TRIBHUVAN UNIVERSITY

Prime College

Nayabazar, Kathmandu, Nepal

A Project Report

On

ROUTINE SCHEDULER

Submitted By

Anjal Pandey (20470/075)

Shirish Mainali (20494/075)

Suraj Sedhain (20502/075)

A Project Report Submitted in partial fulfillment of the requirement of **Bachelor of Science in Computer Science & Information Technology (BSc.CSIT) 7th Semester** of Tribhuvan University, Nepal

March, 2023

ROUTINE SCHEDULER

CSC 412

A project report submitted for the partial fulfillment of the requirement for the degree of Bachelor of Science in Computer science & Information Technology awarded by Tribhuvan University.

Submitted By

Anjal Pandey (20470/075)

Shirish Mainali (20494/075)

Suraj Sedhain (20502/075)

Submitted To

Prime College

Department of Computer science

Affiliated to Tribhuvan University

Khusibun, Nayabazar, Kathmandu

March, 2023

Date: ……

SUPERVISOR'S RECOMMENDATION

It is my pleasure to recommend that a report on "**ROUTINE SCHEDULER"** has been prepared under my supervision by **Anjal Pandey, Shirish Mainali and Suraj Sedhain** in partial fulfillment of the requirement of the degree of Bachelor of Science in Computer Science and Information Technology (BSc.CSIT). Their report is satisfactory to process for the future evaluation.

……

Mr. Sudan Prajapati

Supervisor

Department of Computer Science & IT

Prime College

Date: ……

CERTIFICATE OF APPROVAL

The undersigned certify that he has read and recommended to the Department of Computer Science and Information Technology for acceptance of report entitled "**ROUTINE SCHEDULER**" submitted by **Anjal Pandey, Shirish Mainali and Suraj Sedhain** in partial fulfillment for the degree of Bachelor of Science in Computer Science and Information Technology (BSc.CSIT), Institute of Science and Technology, Tribhuvan University.

…… **…….**

Mr. Narayan Prasad Sharma
Principal

Ms. Rolisha Sthapit
Program Coordinator

…… **……**

Mr. Sudan Prajapati
Supervisor

Mr. Jagdish Bhatta
External Examiner

ACKNOWLEDGEMENT

We owe our deepest gratitude to Prime College for allowing us an opportunity to work on this project as part of our syllabus. We are heartily indebted to our venture supervisor **Mr. Sudan Prajapati** for his constant guide and guidance for the duration of this challenge. It became his precious hints that helped us manage the emerging barriers through the development of this project.

We are also thankful to all our instructors for tips and inspirational lectures that paved the way in the direction of the entirety of this project. Additionally, we would like to thank our pals and colleagues for their valuable feedback and tips for the duration of this mission. Any kind of suggestion for criticism might be pretty preferred and acknowledged.

With respect,

Anjal Pandey (20470/075)

Shirish Mainali (20494/075)

Suraj Sedhain (20502/075)

ABSTRACT

The Routine Scheduler project aims to help full-time teachers create schedules for their daily activities. This involves dealing with various resource constraints such as faculty availability, room allocation, and time slot availability. To address these challenges, the system uses a combination of Genetic Algorithm and Active Rules to create an optimized solution. The system is designed to meet various constraints, but it does not account for lecturer unavailability, small room sizes, and travel time between classes. The Routine Scheduler generates schedules on a semester and teacher basis.

KEYWORDS: *Routine, Genetic Algorithm, Schedule, Course*

TABLE OF CONTENTS

ROUTINE SCHEDULER ii

SUPERVISOR'S RECOMMENDATION iii

CERTIFICATE OF APPROVAL iv

ACKNOWLEDGEMENT v

ABSTRACT vi

TABLE OF CONTENTS vii

LIST OF ABBREVIATIONS ix

LIST OF FIGURES x

LIST OF TABLES xi

CHAPTER 1 INTRODUCTION 1

1.1 Introduction 1

1.2 Problem Statement 2

1.3 Objectives 2

1.4 Scope and Limitation 2

1.5 Development Methodology 3

1.6 Report Organization 4

CHAPTER 2 BACKGROUND STUDY AND LITERATURE REVIEW 5

2.1 Background 5

2.2 Literature Review 5

CHAPTER 3 SYSTEM ANALYSIS 7

3.1 System Analysis 7

3.1.1 Requirement Analysis 7

3.1.2. Feasibility Study 11

3.1.3 Analysis 14

CHAPTER 4 SYSTEM DESIGN 17

4.1. Design 17
4.1.1. Refinement of class sequence and activity diagram 18
4.1.2. Component Diagram 21
4.1.3. Deployment Diagram 22
4.2. Algorithm Details 23
CHAPTER 5 IMPLEMENTATION AND TESTING 24
5.1. Implementation Overview 25
5.1.1. Tools Used 25
5.1.2. Modules Description 26
5.2 Testing 46
5.2.1 Unit Testing 46
5.2.2 System Testing 51
5.3. Result Analysis 55
CHAPTER 6 CONCLUSION AND FUTURE RECOMMENDATIONS 56
6.1. Conclusion 57
6.2. Future Recommendations 57
APPENDICES 60

LIST OF ABBREVIATIONS

CSS Cascading Style Sheet

GA Genetic Algorithm

HTML Hypertext Markup Language

IDE Interactive Development Environment

SQL Structured Query Language

UI User Interface

UML Unified Modeling Language

LIST OF FIGURES

Figure 1.1 Incremental Delivery Model....4
Figure 3.1 Use Case Diagram....9
Figure 3.2 Use case for register....10
Figure 3.3 Use case for Login....10
Figure 3.4 Gantt chart....13
Figure 3.5 Class Diagram....14
Figure 3.6 Sequence Diagram....15
Figure 3.7 Activity Diagram....16
Figure 4.1 System design of routine scheduler....17
Figure 4.2 Refined Class Diagram....18
Figure 4.3 Refined Sequence Diagram....19
Figure 4.4 Refined Activity Diagram....20
Figure 4.5 Component Diagram of Routine Scheduler....21
Figure 4.6 Deployment Diagram of Routine Scheduler....22
Figure 4.7 Working mechanism of genetic algorithm....23
Figure 5.1 Login Module....26
Figure 5.2 Registration Module....27
Figure 5.3 Setup Model....28
Figure 5.4 Routine Generation Model....29
Figure 5.5 Account Controller....30
Figure 5.6 Dashboard Controller....31
Figure 5.7 Course Controller....31
Figure 5.8 Lecturer Controller....32
Figure 5.9 Lecturer Subject Controller....33
Figure 5.10 Time Table Generation Controller....33
Figure 5.11 Semester Wise Controller....34
Figure 5.12 Teacher Wise Controller....34
Figure 5.13 Repository....35
Figure 5.14 Test Case for Register....48
Figure 5.15 Test Case for Login....50
Figure 5.16 System Testing....55

LIST OF TABLES

Table 3.1 Project Schedule 12
Table 5.1 Test case for Register 47
Table 5.2 Test case for Login 49
Table 5.3 Test case for system testing 53
Table 5.4 Test Cases for Routing Generate 54

CHAPTER 1
INTRODUCTION

1.1 Introduction

The class timetabling problem is a scheduling challenge that occurs in every academic institution and involves creating a timetable that accurately allocates classes to various time slots, rooms, and faculty members. In the past, this process was typically done manually, involving a single person or group of people who would spend a significant amount of time and effort on the task. Timetabling is a complex and error-prone application due to the numerous constraints and variables involved, including limited resources such as classrooms and faculty availability, scheduling conflicts, and the need to create a balanced and efficient schedule. Creating a timetable requires careful consideration of all these factors to ensure that classes are scheduled in a way that maximizes the use of available resources while minimizing conflicts. Manual timetabling can be time-consuming, prone to errors, and difficult to modify when changes occur. Therefore, the use of automated systems like the Routine Scheduler project is made to streamline the timetabling process and reduce the workload of administrators and faculty members. By automating the process, institutions can save time, increase accuracy, and create more efficient schedules that meet the needs of faculty.

The purpose of the Routine Scheduler project is to create a schedule for the routine activities of various faculty teachers. The main focus of the project is to create a schedule for routine activities based on the information provided, including the semester, room number, subject, teacher, break time, and lab requirements. The project aims to incorporate these details into the scheduling process to ensure that all necessary information is considered when creating the routine schedule. The project aims to create a platform that is easy for the User to use, which will enable them to register courses, subjects and teachers. Additionally, the platform will allow them to generate routine based on different parameters such as day wise, semester wise and teachers wise.

The Routine Scheduler is a web-based application that enables users to generate a routine by scheduling routine activities. Once the necessary data is provided, the user can use the application to generate a routine. The application is compatible with different platforms.

1.2 Problem Statement

The Routine Scheduler has been created to address the difficulties of scheduling conflicts and the availability of faculty members. Scheduling a routine can be a complex and time-consuming task, especially if done manually. There are many parameters to consider, such as course, classroom availability, instructor schedules. Additionally, entering all of this data manually can lead to errors, which can be difficult and time-consuming to correct.

The Routine Scheduler has been implemented to solve the issues related to scheduling conflicts and limited resources such as classrooms, courses, time slots, and availability of faculty members. It aims to provide an effective solution by creating a routine that takes into account the availability of these resources.

1.3 Objectives

This project aims to meet the following objectives:

- To generate routine, the platform allows for generating it semester wise.
- To generate routine, the platform allows for generating it teacher wise.

1.4 Scope and Limitation

Scope:

A routine scheduler web app can automatically generate class schedules based on a variety of parameters, such as course, Teacher availability, and classroom availability. This can save time and reduce errors compared to manually creating schedules.

Limitation:

- Data input errors: If the input data for course, teacher availability, and classroom availability is inaccurate or incomplete, the app may generate schedules that are not optimal or require significant modifications.
- Lack of flexibility: Routine scheduler web apps may not offer enough flexibility to account for unexpected such as room closures, instructor absences, or student requests.

1.5 Development Methodology

Routine Scheduler has adopted the Incremental Delivery Model, which involves delivering the project in stages, with each stage building on the previous one. By doing so, system can provide regular working versions of the project and enabling us to make adjustments and improvements throughout the development process.

The incremental delivery approach is perfect for complicated projects or projects where the requirements are unclear or may change over time. By dividing the project into smaller parts, the development process becomes more manageable and adaptable. Additionally, the approach enables the project team to receive feedback and make changes throughout the process, rather than waiting until the end to assess the final product. This helps to ensure that the project is completed on time. For the routine scheduler project the first iteration focuses on building the database as database first approach is used to construct this application. Then similarly subsequent iteration are used to make other features and before going into the next iterations each iteration was tested to make sure that the system is reliable and useable throughout the development process.

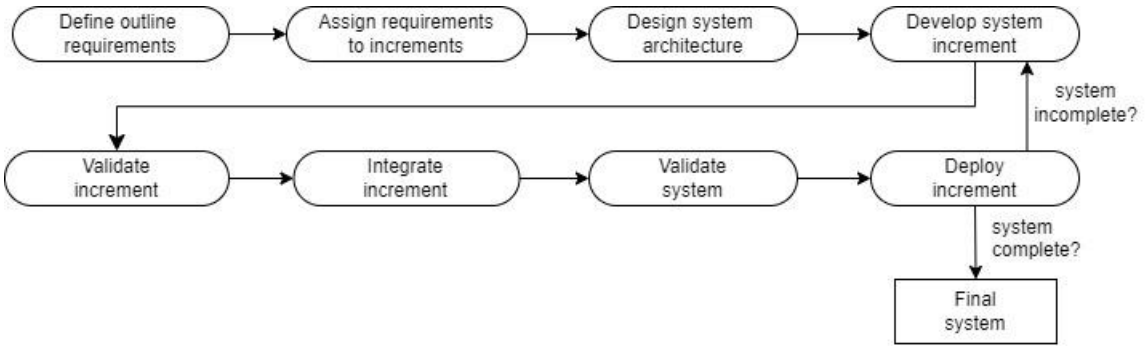


Figure 1.1 Incremental Delivery Model

1.6 Report Organization

Report on "Routine Scheduler" is comprised on six chapter.

Chapter 1 Provides the introduction regarding the project along with its development methodology, scopes and limitations.

Chapter 2 describes background study of the present system available related to the time table management and briefing on the automatic generation of timetable using genetic algorithm.

Chapter 3 provides an overview of all the functional and nonfunctional requirements binding the operation of the system along with analysis of the feasibility study of the system.

Chapter 4 portrays the diagram representing UML which describes the core design of the system alongside outline of the algorithms used to develop the system.

Chapter 5 contains a detailed description of the implementation phase and model phases of different modules. Moreover, this section also defines the testing processes providing an analytical result

Chapter 6 provides summary of the system that has been developed and future possibility for the system to make it more sustainable, reliable and efficient.

CHAPTER 2
BACKGROUND STUDY AND LITERATURE REVIEW

2.1 Background

The current process of creating routines is done manually by teachers, coordinators, or management using an Excel sheet. Unfortunately, this method is encountering various issues such as conflicting schedules, classroom availability, and faculty member availability. As a result, it becomes challenging to manage the routine, leading to an inefficient process. This, in turn, makes it difficult for students to determine their classes. This can result in additional work for both the faculty and students, who may need to be notified of changes.

Using a routine scheduler can be a solution to this problem. Once the data is inserted into the system, the scheduler automatically generate a routine that takes into account all the necessary variables, such as teacher availability, classroom availability, course availability. This eliminates the need for manual input and reduces the likelihood of errors.

By using a routine scheduler, institutions can save time and resources while ensuring that schedules are accurate and up-to-date. This can help to improve overall efficiency and productivity, and can result in a better experience for both faculty and students.

2.2 Literature Review

Early research in the field of routine schedule relied on manual entering of data in excel sheet which required a lot of time and possibility of errors. However the emergence of software through which user can enter correct data and generate routine on one click .Here simply using the resource available in the website user can create a weekly routine by entering data in the website and if users want new schedule then the old schedule is replaced by new one. [1]

This timetable tool will serve as a replacement for your paper timetable and schedule. You can input all of your classes, tutorials, and practical courses, including the classroom

number, teacher's name, time, duration, frequency, and color code. This way, you can easily view your student timetable and access a summary of all your classes. It is a practical and efficient tool that ensures you always have your class schedule with you. You can easily make modifications as needed [2]

With more than six different calendar views, including monthly, weekly, and daily agendas, as well as views of teacher and location availability, finding available spaces is fast and simple. The calendar also allows you to see a teacher's availability, so you can quickly find open spots. Teachers can manage their availability from their personal accounts on Teach works, ensuring their schedules are always up-to-date. When scheduling a new lesson, you can easily check for conflicts by viewing a teacher's lesson and availability conflicts and location conflicts, which will be displayed with detailed information including specific lesson details, availability, and times [3].

