assists users in precisely entering the required data. The required validations are carried out by the server-side code, which also communicates with the database to securely save the inputted data. Overall, the web-based system's input design makes it easy to add information without experiencing any difficulty, which improves the system's usability and functionality. The screens of system will be illustrated in figures below.

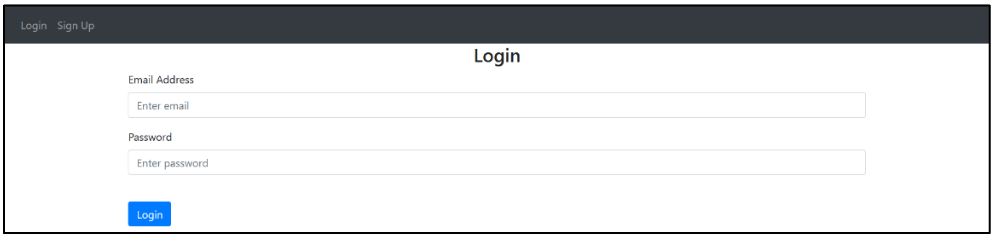

Figure 4.2 Login Screen

In figure 4.2, This screen includes text input fields for the email address and password. The email address and password fields are required, and the system validates the credentials against the database.

Figure 4.3 Sign Up Screen

In figure 4.3, sign up screen includes various input fields such as email address, full name, password for users to create a new account. The email address, full name, and password fields are required. The email field must be in a valid email format, and the password must meet certain requirements like minimum length.

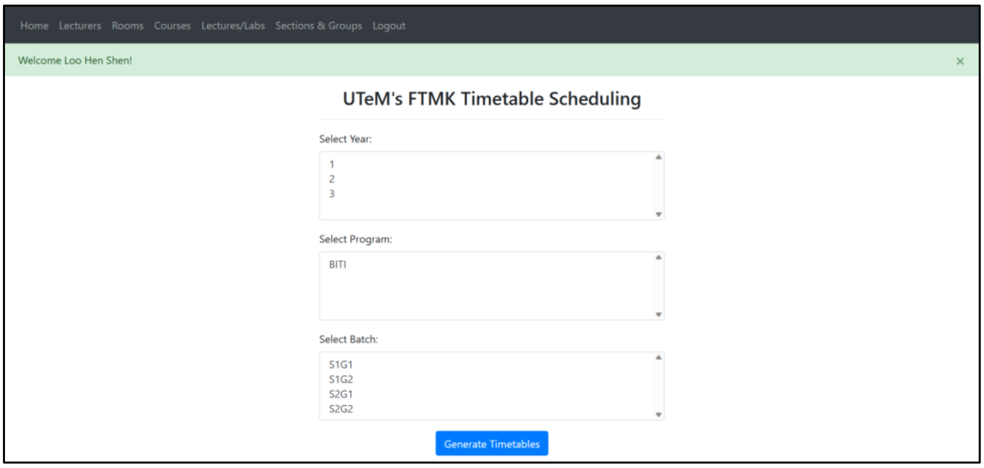

Figure 4.4 Home Screen

In figure 4.4, Home screen allows users to generate and view timetables for different year, program and batch of students. It includes selection boxes for year, program, and batch. The year, program, and batch selection boxes are required. The system validates that the selected combination exists in the database before generating the timetable.

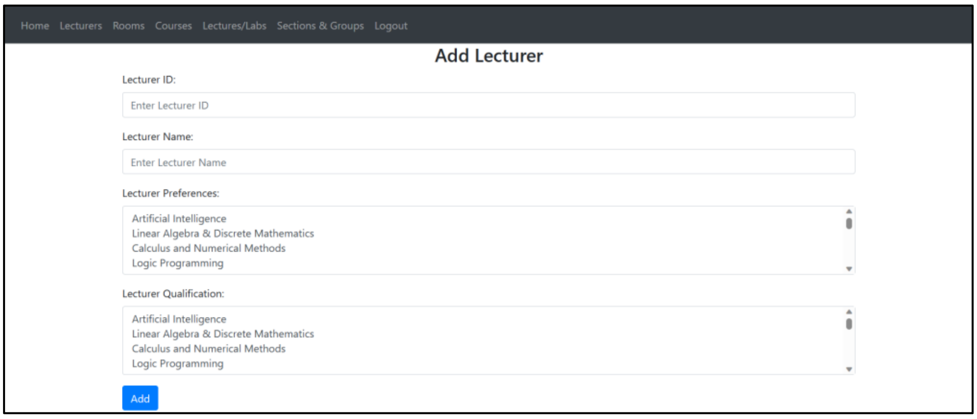

Figure 4.5 Lecturer Screen

In figure 4.5, Lecturer screen includes text input fields such as lecturer's id and lecturer's name and selection boxes for lecturer's preferences and qualifications to add lecturer information along with options for adding, editing, and deleting lecturer. The lecturer's id, lecturer's name, lecturer's preferences and qualifications fields are required. The system verifies the selected courses against the database.

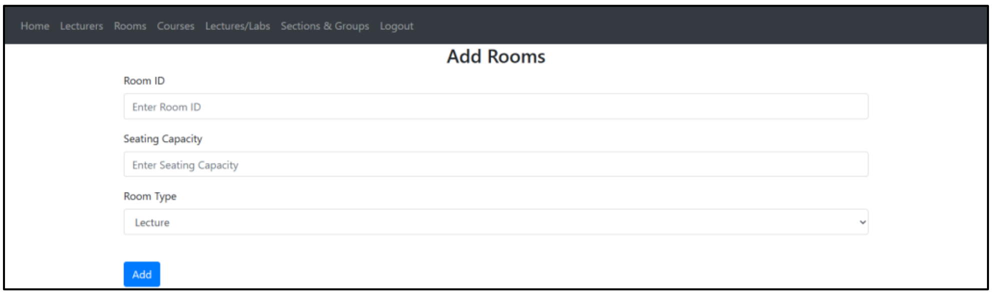

Figure 4.6 Room Screen

In figure 4.6, room screen enables users to manage rooms availability. It includes input fields for room id, seat capacity and selection box for room type. The room id, seat capacity and room type fields are required. The seat capacity must be a positive integer.

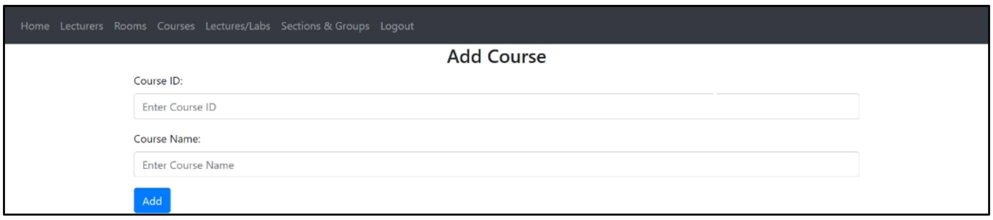

Figure 4.7 Course Screen

In figure 4.7, course screen allows users to manage courses offered by the university. It includes input fields for course ID, course name, as well as options for adding, editing, and deleting courses. The course ID, course name fields are required. The system checks for unique course IDs to prevent duplication.

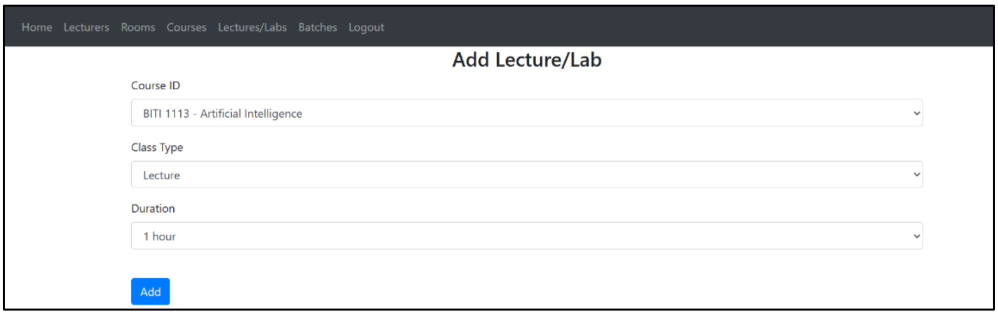

Figure 4.8 Add Lecture/Lab Screen

In figure 4.8, Lecture/Lab screen allows users to add lecture or lab for courses. It includes selection boxes for course ID, class type and duration. The course ID, class type and duration fields is required. The system verifies the selected course ID against the database to ensure their validity.

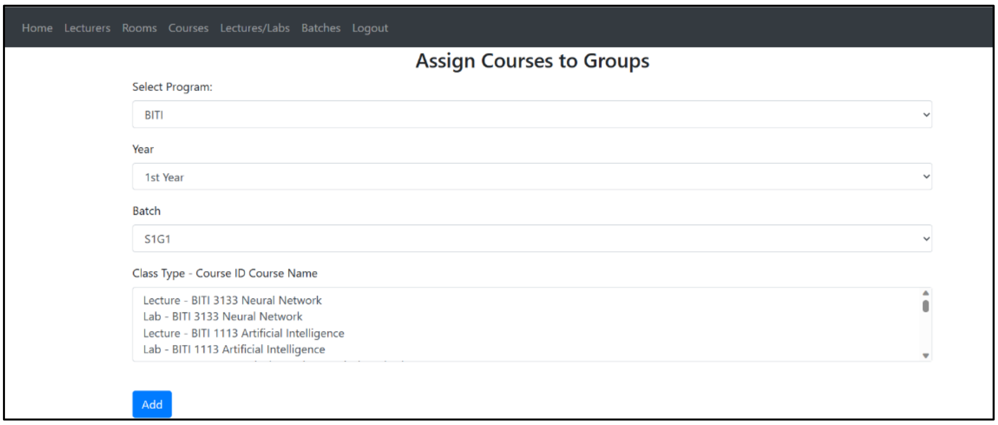

Figure 4.9 Batch Screen

In figure 4.9, batches screen allows users to add batches and assign courses to them. It includes selection boxes for program, year, and batch, as well as a multiple selection box for class types and course IDs. The program, year, and batch selection boxes are required. At least one class type and course ID must be selected from the multiple selection box. The system verifies the selected class types and course IDs against the database to ensure their validity.

4.2.2.3 Technical Design

A genetic algorithm is an artificial intelligence technique used in operations research and computer science to address optimization problems by mimicking the natural selection process. A genetic algorithm uses processes including parent selection, crossover, and mutation in an effort to produce the best result. Therefore, the goal of GA is to begin with a very large population and then gradually diminish size of it until only the best kind of like survival of the fittest. Figure 4.10 shows the process of genetic algorithm in pseudocode.

```
generation = 0;
initialize population;
while generation < max-generation
        evaluate fitness of population members
        for i from 1 to elites
                select best individual
        endfor
        for i from elites to population-size
                for j from 1 to tourmanentsize;
                        select best parents;
                endfor
                for k from elites to population-size*(1-mutationrate)
                        crossover parents -> child;
                endfor
                for k from population-size*(1-mutationrate) to population-size
                        mutate parent->child;
                endfor
                insert child into next generation's population;
        endfor;
        update current population
        generation++;
endwhile;
```

Figure 4.10 Pseudocode of Genetic Algorithm

i. Population Initialization

A group of individuals called the population serves as the process's starting point. Every single individual in the population stands for the solution to the issue that has to be resolved. Each individual has a set of parameters known as genes, which are stick together to form a chromosome. A string which represents the chromosome is used to represent the collection of genes for a certain individual. The figure 4.11 shows how this is done.

