TIGEN

A project report

submitted in partial fulfillment of the requirements

for the degree of

Bachelor of Technology

in

Computer Science & Engineering

by

Tarun Singh(1828410109)

Pragati Srivastava(1828410064)

Under the Guidance of

Dr. Umesh Kumar Pandey

(Assistant Professor)

Department of Computer Science & Engineering

UNITED INSTITUTE OF TECHNOLOGY PRAYAGRAJ

Uttar Pradesh 211010, INDIA.

(Affiliated to Dr. A.P.J. Abdul Kalam Technical University, Lucknow)

2021-2022

Declaration of Academic Ethics

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, We have adequately cited and referenced the original sources. We declare that We have properly and accurately acknowledged all sources used in the production of this project report.

We also declare that We have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. We understand that any violation of the above will be a cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Date:..........................   (Tarun Singh)

Pragati Srivastava)

Certificate from the project guide

This is to certify that the work incorporated in the project report entitled *"TIGEN"* is a record of work carried out by *Tarun Singh(1828410109)* and *Pragati Srivastava(1828410064)* Under my guidance and supervision for the award of Degree of Bachelor of Technology in Computer Science & Engineering.

To the best of my knowledge and belief the project report

1. Embodies the work of the candidates themselves
2. Has duly been completed
3. Fulfils the requirement of the Ordinance relating to the Bachelor of Technology degree of the University and
4. Is up to the desired standard both in respect of contents and language for being referred to the examiners.

Date:...........................   (Dr. Umesh Kumar Pandey)

The project work as mentioned above is here by being recommended and Forwarded for examination and evaluation.

Date:...........................   Head of Department

Certificate from external examiner

This is to certify that the project report entitled *“TIGEN”* which is submitted by *Tarun Singh(1828410109)* and *Pragati Srivastava(1828410064)* has been examined by the under-signed as a part of the examination for the award of Degree of *Bachelor of Technology in Computer Science & Engineering*.

Internal Examiner	External Examiner
Date:...........................	Date:............................

Acknowledgments

We would like to thank **Mr. A.K. Tiwari** and **Dr. Umesh kumar pandey** for granting us the honour of being a member of Biomedical group and their invaluable help and guidance. We would like to express my deep sense of gratitude to **Abhay Shanker Pathak** for their important help and technical suggestions. We are grateful to all our supporters for their unconditional support at all times. We would like to thank all members of CSE lab who have directly or indirectly contributed in our project work and maintained a friendly atmosphere in the lab. Finally We would like to thank our families and our friends for their constant moral support.

Abstract

TIGEN which stands for "Time-Table Generator" is a project which is based on the concept of Genetic Algorithm (Evolutionary Algorithm). Genetic Algorithm are meant to provide solutions for real-life problems. Genetic Algorithm is one of the branches of Artificial Intelligence which is a very hot topic in the current age. Generating a timetable manually is rigorous and error-prone task. This project not only saves time but need minimal manual assistance and assures error free results. We have used Genetic Algorithm which provides accurate optimized solution in less time by implementing various genetic operators and provides feasible solution with satisfying provided constraints.

Apart from this, this project uses some other traditional and non-traditional methods like interaction with database system and also uses custom data-structure for storage of data during program execution etc. for successful interaction and getting result from project. This project can be extended for any type of organization where scheduling is needed. This project will also benefit in time-conservation and reducing manual work of the people who use to make schedules for their department/organizations and they can now put their effort more in other important tasks. Tigen can also be extended to have hardware interaction for minimizing manual assistance.

Contents

List of Figures

List of Tables

Listings

Chapter 1

Introduction

Tigen is a Timetable scheduler developed using the concept of genetic algorithm. Genetic Algorithm is used to find solutions for problem using various operators which is inspired by biological field. Genetic Algorithm (GA) is a search-based optimization technique based on the principles of Genetics and Natural Selection. It is frequently used to find optimal or near-optimal solutions to difficult problems which otherwise would take a lifetime to solve. It is frequently used to solve scheduling problems, and to find feasible solutions.

Generating timetable is also a scheduling problem in which various entities such as Instructors, Courses, Departments etc. has to be assigned in proper slots without causing time-clashing between any two slot. [5] If we schedule a timetable manually, then this task is laborious as well as time taking and there are high chances of error.

Tigen can rapidly generate a non- conflicted timetable which is free of errors and ensures to provide feasible solution. Feasible solutions mean those which do not violate hard constraints as well try to satisfy soft constraints. Hard constraints concern issues that are physically impossible – such as an Instructor being in two places at the same time. Similarly, two teachers are not permitted to teach two separate courses to the same group of students in a given time slot. Further, allocations of two or more classrooms are not made for the same course for a given group of students. The same room is allocated to two different classes. Genetic operators such as selection, crossover, mutation are used to find feasible solutions. We need to choose the most appropriate one from feasible solutions. Adding to more functions, Tigen has the cross-plateform functionality, Database integration feature for data management.

Tigen can run on both Graphical user interface and terminal user interface. From the main perspective, the ultimate goal is to design the format of the timetable and utilize genetic algorithm to get desired output in the form of timetable. [1]

Technologies Used

Language : C++

Modules : ncurses, Qt

VCS : Git

Database: MySQL

1.1 Existing Systems

The existing Systems which are currently in use are Either an individual performs timetable scheduling manually or some of the existing model which is present on the internet like a Class schedule generation using genetic algorithm is developed by Mohit Kumar and Jaggi Singh. They used groups of students, Courses, Instructors as entities. Their model generates Intital population of size 100 and forwards toward genetic algorithm cycle. [2] An Automated timetable scheduler using genetic algorithm by Shraddha Thakare , Tejal Nigam , Prof. Mamta Patil. Their model generates 60 - 80 considers labs,lectures, courses, sems etc entities. [3]In Dipesh mittal's timetable scheduling system they have used entities like Professor, Subject, Room, Time intervals. Hard constrains used by them are: A student should have only one class at a Time, Some classes require classes to have particular equipment. For example, audio visual equipment, projectors etc. [4]
A very close solution to our proposed approach is Timetable scheduler on java by Pranav Khurana on GitHub [5] pranav's model is the most appropriate model currrently present on internet. A survey on automatic timetable was presented by Shraddha Shinde, Saraswati Gurav and Sneha karme. They applied Machine Learning to solve Timetabling problem for university. [6] All the above-mentioned models use different approaches. Tigen is developed by taking some fixed entities into the consideration which can be easily varied while extension of model. Entities such as Instructors, Courses, Departments, class time, room are used in order to generate a timetable. [7]

There are many other schedulers present on the internet that do a job similar to that of our software. But they are not what our project is intended to be.
The solutions below-mentioned are mostly used as purpose of demonstrating a use case for Ge-

netic (Evolutionary) Algorithm.

According to us, best solutions from the below mentioned existing solutions is by **pranavkhurana**, but it is still far from what our project is ought to be. [5]

Some of the exisiting solutions on Github are:

- nuhu-ibrahim/time-table-scheduling (https://github.com/nuhu-ibrahim/time-table-scheduling.git)
- akazuko/timetablescheduler (https://github.com/akazuko/timetablescheduler.git)
- Baksonator/evolutionary-timetable-scheduling (Baksonator/evolutionary-timetable-scheduling.git)
- pranavkhurana/Time-table-schedular (https://github.com/pranavkhurana/Time-table-schedular.git) [5]

Chapter 2

SDLC Model

Software Development Life Cycle (SDLC) is a cycle inculcated by the product business for configuration, create and test top notch programming projects. The SDLC intends to deliver a top-notch cycle that meets or surpasses client assumptions, arrives at fruition inside times and cost estimates.
SDLC is a cycle followed for a product assistance, inside a product association. It comprises of an itemized plan depicting how to create, keep up with, supplant and adjust or improve explicit programming. The existence cycle characterizes an approach for working on the nature of programming and the general improvement process.

2.1 SDLC Model used

Agile Model of software development life cycle models is used for developing our project.
REASONS:

- The fundamental reason for this is that in the Agile methodology, software is broken down into small pieces and produced separately.
- Any flaws or errors can be easily discovered and fixed due to individual development.
- If any modifications are required,they may be made quickly and easily.

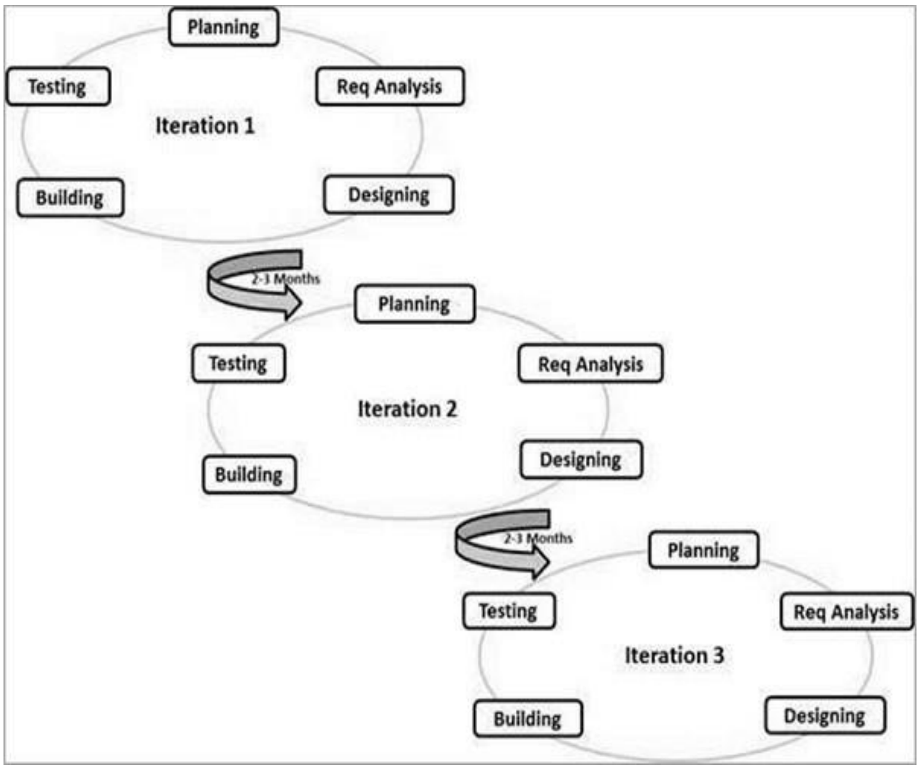


Figure 2.1: Agile phases

2.1.1 Description of Agile Model

The coordinated model recognises that each project should be approached differently and that current tactics should be tailored to the task requirements. The errands are divided into time boxes (humble enclosures) in Agile to communicate precise highlights for a release.

After each emphasis, an iterative process is used, and a working programming assembly is delivered. Each form has the same highlights; the last form has all the features that the customer expects.

The Agile approach to product development began from the beginning and grew in popularity over time due to its versatility and adaptability.

Rational Unified Process (1994), Scrum (1995), Crystal Clear, Extreme Programming (1996), Adaptive Software Development, Feature Driven Development, and Dynamic Systems Development Method (DSDM) are some of the most well-known Agile methodologies (1995). Following the distribution of the Agile Manifesto in 2001, these are now collectively referred to as Agile Methodologies.

Following are the Agile Manifesto principles

- Individual and their interactions: Self-organization and motivation are key in Agile development, as are interactions like co-location and pair programming.
- Working software: Demo working software is considered the best means of communication with the customers to understand their requirements, instead of just depending on documentation.
- Customer collaboration: Because requirements cannot be fully acquired at the start of the project owing to a variety of variables, ongoing customer involvement is critical to obtaining accurate product specifications.
- Responding to change: Agile Development is focused on quick responses to change and continuous development.

Phase 1- Requirements

Partners lead an overall project evaluation to determine the amount of time and resources required for the development cycle. At the same time, the proprietor assesses the threats and concentrates on the various capacities based on their business worth.

Phase 2- Design

The product owner meets with the product development team and informs them of the requirements outlined in the first phase.
The gathering then investigates the sequence for presenting capabilities and identifies the basic apparatuses - programming languages, linguistic structure libraries, and critical systems. Programming development teams can model the standard user interface at the same time.