As discussed, an evolutionary algorithm, genetics algorithm for time tabling has been proposed. The intention of the algorithm to generate a time-table schedule automatically is satisfied. The algorithm incorporates a number of techniques, aimed to improve the efficiency of the search operation. By automating this process with the help of computer assistance timetable generator can save a lot of precious time of administrators who are involved in creating and managing various timetables of the institutes [4].

The GA in timetabling framework has been shown to be successful on several real problems of 'University Department size. It has been shown that the genetic algorithm perform better in finding areas of interest even in a complex, real-world scene. This paper described how set of active rules can be used to express the knowledge of intelligent and how a genetic algorithm can be used to dynamically prioritize rules in the face of dynamically evolving environments. This paper illustrated the applicability of the above method by using it to optimize the performance. The advantages of this approach to optimizing the solution for time table are apparent: distributed solution, load balancing and fault situations. [5]

CHAPTER 3
SYSTEM ANALYSIS

3.1 System Analysis

Systems analysis is indeed a process of examining the components of a system or organization, how they interact, and how they can be improved. It involves looking at the system as a whole, identifying its strengths and weaknesses, and proposing solutions for improvement.

3.1.1 Requirement Analysis

The purpose of the Requirements Analysis is to transform the needs and high-level requirements specified in earlier phases into unambiguous (measurable and testable), traceable, complete, consistent requirements.

Web Application

Software Requirements

- IDE
    - Visual Studio 2022
- Database
    - Sql Server
- Languages
    - C#
- Web browser:
    - Google Chrome
    - Internet Explorer
    - Brave
    - Microsoft Edge
    - Safari
    - Mozilla Firefox

Hardware Requirements

- Internet Connection
- Monitor, keyboard, Mouse

3.1.1.1. Functional Requirements

The functional requirement defines the system. It specifies what the system should do. Functional requirements of the system include the followings.

- Register by the user
- Login of user
- Add days
- Register course
- Register Semester
- Register Teacher
- Generate Routine

Use Case Diagram

Use case diagram is a graphical depiction of a user's possible interactions with a system. It is used to gather the requirements of a system including internal and external influences. In addition, the use case diagram can help to identify external and internal factors that influence the system's behavior. External factors can include user needs, external systems, or external regulations. Internal factors can include the system's architecture, constraints, or business rules.

By using a use case diagram, stakeholders can gain a better understanding of the system's requirements and dependencies, and can help to ensure that the system is designed to meet the needs of its users. It can also serve as a communication tool between developers, designers, and other stakeholders in the project, helping to ensure that everyone is on the same page regarding the system's functionality and requirements.

Figure 3.1 Use Case Diagram

The use case diagram describes a scenario where an admin user needs to register first to access a web application. The registration process involves saving the user's data into the system's database. Once registered, the user can log in to the web application using their validated credentials.

Upon successful login, the user is directed to a dashboard where they can input information about a school, such as the number of rooms, subject teachers, available labs, programs, semesters, courses, and time slots. The user can then generate a routine using the data they inputted.

Register

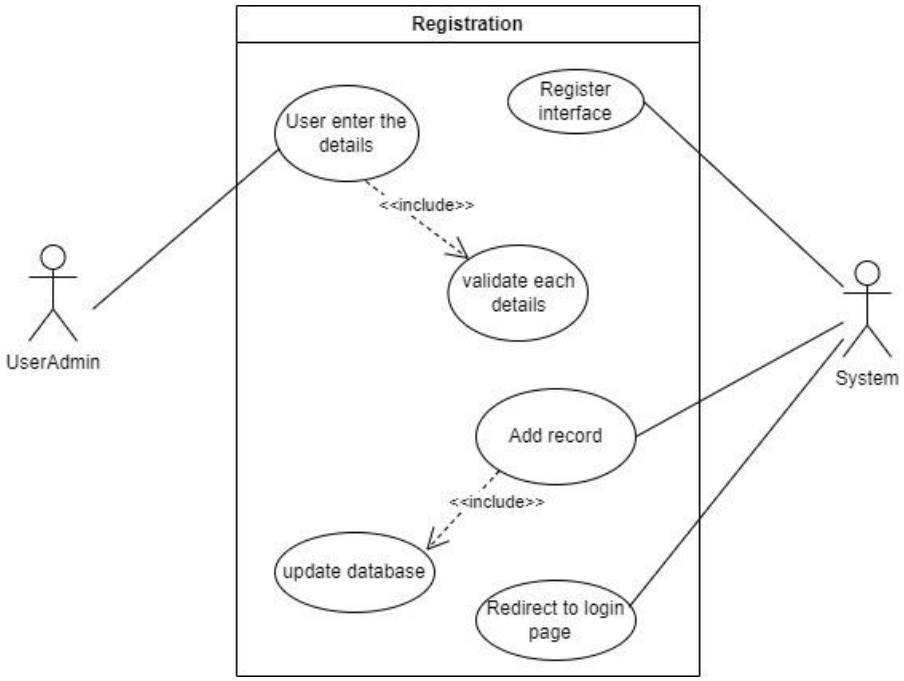


Figure 3.2 Use case for register

Login

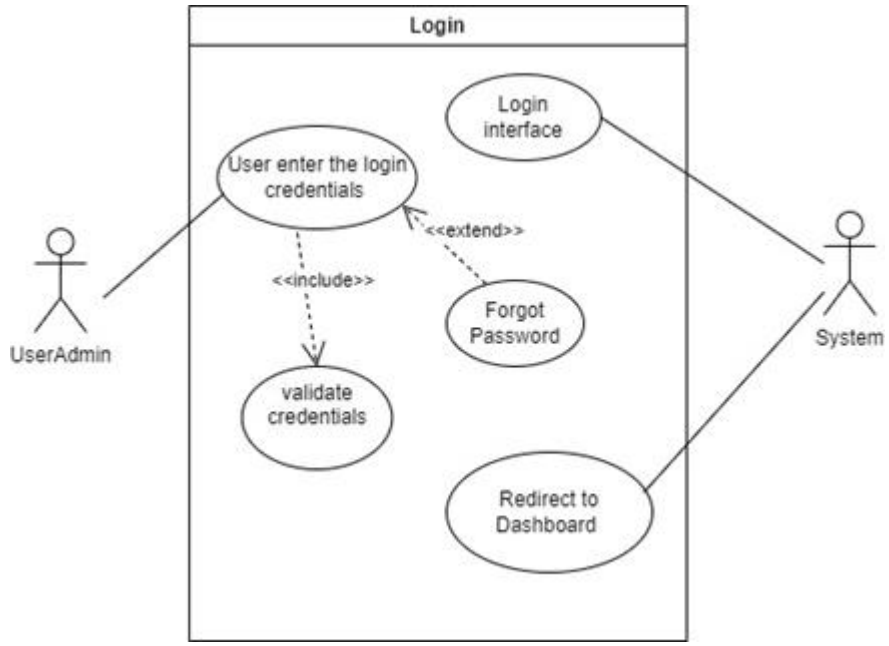


Figure 3.3 Use case for Login

3.1.1.2. Non - Functional Requirements

It specifies how the system performs. The non-functional requirements include the following:

- Performance: The system needs to work really well and respond quickly, without much time delay, especially when it comes to generating routines.
- Usability: The system should be easy to use and navigate, with a clear and intuitive interface that appeals to a wide range of users.
- Scalability: The system should be scalable and able to handle large volumes of data.
- Availability: The system should be highly available and able to handle high volumes of traffic without downtime or interruption, to ensure a smooth user experience.

3.1.2. Feasibility Study

The feasibility study is performed to determine whether the proposed system is viable considering the Technical, Operational and Economical factors. After going through feasibility study, system's benefits and drawbacks can be seen clearly..

3.1.2.1. Technical Feasibility

The system being developed using .NET and SQL is technically feasible, as both technologies are widely used and have been proven to be reliable and efficient in developing web-based applications. .NET is a robust and versatile framework that provides a wide range of features for developing scalable and secure applications. SQL, on the other hand, is a widely used database management system that provides efficient storage and retrieval of data. However, given the popularity and reliability of .NET and SQL, and the availability of suitable hosting environments, the development of the system is technically feasible.

3.1.2.2. Operational Feasibility

The system reduces the time required to create a routine, which results in increased speed and efficiency. The routine scheduling system is operationally feasible because it offers tangible benefits to users and can be easily integrated into existing processes and operations.

3.1.2.3. Economic Feasibility

The proposed system is economically feasible as the required hardware and software are readily available in the market at a low cost. The initial investment is the only expense, and there is no need for further upgrades or investments. This makes the system financially viable, and it encourages the adoption of the system design as it offers a good return on investment. Additionally, the system's feasibility in all aspects further supports its economic viability.

3.1.2.4. Schedule Feasibility

The below Gantt chart displays the overall timeline of the project, presenting a sequential breakdown of the tasks involved along with the time taken for each task. During the first two desk, requirements were gathered, and system design was carried out. After that, the system was implemented over a period of two months, and system documentation ran in parallel with all the processes. During this time, the testing phase was performed, and documentation was updated in accordance with the testing.

Table 3.1 Project Schedule

S.N.	Tasks	Start Date	End Date	Duration
1	Requirement Gathering	11/20/2022	11/29/2022	9
2	System Design	11/25/2022	12/5/2022	10
3	Coding	12/1/2022	1/10/2023	40
4	System Testing	1/11/2023	1/20/2023	9
5	Implementation	1/21/2023	1/30/2023	9
6	System Documentation	11/20/2023	2/10/2023	87

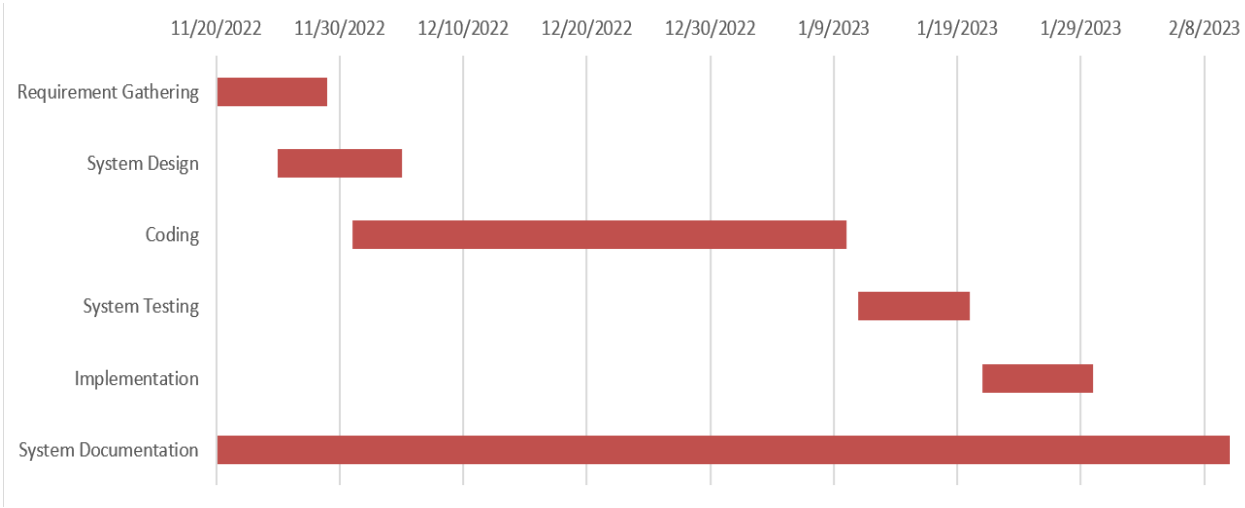


Figure 3.4 Gantt chart

The project schedule outlines the estimated time required to complete each stage of a software development project. The first stage, Requirement Gathering, involves gathering and analyzing the requirements for the software system, and the project schedule estimates that it will take 9 days to complete. The System Design stage comes next, and this is where the architecture of the software system is designed, including its components, interfaces, and data structures. The schedule estimates that this stage will take 10 days. After the System Design stage, the coding stage begins, where the developers will write the code that will form the basis of the software system. The project schedule estimates that the coding stage will take 40 days. Once the coding stage is complete, the software system will need to be tested to ensure that it functions correctly. The System Testing stage is expected to take 9 days. Following System Testing, the software system can be implemented, which includes activities such as user training and data migration, and the Implementation stage is estimated to take 9 days. Finally, the System Documentation stage is expected to take 87 days to complete. The project schedule provides an estimate of the time required for each stage of the software development project, which can serve as a roadmap for completing the project within the specified timeframe.