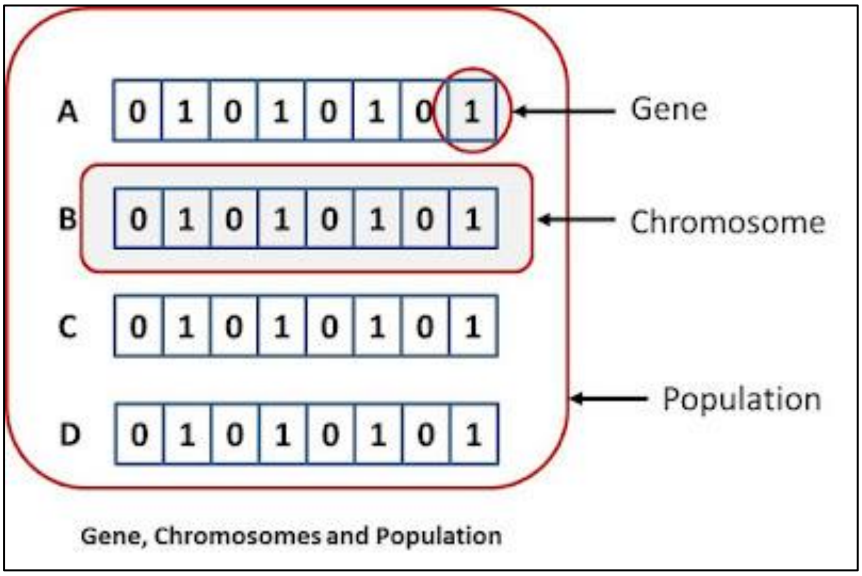

Figure 4.11 Genes, Chromosomes and Population

ii. Evaluation of Fitness

The relation that quantifies a population individual's level of fitness is called the fitness function. Each individual has a fitness score assigned to them. An individual's fitness score will increase or decrease depending on how fit they are. Additionally, the likelihood that an individual will be chosen for reproduction to pass on to the next generation increases with their fitness score.

iii. Parent Selection

This concept decides around choosing a individual of the current generation based on that individual's fitness to breed the next generation. The likelihood of it being chosen increased with increased fitness. Two individuals are chosen throughout the selection procedure to act as the "parents" who will give birth to the following generation.

iv. Crossover

Another name for crossover is recombination. It is a procedure that combines the genetic material from the father and mother to create new offspring. Figure 4.12 provides an illustration of this. As seen in the picture, some of the gene’s crossover to create new sets of genes.

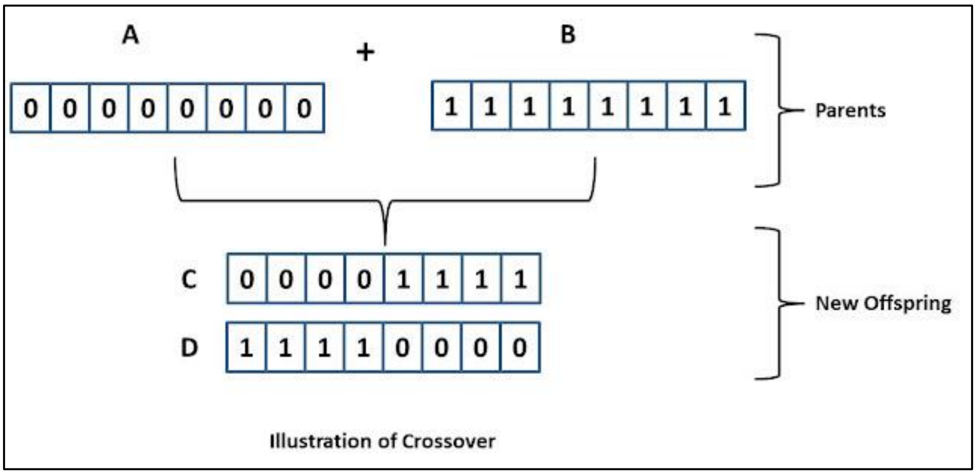

Figure 4.12 Illustration of crossover

v. Mutation

Mutation is just like crossover; both are genetic processes. When there is a mutation, the chromosome's gene organization sort of changes, creating a completely new chromosome. It is quite unlikely that this will occur. In order to explain how species, evolve into the current species, evolutionists tend to capitalize on mutation. But there is no scientific law on which this is founded.

vi. Termination Condition

An algorithm must end at some point. Convergence is another name for this procedure. Thus, unless a set of termination requirements is met, the algorithm iterates until a new generation is produced that is not significantly different from the one before. At this point, not much changes, and the process is said to be terminated.

4.2.2.4 Output Design

A table containing the current data, such as lectures and labs, is displayed upon submission after the user has selected the necessary information. The table includes the course ID, lecture/lab ID, lecture/lab name, and duration. The table is filled with information retrieved from the database, with each entry denoting a lecture or lab. The forms on those pages allow users to add new records to the corresponding database tables. Additionally, they present current records in a tabular manner so that users may view and control the data. Error messages are shown when it is important to alert users to any problems or limitations relating to data input and database activities.

4.2.3 Database Design

4.2.3.1 Conceptual and Logical Database Design

This system's database has the following 8 entities: Lecturer, Course, Room, Lecture/Lab, Section & Group, Lecturer's Preferences, Lecturer's Qualification, and User. To set each entity apart from other records in the same database, each one has a primary key. The lecturer entity contains two fields: lecturer_name and lid. The lid stores the lecturer's ID in integer format. A text field called lecturer_name contains the lecturer's name. There are two fields in the Course entity: cid and course_name. The cid naturally records the id in integer format. The text parameter course_name contains the name of the course. Email, password, and full_name are the only three fields of the User entity. The user's email is kept on file. The user's password is stored in the password box. The entire name of the user is indicated via the text field full_name.

The three fields of the Room entity are rid, seat, and room_type. The room's id is recorded in the rid in integer format. The seat field is an integer text field that represents the number of seats in the room. A varchar value for room_type specifies whether the room has been designated for a lecture or lab. Gid, cid, lecture/lab_name, and duration are the four fields that make up the Lecture/Lab entitiy. The gid stores the lecture or lab ID in integer format. A foreign key referencing the Course entity is the cid field. The class_type field is a selection box that contains the lecture or lab. The duration value keeps track of how long a lecture or lab lasts in hours.

There are five fields in the Section & Group entity: sgid, gid, section/group_name, year_level, and program. The SGID keeps track of section and group ids in integer format. A foreign key referencing the Lecture/Lab entity is the gid field. The student's section and group are indicated in the section/group_name field, which is an integer text field. The year_level identifies the student's level of study. The student's program is recorded in the program value. There are three fields in the Lecture's Preferences entity and Lecture's Qualifications entity: id, lid, and cid. The integer-formatted id of the lecture's preferences is recorded in the id. The lecturer and course entities are referenced by the lid and cid fields, respectively.

A many-to-many relationship exists between the lecturer and course elements. A lecturer may be connected to several different courses, and a course may have several lecturers. By referencing the main keys (lid and cid) of the lecturer and course entities in a junction table, this relationship is built. One-to-many relationships define the relationship between the room and lecture or lab entities. Although a lecture or lab can be assigned to more than one room, a particular lecture or lab can only be assigned to one room. In the Lecture/Lab entity, a foreign key that refers to the main key (rid) of the Room object serves as a representation of this relationship. A one-to-many relationship exists between the lecture/lab and section and group entities. A section or group can be a part of more than one lecture or lab, but only one lecture or lab is linked to that section or group. A foreign key in the Section and Group entity that refers to the main key (gid) of the Lecture/Lab entity establishes this relationship. The user and lecturer entities have a one-to-one relationship with one another. Each lecturer relates to a single user, and each user relates to a single lecturer. A foreign key in the Lecturer entity that refers to the main key (lid) of the User entity serves as a representation of this relationship.

A one-to-many relationship exists between the lecturer's preferences and lecturer entities. A lecturer may have many preferences, but only one lecturer is connected to each preference. A foreign key in the Lecturer's Preferences object that refers to the primary key (lid) of the Lecturer entity establishes this link. A many-to-many relationship exists between the lecturer's preferences and course entities. A course may have many preferences, and a preference may be related to multiple courses. A junction table that makes use of the primary keys (id and cid) of the lecturer's preferences and course entities serves as a representation of this relationship.

A one-to-many relationship exists between lecturer entities and the lecture's qualifications. Multiple qualifications are possible for a professor, but only one lecturer is assigned to each qualification. This connection is made possible through a foreign key that points to the primary key (lid) of the Lecturer entity in the Lecture's Qualifications entity. A many-to-many relationship exists between the course entities and the qualifications of the lecture. A course may have numerous credentials, and a qualification may be related to multiple courses. A junction table that makes use of the primary keys (id and cid) of the Qualifications and Course entities of the lecture serves as a representation of this link.

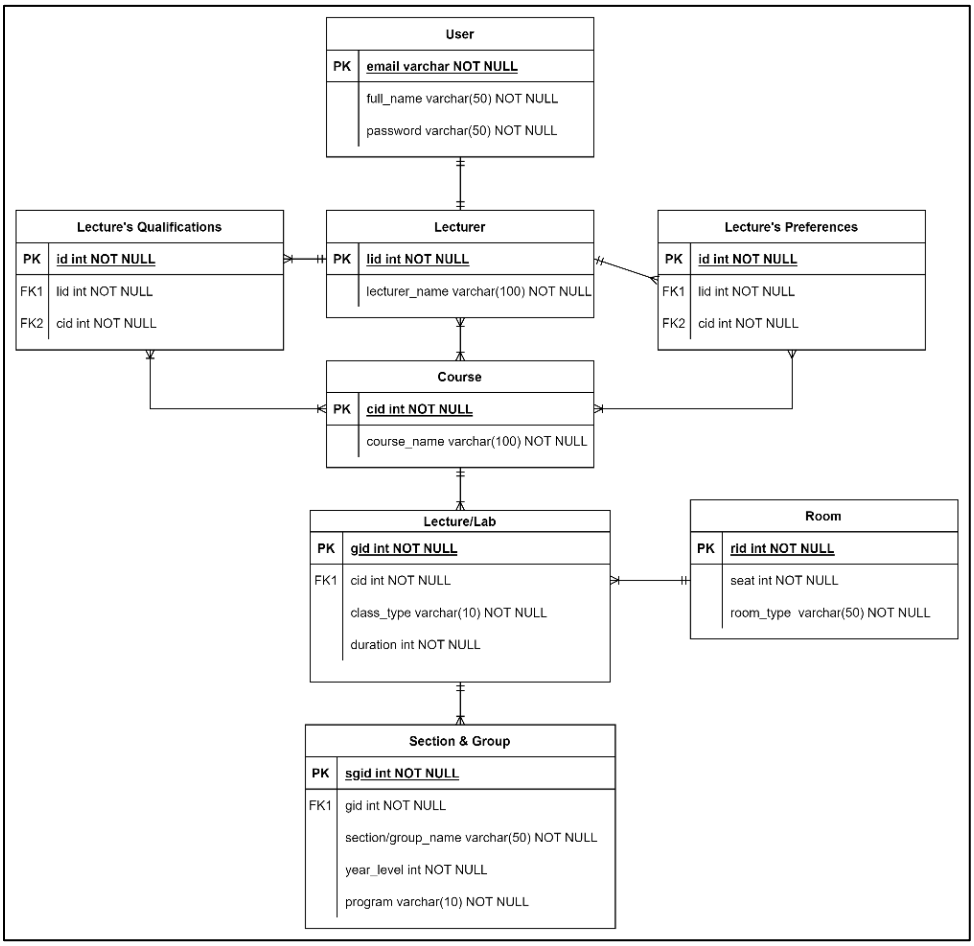

Figure 4.13 Entity Relationship Diagram

4.3 Detailed Design

The broader category of evolutionary algorithms (EA) includes genetic algorithms (GA), which are metaheuristics modelled after the process of natural selection. By relying on biologically inspired operators such as mutation, crossover, and selection, genetic algorithms are frequently employed to produce high-quality solutions to optimization and search problems.

4.3.1 Software Design

4.3.1.1 Chromosome Representation

The chromosome serves as a possible timetable or schedule for the problem of scheduling at a university. It encrypts the scheduling of lectures and laboratories for each day of the week at particular times and rooms. Each element of the chromosome's list or array, which is used to represent it, represents a specific hour of the day. The number of timeslots in a week is equal to the length of a chromosome. Each chromosome element holds details about the room, lecturer, and lecture or lab allocated for that time slot.