Phase 3- Development and Coding

After reaching an agreement with the client, the team works on the final product. The item is delivered in phases, in separate runs, each one meant to work on the item’s ongoing adaption.

The underlying delivery is likely to undergo numerous changes in order to provide enhanced functionality and additional aspects.

Each step includes testing, and the final product should also be thoroughly tested. Scrum and the Kanban approach, which allows for development interaction based on specific projects, can be used at this stage.

Phase 4- Testing

Because the product is now available to consumers, the group should oversee a series of tests to ensure that it is completely functional. If any expected faults or defects are discovered, the engineers will immediately remedy them. They also received customer feedback at this point.

Phase 5- Deployement

The product is presently completely sent and accessible to clients. This activity places him in the support stage. During this stage, the product advancement group offers continuous help to keep the framework chugging along as expected and fix any new bugs. Over the long haul, further cycles are feasible to refresh a current item or add other usefulness.

Phase 6- Review

That is the last phase of the Agile advancement cycle. Subsequent to finishing every one of the past transformative phases, the improvement group presents to the proprietor the outcome accomplished in gathering the prerequisites. From that point onward, the Agile programming advancement stages begin once again - either with another cycle or moving to the following stage and scaling Agile.

Chapter 3

Requirement Analysis

Programming necessity is a practical or non-useful should be executed in the framework. Practical means offering specific support to the user.
For instance, in setting the banking application the utilitarian necessity will be when client chooses "View Balance" they should have the option to take a gander at their most recent record balance.

Programming necessity can likewise be a non-practical, it tends to be a presentation prerequisite. For instance, a non-utilitarian prerequisite is where each page of the framework ought to be apparent to the clients inside 5 seconds.
So basically, software requirement is a

- Functional
- Non-functional

need that must be carried out into the framework. Programming necessity are generally communicated as proclamations.

An engaged and point by point necessities investigation can assist you with keeping away from issues like these. This is the most common way of discovering,defining, and reporting the necessities that are connected with a particular business objective. Also, it’s the interaction by which you plainly and unequivocally characterize the extent of the task, so you can evaluate the timescales and assets expected to finish it.

3.1 Functional Requirements

The project should be able to provide these following functional requirements:

- To maintain login database with correct login credentials
- To be able to show the schedule any time to end user
- To perform user interaction successfully and take input values correctly
- To provide a schedule from the provided data
- To show the schedule in correct format to end user
- To insert, update or delete data to and from database successfully

3.2 Operational Requirements

The project should be able to perform these following operational requirements:

- To correctly output log and debug symbols to the file for debugging purpose
- To make a backup and preserve these log and debug files on the event of crash
- To perform schedule generation algorithm process in case of update
- To be able to synchronize the updated schedule with different users

3.3 Technical Requirements

Following are the technical requirements which need to be fulfilled for the project:

- Availability of a "C/C++ compiler"
- Availability of a "MySQL DBMS system" in machine
- Availability of "Python" and "Pip"
- Availability of build tools like "CMake, Nmake, Unix-Makefiles"
- Availability of "git" if user wants the latest development support of project

3.4 System Requirements

Following are the system requirements that need to be fulfilled to run project successfully:

- Windows or Linux/Unix Operating System
- CPU: Dual Core or above (Intel/AMD)
- RAM: OS dependent (but preferrably 2GB or above)
- RAM: OS dependent (but preferrably 2GB or above)

Chapter 4

Design

Programming plan investigation incorporates movements of every kind, which help the change of necessity determination into execution. Programming plan particular necessity determine all useful and non-practical assumptions from the product. These prerequisite details come looking like comprehensible and justifiable records, to which a PC doesn't have anything to do.Project supervisors depend on savvy plan to stay away from slip-ups and offer boundaries to keep key parts of the venture, like the task timetable and spending plan, on target.

Various Diagrams which are used in software developement life cycle are used in this projects for example Dataflow diagram is designed for tracing the flow of the data it is divided in two parts LEVEL ZERO DFD and LEVEL ONE DFD in which level zero dfd act as the context diagram and level one is the elaborative form. Then Entity relationship diagram is also used for the description of database table and the flow of data from the database to the algorithm. Use case is also designed for defining the actors on this project we have three actors ADMIN, USER(can be instructor) and STUDENT these three have different access Admin can add or remove entities into the database, User can select the stored data and can generate timetable and student can view timetable by using the provided credentials.

Process flow chart is used for analysing the complete flow of the project how the data input and structuring is done, flow of all the genetic operator is also defined in the process flow chart and the last one is the class diagram which shows aggregation and the composition of the objects depicted in the project all the classes is being defined in the class diagram.

4.1 Data Flow Diagrams

We have designed till level one of data flow diagram in which complete flow of data is defined.

4.1.1 Level Zero Data Flow Diagram

Level Zero DFD acts as a context diagram for the project.It shows three entities which are working on the algorithm one is admin who is having accessibility of adding or removing data. whereas an user can only select data for generating timetable and the third entity student can login and see the generated time table.which is shown in figure 4.1

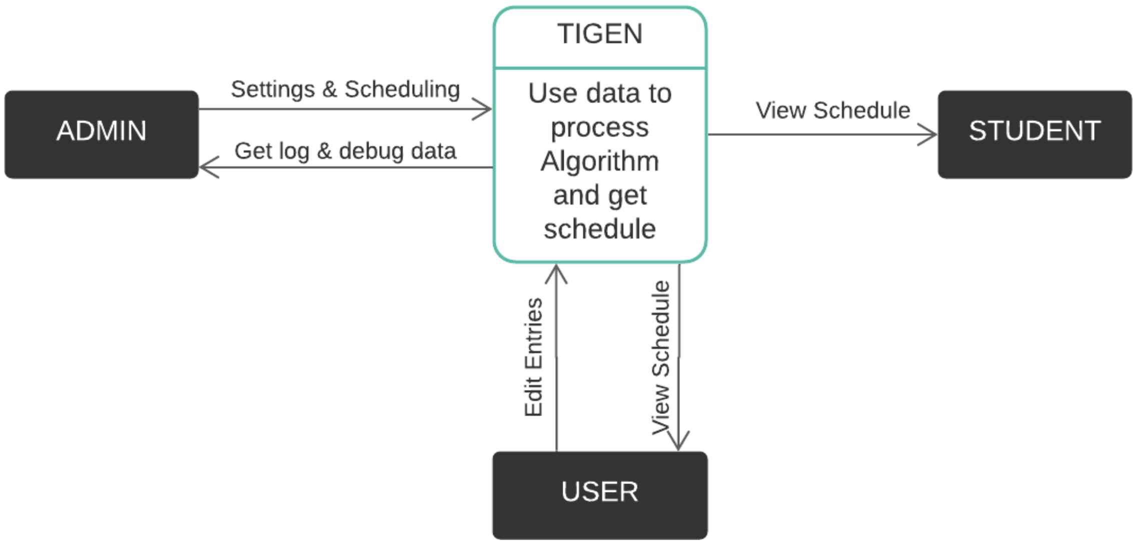


Figure 4.1: DFD-level 0

4.1.2 Level One Data Flow Diagram

Level One DFD Defines the deep knowledge about the flow of data with in the entities of the project.It also elaborates the Algorithm process from context diagram in detail all the data flow into the algorithm is depicted in Level one DFD. Which is shown in figure 4.2

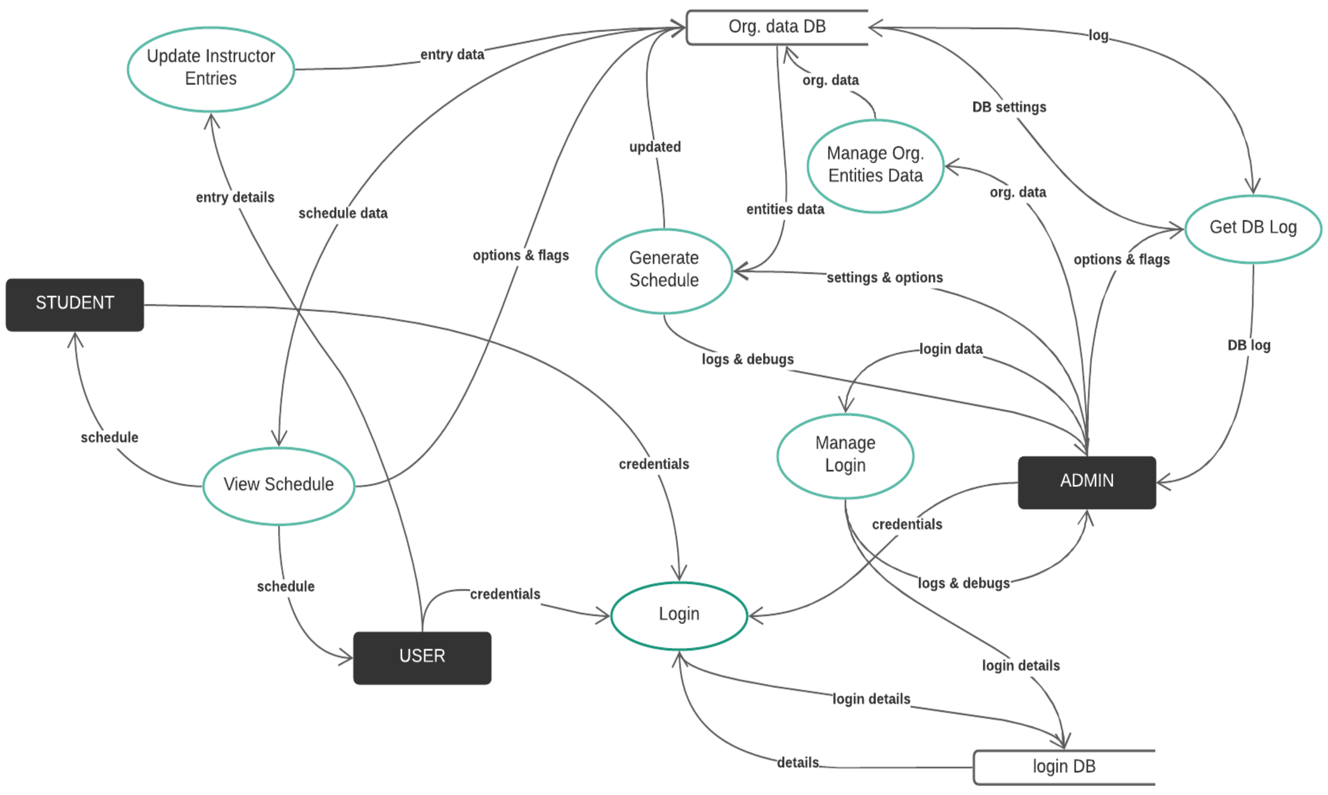


Figure 4.2: DFD-Level 1

4.2 ER diagram

Entity Relationship Diagram contains information about the database used in the project like how many tables are there in the database how they are related to each other. In this project we have five tables which are connected to each other through some relation ship there are some tables too which acts as a foreign key for other table. Figure 4.3 represents the Entity relationship diagram for the project.