3.1.3 Analysis

3.1.3.1 Object Modeling using class diagram

Class diagram in the UML is a type of static structure diagram that describes the structure of a system, it shows system's classes along with their attributes, operations and relationships among objects

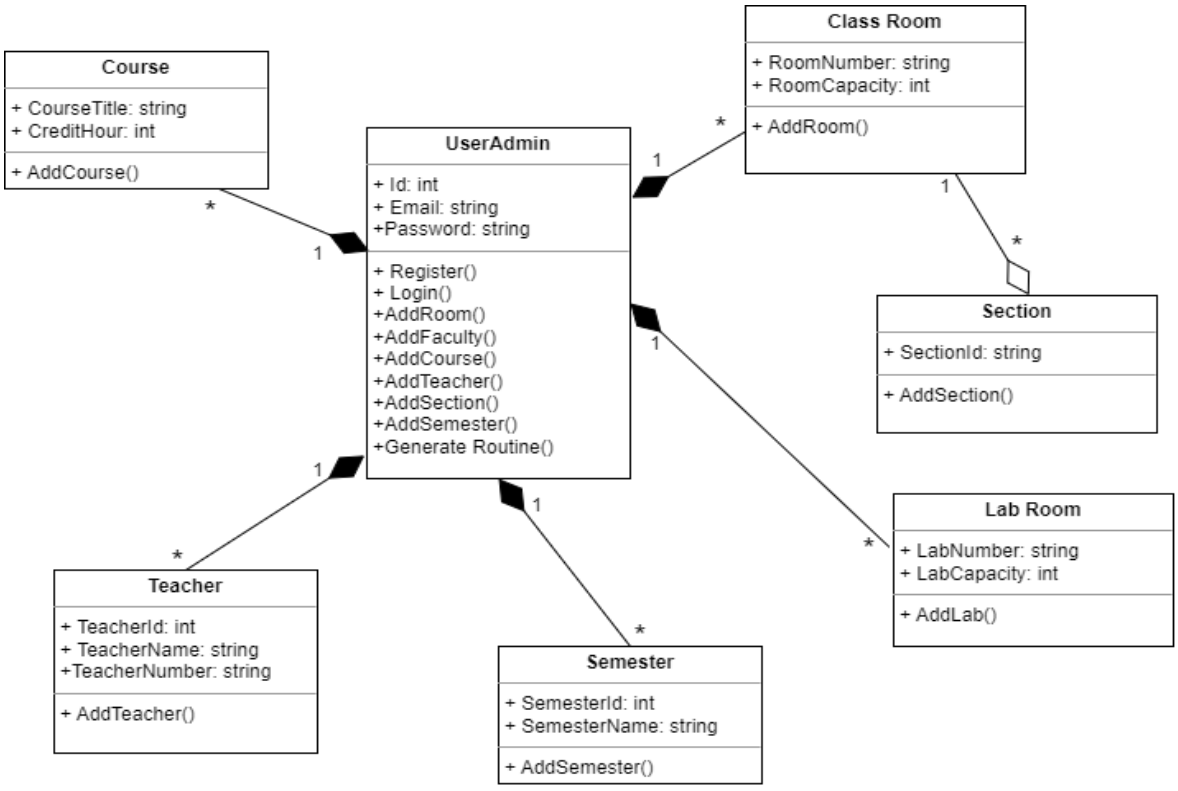


Figure 3.5 Class Diagram

The above class diagram represents relationships between classes, attributes, and operations. The system includes classes: Course, UserAdmin, Teacher, Semester, Class Room, Section and Lab Room. Various relationship among these classes can be seen in the diagram above.

3.1.3.2 Dynamic Modeling using sequence diagram

Sequence diagram in the UML is a type diagram that illustrates the sequence of messages between objects in an interaction. It consists of a group of objects represented by lifelines and message they exchange during the interaction denoted by arrow symbols.

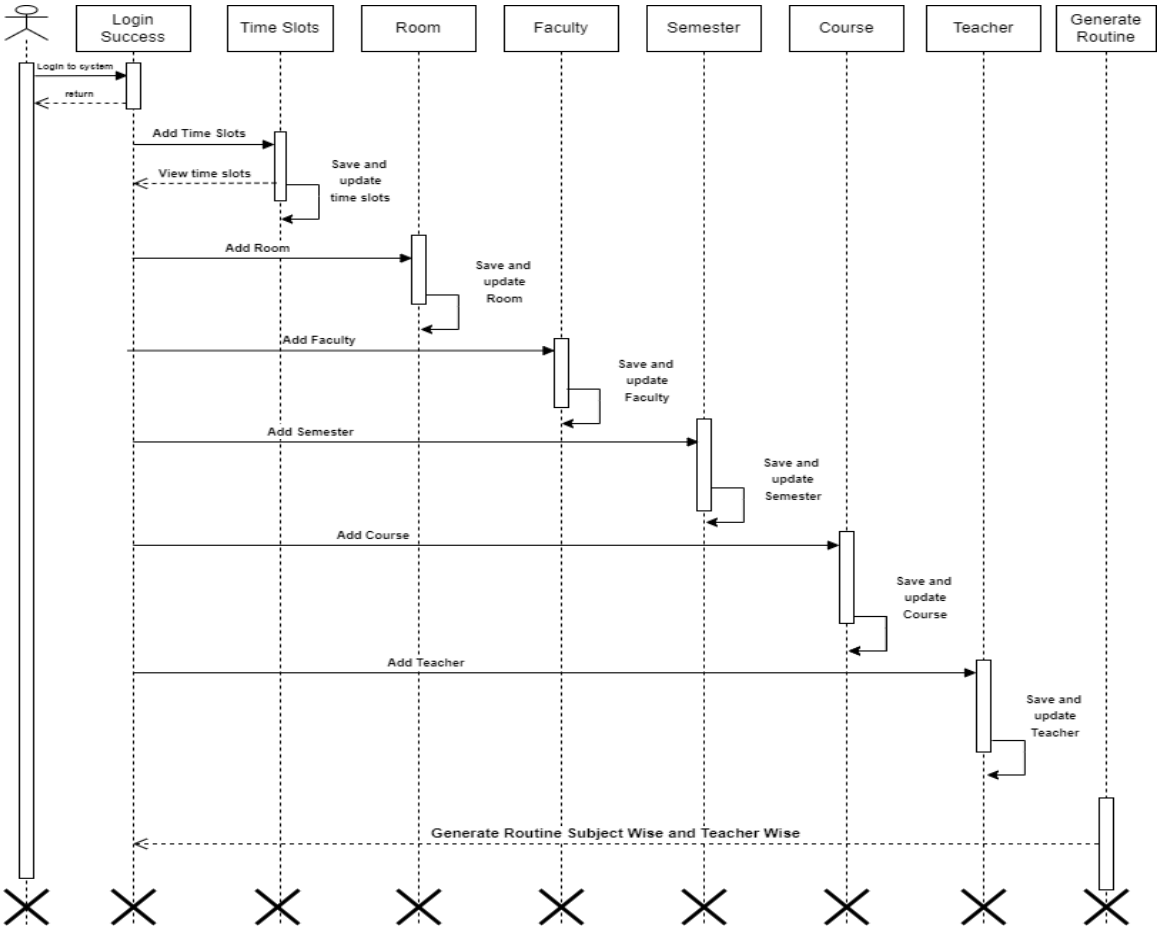


Figure 3.6 Sequence Diagram

The above diagram shows how the user interacts with the system. The first process in to login into the system and only the further activities can be carried out. The admin can add time slots, rooms, faculty, semester, course and teacher. According to the information fed into the system, the system generates routine subject wise and teacher wise.

3.1.3.3 Process modelling using Activity Diagram

Activity diagram in the UML is a type of diagram that visually presents a series of actions or flow of control in a system. It shows workflows of stepwise activities with support for choice, iteration and concurrency.

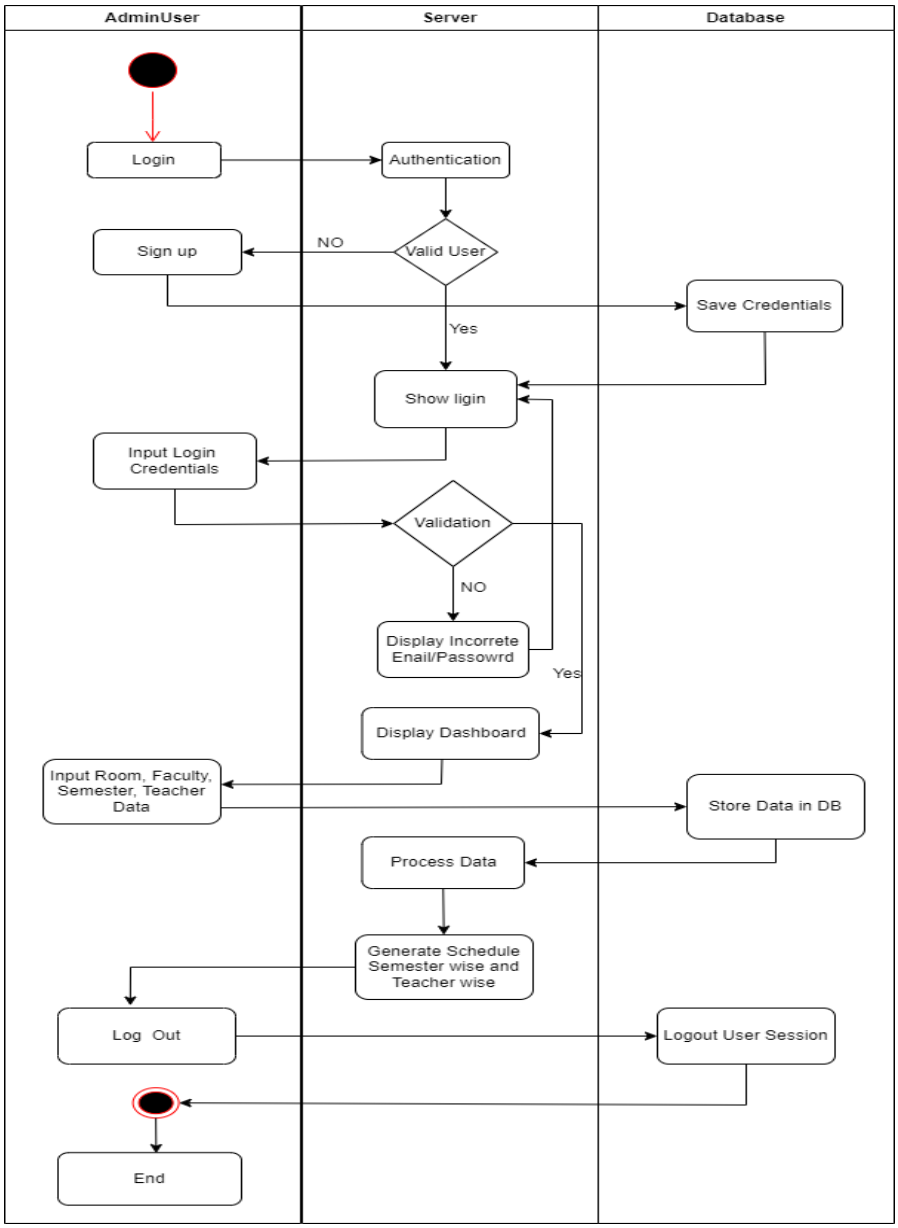


Figure 3.7 Activity Diagram

The above diagram shows how the control flows from user to server and database. The first activity that admin has to do is log in to the system then the admin can manage time slots, rooms, faculty, semester, course and teacher. The validation processes occurs in the server and the data are fetched and stored in the database.

CHAPTER 4
SYSTEM DESIGN

4.1. Design

System design is the process of representing architecture, interfaces, components that are included in the system. i.e., system design can be seen as the application of system theory to product development.

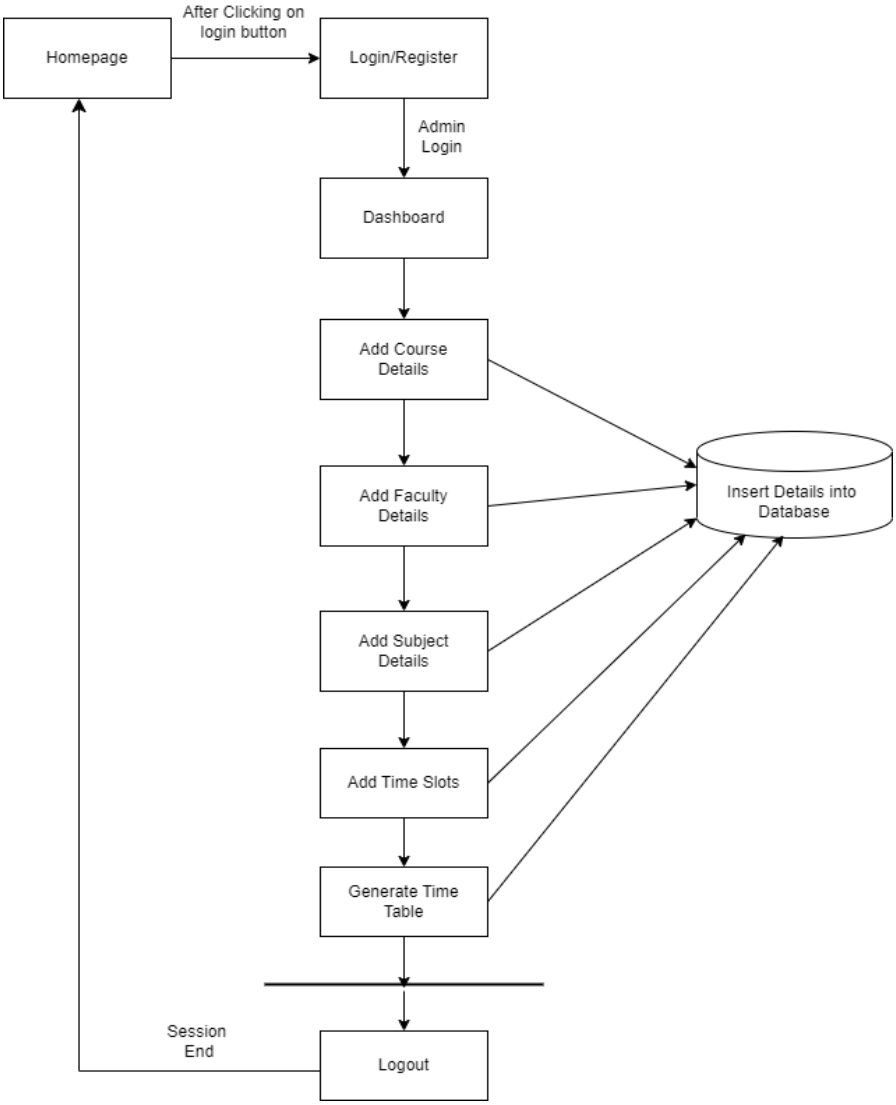


Figure 4.1 System design of routine scheduler

The system design for the routine generation system includes user authentication, database management, and an admin interface for managing data entities, a genetic algorithm that generates timetable on the basis of data entered by the admin, and testing and maintenance. The admin is responsible for inserting time slots, details of rooms, teacher and subjects. The timetable is generated randomly and unique every time.

4.1.1. Refinement of class sequence and activity diagram

The UML diagrams are now refined to shows more detail description of the system component which makes it easier to understand the overall working of the system. Refined UML diagrams include class diagram, sequence diagram and activity diagram of different system modules.