The chromosomes in Figure 4.14 illustrates the chromosome representation. Each chromosome includes 92 genes or called classes. Each genes have 11 list values, which are day, hour, consecutive_hour, gid, cid, duration, room_id, lecturer_name, year, program, student_batch. For instance, {"Monday", 9, 10, 5, "Lecture", "Statistic & Probability", "BK3", "Dr Yaakob", "1", "BITI", "S1G2"}.

Chromosome	**{"Friday", 11, 12, 7, "Lecture", "Evolutionary Computing", "BK1", "Dr Zeratul", 2, "BITI", "S1G1"}, …. (90 classes) …., {"Monday", 9, 10, 4, "Lab", "Artificial Intelligence", "AI3", "Dr Elle", 1, "BITI", "S1G1"}**

Figure 4.14 Chromosome Representation

4.3.1.2 Initialization Method

The algorithm begins with an initialization step that creates an initial population of timetables. Each timetable is represented as a list of timeslots, where each timeslot corresponds to a scheduled class with details such as day, hour, room, course, and other constraints. The initialization process involves generating random schedules for classes while adhering to the given constraints. This diverse initial population serves as the starting point for the evolutionary process of selecting, recombining, and mutating solutions to iteratively improve the quality of the timetables.

4.3.1.3 Fitness Evaluation

The fitness evaluation of the genetic algorithm described involves assessing the quality of candidate timetables generated within the algorithm's population. The goal is to measure how well a timetable satisfies both hard and soft constraints inherent to a university timetable scheduling problem. The fitness function computes a fitness score for each individual timetable, quantifying its suitability as a solution. Formula are shown below.

Fitness Score = (1 - (VIOLATED_HARD_CONSTRAINTS / TOTAL_HARD_CONSTRAINTS)) * 0.85 + (FULFILLED_SOFT_CONSTRAINTS / TOTAL_SOFT_CONSTRAINTS) * 0.15

Hard constraints are crucial scheduling conditions that must not be violated. The first hard constraint ensures that rooms are not allocated to different classes at the same day and time. Additionally, consecutive classes in the same room are accounted for. Another constraint prevents classes from being scheduled during the 13:00 - 14:00 period. Another hard constraint enforces that students and lecturers are not simultaneously scheduled. Lastly, the class's room type should align with its class type. Soft constraints are desirable conditions that improve a timetable's quality but may be sacrificed to a degree. Soft constraints involve preferences, like preferred lecturers for a class. These are not mandatory but contribute positively to the fitness score if fulfilled. The hard constraints and soft constraint equations are demonstrated as below:

Hard Constraints:

1. Room Assignment Constraint:

   For each timeslot, ensure that a room is not assigned on the same day and hour: $\sum_{r \in Rooms} x_{r,t} \leq 1$, where $x_{r,t}$ is a binary variable representing whether room r is assigned at timeslot t.

2. No Class During 1:00 pm Constraint:

   Ensure that a class is not assigned during specific hours: $\sum_{t \in hour} \sum_{r \in Rooms} x_{r,t} = 0$, where t represents specific hours.

3. Student-Lecturer Clash Constraint:

   Ensure that students and lecturers are not assigned at the same time: $\sum_{t \in Timeslots} \sum_{r \in Rooms} \sum_{c \in Classes} y_{c,t,r} + \sum_{t \in Timeslots} \sum_{r \in Rooms} \sum_{l \in Lecturers} z_{l,t,r}$

   ≤ 1, where $y_{c,t,r}$ is a binary variable indicating if class c is in room r at timeslot t, and $z_{l,t,r}$ is a binary variable indicating if lecturer l in room r at timeslot t.

4. Room Type Compatibility Constraint:

   Ensure that a class is assigned to a room with a compatible type: $x_{r,t} \leq y_{r,c}$, where $y_{r,c}$ is a binary variable representing room r's compatibility with class type c.

5. Qualified Lecturer Constraint:

   Ensure that a class is assigned by a qualified lecturer: $\sum_{t \in Timeslots} \sum_{r \in Rooms} \sum_{l \in Lecturers} z_{l,t,r} \geq 1$, where $z_{l,t,r}$ is a binary variable indicating if lecturer l is in room r at timeslot t.

Soft Constraint:

1. Preferred Lecturer Constraint:

   Encourage assigning a class to a preferred lecturer: $\sum_{t\in Timeslots}\sum_{r\in Rooms}\sum_{l\in Preferred\ Lecturers} z_{l,t,r} \geq 1$, where $z_{l,t,r}$ is a binary variable indicating if preferred lecturer l is in room r at timeslot t.

The fitness function computes the fitness score by incorporating the ratio of violated hard constraints to the total hard constraints. This ratio is multiplied by a weight (0.85) to emphasize the importance of hard constraints. Similarly, the ratio of fulfilled soft constraints to total soft constraints is computed and multiplied by a weight (0.15). The two weighted scores are combined to yield the final fitness score. A higher fitness score indicates a timetable with fewer constraint violations and better adherence to preferences.

4.3.1.4 Parent Selection

The parent selection process in the given genetic algorithm involves the use of a tournament selection strategy. This strategy aims to select individuals from the current population to serve as parents for generating the next generation of offspring. The selection process is performed iteratively for each individual in the population, with each iteration constituting a tournament.

During a single tournament, a fixed number of individuals (tournament size) are randomly chosen from the current population. These individuals then compete against each other to determine the winner of the tournament. The winning individual, also known as the "winner," is the one with the highest fitness score among the tournament participants. The fitness score of an individual indicates how well it satisfies the problem's constraints and objectives.

Importantly, individuals with empty timetables (no assigned classes) are removed from the tournament since they represent invalid solutions. Once the winner of the tournament is determined, it is added to the list of selected parents. This process is repeated for all individuals in the population, resulting in a list of selected parents that will be used for the genetic operators of crossover and mutation to create the next generation of offspring.

4.3.1.5 Crossover & Mutation

The crossover and mutation operations are essential components of the evolutionary process that drives the search for optimal solutions. These operations simulate the natural processes of genetic recombination and variation, allowing the algorithm to explore and exploit the solution space more effectively.

Crossover is a genetic operator that combines genetic material from two parent individuals to create new offspring. In the context of this algorithm, parents are selected from the population based on their fitness scores. The algorithm employs uniform crossover, where each gene (or element) of the offspring is randomly chosen from one of the corresponding genes of the parents. A binary mask is used to determine which genes come from which parent. This introduces diversity into the population by recombining promising features from different individuals, potentially leading to offspring with improved fitness.

Mutation is another genetic operator that introduces small, random changes into an individual's genetic makeup. In this algorithm, mutation is applied to the offspring with a certain probability (mutation rate). If a gene is selected for mutation, the algorithm swaps the values of two randomly chosen genes within the individual. This introduces exploration into the search process, allowing the algorithm to escape local optima and discover novel solutions. Mutation can be crucial for maintaining genetic diversity within the population and preventing premature convergence to suboptimal solutions.

4.3.1.6 Survival Selection

The survival selection process involves identifying a subset of individuals that exhibit higher fitness scores, favoring the preservation of the most promising solutions while promoting diversity and evolution.

The process starts by merging the current population with the offspring generated through crossover and mutation. The fitness scores of both the original population and the offspring are combined into a single list. This combined pool of individuals and their associated fitness scores is then used to select the next generation.

During selection, a predetermined number of individuals, usually equal to the population size, are chosen as survivors. The selection process employs a straightforward strategy known as "maximization selection." In this strategy, individuals are chosen one by one, iteratively. In each iteration, the individual with the highest fitness score is selected from the combined pool. The selection is repeated until the desired number of survivors is reached.

This approach is advantageous for several reasons. First, it ensures that individuals with higher fitness scores are more likely to be selected, improving the overall quality of the population. Second, it allows for the preservation of genetic material that has contributed to successful solutions in previous generations. Finally, by considering both the original population and the offspring, the algorithm avoids prematurely discarding potentially valuable genetic material and promotes genetic diversity, which is essential for avoiding convergence to local optima.

4.3.1.7 Iteration of Generations and Termination

The genetic algorithm iterates through a series of generations, gradually improving the population's solutions through selection, crossover, and mutation operations. The algorithm's termination is based on a predefined number of generations, and the final output is the best timetable solution found during the iterations. Each generation involves a set of steps that collectively aim to improve the quality of the generated timetables. The algorithm's goal is to evolve a population of timetables over multiple generations to find the best possible timetable that satisfies various constraints and preferences.

4.3.2 Physical Database Design

A number of actions must be completed in order to implement the logical design of the MySQL database in the DBMS. In order to define the basis tables and their structure, Data Definition Language (DDL) statements that correspond to the logical design must first be translated into DDL statements. In our instance, the foundation tables comprise lecturer, room, course, section&group, lecture/lab, lecturer_preferences, lecturer_qualifications, and user are shown as figures below.

Designing and enforcing additional business rules using constraints and validations comes after the base tables have been developed. The guidelines and requirements that the tables' data must follow must be specified in order to accomplish this. For instance, we can apply constraints to guarantee that a lecturer's ID (lid) in the lecturer_preferences and lecturer_qualifications columns refer to a legitimate lecturer ID from the lecturer table. To guarantee that a field includes the correct value, we can similarly establish constraints on fields like seat at the room table.

```
-- Create the lecturer table
CREATE TABLE IF NOT EXISTS lecturer (
    lid INT PRIMARY KEY AUTO_INCREMENT,
    lecturer_name VARCHAR(255)
);
```

Figure 4.15 DDL of Creating Lecturer Table

```
-- Create the room table
CREATE TABLE IF NOT EXISTS room (
  rid INT PRIMARY KEY AUTO_INCREMENT,
  room_type VARCHAR(50) NOT NULL,
  seat INT NOT NULL
);
```

Figure 4.16 DDL of Creating Room Table

```sql
-- Create the course table
CREATE TABLE IF NOT EXISTS course (
    cid INT PRIMARY KEY AUTO_INCREMENT,
    course_name VARCHAR(255)
);
```

Figure 4.17 DDL of Creating Course Table

```sql
-- Create the section&group table
CREATE TABLE IF NOT EXISTS sectiongroup (
    sgid INT PRIMARY KEY AUTO_INCREMENT,
    gid INT,
    sectiongroup_name VARCHAR(255),
    year_level INT,
    program VARCHAR(255),
    FOREIGN KEY (gid) REFERENCES lecturelab(gid)
);
```

Figure 4.18 DDL of Creating Section & Group Table

```sql
-- Create the lecturelab table
CREATE TABLE IF NOT EXISTS lecturelab (
  gid INT PRIMARY KEY AUTO_INCREMENT,
  cid INT,
  duration INT NOT NULL,
  class_type VARCHAR(10) NOT NULL,
  FOREIGN KEY (cid) REFERENCES course(cid)
);
```

Figure 4.19 DDL of Creating Lecture/Lab Table

```sql
-- Create the lecturer_preferences table
CREATE TABLE IF NOT EXISTS lecturer_preferences (
    id INT PRIMARY KEY AUTO_INCREMENT,
    lid INT,
    cid INT,
    FOREIGN KEY (lid) REFERENCES lecturer(lid),
    FOREIGN KEY (cid) REFERENCES course(cid)
);
```

Figure 4.20 DDL of Creating Lecturer_Preferences Table

```sql
-- Create the lecturer_qualifications table
CREATE TABLE IF NOT EXISTS lecturer_qualifications (
    id INT PRIMARY KEY AUTO_INCREMENT,
    lid INT,
    cid INT,
    FOREIGN KEY (lid) REFERENCES lecturer(lid),
    FOREIGN KEY (cid) REFERENCES course(cid)
);
```

Figure 4.21 DDL of Creating Lecturer_Qualifications Table

```sql
-- Create the user table
CREATE TABLE IF NOT EXISTS user (
    email VARCHAR(255) PRIMARY KEY,
    full_name VARCHAR(255),
    password VARCHAR(255)
);
```

Figure 4.22 DDL of Creating User Table

4.4 Conclusion

In both high-level and specific contexts, this chapter provides a summary of the system's design. The implementation of the system is covered in the following chapter.