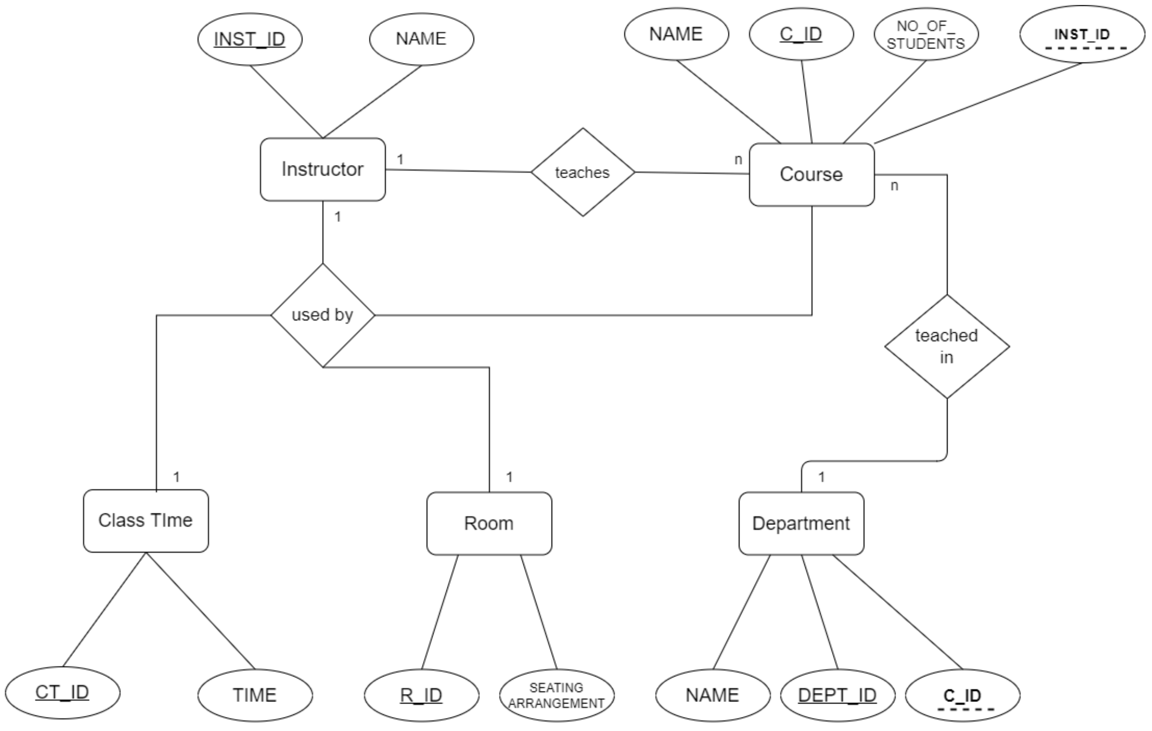


Figure 4.3: ER Diagram

4.3 Usecase Diagram

There are two types of users which are dealing with out project one is admin and other one is user(or an instructor) an admin has all the privileged where as an user can modify details of data provided such as Instructors , courses etc.

There is one more use case which is student who is Viewing timetable. Use case diagram of the project is shown in figure 4.4

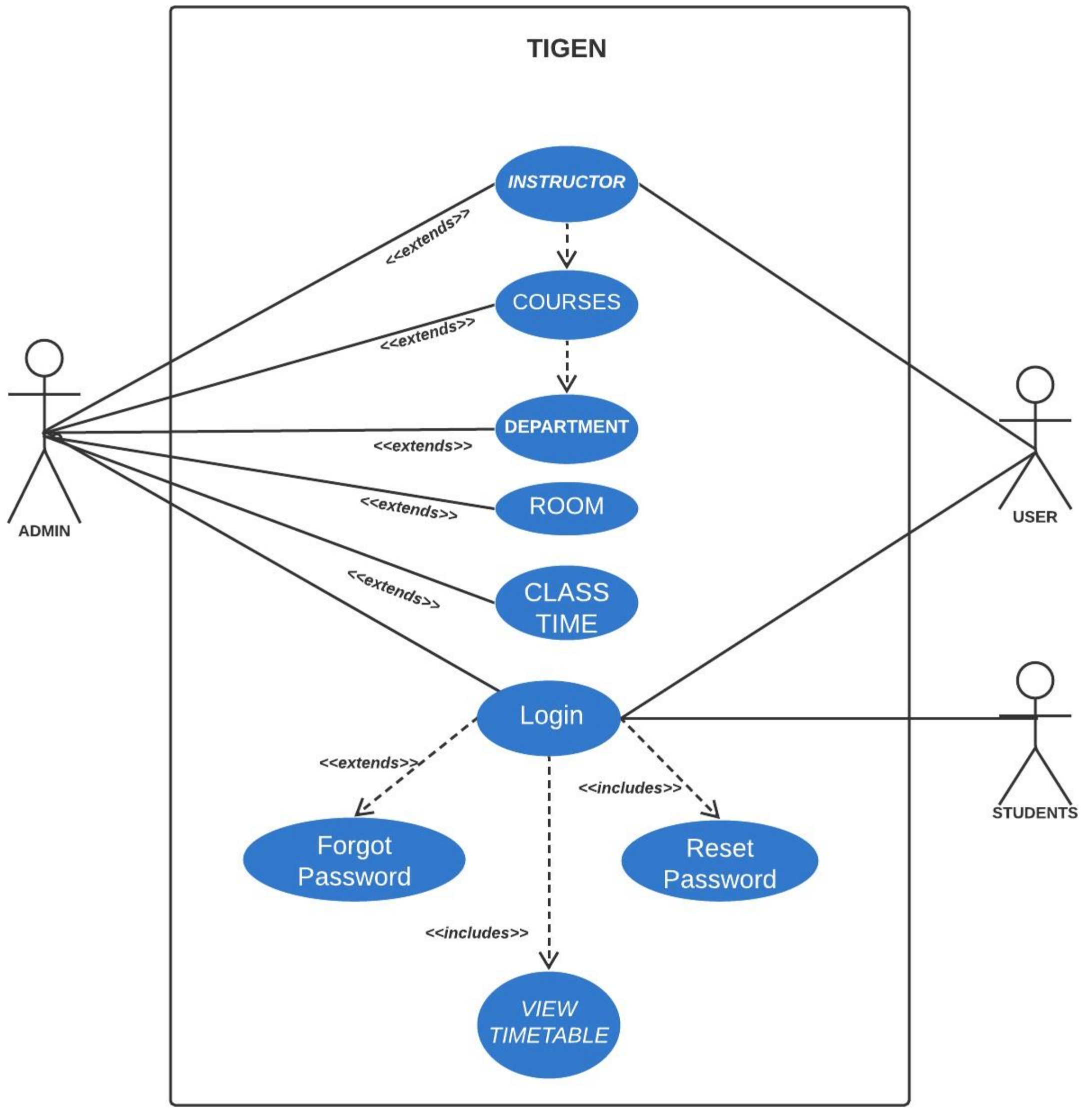


Figure 4.4: Usecase Diagram

4.4 Process Flow Chart

Process flow chart depicts the complete flow of the process as in our case it starts with the database then moving on it has to run the complete flow of genetic algorithm then after it provides out in an user understandable by processing it in front end.

In figure 4.5 Process Flow cart is represented.

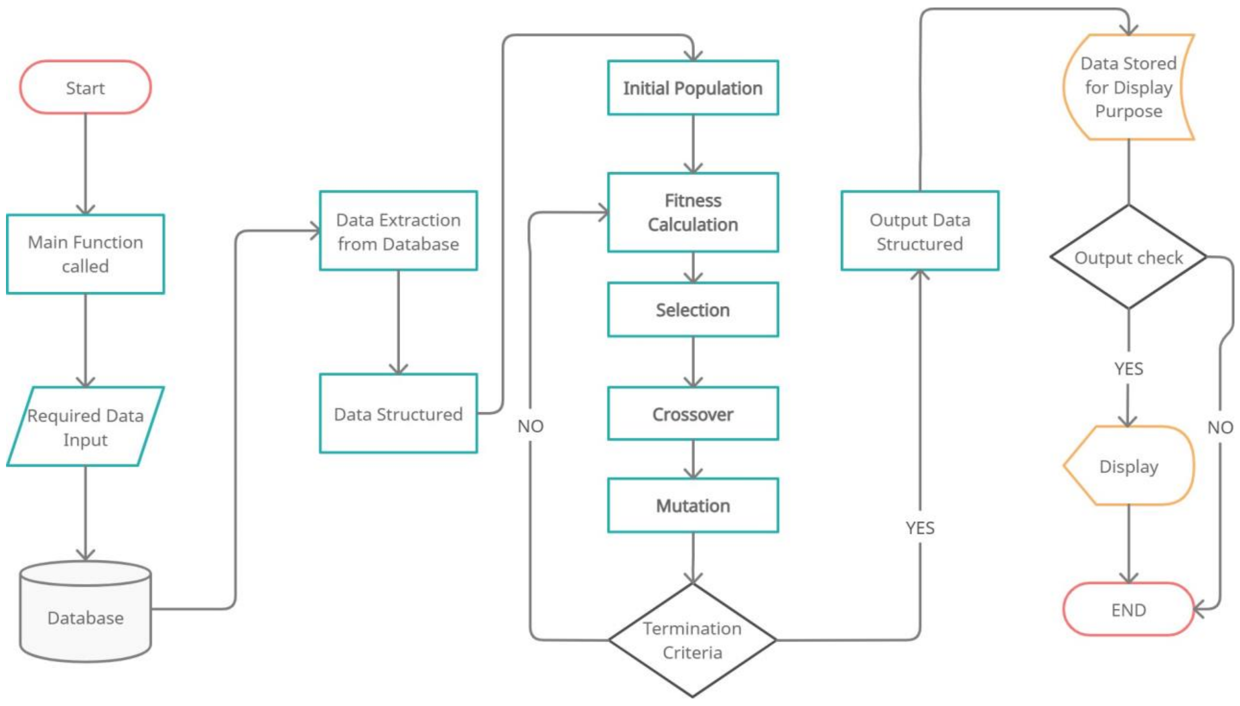


Figure 4.5: Process Flow chart

4.5 Class Diagram

Class Diagram contains all the functions and variable used in this project it also shows the composition and aggregation amongst various classes while cardinalty between classes is also mentioned in the class Diagram.
which is shown in figure 4.6