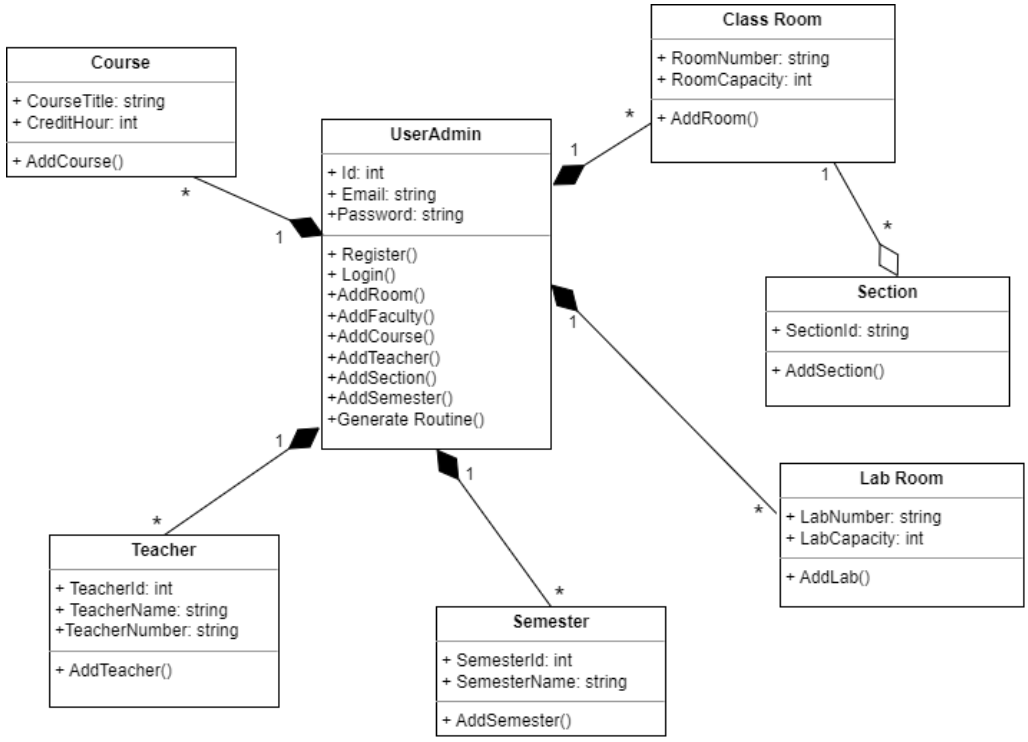


Figure 4.2 Refined Class Diagram

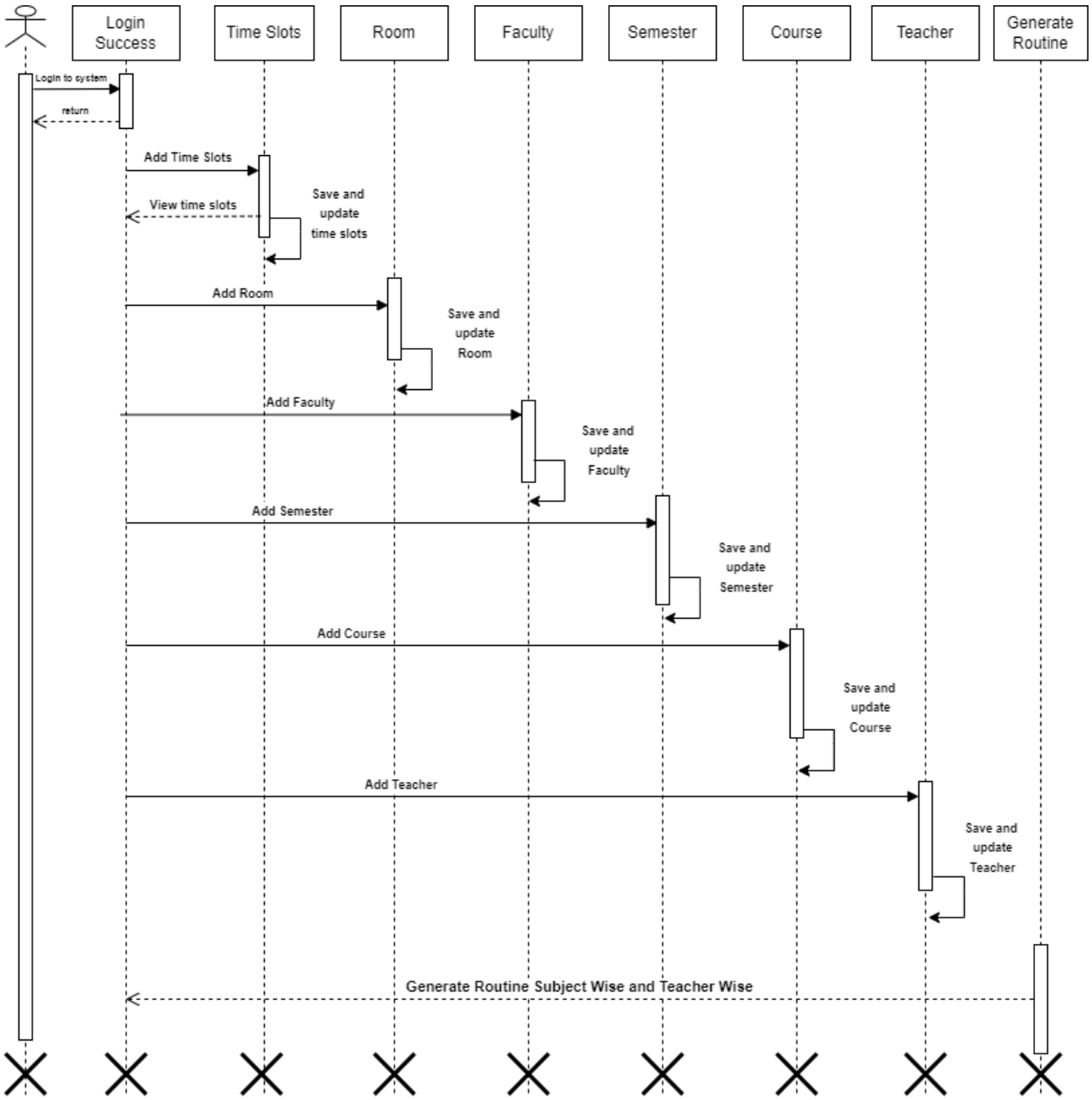


Figure 4.3 Refined Sequence Diagram

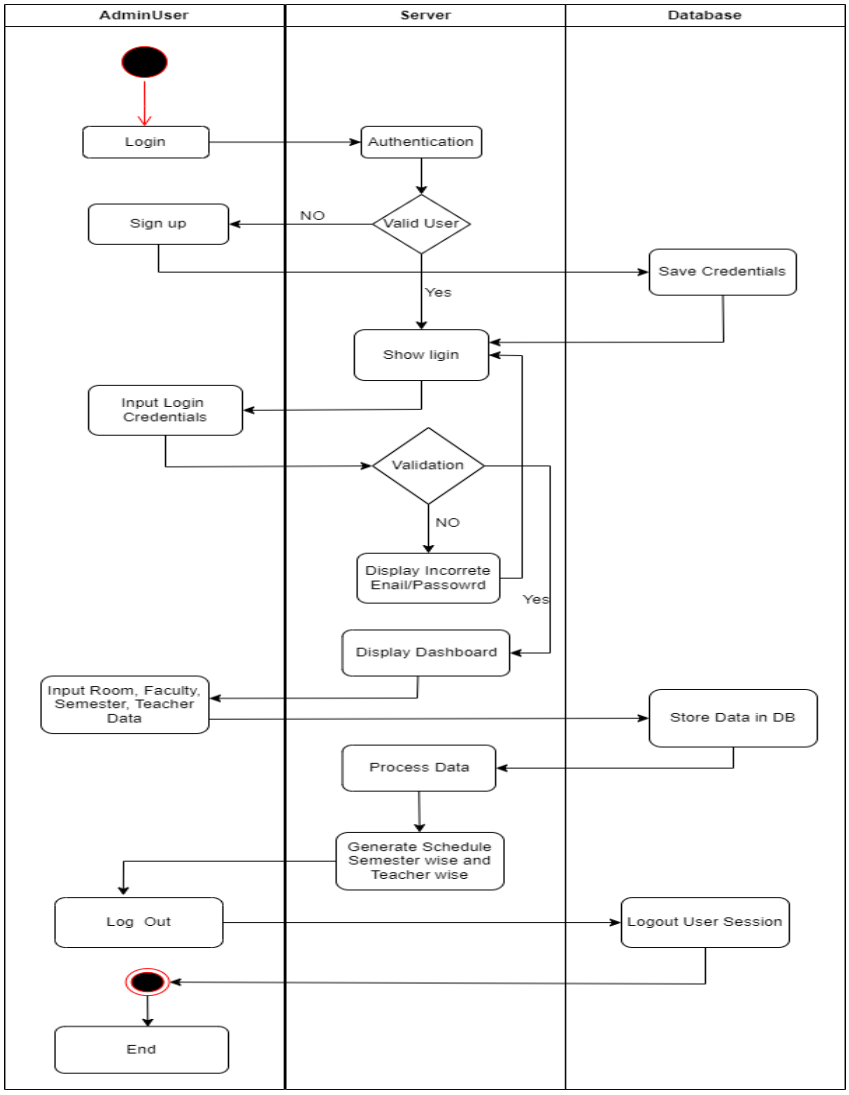


Figure 4.4 Refined Activity Diagram

4.1.2. Component Diagram

Component diagram are UML structural diagrams that is used in modeling the physical aspects of object-oriented systems. It shows the relationship between various components of the system which helps to visualize, specify and document those components-based systems.

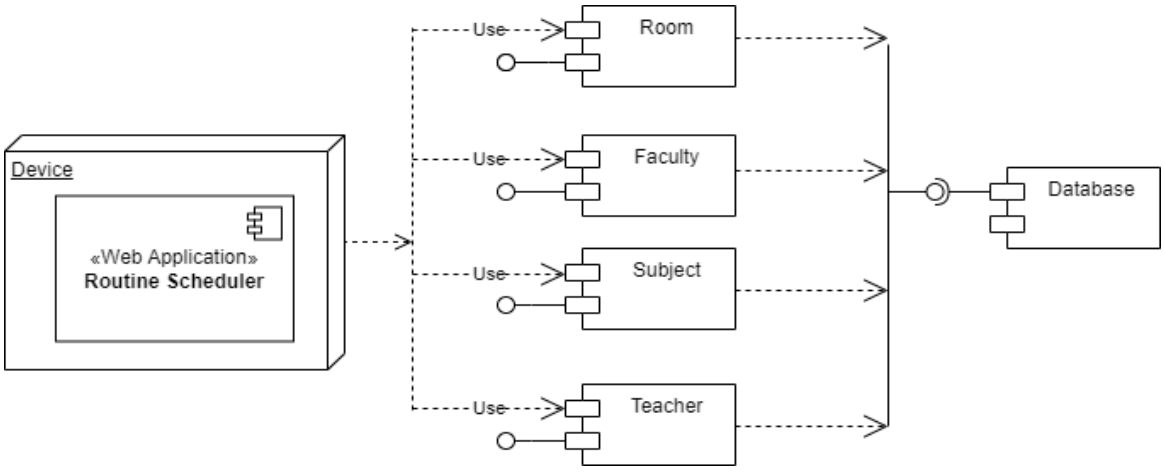


Figure 4.5 Component Diagram of Routine Scheduler

The main components in this system are the room, faculty, subject, and teacher. Each of these components is connected to the database, which stores information about them, such as their names, IDs, and schedules.

The room component represents a physical room where classes are held and has a connection to the database, which stores information about the room, such as its capacity, availability.

The faculty component represents a group of teachers who are assigned to teach in the classes and has a connection to the database, which stores information about the faculty, such as their names, IDs.

The subject component represents a course or a subject that is taught in the classes and has a connection to the database, which stores information about the subject, such as its name, course ID, and credit hours.

The teacher component represents an individual teacher who is assigned to teach a subject in a specific room and has connections to the faculty, subject, and room components as well as the database.

4.1.3. Deployment Diagram

Deployment diagram are UML structural diagrams that shows the relationships between the hardware and software components in the system and the physical distribution of the processing i.e., Deployment diagram are used to visualize the topology of the physical components of the system where software components are deployed.

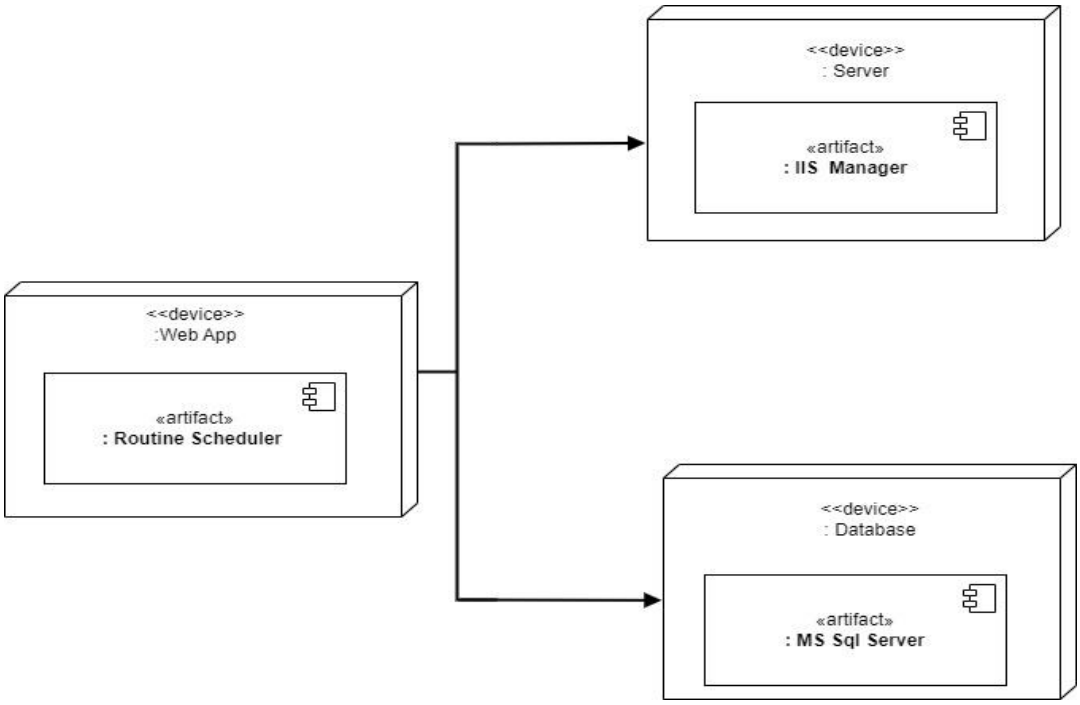


Figure 4.6 Deployment Diagram of Routine Scheduler

The deployment diagram would illustrate the relationship between the IIS manager node and the SQL Server node. It would show that the routine scheduler application communicates with the routine scheduler database through the SQL Server node. It would also show that the IIS manager node and the SQL Server node are connected through a network connection.