CHAPTER 5: IMPLEMENTATION

5.1 Introduction

In this chapter, the project delves into the core activity, using a genetic algorithm to construct the university timetable. In this stage, theoretical ideas are translated into practical solutions, which leads to the creation of academic schedules that are optimized. Thorough review of the implementation process in this chapter, emphasizing its essential elements, methodology, and anticipated results. This chapter's main goal is to provide a thorough explanation of how the genetic algorithm described in Chapter 4 is transformed into a useful system that can resolve the challenging university scheduling problem. The system's expected output is a description of the ideal class schedule for the university, along with the creation of Excel files to track fitness trends and results.

5.2 Software Development Environment Setup

Software development environment is designed to support the efficient implementation of the genetic algorithm while ensuring seamless interaction with the underlying database. This environment involves the following key elements:

1. **Programming Language:** Python

   The genetic algorithm will be implemented using the Python programming language due to its versatility and extensive libraries support.

2. **Database Management System:** MySQL

Utilizing MySQL as the relational database management system to store and manage data related to class groups, rooms, lecturers, courses, and constraints.

3. **Integrated Development Environment (IDE):** Visual Studio Code

   Visual Studio Code will serve as the primary IDE for coding, debugging, and testing the algorithm.

The client software, which was created using the Python programming language, lies at the heart of this system. The genetic algorithm's algorithmic logic is embodied in the client program, which includes procedures including population initialization, fitness assessment, selection, crossover, mutation, and survivor selection. This element serves as the solution's brain, coordinating the generational evolution of schedules.

The server software, which is built on the MySQL database management system, is a complement to the client software. The MySQL Server is in charge of holding the database that houses crucial information on class, rooms, lecturers, courses, and constraint data. This server-side component offers the algorithm a structured repository from which it can retrieve and modify data as needed.

Secure network communication channels make it easier for client and server software to interact. The client program connects to the MySQL server through a designated IP address and port number while being hosted within an integrated development environment (IDE) like Visual Studio Code. Through this link, the algorithm may obtain the information it needs to create the best possible timetable, such as information about each class, room assignments, and lecturer preferences.

5.3 Software Configuration Management

5.3.1 Configuration environment setup

For this project to maintain control over its development lifecycle and guarantee that changes are tracked, managed, and documented in a methodical way, an efficient configuration management system must be set up. A well-organized configuration management system aids in preserving stability, streamlining teamwork, and enabling trustworthy version control.

This project makes use of a suitable version control system, Git, to track changes made to the project's source code, documents, and other materials. Git offers a branching technique that enables concurrent work on many features, fixes, or experiments while maintaining the integrity of the main codebase. It establishes rules for branching off from a default branch called "main" at the beginning. Make branches for releases, bug fixes, and certain features. To guarantee stability, refrain from making changes directly to the main branch. Backups of the version control repository are performed often. The project's history and data are protected from possible losses thanks to this measure.

5.3.2 Version Control Procedure

Through a version control system (VCS), the source code is systematically managed as part of the version control process. This involves choosing a suitable VCS, Git, setting up a central repository, deciding on a branching strategy, making and working in branches for features and fixes, committing changes with helpful messages, performing code reviews, integrating continuous integration for automated testing, and releasing versions by merging approved changes. Throughout the development lifecycle, access control, dispute resolution, and thorough documentation enable efficient cooperation, high-quality code, and project stability.

Setting up organized systems to manage the project's source code versions is known as version control. These entails putting in place access controls and permissions to control code contribution and merging, enforcing code approval through review processes, resolving conflicts that may arise during branch merging, managing the release of new versions with obvious documentation and tagging, maintaining thorough documentation outlining version control procedures and guidelines, and making sure that routine backup and recovery procedures are in place to protect the integrity of the codebase. Together, these controls guarantee the development process's security, stability, and effectiveness while maintaining a well-documented history of code changes and supporting successful project management.

5.4 Implementation Status

5.4.1 Website-based Information System

The front-end of website-based information system is the user interface module combined with database module. It offers a user-friendly interface that enables users to access different features and modules and manage their data. The implementation tasks for the website-based information systems are utilized less time compare to the university timetabling using genetic algorithm module. It is completed in June, 2023, awaiting integration from next module.

Table 5.1 Website-based Information System's Implementation

Task	Description	Duration (day)	Date Completed
Database design	Identify the entities and attributes, define relationships, normalize data.	7	1st April 2023
Database creation	Execute SQL commands to create the database and tables.	3	7th April 2023
Administration	Create sign up, log in, and forgot password function.	2	9th April 2023
Basic CRUD to database	Create web pages, functions and forms for CRUD to database.	7	16th April 2023
Form validations	Create form validations functions.	2	18th April 2023
Bootstrap	Enhance appearance of web system.	1	19th May 2023

Data transfer	Inserting user information into MySQL database table.	1	20th May 2023
Integrate with genetic algorithm	Connect the web system with genetic algorithm running in background.	3	1st June 2023

5.4.2 University Timetabling using Genetic Algorithm

The main part of the system that creates the university timetables is the genetic algorithm module. The implementation task of this module is used the most of the time in the last step of fine-tuning the algorithm. In this step, trial-and-error, observation on graphs, adjusting operators and strategies combinations are needed to produce more satisfactory result. This module has been completed in August, 2023.

Table 5.2 University Timetabling using Genetic Algorithm's Implementation

Task	**Description**	**Duration (day)**	**Date Completed**
Problem formulation	Formulate the problem scope in mathematical notation.	5	21th April 2023
Algorithm design	Design algorithm in detail.	3	24th April 2023
Initialization implementation	Create population initialization function.	7	1st May 2023
Fitness function design	Design fitness function.	5	6th May 2023

Fitness function implementation	Create fitness function evaluation function.	7	13th May 2023
Parent selection, Crossover, mutation, survival selection	Create parent selection, crossover, mutation, and survival selection function	7	20th May 2023
Termination criteria, and generation looping	Create termination criteria function, link the functions together in loops.	3	20th May 2023
Line graph	Create function to display graphs. Conduct analysis.	2	10th June 2023
Fine tune algorithm	Tweak the algorithm's operators, repeat above task for better result	30	20th August 2023

5.5 Conclusion

This chapter examined the version control process that protects code integrity and collaborative development, developed a strong software configuration management system, and outlined how to set up a software development environment. Each module's implementation status was displayed, including the development, due dates, and software specifications of important components. The fundamental principles of the genetic algorithm, database interactions, constraint translation, fitness evaluation mechanism, evaluation and analysis, result output, integration, and testing, and all of these, were carefully developed and integrated. The emphasis now changes to the activities that will carry the project through to completion after this chapter is complete. The robustness, scalability, and solution quality of the algorithm will be rigorously tested using a variety of datasets.

CHAPTER 6: TESTING

6.1 Introduction

In this chapter, this project delves into the crucial phase of testing, which is a pivotal step in ensuring the effectiveness and reliability of the developed university timetable using genetic algorithm system. The testing phase serves as a critical evaluation of the system's functionality, performance, and adherence to the hard and soft constraints. By subjecting the system to various scenarios and input data, it aims to validate its ability to produce accurate and feasible timetables while maintaining the integrity of the imposed constraints.

To ensure comprehensive testing, a well-defined testing strategy has been devised. This strategy encompasses a range of testing methodologies, including unit testing, integration testing, and system testing, each tailored to address specific aspects of the system's functionality. Additionally, the testing strategy outlines the procedures for testing the system's adherence to both hard and soft constraints, ensuring that the timetables generated are not only feasible but also optimized according to user preferences.

6.2 Test Plan

6.2.1 Test Organization

The testing phase of the university timetable using genetic algorithm system was conducted under the management and supervision of a single individual, namely myself. As the primary developer and project lead, the developer undertook the responsibility of planning, executing, and overseeing the university timetable using genetic algorithm module’s performance. This role encompassed the formulation of various operators and strategies, and needed to be tested by different combinations, and the execution of these tests against the system. By assuming a hands-on approach to testing, the developer ensured that the testing process remained consistent, thorough, and aligned with the project's objectives. Throughout this phase, the developer also documented the graphs of best fitness score results, and average fitness score of the module and improvise the algorithm.

6.2.2 Test Environment

The testing for the university timetable using genetic algorithm system was conducted in a controlled and dedicated environment. The testing environment consisted of a standard laptop computer with the following hardware specifications: Intel Core i5 processor, 16GB RAM, and a solid-state drive. The system was developed to be platform-independent, thus allowing testing to be carried out on both Windows and Linux operating systems.

Prior to the testing phase, the required database management system (DBMS) was set up to store the necessary data, including class group information, course details, room specifications, and lecturer preferences. The DBMS was configured to simulate a real-world scenario with various classes, courses, lecturers, and rooms, ensuring the system's capacity to handle diverse data sets.

In addition to empirical testing, statistical significance tests were conducted to assess the system's performance in terms of fitness evaluation and constraint fulfillment. This involved running multiple iterations of the system and analyzing the distribution of fitness scores and constraint violations to ensure consistency and accuracy in the results.

6.2.3 Test Schedule

The testing of the university timetable using genetic algorithm system was executed in a series of iterative cycles, each designed to assess the system's performance, identify areas of improvement, and refine its functionality. The testing process comprised a total of five cycles, with five cycles lasting approximately one week. This duration allowed for comprehensive testing and analysis while also accommodating any necessary adjustments to the system.

During each testing cycle, a set of diverse and representative test cases was selected to cover between 100 and 550 populations, different combinations of crossover and mutation operators, both constraints, and input data combinations. The system was then subjected to these test cases to evaluate its ability to generate optimal timetables while satisfying both hard and soft constraints. The outcomes of each cycle were meticulously analyzed to determine the system's efficiency, accuracy, and adherence to the defined constraints.

Tables below illustrates the test schedules of both modules based on the type of test conducted and the order they are conducted. In each of the 4 tests below, various functionality tests are conducted.

Table 6.1 Test Schedule

Type of tests	**Requirement**	**Status**
Incremental Integration	Throughout implementation phase	Completed
Unit	After implementation phase is complete.	Completed
Integration	After unit testing is complete.	Completed
System	After integration testing is complete.	Completed

6.3 Test Strategy

The testing strategy employed for the system utilized the bottom-up testing strategy. The bottom-up testing strategy involves testing individual components or units of the system before progressively integrating and testing higher-level components or modules. This approach ensures that the system's building blocks are robustly tested and validated before being combined into more complex structures.

6.3.1 Classes of tests

The testing strategy employed for the system incorporated a comprehensive approach to ensure the system's reliability, functionality, and user-friendliness. This strategy encompassed various classes of tests, including Graphical User Interface (GUI) test, Functional test, Comparison test, and Usability test, all aimed at thoroughly evaluating different aspects of the system's performance.

1. **Graphical User Interface (GUI) test:** GUI tests focused on validating the user interface's responsiveness, layout, and interaction design. These tests assessed the system's visual elements, ensuring that they are properly aligned, appropriately labeled, and consistent across different screens. Interaction scenarios were simulated to confirm that buttons, menus, and other interactive components operated as intended. Furthermore, GUI tests examined the system's error handling and feedback mechanisms to provide a user-friendly and intuitive experience.

2. **Functional test:** Functional tests aimed to validate the core functionalities of the system. These tests encompassed the entire timetable generation process, from input data validation to final timetable output. Test cases were designed to cover various parameter, including all year class combinations, constraints, and preferences. The system's ability to generate accurate and optimized timetables while adhering to both hard and soft constraints was rigorously evaluated during functional testing.

3. **Comparison test:** Comparison tests involved comparing the current system's generated timetables with manually created timetables for the same input data. This approach allowed for a direct assessment of the system's efficiency and accuracy in producing schedules that meet the specified requirements. Discrepancies between the two sets of timetables were meticulously analyzed to identify areas for improvement and potential constraint violations.