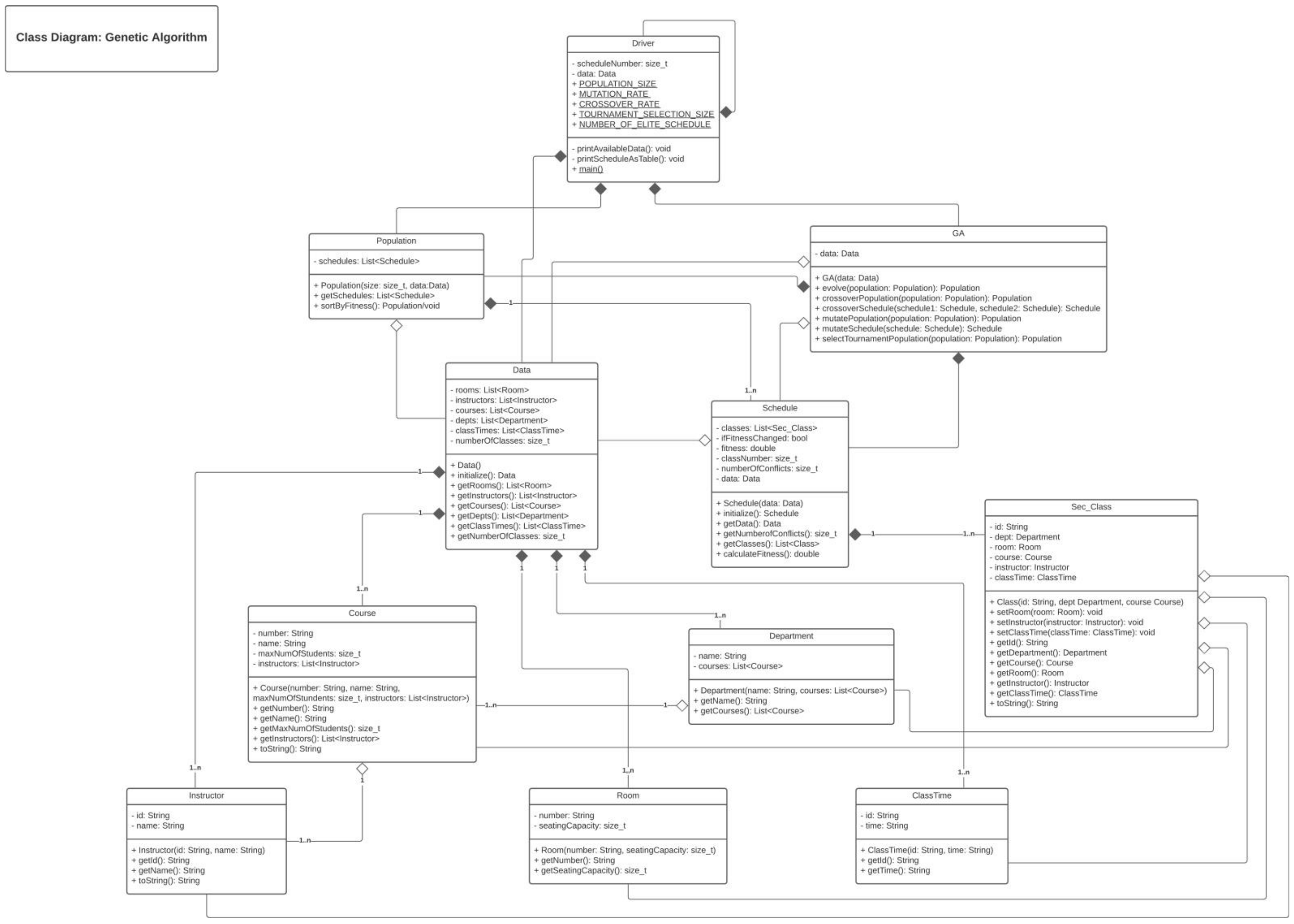

Class Diagram: Genetic Algorithm
Driver
- scheduleNumber: size_t
- data: Data
+ POPULATION_SIZE
+ MUTATION_RATE
+ CROSSOVER_RATE
+ TOURNAMENT_SELECTION_SIZE
+ NUMBER_OF_ELITE_SCHEDULE
- printAvailableData(): void
- printScheduleAsTable(): void
+ main()
Population
- schedules: List<Schedule>
+ Population(size: size_t, data:Data)
+ getSchedules: List<Schedule>
+ sortByFitness(): Population/void
GA
- data: Data
+ GA(data: Data)
+ evolve(population: Population): Population
+ crossoverPopulation(population: Population): Population
+ crossoverSchedule(schedule1: Schedule, schedule2: Schedule): Schedule
+ mutatePopulation(population: Population): Population
+ mutateSchedule(schedule: Schedule): Schedule
+ selectTournamentPopulation(population: Population): Population
Data
- rooms: List<Room>
- instructors: List<Instructor>
- courses: List<Course>
- depts: List<Department>
- classTimes: List<ClassTime>
- numberOfClasses: size_t
+ Data()
+ initialize(): Data
+ getRooms(): List<Room>
+ getInstructors(): List<Instructor>
+ getCourses(): List<Course>
+ getDepts(): List<Department>
+ getClassTimes(): List<ClassTime>
+ getNumberOfClasses: size_t
Schedule
- classes: List<Sec_Class>
- ifFitnessChanged: bool
- fitness: double
- classNumber: size_t
- numberOfConflicts: size_t
- data: Data
+ Schedule(data: Data)
+ initialize(): Schedule
+ getData(): Data
+ getNumberofConflicts(): size_t
+ getClasses(): List<Class>
+ calculateFitness(): double
Sec_Class
- id: String
- dept: Department
- room: Room
- course: Course
- instructor: Instructor
- classTime: ClassTime
+ Class(id: String, dept Department, course Course)
+ setRoom(room: Room): void
+ setInstructor(instructor: Instructor): void
+ setClassTime(classTime: ClassTime): void
+ getId(): String
+ getDepartment(): Department
+ getCourse(): Course
+ getRoom(): Room
+ getInstructor(): Instructor
+ getClassTime(): ClassTime
+ toString(): String
Course
- number: String
- name: String
- maxNumOfStudents: size_t
- instructors: List<Instructor>
+ Course(number: String, name: String, maxNumOfStundents: size_t, instructors: List<Instructor>)
+ getNumber(): String
+ getName(): String
+ getMaxNumOfStudents(): size_t
+ getInstructors(): List<Instructor>
+ toString(): String
Department
- name: String
- courses: List<Course>
+ Department(name: String, courses: List<Course>)
+ getName(): String
+ getCourses(): List<Course>
Instructor
- id: String
- name: String
+ Instructor(id: String, name: String)
+ getId(): String
+ getName(): String
+ toString(): String
Room
- number: String
- seatingCapacity: size_t
+ Room(number: String, seatingCapacity: size_t)
+ getNumber(): String
+ getSeatingCapacity(): size_t
ClassTime
- id: String
- time: String
+ ClassTime(id: String, time: String)
+ getId(): String
+ getTime(): String
1
1..n

Figure 4.6: Class Diagram

Chapter 5

Algorithm

During the development of tigen we have followed sprint cycles according to our proposed SDLC model. We have developed this project in various sprints of more than fifteen days each. Initial sprint is all about the collection of requirements and to understand the need of our project. After collecting all the requirements and understanding the need we discussed the various entities which we have to manage in tigen. Entities such as Instructors , Courses , class times, departments , rooms all these have to be managed which will act as data in our algorithm.

Data Encoding: After collecting all the entity data then we structured data according to our need such that we can pass into our algorithm all the data is encoded in proper format before we pass it to the algorithm. An DATA class is written in order to maintain structuring and encoding of data. Data stored by the user is called when we invoke create data function.

Listing 5.1: Data class

```
void data::create_data() {
entities::room r1("R1", 155);
entities::room r2("R2", 170);
entities::room r3("R3", 160);
entities::room r4("R4", 180);
entities::room r5("R5", 150);
this->_rooms = std::vector<entities::room>{r1, r2, r3, r4, r5};
```

```
// classtimings
entities::class_time ct1("CT1", "MWF␣09:00␣−␣11:00");
entities::class_time ct2("CT2", "MWF␣12:00␣−␣14:00");
entities::class_time ct3("CT3", "MWF␣15:00␣−␣17:00");
entities::class_time ct4("CT4", "TTS␣09:00␣−␣11:00");
entities::class_time ct5("CT5", "TTS␣12:00␣−␣14:00");
entities::class_time ct6("CT6", "MWF␣15:00␣−␣17:00");
this->_class_times = std::vector<entities::class_time>
            {ct1, ct2, ct3, ct4, ct5, ct6};

// instructors for the batch
entities::instructor i1("I1", "Dr.␣Markandeya");
entities::instructor i2("I2", "Dr.␣Gautam");
entities::instructor i3("I3", "Dr.␣Bharadwaj");
entities::instructor i4("I4", "Dr.␣Shandilya");
entities::instructor i5("I5", "Dr.␣Angiras");
entities::instructor i6("I6", "Dr.␣Virajas");
entities::instructor i7("I7", "Dr.␣Kashyap");
entities::instructor i8("I8", "Dr.␣Satya");
entities::instructor i9("I9", "Dr.␣Kavya");
entities::instructor i10("I10", "Dr.␣Garg");
entities::instructor i11("I11", "Dr.␣Kapil");
entities::instructor i12("I12", "Dr.␣Kanad");
entities::instructor i13("I13", "Dr.␣Anusuya");
entities::instructor i14("I14", "Dr.␣Agastya");
entities::instructor i15("I15", "Dr.␣Vajashrava");
this->_instructors = std::vector<entities::instructor>
            {i1, i2, i3, i4, i5, i6, i7, i8, i9,
            i10, i11, i12, i13, i14, i15 };

// courses running
entities::course c1("C1", "Automobile", 120,
std::vector<entities::instructor>{i3, i5});
    entities::course c2("C2", "Aeronautics", 150,s
td::vector<entities::instructor>{i5, i4});
    entities::course c3("C3", "Metullurgy", 150,
std::vector<entities::instructor>{i10});
    entities::course c4("C4", "Biological", 140,
std::vector<entities::instructor>{i1, i7});
    entities::course c5("C5", "Chemical", 150,
std::vector<entities::instructor>{i1, i7});
    entities::course c6("C6", "Embedded␣Programming", 120,
std::vector<entities::instructor>{i2});
    entities::course c7("C7", "Web␣Development", 170,s
td::vector<entities::instructor>{i6, i9});
    entities::course c8("C8", "Networking␣Security", 140,
std::vector<entities::instructor>{i2, i8});
    entities::course c9("C9", "Electronics", 130,
std::vector<entities::instructor>{i2, i11});
    entities::course c10("C10", "Robotics", 180,
std::vector<entities::instructor>{i11, i2, i14});
    entities::course c11("C11", "GIS␣&␣Remote␣Sensing", 120,
std::vector<entities::instructor>{i11, i14});
    entities::course c12("C12", "Construction", 110,
std::vector<entities::instructor>{i12, i15});
    entities::course c13("C13", "Ceramic", 90,s
td::vector<entities::instructor>{i13});
    entities::course c14("C14", "Mining", 100,s
td::vector<entities::instructor>{i12, i15});
    this->_courses = std::vector<entities::course>
    {c1, c2, c3, c4, c5, c6, c7, c8, c9, c10, c11, c12, c13, c14};

    // departments available
    entities::department d1("Dept.␣of␣Mechanincal␣Engineering",
                std::vector<entities::course>{c1, c2, c3});
    entities::department d2("Dept.␣of␣Nuero Science"
                std::vector<entities::course>{c4, c5});
    entities::department d3("Dept.␣of␣Information␣Technology",
                std::vector<entities::course>{c6, c7, c8});
    entities::department d4("Dept.␣of␣Electrical␣Engineering",
                std::vector<entities::course>{c9, c10, c11});
    entities::department d5("Dept.␣of␣Civil␣Engineering",
                std::vector<entities::course>{c12, c13, c14});
    this->_depts = std::vector<entities::department>
                {d1, d2, d3, d4, d5};
    for (const entities::department& dept: this->_depts) {
        _classes_count += dept.get_courses().size(); }
    }
    std::vector<entities::department> data::get_deparatments()
                const { return this->_depts; }
    std::vector<entities::course> data::get_courses()
                const { return this->_courses; }
    std::vector<entities::room> data::get_rooms()
                const { return this->_rooms; }
    std::vector<entities::instructor> data::get_instructors()
                const { return this->_instructors; }
    std::vector<entities::class_time> data::get_class_times()
                const { return this->_class_times; }
    size_t data::get_classes_count()
                const { return this->_classes_count; }
}
```

[7]

After this data is encoded and ready to pass to our algorithm then genetic algorithm class is invoked. This class contains methods of various genetic operators such as method for fitness function which generates population according to the rule of ’SURVIVAL OF THE FITTEST’ , selection, crossover and mutation and a pre defined termination criteria.

Initial Population: Creating an initial population is the first step of genetic algorithm cycle. A number of chromosomes is created on the given data set, which combines to form a population set. Population is created having hard constraints into consideration.

Listing 5.2: Population Method

```
population* genetic_algo::crossover_population(population* pop) {
population* crs_pop = new population
            (pop->get_schedules().size(), this->_gene_data);

for (size_t i = 0; i < NUMBER_OF_ELITE_SCHEDULES; ++i) {
        crs_pop->get_schedules()[i] = pop->get_schedules().at(i);
}

for (size_t i = NUMBER_OF_ELITE_SCHEDULES; i
pop->get_schedules().size(); ++i){
static_cast<double>(RAND_MAX))) {
        if (CROSSOVER_RATE > util::rngr(2.0)) {
population* trnmnt_pop1 = this->select_tournament_population(pop);
trnmnt_pop1->sort_by_fitness();
population* trnmnt_pop2 = this->select_tournament_population(pop);
trnmnt_pop2->sort_by_fitness();
schedule sch1 = trnmnt_pop1->get_schedules().at(0);
schedule sch2 = trnmnt_pop2->get_schedules().at(0);
// try deleting *trnmnt_pop here, since sch1 and sch2 are copied
if (trnmnt_pop1)
        delete trnmnt_pop1;
if (trnmnt_pop2)
        delete trnmnt_pop2;
schedule* crs_sch = crossover_schedule(sch1, sch2);
crs_pop->get_schedules()[i] = crs_sch;
if (crs_sch)
delete crs_sch;  // trying to free crs_sch here
}
else {
crs_pop ->get_schedules ()[ i ] = pop->get_schedules (). at ( i );
    }
}
```

Fitness Function: Fitness function uses below formula for calculating fittest chromosome.

$$Fitnessvalue = 1/Numberofconflicts. \tag{5.1}$$

Conflicts is defined as the clash between two chromosomes or interrupting any constraints in this model.Conflicts can be generated in several situations due to violation of constraints.Hard constraints in which no timetable generation is possible as hard constraints can not be compromised. Soft constraints such as two instructors are assigned to same class time, room is pre-occupied, same instructors are assigned to two different courses at same time, number of students enrolled in a course is greater than room capacity these all constrains generates conflicts in a chromosome which is used in fitness function.