4.2. Algorithm Details

A genetic algorithm is a search-based algorithm used for solving optimization problems. This algorithm is important because it solves difficult problems that would take a long time to solve. Genetic algorithms use the evolutionary generational cycle to produce high-quality solutions. They use various operations that increase or replace the population to provide an improved fit solution.

Genetic algorithm approach is used in project to optimize the creation of a timetable. In this context, the population represents all of the available time slots in the timetable, while the chromosomes represent the various subjects to be scheduled. The genic element refers to the application of genetic algorithm operators, such as selection and crossover, to find the optimal solution for the timetable. [6]

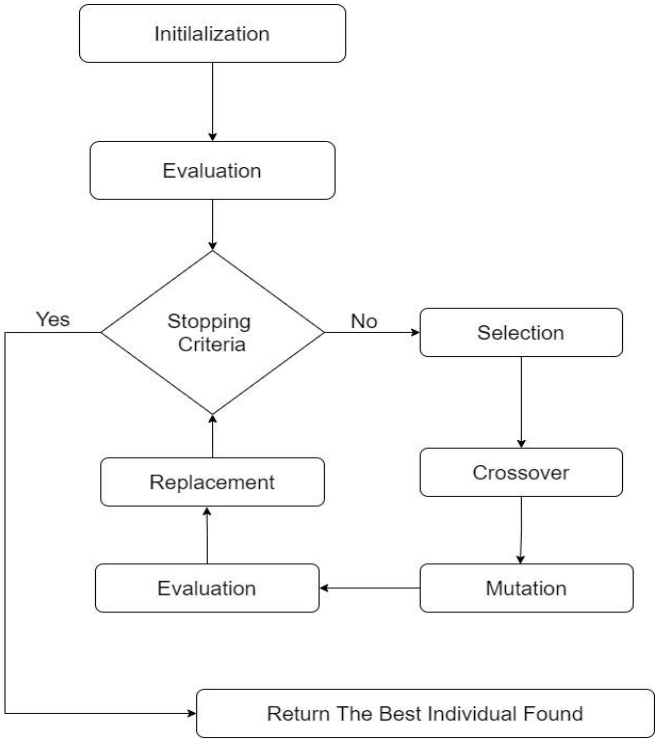


Figure 4.7 Working mechanism of genetic algorithm

The first stage of a genetic algorithm (GA) is to create an initial population, where each member of the population represents a potential solution to the problem at hand. The fitness

function is then used to evaluate each individual in the population and assign them a fitness score. It has been acknowledged that the success of the GA largely depends on the quality of the initial population. If the initial population is of high quality, the GA has a greater chance of finding a good solution. Conversely, if the initial population is insufficient or of poor quality, it will be challenging for the GA to find a good solution.

The selection operator in a genetic algorithm chooses chromosomes from the population to reproduce. Chromosomes with higher fitness scores have a greater probability of being selected for reproduction multiple times.

Crossover is a genetic operator utilized in genetic algorithms to introduce variation in the programming of one or more chromosomes across generations. This process is similar to biological reproduction and crossover, which is the foundation of genetic algorithms. During crossover, more than one parent solution is used to generate a new child solution. The chromosomes are selected through specific methods, and a random locus is chosen for exchanging subsequences before and after that locus between two chromosomes, resulting in the creation of two offspring.

CHAPTER 5
IMPLEMENTATION AND TESTING

5.1. Implementation Overview

Process model used

The Incremental Delivery Model is a software development approach that delivers the given project in small, incremental releases through a series of iterations. This model allows for changes and improvements to be made based on feedback, ensuring that the changes in the requirement can be deal with this model.

The Incremental Delivery Model is a proven and effective method for developing software projects like. By delivering small, manageable portions of the project, feedback can be obtained early on and ensure that the final product meets their needs. This approach also allows for a more flexible and responsive development process, as changes can be made quickly and easily based on feedback and changing requirements. The result is a highly optimized routine is generated.

5.1.1. Tools Used

5.1.1.1. Front End Tools

The front-end of the "Routine Scheduler" system was developed using a combination of technologies that enable the creation of a user-friendly and interactive interface. For web development, the team utilized HTML, CSS, and JS. HTML provides the structure and content of the web pages, while CSS is used to control the appearance and layout of the pages.

5.1.1.2. Back End Tools

The back-end of the system was developed using a combination of technologies that provide robust and scalable support for the application. For web development, the team utilized ASP.NET core with Controller, and code-behind.

MSSQL is a database server used in the project that is installed in the SSMS software for writing different database queries. The tables used in the project are created with MSSQl server queries and different stored procedures are also written with these queries.

5.1.2. Modules Description

This project implements following modules:

1. Description of Modules
    a) **Login/Registration module**: This model is use for the registration and login of the users. After login to the system user can further setup the required setup to generate the routine.

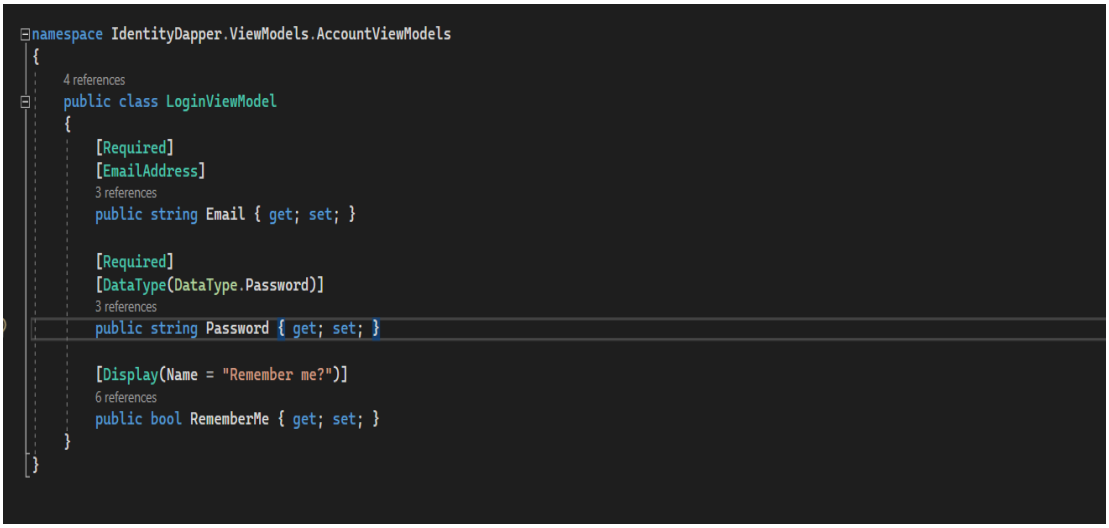


Figure 5.1 Login Module

```
namespace IdentityDapper.Models
{
    94 references
    public class ApplicationUser
    {
        30 references
        public int Id { get; set; }

        8 references
        public string UserName { get; set; }
        1 reference
        public string FullName { get; set;}
        4 references
        public string NormalizedUserName { get; set; }

        10 references
        public string Email { get; set; }

        4 references
        public string NormalizedEmail { get; set; }

        5 references
        public bool EmailConfirmed { get; set; }

        5 references
        public string PasswordHash { get; set; }

        6 references
        public string PhoneNumber { get; set; }

        4 references
        public bool PhoneNumberConfirmed { get; set; }

        7 references
```

Figure 5.2 Registration Module

b) **Setup model**: Setup model is design to do required setup to generate the routine. After login in to the system the user can setup the required data like Adding the classroom, Adding teacher, adding Courses, Adding Semester and other data to generate the routine.

```
Day.cs  Config.cs  Course.cs  GenerateTimeTable.cs  DayWise.cs  DayTimeSlot.cs  TeacherWiseController.cs  SemesterSectionController.cs
Routine.Core  Routine.Core.Model.DayTimeSlot  DayTimeSlotId
1   using System;
2   using System.Collections.Generic;
3   using System.ComponentModel.DataAnnotations;
4   using System.Text;
5
6   namespace Routine.Core.Model
7   {
        28 references
8       public class DayTimeSlot
9       {
10          [Required]
            2 references
11          public int DayTimeSlotId { get; set; }
12          [Required]
            1 reference
13          public int DayId { get; set; }
14          [Required]
            1 reference
15          public string SlotTitle { get; set; }
16          [Required]
            1 reference
17          public DateTime StartTime { get; set; }
18          [Required]
            1 reference
19          public DateTime EndTime { get; set; }
20          [Required]
            1 reference
21          public bool IsActive { get; set; }
22      }
23  }
24
```

```
using System.Collections.Generic;
using System.Text;

namespace Routine.Core.Model
{
    8 references
    public class DayWise
    {
        0 references
        public int TimeTableId { get; set; }
        2 references
        public string DayTitle { get; set; }
        2 references
        public string LabRoomTitle { get; set; }
        2 references
        public string Slot1 { get; set; }
        2 references
        public string Slot2 { get; set; }
        2 references
        public string Slot3 { get; set; }
        2 references
        public string Slot4 { get; set; }
        2 references
        public string Slot5 { get; set; }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Text;

namespace Routine.Core.Model
{
    34 references
    public class Lecturer
    {
        [Required]
        4 references
        public int LecturerId { get; set; }
        [Required]
        [Display(Name = "Full Name")]
        10 references
        public string FullName { get; set; }
        [Required]
        [Display(Name = "Contact Number")]
        10 references
        public string ContactNo { get; set; }
        [Required]
        [Display(Name = "Lecturer Status")]
        10 references
        public bool IsActive { get; set; }
    }
}
```

Figure 5.3 Setup Model

c) **Routine Generate Model:** This model is use to generate routine after the completion of setup process.

```
namespace Routine.Core.Model
{
    5 references
    public class ShowSemesterWise
    {
        0 references
        public int TimeTableId { get; set; }
        0 references
        public string TimeTableTitle { get; set; }
    }
    0 references
    public class ShowSemesterWise1
    {
        0 references
        public string SemesterId { get; set; }
        0 references
        public string Semester { get; set; }
        0 references
        public string TimeTableTitle { get; set; }


    }
}
```

Figure 5.4 Routine Generation Model

2. Controller Description:

Here are the some main controller that are implements into the system.

a) **Account Controller:** This Controller consists of Login, Register, and Logout methods. Register Method is use to register the user. Login method is use for login in to the system. And Logout is use to exist from the system.

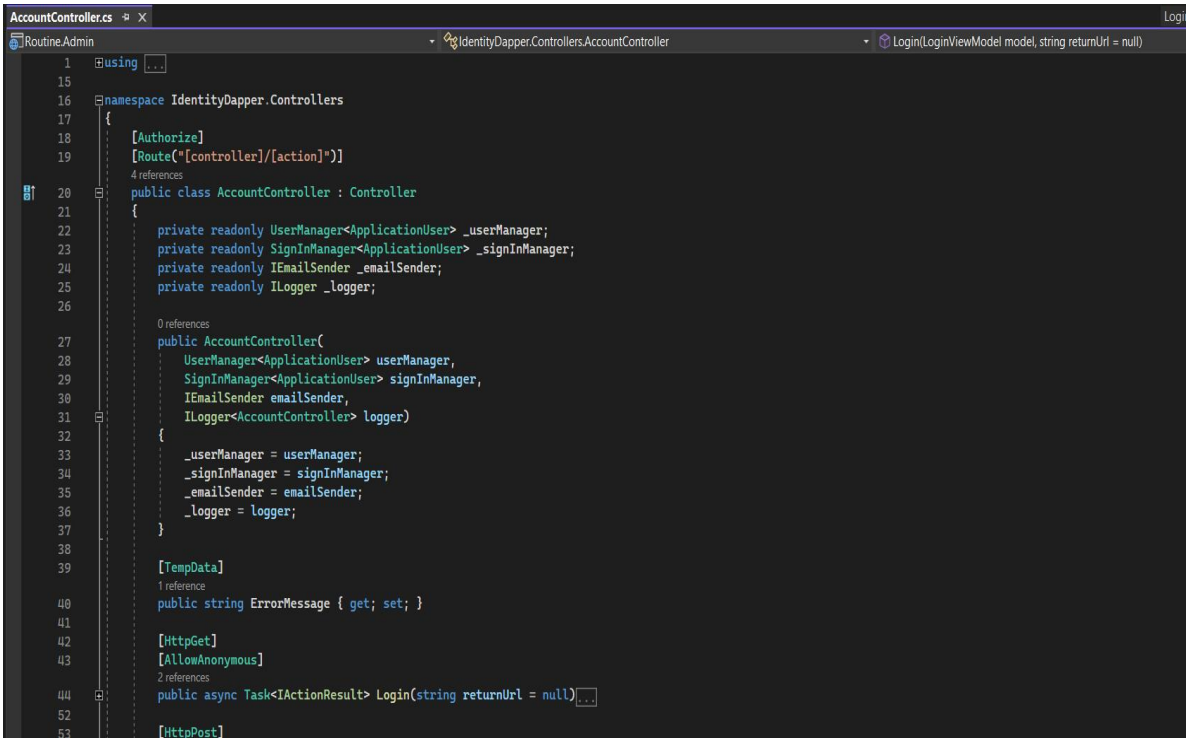
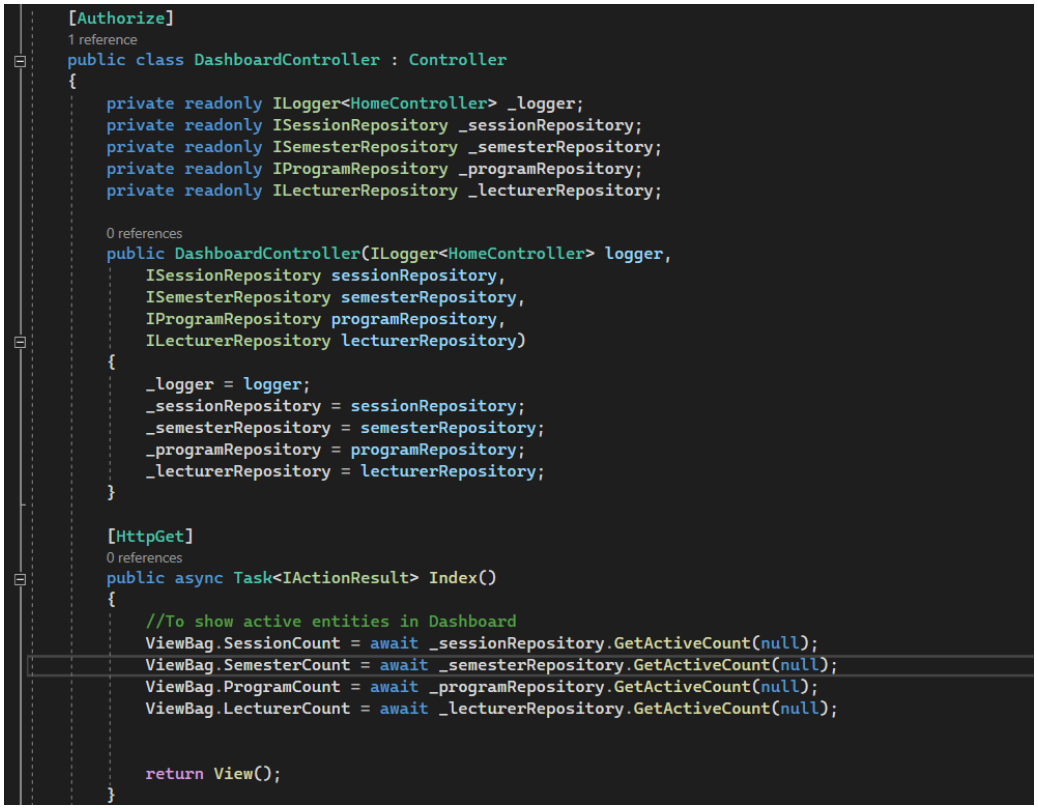