4. **Usability Tests:** Usability tests focused on assessing the system's user-friendliness and overall user experience. Test scenarios were designed to replicate real-world usage, involving users with varying degrees of familiarity with the system. The aim was to ensure that the system is accessible and understandable to users, regardless of their technical expertise.

6.4 Test Implementation

6.4.1 Experimental / Test Description

In the test implementation phase, a comprehensive set of test cases was designed and documented to evaluate the performance and accuracy of the key modules within the system. The three main modules subjected to testing were the Graphical User Interface (GUI) module, the Database module, and the AI (Genetic Algorithm) module. For each module, specific test cases were identified, and their expected results were defined to ensure rigorous testing and validation.

6.4.1.1 Graphical User Interface (GUI) Module Test Cases

1. **Input Data Validation**

   Verify that the UI properly validates input data, such as years, programs, and student batches. Input validation should prevent invalid or incomplete data from being submitted, and appropriate error messages should be displayed for incorrect inputs.

Expected Result: Error messages should appear for missing or invalid inputs, and the user should be prompted to correct them.

2. **Preference Selection**

   Test the UI's ability to handle user preferences for courses and lecturers. Check if preferences are correctly captured and displayed for further processing.

   Expected Result: Selected preferences should be accurately displayed and stored for later use in the scheduling process.

3. **Timetable Display**

   Evaluate the UI's capability to display generated timetables. Confirm that the displayed timetable adheres to the selected preferences and constraints.

   Expected Result: Timetable should be presented in a readable format with all relevant information and properly aligned data.

6.4.1.2 Database Module Test Cases

1. **Data Retrieval**

   Verifying the accuracy of data retrieval from the database, including class details, room information, lecturer qualifications, and preferences.

   Expected Result: Retrieved data should match the expected values and reflect the current state of the database.

2. **Database Consistency**

   Test the integrity of the database by ensuring that relationships between different entities (e.g., rooms, lecturers) are correctly maintained and reflected in the retrieved data.

Expected Result: Data retrieved from different tables should be consistent and aligned.

6.4.1.3 AI (Genetic Algorithm) Module Test Cases

1. **Initial Population Generation**

   Evaluate the GA module's ability to create an initial population of timetables with valid and diversified population and generations.

   Expected Result: Initial population should consist of valid timetables with diverse combinations of student batches, rooms, and days/times.

2. **Fitness Evaluation**

   Validate the GA's fitness evaluation process by assessing the fitness scores of generated timetables against the defined both constraints and preferences.

   Expected Result: Fitness scores should accurately and improved to represent how well each timetable adheres to hard and soft constraints.

3. **Genetic Operators**

   Test the correctness of genetic operators, including crossover and mutation, by applying them to selected parent timetables and verifying the resulting offspring timetables.

   Expected Result: Offspring timetables should demonstrate combinations of traits inherited from their parents, reflecting the genetic operators' effects.

4. **Survival Selection**

   Verify the survival selection process by selecting individuals with higher fitness scores for the next generation.

Expected Result: The selected individuals should have higher fitness scores compared to others, ensuring the preservation of favorable traits.

6.4.2 Test Data

The data collection process involved gathering information from the UTeM FTMK of 2 main programs which is BITI and BITS actual class schedules, lecturer qualifications & preferences, courses, room details, and student batch. The data will be stored into the database by the user recorded in the website-based information system. On top of that, the data is utilized for the AI module to generate an optimized timetable based on both constraints and preferences. In this test data, there are 19 lecturers, 21 rooms and labs, 18 courses, with all years, BITI & BITS program, section 1 & 2, and group 1 & 2 of student batches. The complete data is included in Appendix B for reference.

6.5 Test Results and Analysis

6.5.1 Graphical Analysis

Graphical analysis of the university timetable using genetic algorithm module’s performance is conducted at current stage. Figure below shows the graph is yielding a satisfactory result. The graph depicts the performance of the genetic algorithm in optimizing the university timetable. The blue line represents the best fitness score achieved by the algorithm during its iterations. This score measures how well the timetable satisfies certain criteria, such as minimizing conflicts and maximizing resource utilization. The red line represents the average fitness score across multiple runs of the genetic algorithm. It gives an idea of the algorithm's overall performance and stability. The green line represents the threshold, which likely indicates a predetermined level of fitness that the timetable needs to achieve to be considered satisfactory. The fitness score ranges are between 0 and 1, the higher indicating a better result. Therefore, the fitness score reached 1 at 75th generation means a strong indication of the algorithm's success in creating an efficient university timetable. This implies that the timetable effectively minimizes conflicts, maximizes resource utilization, and meets all specified constraints and preferences.

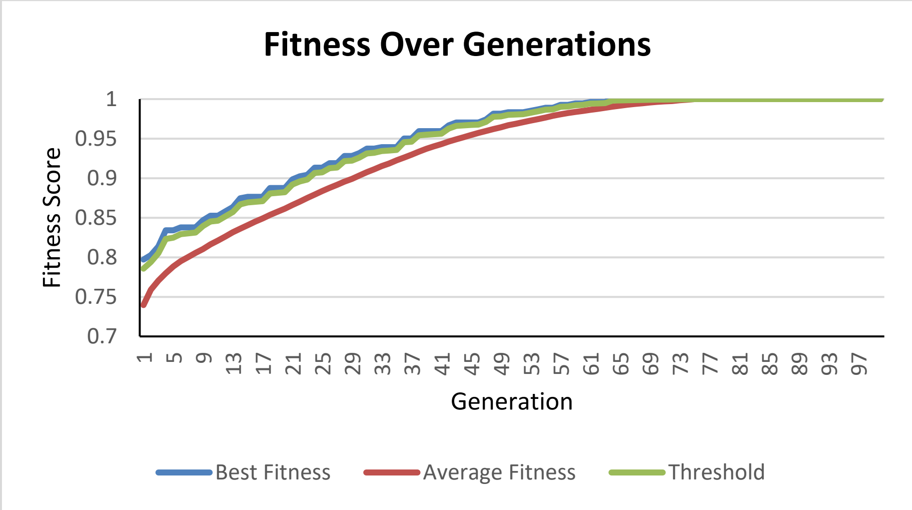

Figure 6.1 100 Generations with 550 Population Size

6.5.2 Optimal University Timetable Generated

One of the optimal university timetables is generated by the data mentioned in section 6.4.2 that no hard constraints are violated and soft constraint is satisfied. The complete and optimal output is included in Appendix UTeM FTMK Timetable for reference.

6.5.3 Algorithm Performance

The parameter setting for the genetic algorithm is 550 of population size with 100 generations. The algorithm performance is evaluated by the minimum, maximum, average and standard deviation of 3 values which are best fitness score, average fitness score, time taken. Both of fitness score ranges are from 0 until 1, the higher score indicating a better result. Time taken refers to the total time needed to complete the generation of timetable.

The table below shows the best fitness score and average fitness score with 3 runs are 1, indicating the algorithm is able to produce the most optimal results without constraints violation in every run. A standard deviation of 0 for both fitness scores indicates the quality of the university timetables is generated consistently. The time taken has an average value of 5.772, indicating the average timetable generated will be slightly 6 hours. A standard deviation of 0.115 is higher than the standard deviation of both fitness scores. Therefore, the time taken indicator variates more.

Table 6.2 Minimum, Maximum, Average and Standard Deviation of 3 Runs Algorithm Performance

Runs	Best Fitness	Average Fitness	Time Taken
1	1	1	5.716666666
2	1	1	5.666666666
3	1	1	5.933333333
Min	1	1	5.666666666
Max	1	1	5.9333333333
Average	1	1	5.772222222
Standard Deviation	0	0	0.115737037

6.6 Conclusion

The testing phase serves as a crucial gateway towards validating the genetic algorithm's capabilities for university timetabling. Through numerous iterations of testing and project improvement, the testing carried out had enhanced the project's quality and the algorithm's performance, yielding satisfactory results. The project's end will be covered in the following chapter.

CHAPTER 7: CONCLUSION

7.1 Observation on Weaknesses and Strengths

The university timetable which utilizes genetic algorithm has impressive strengths. One of the aspects is its ability to create a timetable management system. The inclusion of user authentication is praiseworthy as it provides a secure login mechanism, for authorized users to access and manage the system. Moreover, the use of an algorithm to generate timetables demonstrates an innovative approach in solving the complex optimization problem associated with creating university schedules. The integration with a MySQL database is a choice as it allows for storage and retrieval of data. Additionally, the systems implementation of Flask and HTML templates results in a organized and user friendly interface, for managing components of the university timetable. Lastly the codes modularity enhances maintainability and facilitates development efforts.

However, there are some weaknesses to be considered. It is important to strengthen security measures as storing passwords in text within the database can potentially compromise security. To ensure the integrity of user inputs and prevent vulnerabilities such, as SQL injection it is necessary to implement data validation. Additionally, it would be beneficial to enhance error handling mechanisms in order to handle situations gracefully and provide users with error messages. It is also important to consider scalability as the systems performance and responsiveness may be affected as data volume increases over time. Lastly although this system offers a range of features the complexity of the user interface may present challenges for users who're not familiar, with the domain.

7.2 Propositions for Improvement

There are several key propositions for improvement can be considered. First and foremost, the system could provide options for administrators, faculty, and students to input their preferences and constraints, such as preferred class timings and room choices, can further personalize the timetables. Additionally, optimizing the genetic algorithm's execution by employing parallel processing or distributed computing can handle larger datasets more efficiently. Lastly, the system can offer training and support for users (administrators, faculty) to effectively use and understand the timetable generator, making it more likely to meet their needs.

7.3 Project Contribution

The project employs a powerful genetic algorithm to create optimized timetables, ensuring that lecture rooms or labs, lecturers, and every batch student are allocated effectively. By avoiding room and lecturer conflicts, the solution maximizes resource utilization, reducing the need for manual intervention and rescheduling. Additionally, the system is designed to prevent lecture overlaps and to evenly distribute courses across the week, enhancing the learning experience for students. With optimized timetables, students can attend all required classes without scheduling conflicts, reducing stress and improving overall satisfaction.

The timetabling procedure is also automated by this system, so human schedule preparation and adjustment are no longer necessary. This lessens the administrative workload and chance for mistakes, allowing employees to concentrate on duties of higher value, such as curriculum development and student support. Additionally, the system provides information that might help decision-makers by analyzing the preferences and qualifications of lecturers as well as the requirements of various student batches. The course offerings, lecturer assignments, and resource distribution can all be changed by administrators using data-driven decisions.

In conclusion, the use of genetic algorithms in university timetables promotes better resource utilization, improved student and lecturer experiences, simplified administrative procedures, and data-driven decision-making. This project provides an important solution that enables educational institutions to offer a productive, successful, and excellent learning environment.

7.4 Conclusion

The project has made significant strides towards resolutely achieving its goals. The goal of the project is to create an academic schedule using a genetic algorithm-based method. The evolutionary algorithm is a useful method for creating an ideal itinerary because it seeks to optimize the configuration of lectures or laboratories, rooms, lecturers, and other criteria.

Additionally, the project also includes a web application built using Flask that allows users to create timetables using the genetic algorithm. User identification, input selection (year levels, programs, student batches), and the presentation of generated timetables are all aspects of the website application.

In a nutshell, the timetable generation process is optimized by a genetic algorithm that is implemented by the genetic algorithm module. It evolves a population over generations to increase fitness while taking into account a variety of constraints and preferences. The system balanced hard and soft constraints efficiently, producing timetables that could be beneficial.