Listing 5.3: Fitness Function

```
double schedule::calculate_fitness() {
this->_number_of_conflicts = 0;
for (const entities::sec_class& cls: this->_sec_classes) {
    if (cls.get_room().get_capacity()
    cls.get_course().get_max_students())
    this->_number_of_conflicts++;
    [&](const entities::sec_class& tcls) {
    return (&tcls - &(*this->_sec_classes.cbegin())) >= (&cls
    &(*this->_sec_classes.cbegin()));
    });
for (const entities::sec_class& ucls: up_cls) {
    if ((cls.get_class_time() == ucls.get_class_time())
   && (cls.get_id() != ucls.get_id())) {
    if (cls.get_room() == ucls.get_room())
```

```
                this->_number_of_conflicts++;
        if (cls.get_instructor() == ucls.get_instructor())
                this->_number_of_conflicts++;
                                }
                }
}
return 1 / static_cast<double>(this->_number_of_conflicts + 1);
}
```

Selection: Selection is the next step after the creation of initial population. Tigen uses k-Tournament selection for the selection of chromosomes from the population and send them for further operations. It first selects k no of chromosomes from the initial population and the uses the fitness function to find which chromosome is fittest among the k selected chromosome the fittest one moves further and follows rest of the cycle.

Listing 5.4: Population Selection Method

```
population* genetic_algo::select_tournament_population
                (population* pop){
population* tournament_population =
        new population(TOURNAMENT_SELECTION_SIZE, this->_gene_data);
for (size_t i = 0; i < TOURNAMENT_SELECTION_SIZE; ++i)
                {               tournament_population->get_schedules()[i]
                                = pop->get_schedules()
                                [util::rngi(pop->get_schedules().size())];
                }
return tournament_population;
}
```

Crossover: Crossover method is used to create different offspring based on selected chromosomes. Tigen uses Multipoint crossover method for creating offsprings. This crossover uses two chromosomes and creates X new chromosomes. It splits two chromosomes at multiple points and joint with all different points and creates offsprings.

Listing 5.5: Crossover Population Method

```
population* genetic_algo::crossover_population(population* pop) {
```

```
population* crs_pop = new population
              (pop->get_schedules().size(), this->_gene_data);

for (size_t i = 0; i < NUMBER_OF_ELITE_SCHEDULES; ++i) {
          crs_pop->get_schedules()[i] = pop->get_schedules().at(i);
}

for (size_t i = NUMBER_OF_ELITE_SCHEDULES; i
pop->get_schedules().size(); ++i){
static_cast<double>(RAND_MAX))) {
          if (CROSSOVER_RATE > util::rngr(2.0)) {
population* trnmnt_pop1 = this->select_tournament_population(pop);
trnmnt_pop1->sort_by_fitness();
population* trnmnt_pop2 = this->select_tournament_population(pop);
trnmnt_pop2->sort_by_fitness();
schedule sch1 = trnmnt_pop1->get_schedules().at(0);
schedule sch2 = trnmnt_pop2->get_schedules().at(0);
// try deleting *trnmnt_pop here, since sch1 and sch2 are copied
if (trnmnt_pop1)
          delete trnmnt_pop1;
if (trnmnt_pop2)
          delete trnmnt_pop2;
schedule* crs_sch = crossover_schedule(sch1, sch2);
crs_pop->get_schedules()[i]=crs_sch;
if (crs_sch)
delete crs_sch;  // trying to free crs_sch here
}
else {
crs_pop->get_schedules()[i] = pop->get_schedules().at(i);
     }
}
```

Listing 5.6: Crossover method

```
schedule* genetic_algo::crossover_schedule
        (schedule& sch1, schedule& sch2) {
        schedule* sch = new schedule(this->_gene_data);
        size_t count_sec_classes = sch->get_sec_classes().size();
for (size_t i = 0; i < count_sec_classes; ++i) {
if (util::rngr(2.0) > 0.5)
        sch->get_sec_classes()[i] = sch1.get_sec_classes().at(i);
else
        sch->get_sec_classes()[i] = sch2.get_sec_classes().at(i);
    }
return sch;
}
```

Mutation: Mutation operator is used to provide diversity in the initial population. Tigen uses Scramble mutation for this action. This mutation selects number of genes and shuffle their values randomly.

Listing 5.7: Mutation methods

```
population* genetic_algo::mutate_population(population* pop) {
population*mutate_population=
        new population(pop->get_schedules().size(),
        this->_gene_data);
        std::vector<schedule>& schedules
        mutate_population->get_schedules();
for (size_t i = 0; i < NUMBER_OF_ELITE_SCHEDULES; ++i) {
        schedules[i]=pop->get_schedules().at(i);
}
for (size_t i = NUMBER_OF_ELITE_SCHEDULES; i<
    pop->get_schedules().size(); ++i){
        mutate_schedule(pop->get_schedules().at(i));
        schedules[i]=pop->get_schedules().at(i);
}
if (pop)
        delete pop;
return mutate_population;
}
```

Termination Criteria: A defined criterion is set in Tigen for terminating the algorithm. As the number of conflicts becomes zero as soon as the non-conflicted chromosome is found, algorithm gets terminated.

After all the steps when a timetable is generated successfully and which is encoded in form of chromosomes. we parse the result and decode into human readable format.

Chapter 6

Testing

Tigen is using dynamic data as an input and providing results in basis of Assignment problem. In which number of slots of timetable has to be generated in order to provide feasible results. Genetic algorithm cycles contains various operators which uses iteration for finding feasible solutions. Tigen also uses RNG(Random Number Generation) algorithm which is used in some operators. So after following operation the result provided by tigen has to be tested with several test cases which are discussed in following sections.

6.1 Test case 1

All the test cases of tigen is the variation in the number of entities provided to the algorithm. Like in test case 1 we are using a number of entities which is shown in 6.1

Table 6.1: Testcase 1

Entity Name	**No. of entities**
Instructor	5
Courses	10
Department	2
Room	3
ClassTime	6

Test case1 contains the data which is sufficient in order to generate a optimal solution through the algorithm. Code for above test case is written in listing 6.1

Listing 6.1: Testcase 1

```
void data::create_data() {
entities::room r1("R1", 155);
entities::room r2("R2", 170);
entities::room r3("R3", 160);
this->_rooms = std::vector<entities::room>{r1, r2, r3};

// classtimings
entities::class_time ct1("CT1", "MWF␣09:00␣−␣11:00");
entities::class_time ct2("CT2", "MWF␣12:00␣−␣14:00");
entities::class_time ct3("CT3", "MWF␣15:00␣−␣17:00");
entities::class_time ct4("CT4", "TTS␣09:00␣−␣11:00");
entities::class_time ct5("CT5", "TTS␣12:00␣−␣14:00");
entities::class_time ct6("CT6", "MWF␣15:00␣−␣17:00");
this->_class_times = std::vector<entities::class_time>
              {ct1, ct2, ct3, ct4, ct5, ct6};

// instructors for the batch
entities::instructor i1("I1", "Dr.␣Markandeya");
entities::instructor i2("I2", "Dr.␣Gautam");
entities::instructor i3("I3", "Dr.␣Bharadwaj");
entities::instructor i4("I4", "Dr.␣Shandilya");
entities::instructor i5("I5", "Dr.␣Angiras");
this->_instructors = std::vector<entities::instructor>
              {i1, i2, i3, i4, i5};

// courses running
entities::course c1("C1", "Automobile", 120,
std::vector<entities::instructor>{i3, i5});
```

```
    entities::course c2("C2", "Aeronautics", 150,s
td::vector<entities::instructor>{i5, i4});
    entities::course c3("C3", "Metullurgy", 150,
std::vector<entities::instructor>{i10});
    entities::course c4("C4", "Biological", 140,
std::vector<entities::instructor>{i1, i7});
    entities::course c5("C5", "Chemical", 150,
std::vector<entities::instructor>{i1, i7});
    entities::course c6("C6", "Embedded␣Programming", 120,
std::vector<entities::instructor>{i2});
    entities::course c7("C7", "Web␣Development", 170,s
td::vector<entities::instructor>{i6, i9});
    entities::course c8("C8", "Networking␣Security", 140,
std::vector<entities::instructor>{i2, i8});
    entities::course c9("C9", "Electronics", 130,
std::vector<entities::instructor>{i2, i11});
    entities::course c10("C10", "Robotics", 180,
std::vector<entities::instructor>{i11, i2, i14});
    {c1, c2, c3, c4, c5, c6, c7, c8, c9, c10};

    // departments available
    entities::department d1("Dept.␣of␣Mechanincal␣Engineering",
                std::vector<entities::course>{c1, c2, c3});
    entities::department d2("Dept.␣of␣Nuero Science"
                std::vector<entities::course>{c4, c5});
    this->_depts = std::vector<entities::department>
                {d1, d2};
    for (const entities::department& dept: this->_depts) {
        _classes_count += dept.get_courses().size(); }
    }
    std::vector<entities::department> data::get_deparatments()
                const { return this->_depts; }
    std::vector<entities::course> data::get_courses()
                const { return this->_courses; }
    std::vector<entities::room> data::get_rooms()
                const { return this->_rooms; }
    std::vector<entities::instructor> data::get_instructors()
                const { return this->_instructors; }
    std::vector<entities::class_time> data::get_class_times()
                const { return this->_class_times; }
    size_t data::get_classes_count()
                const { return this->_classes_count; }
}
```

6.2 Testcase 2

Variation in the number of entities provided to the algorithm which is different from testcase 1 . In testcase 2 we are using number of entities which is shown in 6.2

Table 6.2: Testcase 2

Entity Name	**No. of entities**
Instructor	15
Courses	20
Department	4
Room	6
ClassTime	6

In testcase 2, we are using large number of entities such that the algorithm is tested for the situation where the data is very dense to handle. Here Dense data means we have number of instructors more than 15 or for each courses less number of instructors is available and each department is number of courses which is grater than 10 in case of each department, Classtime entity is fixed for our case as such we have to also limit slots for the algorithm to provide optimal solution. Code for testcase 2 is written in listing 6.2