Figure 5.5 Account Controller

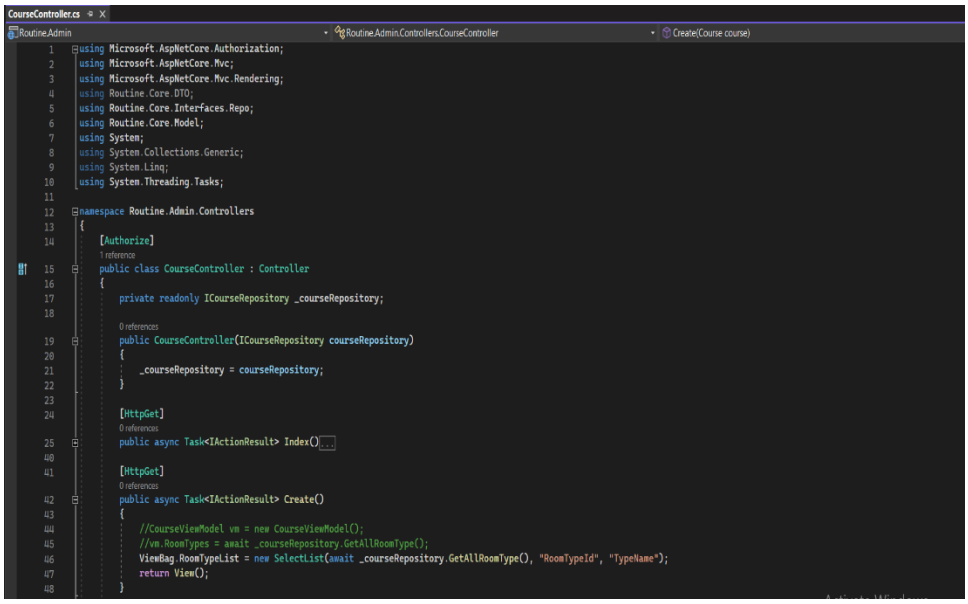
b) **Dashboard Controller**: This controller holds the method to show various data in dashboard.



```
[Authorize]
1 reference
public class DashboardController : Controller
{
    private readonly ILogger<HomeController> _logger;
    private readonly ISessionRepository _sessionRepository;
    private readonly ISemesterRepository _semesterRepository;
    private readonly IProgramRepository _programRepository;
    private readonly ILecturerRepository _lecturerRepository;

    0 references
    public DashboardController(ILogger<HomeController> logger,
        ISessionRepository sessionRepository,
        ISemesterRepository semesterRepository,
        IProgramRepository programRepository,
        ILecturerRepository lecturerRepository)
    {
        _logger = logger;
        _sessionRepository = sessionRepository;
        _semesterRepository = semesterRepository;
        _programRepository = programRepository;
        _lecturerRepository = lecturerRepository;
    }

    [HttpGet]
    0 references
    public async Task<IActionResult> Index()
    {
        //To show active entities in Dashboard
        ViewBag.SessionCount = await _sessionRepository.GetActiveCount(null);
        ViewBag.SemesterCount = await _semesterRepository.GetActiveCount(null);
        ViewBag.ProgramCount = await _programRepository.GetActiveCount(null);
        ViewBag.LecturerCount = await _lecturerRepository.GetActiveCount(null);


        return View();
    }
```

Figure 5.6 Dashboard Controller

c) **Course Controller**: Course Controller is use to add the course of the various semesters.



```
CourseController.cs
Routine.Admin
Routine.Admin.Controllers.CourseController
Create(Course course)
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Routine.Core.DTO;
using Routine.Core.Interfaces.Repo;
using Routine.Core.Model;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace Routine.Admin.Controllers
{
    [Authorize]
    1 reference
    public class CourseController : Controller
    {
        private readonly ICourseRepository _courseRepository;

        0 references
        public CourseController(ICourseRepository courseRepository)
        {
            _courseRepository = courseRepository;
        }

        [HttpGet]
        0 references
        public async Task<IActionResult> Index()...

        [HttpGet]
        0 references
        public async Task<IActionResult> Create()
        {
            //CourseViewModel vm = new CourseViewModel();
            //vm.RoomTypes = await _courseRepository.GetAllRoomType();
            ViewBag.RoomTypeList = new SelectList(await _courseRepository.GetAllRoomType(), "RoomTypeId", "TypeName");
            return View();
        }
```

Figure 5.7 Course Controller

d) **Lecturer Controller:** This controller consists the method to add teacher for the different semester.

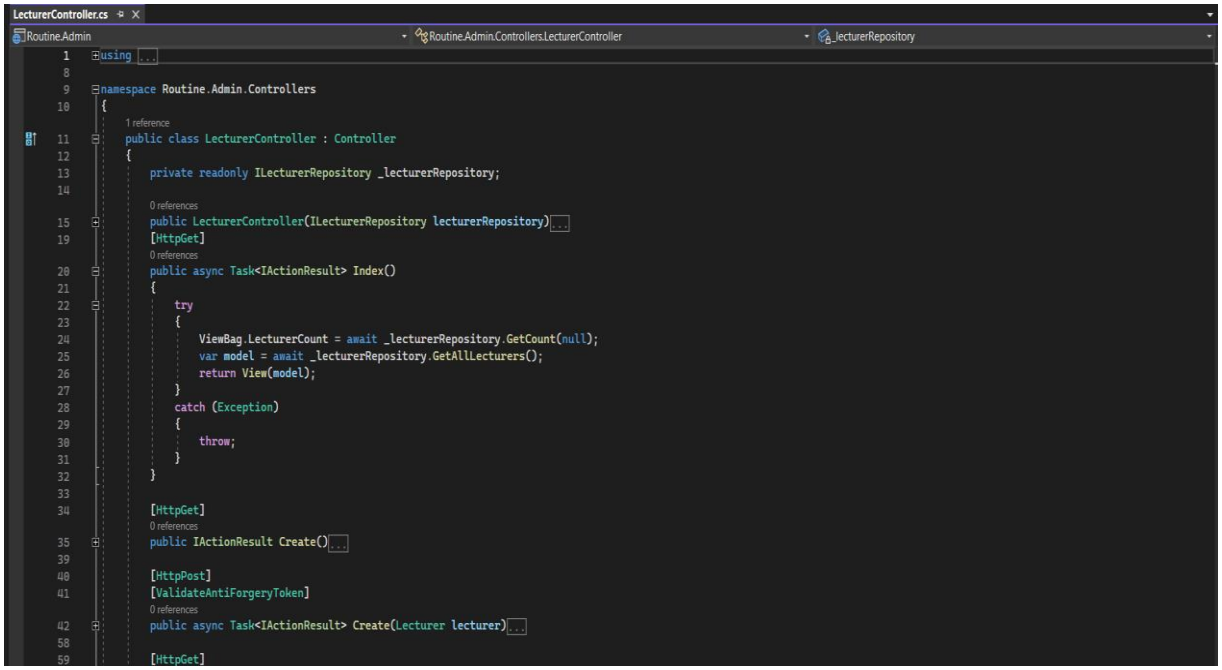


Figure 5.8 Lecturer Controller

e) **LecturerSubject Controller:** This controller holds the methods to map the lecturer with their respective subject for different semester.

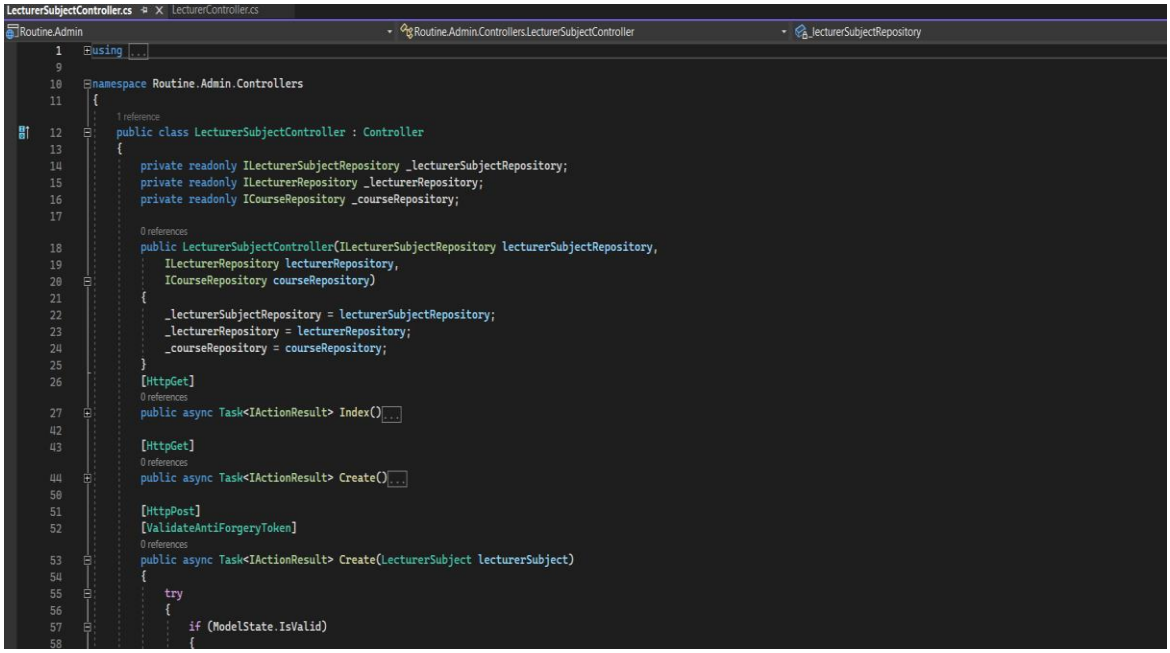


Figure 5.9 Lecturer Subject Controller

f) **GenerateTimeTable Controller:** This controller holds the method to generate the timetable for the various active semester with the help of algorithm that is implemented in the database.

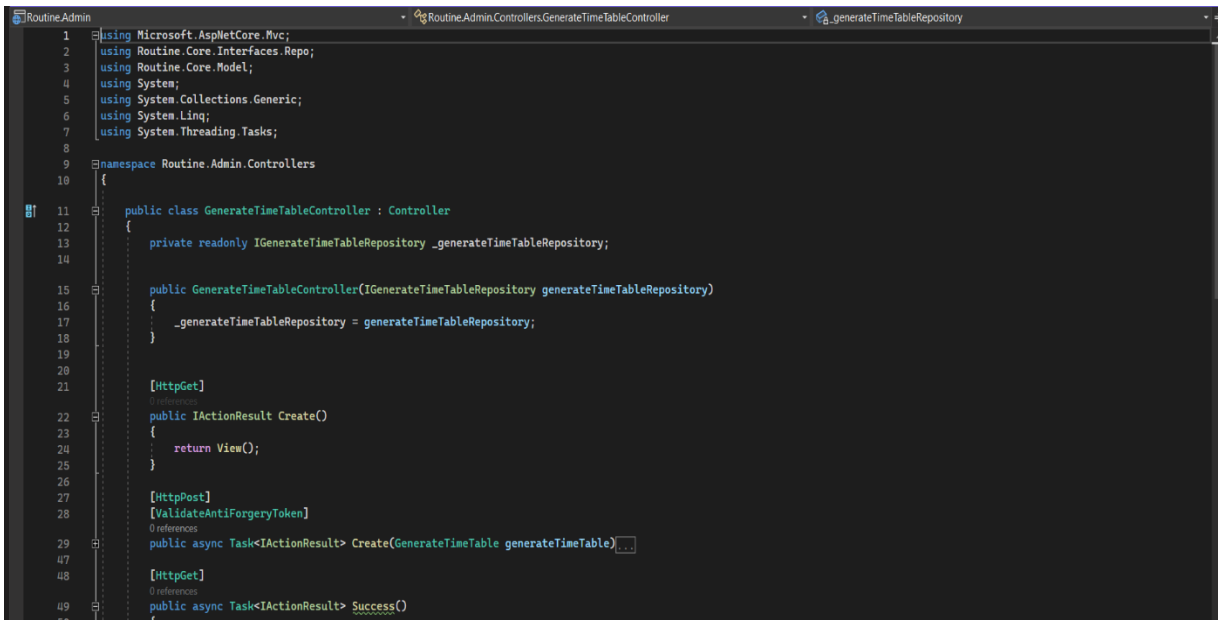
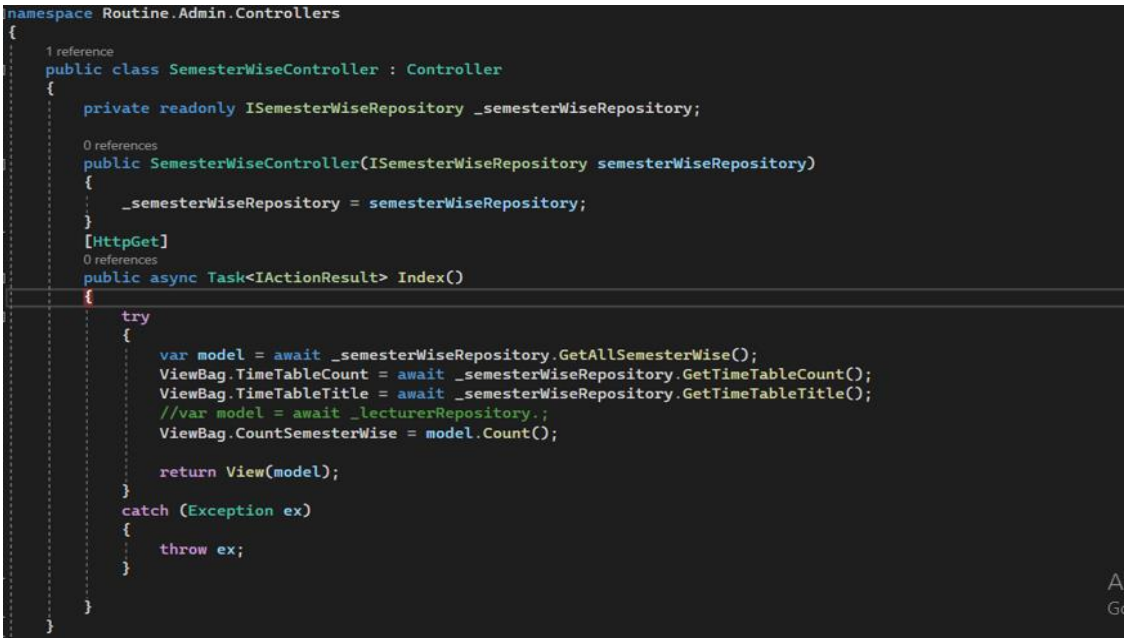