REFERENCES

1. Genetic algorithm for university course timetabling problem. (2018) https://egrove.olemiss.edu/cgi/viewcontent.cgi?article=1442&context=etd.
2. Developing a course timetable system for academic departments using genetic algorithm(2017). https://www.researchgate.net/publication/313487752_Developing_A_Course_Timetable_System_for_Academic_Departments_Using_Genetic_Algorithm.
3. University timetable scheduling using genetic algorithm approach case. (2018). https://www.ijera.com/papers/vol8no12/p2/E0812023035.pdf.
4. Hamed Babaei, Scheduling is one of the problems which so many researches have been conducted on it over the years. The university course timetabling problem which is an NP-hard problem is a type of scheduling problem. Timetabling process must be done for each s, Aladag, C. H., Aubin, J., Daskalaki, S., Deris, S., Werra, D. D., Dimopoulou, M., Al-Betar, M. A., Abdullah, S., Alsmadi, O. MK., Alvarez, R., Amintoosi, M., Aycan, E., Bakir, M. A., Chaudhuri, A.Joudaki, M. (2014, November 21). A survey of approaches for university course timetabling problem. Computers & IndustrialEngineering. https://www.sciencedirect.com/science/article/abs/pii/S0360835214003714.
5. Edceliz. (n.d.). Edceliz/Geneticalgorithmuniversityclassscheduler: A class scheduler using adaptive-elitist genetic algorithm. GitHub. https://github.com/edceliz/GeneticAlgorithmUniversityClassScheduler
6. Class timetable scheduling with genetic algorithm - researchgate. (2013). https://www.researchgate.net/publication/273776640_Class_Timetable_Scheduling_with_Genetic_Algorithm.
7. Using a genetic algorithm optimizer tool to solve university timetable (2008). https://ieeexplore.ieee.org/document/4555397/.
8. G. Alnowaini and A. A. Aljomai, "Genetic Algorithm For Solving University Course Timetabling Problem Using Dynamic Chromosomes," 2021 International Conference of Technology, Science and Administration (ICTSA), Taiz, Yemen, 2021, pp. 1-6, doi:

10.1109/ICTSA52017.2021.9406539. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9406539&isnumber=9406514

9. Aki, Ozan. (2020). University Exam Timetabling Using Genetic Algorithms. https://www.researchgate.net/publication/346969170_University_Exam_Timetabling_Using_Genetic_Algorithms
10. Turton, B.C.H. (1998). Genetic Algorithms and the Timetabling Problem. In: Artificial Neural Nets and Genetic Algorithms. Springer, Vienna. https://doi.org/10.1007/978-3-7091-6492-1_60
11. IvyPanda. (2023, June 16). Timetable Scheduling Using Generic Algorithms. https://ivypanda.com/essays/timetable-scheduling-using-generic-algorithms/
12. Sampebatu, L., & Kamolan, A. (2016). Timetable Management Using Genetic Algorithms. https://www.semanticscholar.org/paper/Timetable-Management-Using-Genetic-Algorithms-Sampebatu-Kamolan/edca2bbd6ae40fa537b513139782740cc4094009
13. Omar Alhuniti, Rawan Ghnemat, and M. Samir Abou El-Seoud. 2021. Smart University Scheduling Using Genetic Algorithms. In Proceedings of the 9th International Conference on Software and Information Engineering (ICSIE '20). Association for Computing Machinery, New York, NY, USA, 235–239. https://doi.org/10.1145/3436829.3436873
14. V. Sapru, K. Reddy and B. Sivaselvan, "Time table scheduling using Genetic Algorithms employing guided mutation," 2010 IEEE International Conference on Computational Intelligence and Computing Research, Coimbatore, India, 2010, pp. 1-4, doi: 10.1109/ICCIC.2010.5705788. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5705788&isnumber=5705719
15. Sampebatu, Limbran, and Aries Kamolan. "Timetable Management Using Genetic Algorithms." Widya Teknik, vol. 15, no. 2, 2016, pp. 67-72, doi:10.33508/wt.v15i2.910. https://www.neliti.com/publications/231884/timetable-management-using-genetic-algorithms#cite

APPENDIX

A: Sample Code

```python
def generate_timetable(year_levels, programs, student_groups):
    cursor = db.cursor()

    # Convert year_levels, programs, and student_groups into a string format for the query
    year_levels_str = ", ".join(map(str, year_levels))
    programs_str = ", ".join(map(lambda x: f"'{x}'", programs))
    student_groups_str = ", ".join(map(lambda x: f"'{x}'", student_groups))

    query = (
        "SELECT classgroup.gid, classgroup.class_type, classgroup.cid, course.course_name, classgroup.duration, "
        "studentgroup.year_level, studentgroup.program, studentgroup.studentgroup_name "
        "FROM classgroup "
        "INNER JOIN studentgroup ON classgroup.gid = studentgroup.gid "
        "INNER JOIN course ON classgroup.cid = course.cid "
        f"WHERE studentgroup.year_level IN ({year_levels_str}) "
        f"AND studentgroup.program IN ({programs_str}) "
        f"AND studentgroup.studentgroup_name IN ({student_groups_str})"
    )

    cursor.execute(query)
    classgroups = cursor.fetchall()
    cursor.nextset()  # Clear the unread result

    if not classgroups:
        return []

    cursor.execute("SELECT rid, seat, room_type FROM room")
    rooms = cursor.fetchall()
    cursor.nextset()  # Clear the unread result

    timetable = []
    for classgroup in classgroups:
        gid, class_type, cid, course_name, duration, class_year_level, program, student_group = classgroup

        assigned = False
        while not assigned:
            day = random.choice(["Monday", "Tuesday", "Wednesday", "Thursday", "Friday"])
            hour = random.randint(9, 16)  # Adjusted to allow for consecutive hours

            room_id = random.choice(rooms)[0]

            lecturer_name = get_preferred_lecturer(cid)

            # Check if there's room for a consecutive hour
            if duration > 1:  # Check if there's room for consecutive hour
                consecutive_hour = hour + 1
            else:
                consecutive_hour = None

            timeslot = {
                "day": day,
                "hour": hour,
                "consecutive_hour": consecutive_hour,
                "gid": gid,
                "class_type": class_type,
                "cid": cid,
                "duration": duration,
                "course_name": course_name,
                "room_id": room_id,
                "lecturer_name": lecturer_name,
                "year_level": class_year_level,
                "program": program,
                "student_group": student_group
            }

            timetable.append(timeslot)
            assigned = True

    return timetable
```

Figure 7.1 Population Initialization Function

```python
def evaluate_fitness(population):
    fitness_scores = []

    for timetable in population:
        violated_hard_constraints = 0
        fulfilled_soft_constraints = 0
        total_hard_constraints = 0
        total_soft_constraints = 0

        # Hard constraint 1: A room should not be assigned on the same day and same hour
        assigned_rooms_per_timeslot = {}  # Track assigned rooms for each day and hour
        for timeslot in timetable:
            day = timeslot["day"]
            hour = timeslot["hour"]
            room_id = timeslot["room_id"]

            if (day, hour) in assigned_rooms_per_timeslot:
                if room_id in assigned_rooms_per_timeslot[(day, hour)]:
                    violated_hard_constraints += 1
                else:
                    assigned_rooms_per_timeslot[(day, hour)].add(room_id)
            else:
                assigned_rooms_per_timeslot[(day, hour)] = {room_id}

            if timeslot["consecutive_hour"] is not None:
                consecutive_room_id = timeslot["room_id"]
                if (day, timeslot["consecutive_hour"]) in assigned_rooms_per_timeslot:
                    if consecutive_room_id in assigned_rooms_per_timeslot[(day, timeslot["consecutive_hour"])]:
                        violated_hard_constraints += 1
                    else:
                        assigned_rooms_per_timeslot[(day, timeslot["consecutive_hour"])].add(consecutive_room_id)
                else:
                    assigned_rooms_per_timeslot[(day, timeslot["consecutive_hour"])] = {consecutive_room_id}

            total_hard_constraints += 1

        # Hard Constraint 2: Class should not be assigned on 13:00 - 14:00
        for timeslot in timetable:
            start_time = timeslot["hour"]
            if 12 <= start_time < 13 or 13 <= start_time < 14:
                violated_hard_constraints += 1
        total_hard_constraints += 1
```

```python
        # Hard Constraint 4: Room id's room type must match class type
        for timeslot in timetable:
            room_id = timeslot["room_id"]
            class_type = timeslot["class_type"]
            room_type = get_room_type_for_room(room_id)
            if room_type != class_type:
                violated_hard_constraints += 1
            total_hard_constraints += 1

        # Hard Constraint 5: A class must be assigned by a qualified lecturer
        for timeslot in timetable:
            lecturer_name = timeslot["lecturer_name"]
            class_group_id = timeslot["gid"]  # Use the class group ID instead of course_id
            qualified_lecturers = get_qualified_lecturers_for_group(class_group_id)
            if lecturer_name not in qualified_lecturers:
                violated_hard_constraints += 1
            total_hard_constraints += 1

        # Soft Constraint: A class can be assigned by a preference lecturer
        for timeslot in timetable:
            lecturer_name = timeslot["lecturer_name"]
            class_group_id = timeslot["gid"]
            qualified_lecturers = get_preferred_lecturers_for_group(class_group_id)  # Use the function for preferred lecturers
            if lecturer_name in qualified_lecturers:
                fulfilled_soft_constraints += 1
            total_soft_constraints += 1

        fitness = (1 - (violated_hard_constraints / total_hard_constraints)) * 0.85 + (fulfilled_soft_constraints / total_soft_constraints) * 0.15
        fitness_scores.append(fitness)

    return fitness_scores
        # Hard Constraint 3: Students and lecturers must not be assigned on the same day and hour
        assigned_groups_and_lecturers_per_timeslot = {}  # Track assigned groups and lecturers for each day and hour
        for timeslot in timetable:
            day = timeslot["day"]
            hour = timeslot["hour"]
            year_level = timeslot["year_level"]
            program = timeslot["program"]
            student_group = timeslot["student_group"]
            lecturer_name = timeslot["lecturer_name"]

            key = (day, hour)
            if key in assigned_groups_and_lecturers_per_timeslot:
                if (year_level, program, student_group) in assigned_groups_and_lecturers_per_timeslot[key][0]:
                    violated_hard_constraints += 1
                if lecturer_name in assigned_groups_and_lecturers_per_timeslot[key][1]:
                    violated_hard_constraints += 1
            else:
                assigned_groups_and_lecturers_per_timeslot[key] = (set(), set())
            assigned_groups_and_lecturers_per_timeslot[key][0].add((year_level, program, student_group))
            assigned_groups_and_lecturers_per_timeslot[key][1].add(lecturer_name)

            if timeslot["consecutive_hour"] is not None:
                consecutive_key = (day, timeslot["consecutive_hour"])
                if consecutive_key in assigned_groups_and_lecturers_per_timeslot:
                    if (year_level, program, student_group) in assigned_groups_and_lecturers_per_timeslot[consecutive_key][0]:
                        violated_hard_constraints += 1
                    if lecturer_name in assigned_groups_and_lecturers_per_timeslot[consecutive_key][1]:
                        violated_hard_constraints += 1
                else:
                    assigned_groups_and_lecturers_per_timeslot[consecutive_key] = (set(), set())
                assigned_groups_and_lecturers_per_timeslot[consecutive_key][0].add((year_level, program, student_group))
                assigned_groups_and_lecturers_per_timeslot[consecutive_key][1].add(lecturer_name)

            total_hard_constraints += 2  # Count both group and lecturer constraints
```

Figure 7.2 Fitness Functions

```python
def selection(population):
    tournament_size = 2
    selected_parents = []

    for _ in range(len(population)):
        # Select two random individuals from the population for the tournament
        tournament = random.choices(population, k=tournament_size)

        # Remove any timetables with empty classgroups from the tournament
        tournament = [individual for individual in tournament if any(individual)]

        # Check if any individuals are left in the tournament
        if tournament:
            # Calculate fitness scores for the individuals in the tournament
            tournament_fitness = evaluate_fitness(tournament)

            # Select the individual with the highest fitness score as the winner of the tournament
            winner = tournament[tournament_fitness.index(max(tournament_fitness))]
            selected_parents.append(winner)

    return selected_parents
```