Listing 6.2: Testcase 2

```
void data::create_data() {
entities::room r1("R1", 155);
entities::room r2("R2", 170);
entities::room r3("R3", 160);
entities::room r4("R4", 180);
entities::room r5("R5", 150);
entities::room r5("R6", 120);
this->_rooms = std::vector<entities::room>{r1, r2, r3, r4, r5, r6};

// classtimings
entities::class_time ct1("CT1", "MWF␣09:00␣-␣11:00");
entities::class_time ct2("CT2", "MWF␣12:00␣-␣14:00");
entities::class_time ct3("CT3", "MWF␣15:00␣-␣17:00");
entities::class_time ct4("CT4", "TTS␣09:00␣-␣11:00");
entities::class_time ct5("CT5", "TTS␣12:00␣-␣14:00");
entities::class_time ct6("CT6", "MWF␣15:00␣-␣17:00");
this->_class_times = std::vector<entities::class_time>
            {ct1, ct2, ct3, ct4, ct5, ct6};

// instructors for the batch
entities::instructor i1("I1", "Dr.␣Markandeya");
entities::instructor i2("I2", "Dr.␣Gautam");
entities::instructor i3("I3", "Dr.␣Bharadwaj");
entities::instructor i4("I4", "Dr.␣Shandilya");
entities::instructor i5("I5", "Dr.␣Angiras");
entities::instructor i6("I6", "Dr.␣Virajas");
entities::instructor i7("I7", "Dr.␣Kashyap");
entities::instructor i8("I8", "Dr.␣Satya");
entities::instructor i9("I9", "Dr.␣Kavya");
entities::instructor i10("I10", "Dr.␣Garg");
```

```
    entities::instructor i11("I11", "Dr.␣Kapil");
    entities::instructor i12("I12", "Dr.␣Kanad");
    entities::instructor i13("I13", "Dr.␣Anusuya");
    entities::instructor i14("I14", "Dr.␣Agastya");
    entities::instructor i15("I15", "Dr.␣Vajashrava");
    this->_instructors = std::vector<entities::instructor>
                {i1, i2, i3, i4, i5, i6, i7, i8, i9,
                i10, i11, i12, i13, i14, i15 };

    // courses running
    entities::course c1("C1", "Automobile", 120,
std::vector<entities::instructor>{i3, i5});
    entities::course c2("C2", "Aeronautics", 150,s
td::vector<entities::instructor>{i5,  i4});
    entities::course c3("C3", "Metullurgy", 150,
std::vector<entities::instructor>{i10});
    entities::course c4("C4", "Biological", 140,
std::vector<entities::instructor>{i1, i7});
    entities::course c5("C5", "Chemical", 150,
std::vector<entities::instructor>{i1, i7});
    entities::course c6("C6", "Embedded␣Programming", 120,
std::vector<entities::instructor>{i2});
    entities::course c7("C7", "Web␣Development", 170,s
td::vector<entities::instructor>{i6,  i9});
    entities::course c8("C8", "Networking␣Security", 140,
std::vector<entities::instructor>{i2, i8});
    entities::course c9("C9", "Electronics", 130,
std::vector<entities::instructor>{i2, i11});
    entities::course c10("C10", "Robotics", 180,
std::vector<entities::instructor>{i11, i2, i14});
    entities::course c11("C11", "GIS␣&␣Remote␣Sensing", 120,
std::vector<entities::instructor>{i11, i14});
```

```
    entities::course c12("C12", "Construction", 110,
std::vector<entities::instructor>{i12, i15});
    entities::course c13("C13", "Ceramic", 90,s
td::vector<entities::instructor>{i13});
    entities::course c14("C14", "Mining", 100,s
td::vector<entities::instructor>{i12, i15});
    this->_courses = std::vector<entities::course>
    {c1, c2, c3, c4, c5, c6, c7, c8, c9, c10, c11, c12, c13, c14,c15, c

    // departments available
    entities::department d1("Dept.␣of␣Mechanincal␣Engineering",
              std::vector<entities::course>{c1, c2, c3});
    entities::department d2("Dept.␣of␣NueroScience"
              std::vector<entities::course>{c4, c5});
    entities::department d3("Dept.␣of␣Information␣Technology",
              std::vector<entities::course>{c6, c7, c8});
    entities::department d4("Dept.␣of␣Electrical␣Engineering",
              std::vector<entities::course>{c9, c10, c11});
    entities::department d5("Dept.␣of␣Civil␣Engineering",
              std::vector<entities::course>{c12, c13, c14});
    this->_depts = std::vector<entities::department>
              {d1, d2, d3, d4, d5};
    for (const entities::department& dept: this->_depts) {
        _classes_count += dept.get_courses().size(); }
    }
    std::vector<entities::department> data::get_deparatments()
              const { return this->_depts; }
    std::vector<entities::course> data::get_courses()
              const { return this->_courses; }
    std::vector<entities::room> data::get_rooms()
              const { return this->_rooms; }
    std::vector<entities::instructor> data::get_instructors()
```

```
                const { return this->_instructors; }
    std::vector<entities::class_time> data::get_class_times()
                const { return this->_class_times; }
    size_t data::get_classes_count()
                const { return this->_classes_count; }
}
```

6.3 Testcase 3

Test case 3 of tigen is the edge case of tigen in which much fewer entities can be provided such that the constraint can be set for minimum data needed for algorithm to provide results. Classtime entity is fixed for this case too as if we varies classtime then it will not impact result.Number of entities used is shown in 6.3

Table 6.3: Testcase 3

Entity Name	**No. of entities**
Instructor	2
Courses	5
Department	3
Room	2
ClassTime	6

The edge case of tigen is testcase3 which defines the minimun data required for the successfully completion of the algorithm. Code for testcase 3 is written in listing 6.3

Listing 6.3: Testcase 3

```
    void data::create_data() {
    entities::room r1("R1", 155);
    entities::room r2("R2", 170);
    this->_rooms = std::vector<entities::room>{r1, r2};
    // classtimings
    entities::class_time ct1("CT1", "MWF␣09:00␣-␣11:00");
    entities::class_time ct2("CT2", "MWF␣12:00␣-␣14:00");
    entities::class_time ct3("CT3", "MWF␣15:00␣-␣17:00");
    entities::class_time ct4("CT4", "TTS␣09:00␣-␣11:00");
    entities::class_time ct5("CT5", "TTS␣12:00␣-␣14:00");
    entities::class_time ct6("CT6", "MWF␣15:00␣-␣17:00");
    this->_class_times = std::vector<entities::class_time>
                {ct1, ct2, ct3, ct4, ct5, ct6};

    // instructors for the batch
    entities::instructor i1("I1", "Dr.␣Markandeya");
    entities::instructor i2("I2","Dr.␣Gautam");
    this->_instructors = std::vector<entities::instructor>
                {i1, i2};

    // courses running
    entities::course c1("C1", "Automobile", 120,
std::vector<entities::instructor>{i3, i5});
    entities::course c2("C2", "Aeronautics", 150,s
td::vector<entities::instructor>{i5, i4});
    entities::course c3("C3", "Metullurgy", 150,
std::vector<entities::instructor>{i10});
    entities::course c4("C4", "Biological", 140,
std::vector<entities::instructor>{i1, i7});
    entities::course c5("C5", "Chemical", 150,
std::vector<entities::instructor>{i1, i7});
    {c1, c2, c3, c4, c5};

    // departments available
    entities::department d1("Dept.␣of␣Mechanincal␣Engineering",
                std::vector<entities::course>{c1, c2, c3});
    entities::department d2("Dept.␣of␣Nuero Science"
                std::vector<entities::course>{c4, c5});
    entities::department d3("Dept.␣of␣Information␣Technology",
                std::vector<entities::course>{c6, c7, c8});
    this->_depts = std::vector<entities::department>
                {d1, d2, d3};
    for (const entities::department& dept: this->_depts) {
        _classes_count += dept.get_courses().size(); }
    }
    std::vector<entities::department> data::get_deparatments()
                const { return this->_depts; }
    std::vector<entities::course> data::get_courses()
                const { return this->_courses; }
    std::vector<entities::room> data::get_rooms()
                const { return this->_rooms; }
    std::vector<entities::instructor> data::get_instructors()
                const { return this->_instructors; }
    std::vector<entities::class_time> data::get_class_times()
                const { return this->_class_times; }
    size_t data::get_classes_count()
                const { return this->_classes_count; }
}
```

Chapter 7

Deployement

The **tigen** starts its process by calling a *driver* function. This driver function checks the termination criteria for genetic algorithm. The driver function prepares the initial population and then mutate population sub-process is called which works in evolving the initial population to get the desired result.

The evolution of the initial population starts by calling the method *mutate_population* which exists in class *gen_algo*. The interaction between the operators of genetic algorithm determines the performance of the product using genetic algorithm. The mutate population calls the *crossover_population* method of the same class which in turns calls the *tournament_selection* method.

The selection operator used in *tigen* is K-tournament selection. In K-tournament selection, K is the constant which tells the number of chromosomes we need to select randomly from the population of current generation. Fitness function for these chromosomes is calculated and the fittest chromosome from these K chromosomes is returned as resultant chromosome for selection operation.

The *tournament_selection* method is called two times from *crossover_population* method, reason being to get the pair of chromosomes for generating offspring for new generation. Lets focus on how *crossover_population* works.

Chapter 8

Review

8.1 Result

Timetable generates through tigen are the implementation of Genetic algorithm it starts with the evolution of population which is randomly generated and then forwards toward selection of genetic operator using k-tournament selection which randomly selects 3 chromosomes from randomly selected chromosome as the value of k is considered three then the crossover used is multipoint crossover which mutates genes at different points then forwards to mutation operator which is scramble mutation which changes genes randomly then after following complete flow of genetic operators termination criteria is satisfied then the result is generated which is not conflicted.

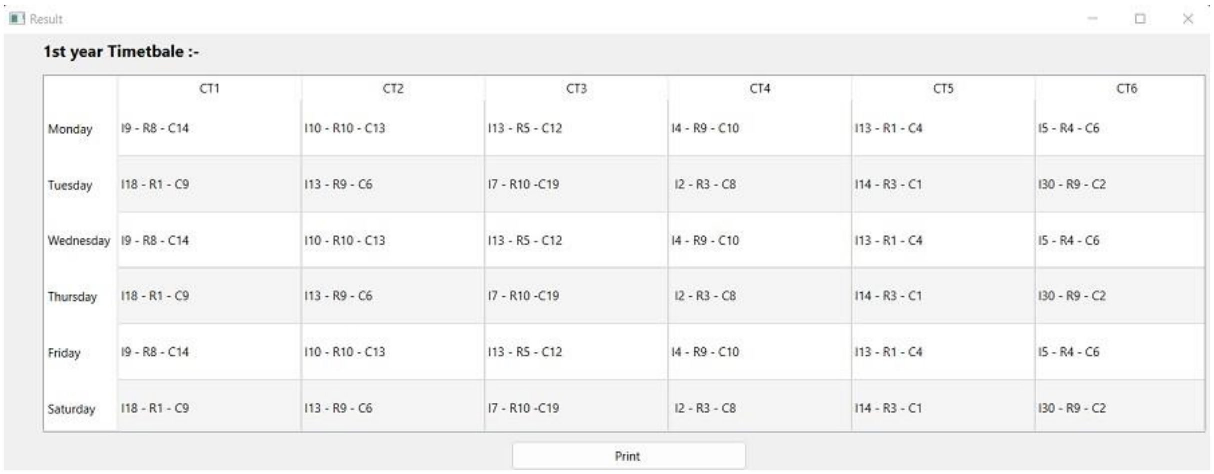
In Figure 8.1, Graphical User Interface result is shown in which all slots are filled using the mnemonic which is used in chromosomes.

In Figure 8.2, Terminal User Interface result is shown which can be parsed into GUI result by using parser.

In Figure 8.3, Selection of Instructor on Terminal User Interface.

In Figure 8.4, Selection of courses on terminal User Interface.