Figure 5.10 Time Table Generation Controller

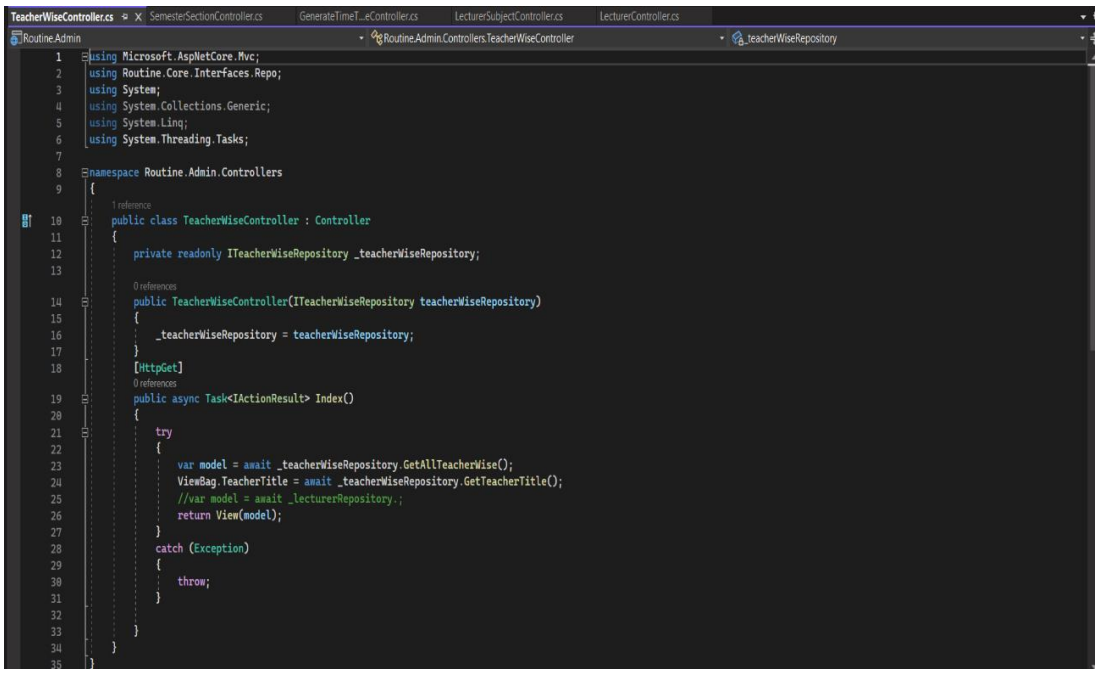
g) **SemesterWise Controller**: This controller is use to generate routine semester wise with the help of previously generated timetable.



```
namespace Routine.Admin.Controllers
{
    1 reference
    public class SemesterWiseController : Controller
    {
        private readonly ISemesterWiseRepository _semesterWiseRepository;

        0 references
        public SemesterWiseController(ISemesterWiseRepository semesterWiseRepository)
        {
            _semesterWiseRepository = semesterWiseRepository;
        }
        [HttpGet]
        0 references
        public async Task<IActionResult> Index()
        {
            try
            {
                var model = await _semesterWiseRepository.GetAllSemesterWise();
                ViewBag.TimeTableCount = await _semesterWiseRepository.GetTimeTableCount();
                ViewBag.TimeTableTitle = await _semesterWiseRepository.GetTimeTableTitle();
                //var model = await _lecturerRepository.;
                ViewBag.CountSemesterWise = model.Count();

                return View(model);
            }
            catch (Exception ex)
            {
                throw ex;
            }

        }
    }
```

Figure 5.11 Semester Wise Controller

h) **TeacherWise Controller:** This controller is use to generate routine Teacherwise with the help of previously generated timetable.



```
using Microsoft.AspNetCore.Mvc;
using Routine.Core.Interfaces.Repo;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace Routine.Admin.Controllers
{
    1 reference
    public class TeacherWiseController : Controller
    {
        private readonly ITeacherWiseRepository _teacherWiseRepository;

        0 references
        public TeacherWiseController(ITeacherWiseRepository teacherWiseRepository)
        {
            _teacherWiseRepository = teacherWiseRepository;
        }
        [HttpGet]
        0 references
        public async Task<IActionResult> Index()
        {
            try
            {
                var model = await _teacherWiseRepository.GetAllTeacherWise();
                ViewBag.TeacherTitle = await _teacherWiseRepository.GetTeacherTitle();
                //var model = await _lecturerRepository.;
                return View(model);
            }
            catch (Exception)
            {
                throw;
            }

        }
    }
}
```

Figure 5.12 Teacher Wise Controller

3. Repository:

This consists of one class and one interface. Repository classes hold all the database operations from database connection to database manipulation and queries for DDL and DML operations. And the corresponding Interface are injected to the controller.

```
2 references
public class DayTimeSlotRepository : IDayTimeSlotRepository
{
    private readonly IRoutineAppContext _routineAppContext;

    0 references
    public DayTimeSlotRepository(IRoutineAppContext routineAppContext)
    {
        _routineAppContext = routineAppContext;
    }
    13 references
    public async Task<DayTimeSlot> Add(DayTimeSlot entity)
    {
        DayTimeSlot dayTimeSlot = new DayTimeSlot();
        try
        {
            string sql = @"INSERT INTO tbl_daytimeslot(DayId, SlotTitle, StartTime, EndTime, IsActive)
                           VALUES (@dayId, @slotTitle, @startTime, @endTime, @isActive)";

            using (IDbConnection db = _routineAppContext.Connection)
            {
                var parameters = new DynamicParameters();
                //parameters.Add("@courseId", entity.CourseId);
                parameters.Add("@dayId", entity.DayId);
                parameters.Add("@slotTitle", entity.SlotTitle);
                parameters.Add("@startTime", entity.StartTime);
                parameters.Add("@endTime", entity.EndTime);
                parameters.Add("@isActive", entity.IsActive);

                dayTimeSlot = await db.QueryFirstOrDefaultAsync<DayTimeSlot>(sql, parameters, commandType: CommandType.Te

                return dayTimeSlot;
            }

        }
        catch (Exception ex)
```

Figure 5.13 Repository

5.1.2.1. Class Description

This project implements many classes, some of them are as follows

a) **HomePage:** This class holds all the methods and function for creating the homepage and dashboard of the application. The user is greeted by this page upon successfully logging into the application.

b) **LoginPage:** This class holds the code for the login form of the application. This class forwards the login information received to the database.

c) **ForgotPassword:** This class holds the code for creating a new password for the user incase the user has forgotten the password for the application

d) **RegisterPage:** This class holds the code for register form for the application.
e) **Dapper:** This class holds the process of all the database operations from database connection to database manipulation.

5.1.2.2. Algorithm Description

In the Routine Scheduler project, a genetic algorithm is used to generate a schedule for routine activities of various faculty teachers. The genetic algorithm works by creating an initial population of possible solutions, which in this case is a set of schedules for the teachers. Next, the genetic algorithm selects the fittest solutions and combines them through genetic operations such as crossover and mutation to create a new population of solutions. The process of evaluating and creating new solutions is repeated over several generations until an optimal solution is reached. In the context of the Routine Scheduler project, the genetic algorithm generates a timetable that includes the date interval and other details such as subject, teacher, classroom, course, and semester. The algorithm takes into account various constraints such as room availability, teacher availability, break time, and lab requirements while generating the timetable. The use of a genetic algorithm allows for an efficient and effective way to generate a schedule that meets the necessary constraints and requirements.

Implement of Genetic Algorithm:

This is a stored procedure named "GenerateTimeTablesForAllSession" in the database. It appears to be implementing a genetic algorithm to generate time tables for different sessions. the procedure takes three input parameters: **StartDate** (start date of the time table), **EndDate** (end date of the time table), and **Message** (an output parameter to return a message). The purpose of this stored procedure is to generate time tables based on the given start and end dates for all sessions.

The stored procedure performs several steps related to generating time tables using a genetic algorithm. Here is a breakdown of the main steps:

1. It declares and inializes various tables to hold subject, time slot and other data.

```
DECLARE @ALLSubjectSemesterTable
    Table(RowNo int, RepeatedRowNo int,ProgramSemesterId int,CrHrs int, ProgramSemesterSubjectId int,SemesterSubjectTitle nvarchar(300),Title nvarchar(200),RoomTypeId int,LecturerId int,SessionId int, SessionTitle NVARCHAR(200))

DECLARE @SubjectSemesterTable
    Table(RowNo int,ProgramSemesterId int,CrHrs int, ProgramSemesterSubjectId int,SemesterSubjectTitle nvarchar(300),Title nvarchar(200),RoomTypeId int,LecturerId int,SessionId int, SessionTitle NVARCHAR(200))

-- Get All Practical Class -- Lab
DECLARE @PracticalSubjectTable
   Table(ProgramSemesterId int,CrHrs int, ProgramSemesterSubjectId int,SemesterSubjectTitle nvarchar(300),Title nvarchar(200),RoomTypeId int,LecturerId int,SessionId int, SessionTitle NVARCHAR(200))

DECLARE @PracticalRandomSubjectTable
   Table(RowNo int,ProgramSemesterId int,CrHrs int, ProgramSemesterSubjectId int,SemesterSubjectTitle nvarchar(300),Title nvarchar(200),RoomTypeId int,LecturerId int,SessionId int, SessionTitle NVARCHAR(200))

-- Get All Non Practical Class --Theory
DECLARE @NonPracticalSubjectTable
   Table(ProgramSemesterId int,CrHrs int, ProgramSemesterSubjectId int,SemesterSubjectTitle nvarchar(300),Title nvarchar(200),RoomTypeId int,LecturerId int,SessionId int, SessionTitle NVARCHAR(200))

DECLARE @NonPracticalRandomSubjectTable
   Table(RowNo int,ProgramSemesterId int,CrHrs int, ProgramSemesterSubjectId int,SemesterSubjectTitle nvarchar(300),Title nvarchar(200),RoomTypeId int,LecturerId int,SessionId int, SessionTitle NVARCHAR(200))

-- Get All Avalibale Slots -- Room and Lab Population
DECLARE @ALLRoomsSlots
   Table(RowNo int, DayTimeSlotId int,SlotTitle nvarchar(200),StartTime Time,EndTime Time,DayId int,DayTitle nvarchar(100),RoomId int,RoomNo nvarchar(200),Capacity int)

DECLARE @ALLLabsSlots
   Table(RowNo int, DayTimeSlotId int,SlotTitle nvarchar(200),StartTime Time,EndTime Time,DayId int,DayTitle nvarchar(100),LabId int,LabNo nvarchar(200),Capacity int)
```

2. It populates the **SubjectSemesterTable** table with subjects retrieved from the v_AllSemesterSectionSubjects view.

```
Insert into @SubjectSemesterTable(RowNo,ProgramSemesterId,CrHrs,ProgramSemesterSubjectId,SemesterSubjectTitle,Title,RoomTypeId,LecturerId,SessionId,SessionTitle)
select Row_Number() over(Order By(SELECT 1)) as RowNo, ProgramSemesterId,CrHrs,ProgramSemesterSubjectId,SemesterSubjectTitle,Title,RoomTypeId,LecturerId,
SessionId,SessionTitle From v_AllSemesterSectionSubjects Order By NewID();
```

3. It duplicates rows in the **SubjectSemesterTable** table based on the value of the CrHrs column, inserting the duplicated rows into the **ALLSubjectSemesterTable**.

```
DECLARE @IndexNo int = 1;
DECLARE @RowNo int = 1;
DECLARE @CountRecord int = (Select Count(*) From @SubjectSemesterTable);
        WHILE @IndexNo <= @CountRecord
        BEGIN
            DECLARE @RepeatPrint AS INT = (SELECT CrHrs From @SubjectSemesterTable where RowNo = @IndexNo);
                    DECLARE @CountCrHrs int = 1;
                          WHILE @CountCrHrs <=@RepeatPrint
                          BEGIN
                            Insert into @ALLSubjectSemesterTable(RowNo,RepeatedRowNo,ProgramSemesterId,CrHrs,ProgramSemesterSubjectId,
                            SemesterSubjectTitle,Title,RoomTypeId,LecturerId,SessionId,SessionTitle)
                                select @RowNo, RowNo, ProgramSemesterId,CrHrs,ProgramSemesterSubjectId,SemesterSubjectTitle,Title,
                                RoomTypeId,LecturerId,SessionId ,SessionTitle
                                From @SubjectSemesterTable WHERE RowNo = @IndexNo;
                                       SET @CountCrHrs = @CountCrHrs + 1;
                                       SET @RowNo = @RowNo + 1;
                                       END
                SET @IndexNo = @IndexNo + 1;
                END
```