Figure 7.3 Parent Selection Function

```python
def crossover(parents):
    offspring = []

    for i in range(0, len(parents), 2):
        if i + 1 < len(parents):
            parent1 = parents[i]
            parent2 = parents[i + 1]

            offspring1, offspring2 = uniform_crossover(parent1, parent2)

            offspring.append(offspring1)
            offspring.append(offspring2)

    return offspring

def uniform_crossover(parent1, parent2):
    mask = [random.choice([0, 1]) for _ in range(len(parent1))]

    offspring1 = [parent1[i] if mask[i] == 0 else parent2[i] for i in range(len(parent1))]
    offspring2 = [parent2[i] if mask[i] == 0 else parent1[i] for i in range(len(parent1))]

    return offspring1, offspring2
```

Figure 7.4 Crossover Functions

```python
def mutation(offspring, mutation_rate):
    mutated_offspring = []

    for individual in offspring:
        if random.random() < mutation_rate:
            # Choose two distinct random indices to swap
            index1, index2 = random.sample(range(len(individual)), 2)

            # Perform the swap mutation
            mutated_individual = individual.copy()
            mutated_individual[index1], mutated_individual[index2] = mutated_individual[index2], mutated_individual[index1]

            mutated_offspring.append(mutated_individual)
        else:
            mutated_offspring.append(individual)

    return mutated_offspring
```

Figure 7.5 Mutation Function

```python
def survival_selection(population, fitness_scores, offspring, offspring_fitness_scores, num_individuals):
    combined_population = population + offspring
    combined_fitness_scores = fitness_scores + offspring_fitness_scores
    next_generation = []

    for _ in range(num_individuals):
        max_fitness_index = combined_fitness_scores.index(max(combined_fitness_scores))
        next_generation.append(combined_population[max_fitness_index])

        # Remove the selected individual from the combined population and fitness scores
        del combined_population[max_fitness_index]
        del combined_fitness_scores[max_fitness_index]

    return next_generation
```

Figure 7.6 Survival Selection Function

```python
def genetic_algorithm(year_levels, programs, student_groups, population_size = 550, generations = 100, mutation_rate = 0.01):
    # Step 1: Initialize the population
    population = []
    best_fitness_scores = []
    avg_fitness_scores = []
    thresholds = []

    for _ in range(population_size):
        timetable = generate_timetable(year_levels, programs, student_groups)
        population.append(timetable)

    # Step 2: Evolutionary loop
    for generation in range(generations):
        print(f"Generation {generation + 1}")

        # Step 2a: Evaluate fitness
        fitness_scores = evaluate_fitness(population)
        print(f"Fitness scores: {fitness_scores}")

        # Step 2b: Select parents for reproduction
        parents = selection(population)
        #print(f"Selected parents: {parents}")

        # Step 2c: Apply genetic operators (crossover and mutation) to create offspring
        offspring = crossover(parents)
        offspring = mutation(offspring, mutation_rate)
        #print(f"Offspring: {offspring}")

        # Step 2d: Evaluate fitness of offspring
        offspring_fitness_scores = evaluate_fitness(offspring)
        #print(f"Offspring fitness scores: {offspring_fitness_scores}")

        # Step 2e: Select survivors for the next generation
        population = survival_selection(population, fitness_scores, offspring, offspring_fitness_scores, num_individuals = 550)
        #print(f"Survivors: {population}")
        #print("-------------------------------------------------------------------------------------------------------------------------------")

        # Calculate best fitness, average fitness, and threshold for the current generation
        best_fitness = max(fitness_scores)
        avg_fitness = sum(fitness_scores) / len(fitness_scores)
        threshold = best_fitness + (avg_fitness - best_fitness) * 0.2  # Adjust the threshold calculation as needed

        best_fitness_scores.append(best_fitness)
        avg_fitness_scores.append(avg_fitness)
        thresholds.append(threshold)

        # Check termination condition
        if generation == generations - 1:
            break

    # Step 3: Return the best solution
    fitness_scores = evaluate_fitness(population)
    best_fitness_score = max(fitness_scores)
    print(best_fitness_score)
    best_timetable = get_best_timetable(population, fitness_scores)

    # Write data to Excel file
    write_to_excel(best_fitness_scores, avg_fitness_scores, thresholds)

    return best_timetable
```

Figure 7.7 Iterations Function

B: Data for University Timetable using Genetic Algorithm

Table 7.1 Lecturer Information

Name	Preferences	Qualifications
PROFESSOR MADYA TS. DR. ZERATUL IZZAH BINTI MOHD YUSOH	Artificial Intelligence, Logic Programming, Knowledge Based System, Statistics & Probability, Evolutionary Computing	Evolutionary Computing
TS. DR. WAN MOHD YA'AKOB BIN WAN BEJURI	Artificial Intelligence, Linear Algebra & Discrete Mathematics, Calculus and Numerical Methods	Artificial Intelligence, Calculus and Numerical Methods
PROFESOR MADYA TS. DR. ZURAIDA BINTI ABAL ABAS	Linear Algebra & Discrete Mathematics, Calculus and Numerical Methods, Statistics & Probability	Calculus and Numerical Methods
PROFESOR MADYA TS. DR. CHOO YUN HUOY	Artificial Intelligence, Machine Learning, Statistics & Probability, Fuzzy Logic	Fuzzy Logic
DR. NOOR FAZILLA BINTI ABD YUSOF	Artificial Intelligence, Machine Learning, Introduction to Data Science	Artificial Intelligence, Introduction to Data Science

TS. DR. SEK YONG WEE	Linear Algebra & Discrete Mathematics, Statistics & Probability	Statistics & Probability
DR. NUR ZAREEN BINTI ZULKARNAIN	Artificial Intelligence, Logic Programming, Intelligent Agents	Artificial Intelligence
PROFESOR MADYA GS. DR. ASMALA BIN AHMAD	Artificial Intelligence, Linear Algebra & Discrete Mathematics, Calculus and Numerical Methods, Image Processing and Pattern Recognition	Image Processing and Pattern Recognition
PROFESOR TS. DR. BURHANUDDIN BIN MOHD ABOOBAIDER	Artificial Intelligence, Linear Algebra & Discrete Mathematics, Calculus and Numerical Methods, Neural Network	Neural Network
TS. DR. ZULKIFLEE BIN MUSLIM	Data Communication and Networking	Data Communication and Networking
TS. ERMAN BIN HAMID	Data Communication and Networking	Data Communication and Networking
TS. ZAKIAH BINTI AYOP	Data Communication and Networking	Data Communication and Networking
TS. HANIZA BINTI NAHAR	Cloud Computing Foundation	Cloud Computing Foundation

DR. NUR FADZILAH BINTI OTHMAN	Information Technology Security	Information Technology Security
TS. AHMAD FADZLI NIZAM BIN ABDUL RAHMAN	Calculus and Numerical Methods	Calculus and Numerical Methods
TS. DR. HALIZAH BINTI BASIRON	Artificial Intelligence, Knowledge Based System	Artificial Intelligence, Knowledge Based System
TS. DR. MOHD NAJWAN BIN MD KHAMBARI	Internet Technology	Internet Technology
TS. ARIFF BIN IDRIS	Internet Technology	Internet Technology
DR. NURUL AZMA BINTI ZAKARIA	Operating System, Data Communication and Networking	Operating System, Data Communication and Networking

Table 7.2 Room Information

Room ID	Room Type
CCNA	Lab
MPD1	Lab
MS	Lab
MTW	Lab

RECAP	Lecture
AI1	Lab
BK1	Lecture
MR1	Lab
AI2	Lab
BK2	Lecture
MK2	Lab
MR2	Lab
AI3	Lab
BK3	Lecture
BK4	Lecture
BK5	Lecture
BK6	Lecture
BK12	Lecture
BK13	Lecture
BK14	Lecture
BK15	Lecture

Table 7.3 Course Information

Course ID	Course Name
BITI 1113	Artificial Intelligence
BITI 1213	Linear Algebra & Discrete Mathematics
BITI 1223	Calculus and Numerical Methods
BITI 2113	Logic Programming
BITI 2213	Knowledge Based System
BITI 2223	Machine Learning
BITI 2233	Statistics & Probability
BITI 2513	Introduction to Data Science
BITI 3113	Intelligent Agents
BITI 3123	Fuzzy Logic
BITI 3133	Neural Network
BITI 3143	Evolutionary Computing
BITI 3313	Image Processing and Pattern Recognition
BITS 1213	Operating System
BITS 1313	Data Communication and Networking
BITS 2513	Internet Technology

BITS 2573	Cloud Computing Foundation
BITS 3423	Information Technology Security

UTeM FTMK Timetable

Time	Monday	Tuesday	Wednesday	Thursday	Friday
9:00 - 10:00	Lecture - BITI 2513 BK6 Dr Elle 3 BITI S2G1 Lecture - BITS 2573 RECAP Dr Haniza 3 BITI S1G1 Lab - BITI 1223 MR2 Dr Fadzli 1 BITS S1G2	Lecture - BITI 1223 BK12 Dr Zuraida 1 BITI S1G1 Lecture - BITI 3123 RECAP Dr Choo 2 BITI S1G1 Lab - BITI 3143 MPD1 Dr Zeratul 2 BITI S1G2 Lecture - BITS 1213 BK3 Dr Azma 1 BITI S1G2 Lab - BITS 2573 CCNA Dr Haniza 3 BITI S2G1	Lecture - BITS 2573 RECAP Dr Haniza 3 BITI S2G1 Lab - BITS 2513 AI3 Dr Ariff 3 BITS S1G2	Lab - BITI 3133 CCNA Dr Burhan 2 BITI S1G2 Lab - BITS 1313 AI1 Dr Erman 1 BITI S1G2 Lecture - BITS 3423 BK4 Dr Fadzilah 3 BITI S1G1 Lab - BITS 2573 MR1 Dr Haniza 3 BITS S2G1 Lab - BITI 1113 MR2 Dr Yaakob 2 BITS S2G2	Lecture - BITI 2513 RECAP Dr Elle 3 BITI S2G2 Lecture - BITS 1313 BK3 Dr Zulkiflee 2 BITS S1G2 Lecture - BITS 2513 BK1 Dr Ariff 3 BITS S2G1 Lab - BITS 2573 AI3 Dr Haniza 3 BITS S2G2

10:00 - 11:00	Lecture - BITI 1113 BK13 Dr Halizah 1 BITI S1G1 Lecture - BITI 2513 BK6 Dr Elle 3 BITI S2G1 Lecture - BITS 2573 RECAP Dr Haniza 3 BITI S1G1 Lecture - BITS 3423 BK15 Dr Fadzilah 3 BITI S2G2 Lab - BITS 2513 MS Dr Ariff 3 BITS S2G1 Lab - BITI 1223 MR2 Dr Fadzli 1 BITS S1G2	Lecture - BITI 1223 BK12 Dr Zuraida 1 BITI S1G1 Lecture - BITI 3123 RECAP Dr Choo 2 BITI S1G1 Lab - BITI 3143 MPD1 Dr Zeratul 2 BITI S1G2 Lecture - BITS 1213 BK3 Dr Azma 1 BITI S1G2 Lab - BITS 2573 CCNA Dr Haniza 3 BITI S2G1 Lab - BITI 1113 MS Dr Halizah 2 BITS S2G1	Lecture - BITS 1313 BK1 Dr Erman 1 BITI S1G1 Lecture - BITS 2573 RECAP Dr Haniza 3 BITI S2G1 Lab - BITS 2513 MS Dr Najwan 3 BITS S1G1 Lab - BITS 2513 AI3 Dr Ariff 3 BITS S1G2	Lab - BITI 3133 CCNA Dr Burhan 2 BITI S1G2 Lab - BITS 1313 AI1 Dr Erman 1 BITI S1G2 Lecture - BITS 3423 BK4 Dr Fadzilah 3 BITI S1G1 Lab - BITS 2573 MR1 Dr Haniza 3 BITS S2G1 Lab - BITI 1113 MR2 Dr Yaakob 2 BITS S2G2	Lab - BITI 3313 MK2 Dr Asmala 2 BITI S1G1 Lecture - BITI 2513 RECAP Dr Elle 3 BITI S2G2 Lab - BITS 3423 MTW Dr Fadzilah 3 BITI S1G2 Lecture - BITS 1313 BK3 Dr Zulkiflee 2 BITS S1G2 Lecture - BITS 2513 BK1 Dr Ariff 3 BITS S2G1 Lab - BITS 2573 AI3 Dr Haniza 3 BITS S2G2