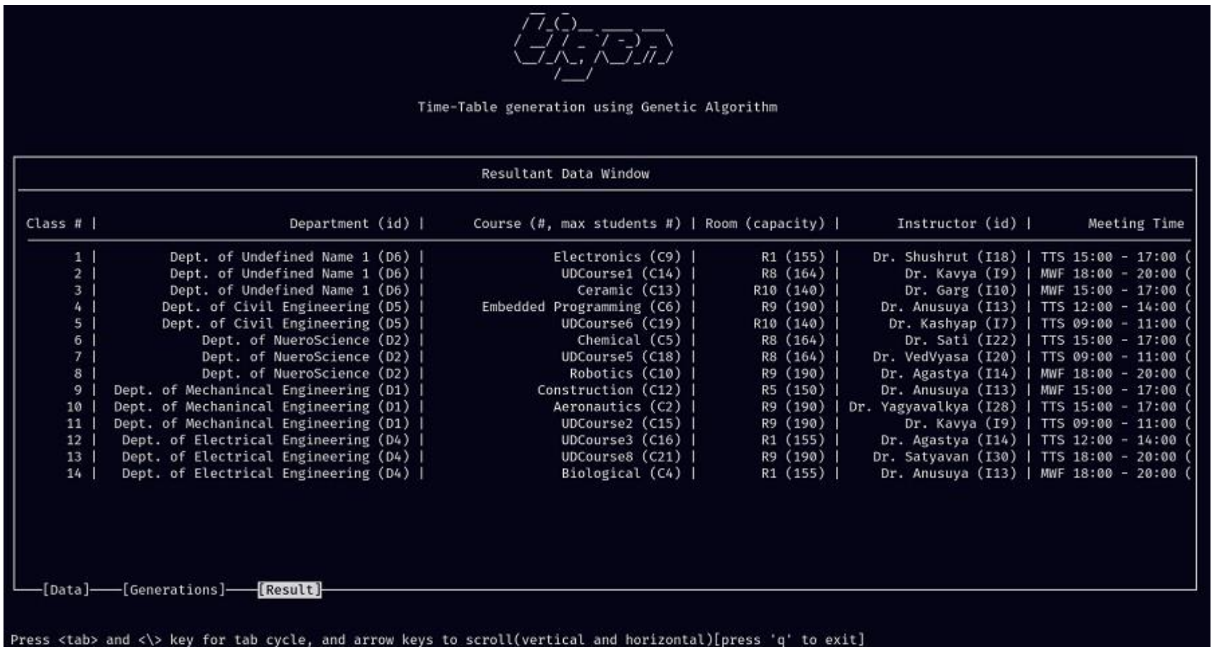
8.2 Screenshots



1st year Timetbale :-

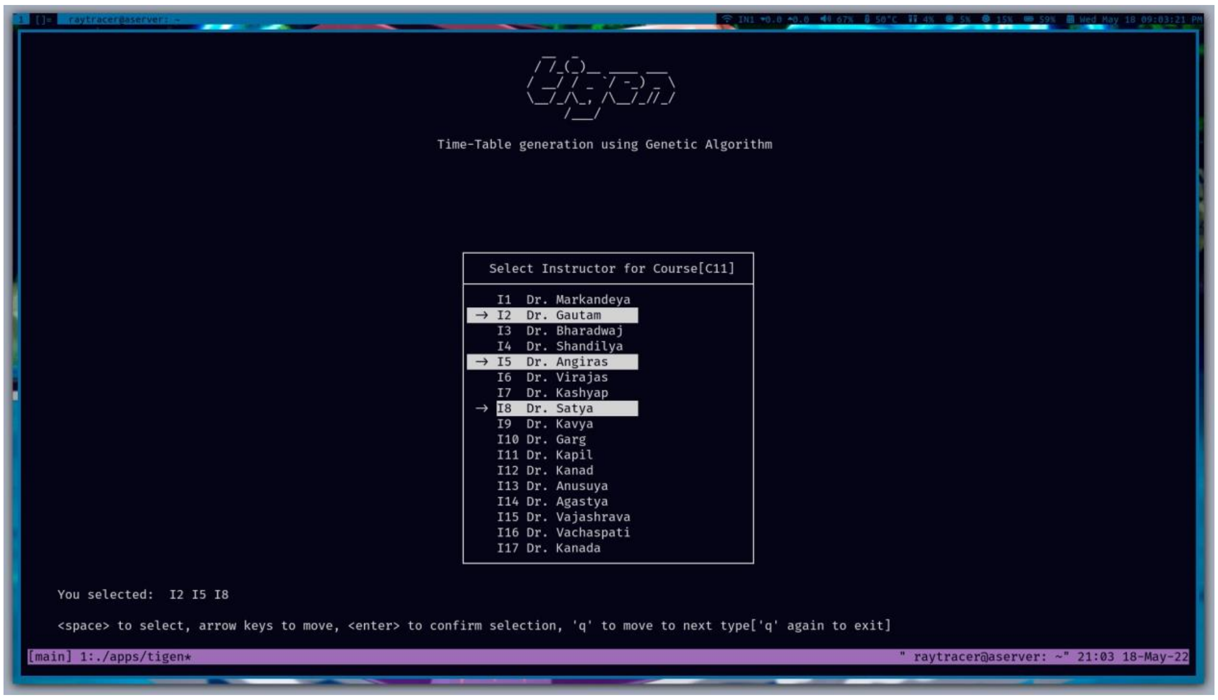
	CT1	CT2	CT3	CT4	CT5	CT6
Monday	I9 - R8 - C14	I10 - R10 - C13	I13 - R5 - C12	I4 - R9 - C10	I13 - R1 - C4	I5 - R4 - C6
Tuesday	I18 - R1 - C9	I13 - R9 - C6	I7 - R10 -C19	I2 - R3 - C8	I14 - R3 - C1	I30 - R9 - C2
Wednesday	I9 - R8 - C14	I10 - R10 - C13	I13 - R5 - C12	I4 - R9 - C10	I13 - R1 - C4	I5 - R4 - C6
Thursday	I18 - R1 - C9	I13 - R9 - C6	I7 - R10 -C19	I2 - R3 - C8	I14 - R3 - C1	I30 - R9 - C2
Friday	I9 - R8 - C14	I10 - R10 - C13	I13 - R5 - C12	I4 - R9 - C10	I13 - R1 - C4	I5 - R4 - C6
Saturday	I18 - R1 - C9	I13 - R9 - C6	I7 - R10 -C19	I2 - R3 - C8	I14 - R3 - C1	I30 - R9 - C2

Figure 8.1: Graphical user Interface result



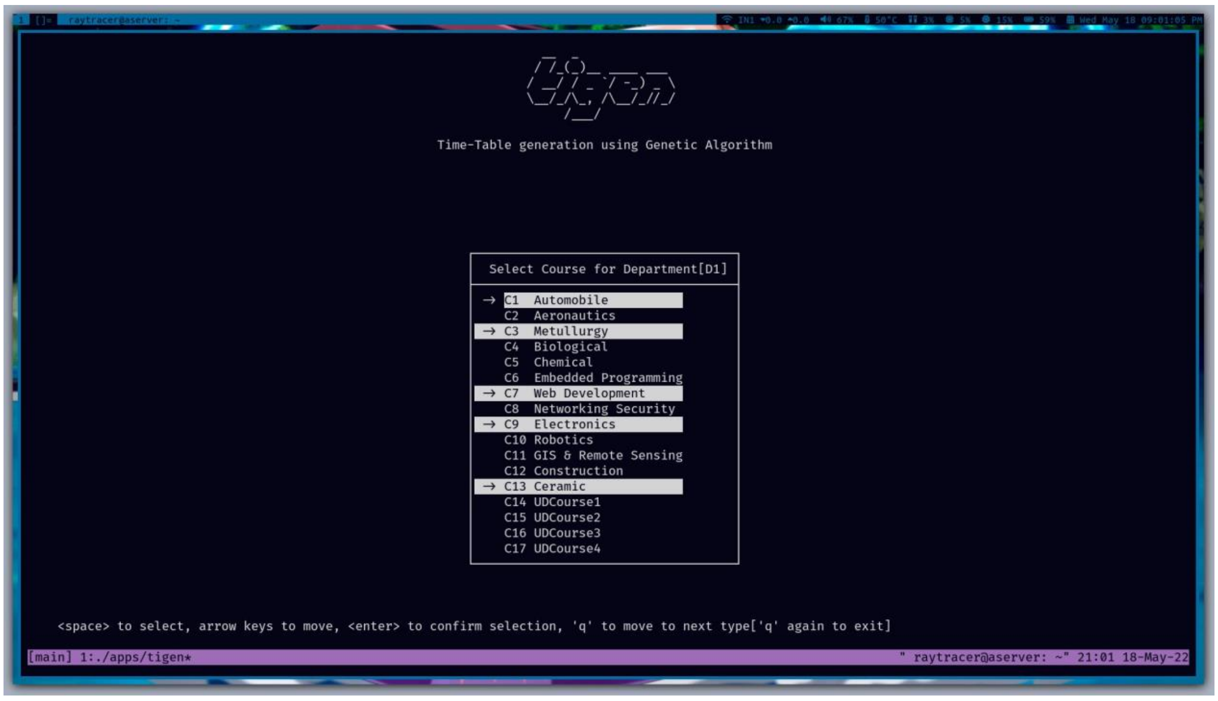
```
                         Time-Table generation using Genetic Algorithm

                                   Resultant Data Window

Class # |                   Department (id) |      Course (#, max students #) | Room (capacity) |      Instructor (id) |      Meeting Time

      1 |      Dept. of Undefined Name 1 (D6) |             Electronics (C9) |       R1 (155) |   Dr. Shushrut (I18) | TTS 15:00 - 17:00 (
      2 |      Dept. of Undefined Name 1 (D6) |             UDCourse1 (C14) |       R8 (164) |      Dr. Kavya (I9) | MWF 18:00 - 20:00 (
      3 |      Dept. of Undefined Name 1 (D6) |               Ceramic (C13) |      R10 (140) |      Dr. Garg (I10) | MWF 15:00 - 17:00 (
      4 |     Dept. of Civil Engineering (D5) |   Embedded Programming (C6) |       R9 (190) |    Dr. Anusuya (I13) | TTS 12:00 - 14:00 (
      5 |     Dept. of Civil Engineering (D5) |             UDCourse6 (C19) |      R10 (140) |     Dr. Kashyap (I7) | TTS 09:00 - 11:00 (
      6 |         Dept. of NueroScience (D2) |                Chemical (C5) |       R8 (164) |       Dr. Sati (I22) | TTS 15:00 - 17:00 (
      7 |         Dept. of NueroScience (D2) |             UDCourse5 (C18) |       R8 (164) |   Dr. VedVyasa (I20) | TTS 09:00 - 11:00 (
      8 |         Dept. of NueroScience (D2) |              Robotics (C10) |       R9 (190) |    Dr. Agastya (I14) | MWF 18:00 - 20:00 (
      9 | Dept. of Mechanincal Engineering (D1) |          Construction (C12) |       R5 (150) |    Dr. Anusuya (I13) | MWF 15:00 - 17:00 (
     10 | Dept. of Mechanincal Engineering (D1) |             Aeronautics (C2) |       R9 (190) | Dr. Yagyavalkya (I28) | TTS 15:00 - 17:00 (
     11 | Dept. of Mechanincal Engineering (D1) |             UDCourse2 (C15) |       R9 (190) |      Dr. Kavya (I9) | TTS 09:00 - 11:00 (
     12 |  Dept. of Electrical Engineering (D4) |             UDCourse3 (C16) |       R1 (155) |    Dr. Agastya (I14) | TTS 12:00 - 14:00 (
     13 |  Dept. of Electrical Engineering (D4) |             UDCourse8 (C21) |       R9 (190) |   Dr. Satyavan (I30) | TTS 18:00 - 20:00 (
     14 |  Dept. of Electrical Engineering (D4) |              Biological (C4) |       R1 (155) |    Dr. Anusuya (I13) | MWF 18:00 - 20:00 (

──[Data]──[Generations]──[Result]──

Press <tab> and <\> key for tab cycle, and arrow keys to scroll(vertical and horizontal)[press 'q' to exit]
```

Figure 8.2: Parsed Terminal user Interface result



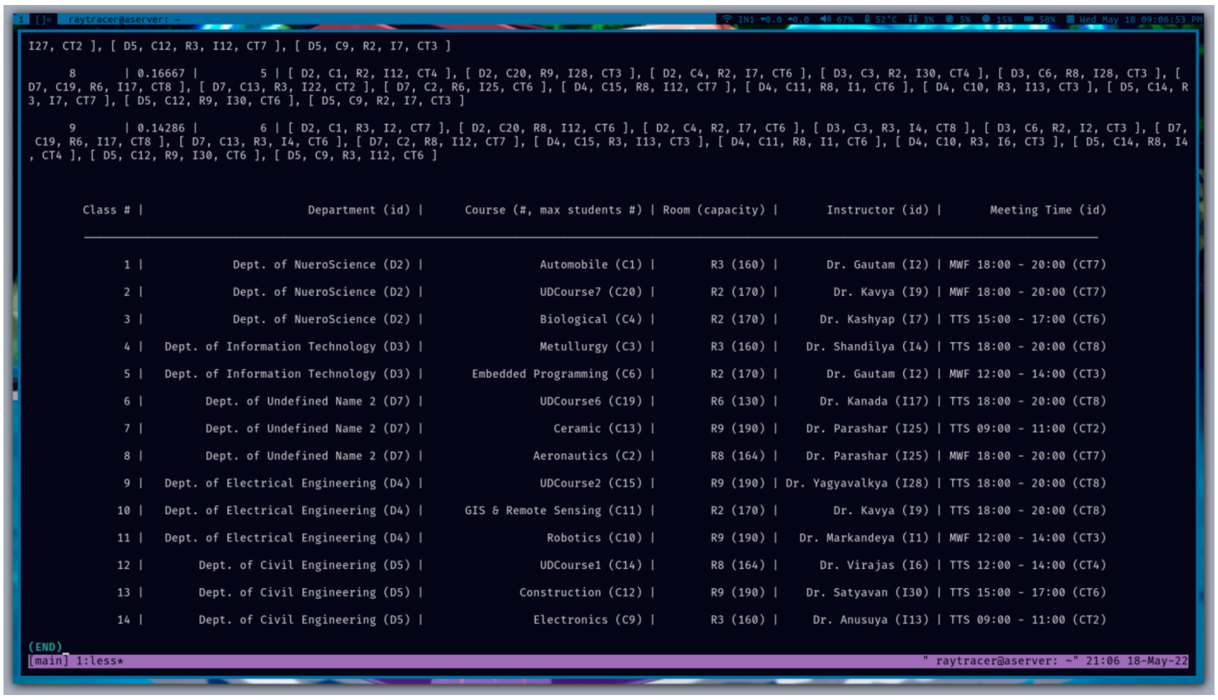
```
1  []=  raytracer@aserver: ~          IN1 0.0 0.0  67%  50°C  4%  5%  15%  59%  Wed May 18 09:03:21 PM

                       __  _
                      / /_(_)__ ____ ___
                     / __/ / _ `/ -_) _ \
                     \__/_/\_, /\__/_//_/
                          /___/

             Time-Table generation using Genetic Algorithm


                ┌──────────────────────────────────────┐
                │ Select Instructor for Course[C11]    │
                ├──────────────────────────────────────┤
                │   I1  Dr. Markandeya                 │
                │ → I2  Dr. Gautam                     │
                │   I3  Dr. Bharadwaj                  │
                │   I4  Dr. Shandilya                  │
                │ → I5  Dr. Angiras                    │
                │   I6  Dr. Virajas                    │
                │   I7  Dr. Kashyap                    │
                │ → I8  Dr. Satya                      │
                │   I9  Dr. Kavya                      │
                │   I10 Dr. Garg                       │
                │   I11 Dr. Kapil                      │
                │   I12 Dr. Kanad                      │
                │   I13 Dr. Anusuya                    │
                │   I14 Dr. Agastya                    │
                │   I15 Dr. Vajashrava                 │
                │   I16 Dr. Vachaspati                 │
                │   I17 Dr. Kanada                     │
                └──────────────────────────────────────┘

  You selected:  I2 I5 I8

  <space> to select, arrow keys to move, <enter> to confirm selection, 'q' to move to next type['q' again to exit]
[main] 1:./apps/tigen*                                              " raytracer@aserver: ~" 21:03 18-May-22
```