11:00 - 12:00	Lecture - BITI 1113 BK13 Dr Halizah 1 BITI S1G1 Lab - BITI 3133 MTW Dr Burhan 2 BITI S1G1 Lecture - BITS 3423 BK15 Dr Fadzilah 3 BITI S2G2 Lab - BITS 2513 MS Dr Ariff 3 BITS S2G1 Lecture - BITI 1223 BK4 Dr Yaakob 1 BITS S1G1	Lab - BITI 1223 AI3 Dr Fadzli 1 BITI S1G1 Lab - BITS 1213 MPD1 Dr Azma 1 BITI S1G2 Lecture - BITS 2573 BK3 Dr Haniza 3 BITS S1G1 Lab - BITI 1113 MS Dr Halizah 2 BITS S2G1	Lab - BITI 1223 MK2 Dr Fadzli 1 BITI S1G2 Lab - BITI 3123 AI2 Dr Choo 2 BITI S1G1 Lecture - BITS 1313 BK1 Dr Erman 1 BITI S1G1 Lecture - BITS 2573 BK15 Dr Haniza 3 BITI S2G2 Lab - BITS 2513 MS Dr Najwan 3 BITS S1G1	Lecture - BITI 3123 BK4 Dr Choo 2 BITI S1G2 Lab - BITI 2513 MPD1 Dr Elle 3 BITI S2G1 Lecture - BITS 1213 BK14 Dr Azma 1 BITI S1G1 Lab - BITS 3423 MR2 Dr Fadzilah 3 BITI S1G1 Lab - BITS 2573 CCNA Dr Haniza 3 BITI S1G2 Lecture - BITS 2513 BK3 Dr Ariff 3 BITS S1G2	Lab - BITI 3313 MK2 Dr Asmala 2 BITI S1G1 Lab - BITS 1313 MR2 Dr Azma 1 BITI S1G1 Lab - BITS 2573 CCNA Dr Haniza 3 BITI S1G1 Lab - BITS 3423 MTW Dr Fadzilah 3 BITI S1G2 Lecture - BITI 1223 BK14 Dr Fadzli 1 BITS S1G2 Lecture - BITI 1113 BK4 Dr Yaakob 2 BITS S1G1 Lecture - BITI 1113 BK12 Dr Zareen 2 BITS S2G2

12:00 - 13:00	Lab - BITI 3133 MTW Dr Burhan 2 BITI S1G1 Lecture - BITI 1223 BK4 Dr Yaakob 1 BITS S1G1	Lab - BITI 1223 AI3 Dr Fadzli 1 BITI S1G1 Lab - BITS 1213 MPD1 Dr Azma 1 BITI S1G2 Lecture - BITS 2573 BK3 Dr Haniza 3 BITS S1G1	Lab - BITI 1223 MK2 Dr Fadzli 1 BITI S1G2 Lab - BITI 3123 AI2 Dr Choo 2 BITI S1G1 Lecture - BITS 2573 BK15 Dr Haniza 3 BITI S2G2	Lecture - BITI 3123 BK4 Dr Choo 2 BITI S1G2 Lab - BITI 2513 MPD1 Dr Elle 3 BITI S2G1 Lecture - BITS 1213 BK14 Dr Azma 1 BITI S1G1 Lab - BITS 3423 MR2 Dr Fadzilah 3 BITI S1G1 Lab - BITS 2573 CCNA Dr Haniza 3 BITI S1G2 Lecture - BITS 2513 BK3 Dr Ariff 3 BITS S1G2	Lab - BITS 1313 MR2 Dr Azma 1 BITI S1G1 Lab - BITS 2573 CCNA Dr Haniza 3 BITI S1G1 Lecture - BITI 1223 BK14 Dr Fadzli 1 BITS S1G2 Lecture - BITI 1113 BK4 Dr Yaakob 2 BITS S1G1 Lecture - BITI 1113 BK12 Dr Zareen 2 BITS S2G2
13:00 - 14:00					

14:00 - 15:00	Lecture - BITI 3143 BK4 Dr Zeratul 2 BITI S1G1 Lab - BITS 3423 AI2 Dr Fadzilah 3 BITI S2G1 Lab - BITS 2573 AI1 Dr Haniza 3 BITI S2G2 Lecture - BITS 2513 BK3 Dr Najwan 3 BITS S1G1 Lab - BITI 1223 MK2 Dr Yaakob 1 BITS S1G1	Lecture - BITI 3313 BK3 Dr Asmala 2 BITI S1G2 Lecture - BITS 2573 BK14 Dr Haniza 3 BITI S1G2 Lab - BITS 2513 MK2 Dr Najwan 3 BITS S2G2	Lab - BITS 2573 AI2 Dr Haniza 3 BITS S1G2	Lecture - BITI 1223 BK3 Dr Yaakob 1 BITI S1G2 Lecture - BITI 3143 BK2 Dr Zeratul 2 BITI S1G2 Lecture - BITS 2573 BK5 Dr Haniza 3 BITS S1G2 Lecture - BITS 2513 BK12 Dr Ariff 3 BITS S2G2 Lab - BITI 1113 MPD1 Dr Halizah 2 BITS S1G1	Lab - BITI 3313 AI3 Dr Asmala 2 BITI S1G2 Lab - BITI 1113 AI2 Dr Halizah 2 BITS S1G2

15:00 - 16:00	Lecture - BITI 1113 BK1 Dr Zareen 1 BITI S1G2 Lecture - BITI 3143 BK4 Dr Zeratul 2 BITI S1G1 Lab - BITS 3423 AI2 Dr Fadzilah 3 BITI S2G1 Lab - BITS 2573 AI1 Dr Haniza 3 BITI S2G2 Lecture - BITS 2513 BK3 Dr Najwan 3 BITS S1G1 Lab - BITI 1223 MK2 Dr Yaakob 1 BITS S1G1	Lab - BITI 3143 CCNA Dr Zeratul 2 BITI S1G1 Lecture - BITI 3313 BK3 Dr Asmala 2 BITI S1G2 Lecture - BITS 2573 BK14 Dr Haniza 3 BITI S1G2 Lab - BITS 2513 MK2 Dr Najwan 3 BITS S2G2	Lab - BITI 1113 MR1 Dr Yaakob 1 BITI S1G2 Lecture - BITI 3133 BK5 Dr Burhan 2 BITI S1G2 Lab - BITS 2573 AI2 Dr Haniza 3 BITS S1G2	Lecture - BITI 1223 BK3 Dr Yaakob 1 BITI S1G2 Lecture - BITI 3143 BK2 Dr Zeratul 2 BITI S1G2 Lab - BITS 1313 AI2 Dr Azma 2 BITS S2G2 Lecture - BITS 2573 BK5 Dr Haniza 3 BITS S1G2 Lecture - BITS 2513 BK12 Dr Ariff 3 BITS S2G2 Lab - BITI 1113 MPD1 Dr Halizah 2 BITS S1G1	Lecture - BITI 3133 BK14 Dr Burhan 2 BITI S1G1 Lab - BITI 3313 AI3 Dr Asmala 2 BITI S1G2 Lecture - BITS 3423 BK15 Dr Fadzilah 3 BITI S1G2 Lecture - BITS 2573 BK13 Dr Haniza 3 BITS S2G2 Lab - BITI 1113 AI2 Dr Halizah 2 BITS S1G2

16:00 - 17:00	Lecture - BITI 1113 BK1 Dr Zareen 1 BITI S1G2 Lab - BITI 2513 MR2 Dr Elle 3 BITI S1G1 Lab - BITS 1213 MR1 Dr Azma 1 BITI S1G1 Lecture - BITS 3423 BK3 Dr Fadzilah 3 BITI S2G1	Lab - BITI 3143 CCNA Dr Zeratul 2 BITI S1G1 Lecture - BITI 2513 BK15 Dr Elle 3 BITI S1G1 Lab - BITS 3423 MK2 Dr Fadzilah 3 BITI S2G2 Lab - BITS 1313 MR1 Dr Zakiah 2 BITS S2G1	Lab - BITI 1113 MR1 Dr Yaakob 1 BITI S1G2 Lecture - BITI 3133 BK5 Dr Burhan 2 BITI S1G2 Lecture - BITI 2513 BK3 Dr Elle 3 BITI S1G2 Lecture - BITS 2573 BK13 Dr Haniza 3 BITS S2G1 Lecture - BITI 1113 BK15 Dr Halizah 2 BITS S2G1	Lecture - BITI 3313 BK1 Dr Asmala 2 BITI S1G1 Lab - BITI 3123 MTW Dr Choo 2 BITI S1G2 Lab - BITI 2513 AI1 Dr Elle 3 BITI S1G2 Lecture - BITS 1313 BK6 Dr Erman 1 BITI S1G2 Lab - BITS 1313 MR1 Dr Zakiah 2 BITS S1G1 Lab - BITS 1313 AI2 Dr Azma 2 BITS S2G2 Lab - BITS 2573 MR2 Dr Haniza 3 BITS S1G1 Lecture - BITI 1113 RECAP Dr Halizah 2 BITS S1G2	Lab - BITI 1113 MPD1 Dr Zareen 1 BITI S1G1 Lecture - BITI 3133 BK14 Dr Burhan 2 BITI S1G1 Lab - BITI 2513 MK2 Dr Elle 3 BITI S2G2 Lecture - BITS 3423 BK15 Dr Fadzilah 3 BITI S1G2 Lecture - BITS 1313 RECAP Dr Zakiah 2 BITS S1G1 Lab - BITS 1313 AI2 Dr Zulkiflee 2 BITS S1G2 Lecture - BITS 1313 BK1 Dr Erman 2 BITS S2G1 Lecture - BITS 1313 BK3 Dr Azma 2 BITS S2G2 Lecture - BITS 2573 BK13 Dr Haniza 3 BITS S2G2

17:00 - 18:00	Lab - BITI 2513 MR2 Dr Elle 3 BITI S1G1 Lab - BITS 1213 MR1 Dr Azma 1 BITI S1G1 Lecture - BITS 3423 BK3 Dr Fadzilah 3 BITI S2G1	Lecture - BITI 2513 BK15 Dr Elle 3 BITI S1G1 Lab - BITS 3423 MK2 Dr Fadzilah 3 BITI S2G2 Lab - BITS 1313 MR1 Dr Zakiah 2 BITS S2G1	Lecture - BITI 2513 BK3 Dr Elle 3 BITI S1G2 Lecture - BITS 2573 BK13 Dr Haniza 3 BITS S2G1 Lecture - BITI 1113 BK15 Dr Halizah 2 BITS S2G1	Lecture - BITI 3313 BK1 Dr Asmala 2 BITI S1G1 Lab - BITI 3123 MTW Dr Choo 2 BITI S1G2 Lab - BITI 2513 AI1 Dr Elle 3 BITI S1G2 Lecture - BITS 1313 BK6 Dr Erman 1 BITI S1G2 Lab - BITS 1313 MR1 Dr Zakiah 2 BITS S1G1 Lab - BITS 2573 MR2 Dr Haniza 3 BITS S1G1 Lecture - BITI 1113 RECAP Dr Halizah 2 BITS S1G2	Lab - BITI 1113 MPD1 Dr Zareen 1 BITI S1G1 Lab - BITI 2513 MK2 Dr Elle 3 BITI S2G2 Lecture - BITS 1313 RECAP Dr Zakiah 2 BITS S1G1 Lab - BITS 1313 AI2 Dr Zulkiflee 2 BITS S1G2 Lecture - BITS 1313 BK1 Dr Erman 2 BITS S2G1 Lecture - BITS 1313 BK3 Dr Azma 2 BITS S2G2