Figure 8.3: Instruction Selection



```
1  []=  raytracer@aserver: ~          IN1 0.0 0.0  67%  50°C  3%  5%  15%  59%  Wed May 18 09:01:05 PM

                       __  _
                      / /_(_)__ ____ ___
                     / __/ / _ `/ -_) _ \
                     \__/_/\_, /\__/_//_/
                          /___/

             Time-Table generation using Genetic Algorithm


                ┌──────────────────────────────────┐
                │ Select Course for Department[D1] │
                ├──────────────────────────────────┤
                │ → C1  Automobile                 │
                │   C2  Aeronautics                │
                │ → C3  Metullurgy                 │
                │   C4  Biological                 │
                │   C5  Chemical                   │
                │   C6  Embedded Programming       │
                │ → C7  Web Development            │
                │   C8  Networking Security        │
                │ → C9  Electronics                │
                │   C10 Robotics                   │
                │   C11 GIS & Remote Sensing       │
                │   C12 Construction               │
                │ → C13 Ceramic                    │
                │   C14 UDCourse1                  │
                │   C15 UDCourse2                  │
                │   C16 UDCourse3                  │
                │   C17 UDCourse4                  │
                └──────────────────────────────────┘

  <space> to select, arrow keys to move, <enter> to confirm selection, 'q' to move to next type['q' again to exit]
[main] 1:./apps/tigen*                                              " raytracer@aserver: ~" 21:01 18-May-22
```

Figure 8.4: Course Selection



```
1  []=  raytracer@aserver: ~                                    IN1 ▾0.0 ▴0.0  67%  52°C  3%  5%  15%  58%  Wed May 18 09:06:53 PM
I27, CT2 ], [ D5, C12, R3, I12, CT7 ], [ D5, C9, R2, I7, CT3 ]

      8       | 0.16667 |         5 | [ D2, C1, R2, I12, CT4 ], [ D2, C20, R9, I28, CT3 ], [ D2, C4, R2, I7, CT6 ], [ D3, C3, R2, I30, CT4 ], [ D3, C6, R8, I28, CT3 ], [
D7, C19, R6, I17, CT8 ], [ D7, C13, R3, I22, CT2 ], [ D7, C2, R6, I25, CT6 ], [ D4, C15, R8, I12, CT7 ], [ D4, C11, R8, I1, CT6 ], [ D4, C10, R3, I13, CT3 ], [ D5, C14, R
3, I7, CT7 ], [ D5, C12, R9, I30, CT6 ], [ D5, C9, R2, I7, CT3 ]

      9       | 0.14286 |         6 | [ D2, C1, R3, I2, CT7 ], [ D2, C20, R8, I12, CT6 ], [ D2, C4, R2, I7, CT6 ], [ D3, C3, R3, I4, CT8 ], [ D3, C6, R2, I2, CT3 ], [ D7,
 C19, R6, I17, CT8 ], [ D7, C13, R3, I4, CT6 ], [ D7, C2, R8, I12, CT7 ], [ D4, C15, R3, I13, CT3 ], [ D4, C11, R8, I1, CT6 ], [ D4, C10, R3, I6, CT3 ], [ D5, C14, R8, I4
, CT4 ], [ D5, C12, R9, I30, CT6 ], [ D5, C9, R3, I12, CT6 ]


        Class # |                    Department (id) |     Course (#, max students #) | Room (capacity) |       Instructor (id) |       Meeting Time (id)
        ____________________________________________________________________________________________________________________________________________________

              1 |           Dept. of NueroScience (D2) |                Automobile (C1) |        R3 (160) |       Dr. Gautam (I2) | MWF 18:00 - 20:00 (CT7)

              2 |           Dept. of NueroScience (D2) |                UDCourse7 (C20) |        R2 (170) |        Dr. Kavya (I9) | MWF 18:00 - 20:00 (CT7)

              3 |           Dept. of NueroScience (D2) |                Biological (C4) |        R2 (170) |      Dr. Kashyap (I7) | TTS 15:00 - 17:00 (CT6)

              4 |   Dept. of Information Technology (D3) |                Metullurgy (C3) |        R3 (160) |    Dr. Shandilya (I4) | TTS 18:00 - 20:00 (CT8)

              5 |   Dept. of Information Technology (D3) |      Embedded Programming (C6) |        R2 (170) |       Dr. Gautam (I2) | MWF 12:00 - 14:00 (CT3)

              6 |         Dept. of Undefined Name 2 (D7) |                UDCourse6 (C19) |        R6 (130) |      Dr. Kanada (I17) | TTS 18:00 - 20:00 (CT8)

              7 |         Dept. of Undefined Name 2 (D7) |                  Ceramic (C13) |        R9 (190) |    Dr. Parashar (I25) | TTS 09:00 - 11:00 (CT2)

              8 |         Dept. of Undefined Name 2 (D7) |               Aeronautics (C2) |        R8 (164) |    Dr. Parashar (I25) | MWF 18:00 - 20:00 (CT7)

              9 |   Dept. of Electrical Engineering (D4) |                UDCourse2 (C15) |        R9 (190) | Dr. Yagyavalkya (I28) | TTS 18:00 - 20:00 (CT8)

             10 |   Dept. of Electrical Engineering (D4) |     GIS & Remote Sensing (C11) |        R2 (170) |        Dr. Kavya (I9) | TTS 18:00 - 20:00 (CT8)

             11 |   Dept. of Electrical Engineering (D4) |                 Robotics (C10) |        R9 (190) |   Dr. Markandeya (I1) | MWF 12:00 - 14:00 (CT3)

             12 |        Dept. of Civil Engineering (D5) |                UDCourse1 (C14) |        R8 (164) |      Dr. Virajas (I6) | TTS 12:00 - 14:00 (CT4)

             13 |        Dept. of Civil Engineering (D5) |             Construction (C12) |        R9 (190) |    Dr. Satyavan (I30) | TTS 15:00 - 17:00 (CT6)

             14 |        Dept. of Civil Engineering (D5) |               Electronics (C9) |        R3 (160) |     Dr. Anusuya (I13) | TTS 09:00 - 11:00 (CT2)

(END)
[main] 1:less*                                                                                         " raytracer@aserver: ~" 21:06 18-May-22
```

Figure 8.5: Terminal user Interface result

Chapter 9

Conclusion and Future Work

Following conclusions have been drawn so far from current state of our project :

Timetable Scheduling is the very core of the working of any department or organization to perform well. **TIGEN** is the project which is going to provide a great help and benefit regarding this task. It will help in doing this important as an automated process with very minimal manual interaction and the user can just sit back and relax or can do some other important task with the time he just saved. So, in this era of increasing technology and automation, TIGEN is another automation tool and helper for human so that they can leave this important task to the project and just focus on other important aspect of their jobs and lives.

Future Work: Tigen can be extended to have hardware interaction in order to reduce manpower needed to use tigen. It can also be extended to provided more constraints functionality to the user such that it can provide more needed results to the user.Efficient data structures can be uses to provide faster results.

Appendix A

Publication

The part of the project is published in *International Journal for Scientific Research and Development(IJSRD)*.

1. T. Singh, P.srivastava, A.S. Pathak, U.K. Pandey and M. singh "TIGEN: Genetic Algorithm for timetable generation," in 2022 International Journal for Scientific Research and Development(IJSRD)

Appendix B

Software Developement Life Cycle Models

Types of SDLC models

- WATERFALL MODEL:

  The Waterfall Model was the principal Process Model to be presented. It is likewise alluded to as a direct successive life cycle model. It is extremely easy to comprehend and utilize. In a Waterfall model, each stage should be finished before the following stage can start and there is no covering in the stages.

- SPIRAL MODEL:

  The spiral model joins the possibility of iterative improvement with the orderly, controlled parts of the cascade model. This Spiral model is a mix of iterative advancement process model and consecutive direct improvement model for example the cascade model with an extremely high accentuation on risk examination. It permits gradual arrivals of the item or steady refinement through every cycle around the spiral.

  Spiral Model - Design

  The spiral model has four stages. A product project over and again goes through these stages in emphasis called Spirals.

- RAD MODEL:

  The RAD (Rapid Application Development) model depends on prototyping and iterative advancement with no particular arranging included. The most common way of composing the actual product includes the arranging expected for fostering the item.

Rapid Application Development centers around get-together client prerequisites through studios or center gatherings, early testing of the models by the client utilizing iterative idea, reuse of the current models (parts), nonstop incorporation and fast delivery.

What is RAD?

Quick application advancement is a product improvement philosophy that involves insignificant preparation for fast prototyping. A model is a functioning model that is practically identical to a part of the product.

In the RAD model, the useful modules are created in lined up as models and are incorporated to make the total item for quicker item conveyance. Since there is no nitty-gritty preplanning, it makes it more straightforward to consolidate the progressions inside the advancement process.

Bibliography

[1] Mrs.G.Maneesha, T.Deepika, S.BhanuSri, N.RaviKumar, and P.SivaNagama, “”time table generation using genetic algorithm,” *JETIRJournal of Emerging Technologies and Innovative Research*, vol. 8, no. 7, 2021.

[2] D. Mittal, H. Doshi, M. Sunasra, and R. Nagpure, “Automatic timetable generation using genetic algorithm,” *IJARCInterna- tional Journal of Advanced Research in Computer and Communication Engineering*, vol. 4, no. 2, 2015.

[3] S. Thakare, T. Nikam, and P. M. Patil, “Automated timetable generation using genetic algorithm,” in *International Journal of Engineering Research Technology (IJERT)*, vol. 9, 2020.

[4] M. K. Kakkar, J. Singla, N. Garg, G. Gupta, P. Srivastava, and A. Kumar, “Class schedule generation using evolutionary algorithms,” *ICMAIInstitute of Cost Accountants of India*, vol. 31, no. 5, 2018.

[5] pranav khurana, *Time-table-scheduler*. Github, 2017.

[6] S. Shinde, S. Gurav, and S. karme, “Automatic timetable generation using genetic algorithm,” *IJSERInternational Journal of Scientific Engineering Research*, vol. 9, no. 4, 2018.

[7] “Tigen.” `https://github.com/thkrts/tigen`. [Online; accessed 23-May-2022].

[8] Schaerf , A. (1999) A Survey of Automated Timetabling. Artificial Intelligence, 13, 87-127.

[9] Kenekayoro , P. (2020). Incorporating Machine Learning to Evaluate Solutions to the University Course Timetabling Problem. arXiv preprint arXiv:2010.00826.