4. It selects Non practical subjects from the **ALLSubjectSemesterTable** and inserts them into the **NonPracticalSubjectTable** and **PracticalRandomSubjectTable** tables.

```
-- Getting Non Practical Class

INSERT INTO @NonPracticalSubjectTable(ProgramSemesterId,CrHrs,ProgramSemesterSubjectId,SemesterSubjectTitle,
Title,RoomTypeId,LecturerId,SessionId,SessionTitle)

SELECT ProgramSemesterId,CrHrs,ProgramSemesterSubjectId,SemesterSubjectTitle,Title,RoomTypeId,LecturerId,SessionId,SessionTitle
From @ALLSubjectSemesterTable
WHERE RoomTypeId = 1  Order By NewID();

INSERT INTO @NonPracticalRandomSubjectTable(RowNo,ProgramSemesterId,CrHrs,ProgramSemesterSubjectID,
SemesterSubjectTitle,Title,RoomTypeId,LecturerId,SessionId,SessionTitle)

SELECT ROW_NUMBER() over (Order By (Select 1)), ProgramSemesterId,CrHrs,ProgramSemesterSubjectId,
SemesterSubjectTitle,Title,RoomTypeId,LecturerId,SessionId,SessionTitle
From @NonPracticalSubjectTable;
```

5. It populates the **AllRoomsSlots** and **AllLabsSlots** tables with time slot data for rooms and labs, respectively.

```
-- Getting All Rooms Time Slots  -- It's Room Slots Population

INSERT INTO @ALLRoomsSlots(RowNo,DayTimeSlotId,SlotTitle,StartTime,EndTime,DayId,DayTitle,RoomId,RoomNo,Capacity)
SELECT Row_Number() over (Order By(SELECT 1))as RowNo,DayTimeSlotId,SlotTitle,StartTime,EndTime,DayId,Name,RoomId,RoomNo,Capacity  FROM v_AllActiveTimeSlots
CROSS JOIN (SELECT * FROM tbl_room where IsActive = 1) RT WHERE SlotStatus = 1 ORDER BY RT.RoomId


-- Getting All Labs Time Slots
-- It's Lab Slots Population

Insert into @ALLLabsSlots(RowNo,DayTimeSlotId,SlotTitle,StartTime,EndTime,DayId,DayTitle,LabId,LabNo,Capacity)
SELECT Row_Number() over (Order By(SELECT 1))as RowNo,DayTimeSlotId,SlotTitle,StartTime,EndTime,DayId,Name,LabId,LabNo,Capacity FROM v_AllActiveTimeSlots
CROSS JOIN (SELECT * FROM tbl_lab where IsActive = 1) LT WHERE SlotStatus = 1 ORDER BY LT.LabId
```

6. It calculates various variables and checks if the number of slots and subjects is sufficient.

```
-- DECLARE Variables For Check Time Slots is Enough

DECLARE @AllPracticalClass int = (SELECT COUNT(*) FROM @PracticalRandomSubjectTable);
DECLARE @AllNonPracticalClass int = (SELECT COUNT(*) FROM @NonPracticalRandomSubjectTable);
DECLARE @TotalLabsTimeSlot int = (SELECT COUNT(*) FROM @ALLLabsSlots);
DECLARE @TotalRoomsTimeSlot int = (SELECT COUNT(*) FROM @ALLRoomsSlots);
DECLARE @RemainingSlotsLabs int =  @TotalLabsTimeSlot - @FitnessValueAllPracticalClass;
DECLARE @TotalNonPracticalTimeSlots int = @TotalRoomsTimeSlot + @RemainingSlotsLabs;
DECLARE @TotalTimeSlots int = @TotalRoomsTimeSlot + @TotalLabsTimeSlot
DECLARE @TotalClass int = @FitnessValueAllPracticalClass + @FitnessValueAllNonPracticalClass;
```

7. It initializes break duration variables for labs and rooms based on the available slots.

```
-- Initialization Slots Break Duration Variables

   IF @TotalTimeSlots = @TotalClass
    BEGIN
        SET @RoomBreak = 0;
        SET @LabBreak = 0;
    END
    ELSE IF @FitnessValueAllNonPracticalClass < @TotalRoomsTimeSlot OR @FitnessValueAllNonPracticalClass = @TotalRoomsTimeSlot
    BEGIN
    -- Set Room Break
     SET @RoomDifference = @TotalRoomsTimeSlot - @FitnessValueAllNonPracticalClass;

     SET @SetRoomDifference = @TotalRoomsTimeSlot / @RoomDifference;
         SET @RoomBreak = Cast(@SetRoomDifference as int);

     -- Set Lab Break

     SET @LabDifference = @TotalLabsTimeSlot - @FitnessValueAllPracticalClass;

     SET @SetLabDifference = @TotalLabsTimeSlot / @LabDifference;
         SET @LabBreak = Cast(@SetLabDifference as int);

        SET @Message = 'Both Lab Slots and Room Slots Break Varaible Initilize';
    END
    ELSE IF @FitnessValueAllNonPracticalClass > @TotalRoomsTimeSlot
    BEGIN
    -- Set Room Break
     SET @RoomDifference = @TotalNonPracticalTimeSlots - @FitnessValueAllNonPracticalClass;
     SET @SetRoomDifference = @TotalNonPracticalTimeSlots / @RoomDifference;
     SET @RoomBreak = Cast(@SetRoomDifference as int) + 1;

     -- Set Lab Break
         SET @LabBreak = 0;
         SET @Message = 'Only Room Slots Break Varaible Initilize';
    END
```

8. It creates tables (**TimeTableHeader** and **TimeTableDetails**) to store the generated time table data.

```
  -- Generate Time Table Header and Details Tables are Polulation Table

DECLARE @TimeTableHeader Table(TimeTableId int,SessionId int,SessionTitle NVARCHAR(500), ProgramSemesterId int,TimeTableTitle nvarchar(200),
SemesterTitle nvarchar(200) ,StartDate Date, EndDate Date, IsActive bit);

DECLARE @TimeTableDetails Table(TimeTableId int,SessionId int,SessionTitle NVARCHAR(500), ProgramSemesterSubjectId int, SubjectTitle nvarchar(400),
RoomId int, LabId int, DayTimeSlotId int, LecturerId int, DayId int, IsActive bit);
```

9. It generates time tables for each semester by iterating over the **AllSemesters** table.

```
-- Create Time Table for one by one Semester
DECLARE @CAllSemester int = (Select Count(*) FROM @AllSemesters);
DECLARE @OneByOneSemester int = 1;
WHILE @OneByOneSemester <= @CAllSemester
BEGIN
-- Get SEMESTER HEADER
DECLARE @SessionTitle nvarchar(150) = (SELECT TOP 1 SessionTitle FROM @AllSemesters WHERE RowNo = @OneByOneSemester);

DECLARE @SessionID nvarchar(150) = (SELECT TOP 1 SessionId FROM @AllSemesters WHERE RowNo = @OneByOneSemester);

DECLARE @ProgramSemesterID int = (SELECT TOP 1 ProgramSemesterId FROM @AllSemesters WHERE RowNo = @OneByOneSemester);

DECLARE @Title nvarchar(200) = (SELECT TOP 1 Title FROM @AllSemesters WHERE RowNo = @OneByOneSemester);

SET @TimeTableTitle = (SELECT Title FROM @AllSemesters WHERE RowNo = @OneByOneSemester); -- + ' - '+@SessionTitle;

INSERT INTO @TimeTableHeader(TimeTableId,SessionId,SessionTitle,ProgramSemesterId,TimeTableTitle, SemesterTitle ,StartDate, EndDate, IsActive)
VALUES(@OneByOneSemester, @SessionID,@SessionTitle, @ProgramSemesterID,@TimeTableTitle,@Title,@StartDate, @EndDate,1);
SET @OneByOneSemester = @OneByOneSemester + 1;
END
```

10. It performs selection and crossover operations using a genetic algorithm.

```
-- Assign Non Practical Class
DECLARE @OneByOneNonPracticalSubject int  = 1;
DECLARE @FitnessValueAllNonPraticalClass int = (SELECT  dbo.CalculateFitnessValue());
-- SELECTION/CROSSOVER OPERATOR
WHILE @OneByOneNonPracticalSubject <= @FitnessValueAllNonPracticalClass
BEGIN
    SET @TimeTableTitle = (SELECT Title FROM @AllSemesters WHERE RowNo = @OneByOneSemester) + ' - '+@SessionTitle;

    DECLARE @RoomDProgramSemesterID int = (SELECT TOP 1 ProgramSemesterId FROM @NonPracticalRandomSubjectTable WHERE RowNo = @OneByOneNonPracticalSubject);

    DECLARE @RoomSemesterTitle nvarchar(150) = (SELECT TOP 1 Title FROM @NonPracticalRandomSubjectTable WHERE RowNo = @OneByOneNonPracticalSubject);

    DECLARE @RoomTimeTableID int = (SELECT TOP 1 TimeTableId FROM @TimeTableHeader WHERE ProgramSemesterId = @RoomDProgramSemesterID
    AND SemesterTitle = @RoomSemesterTitle)

    DECLARE @RoomProgramSemesterSubjectID int = (SELECT TOP 1 ProgramSemesterSubjectId FROM @NonPracticalRandomSubjectTable
    WHERE RowNo = @OneByOneNonPracticalSubject);

    DECLARE @RoomSubjectTitle nvarchar(300)=  (SELECT TOP 1 Title + ' ' + SemesterSubjectTitle FROM @NonPracticalRandomSubjectTable
    WHERE RowNo = @OneByOneNonPracticalSubject);

    DECLARE @RoomTypeID int  =  (SELECT TOP 1 RoomTypeId FROM @NonPracticalRandomSubjectTable WHERE RowNo = @OneByOneNonPracticalSubject);

    DECLARE @RoomLecturerID int = (SELECT TOP 1 LecturerId FROM @NonPracticalRandomSubjectTable WHERE RowNo = @OneByOneNonPracticalSubject);

    DECLARE @RoomLabID int = 0;
    DECLARE @RoomID int = 0;
    DECLARE @RoomDayID int = 0;
    DECLARE @RoomDayTimeSlotID int  = 0;
    DECLARE @RoomIsActive bit = 1;
```

11. It assigns practical subjects to time slots based on the selected crossover data.

```
    IF @LabBreakDurationNo = @LabBreak
        BEGIN
            SET @LabTimeSlotNo = @LabTimeSlotNo + 2;
            SET @LabBreakDurationNo = 1;
        END
        ELSE
        BEGIN
            SET @LabTimeSlotNo = @LabTimeSlotNo + 1;
            SET @LabBreakDurationNo = @LabBreakDurationNo + 1;
        END

    INSERT INTO @TimeTableDetails(TimeTableId,SessionTitle,SessionId, ProgramSemesterSubjectId, SubjectTitle, RoomId,
    LabId, DayTimeSlotId, LecturerId, DayId, IsActive)

    Values(@LabTimeTableID,@SubjectSessionTitle,@SubjectSessionID,@LabProgramSemesterSubjectID,@LabSubjectTitle,0,@LabID,
    @LabDayTimeSlotID,@LabLecturerID,@LabDayID,@LabIsActive);

    IF @LabBreakDurationNo = 1
    BEGIN
     INSERT INTO @TimeTableDetails(TimeTableId,SessionTitle,SessionId, ProgramSemesterSubjectId, SubjectTitle, RoomId, LabId,
                                   DayTimeSlotId, LecturerId, DayId, IsActive)
     Values(@LabTimeTableID,'',0,0,'Break',0,@LabID,@LabDayTimeSlotID,0,@LabDayID,@LabIsActive);
    END

SET @OneByOnePracticalSubject = @OneByOnePracticalSubject + 1;
END
SET @Message = 'All Practical Time Table Details initialize';
```

12. It assigns non-practical subjects to time slots based on available rooms and labs.

```
BEGIN
    SET @RoomLabID  = (SELECT TOP 1 LabId FROM @ALLLabsSlots WHERE RowNo = @LabTimeSlotNo);
    SET @RoomDayTimeSlotID = (SELECT TOP 1 DayTimeSlotID FROM @ALLLabsSlots WHERE RowNo = @LabTimeSlotNo);
    SET @RoomDayID = (SELECT TOP 1 DayId FROM @ALLLabsSlots WHERE RowNo = @LabTimeSlotNo);

    IF @RoomBreakDurationNo = @RoomBreak
    BEGIN
        SET @LabTimeSlotNo = @LabTimeSlotNo + 2;
        SET @RoomBreakDurationNo = 1;
    END

    ELSE
    BEGIN
        SET @LabTimeSlotNo = @LabTimeSlotNo + 1;
        SET @RoomBreakDurationNo = @RoomBreakDurationNo + 1;
    END
END
INSERT INTO @TimeTableDetails(TimeTableId,SessionTitle,SessionId, ProgramSemesterSubjectId, SubjectTitle, RoomId, LabId, DayTimeSlotId, LecturerId, DayId, IsActive)
Values(@RoomTimeTableID,@SubjectSessionTitle,@SubjectSessionID,@RoomProgramSemesterSubjectID,@RoomSubjectTitle,@RoomID,@RoomLabID,@RoomDayTimeSlotID,@RoomLecturerID,
@RoomDayID,@RoomIsActive);

IF @RoomBreakDurationNo = 1
BEGIN
    INSERT INTO @TimeTableDetails(TimeTableId,SessionTitle,SessionId, ProgramSemesterSubjectId, SubjectTitle, RoomId, LabId,
                                  DayTimeSlotId, LecturerId, DayId, IsActive)
    Values(@RoomTimeTableID,'',0,0,'Break',@RoomID,0,@RoomDayTimeSlotID,@RoomLabID,@RoomDayID,@RoomIsActive);
END

SET @OneByOneNonPracticalSubject = @OneByOneNonPracticalSubject + 1;
END
SET @Message = 'All Non Practical Time Table Details initialize';
```

13. It inserts the generated time table data into the **TimeTableHeader** and **TimeTableDetails** tables.

```
DELETE FROM tbl_timetabledetails;
DELETE FROM tbl_timetable;

INSERT INTO tbl_timetable(TimeTableId ,SessionId,SessionTitle,ProgramSemsterId ,TimeTableTitle , SemesterTitle ,StartDate , EndDate , IsActive )

SELECT TimeTableId ,SessionId,SessionTitle ,ProgramSemesterId ,TimeTableTitle , SemesterTitle ,StartDate , EndDate , IsActive
FROM @TimeTableHeader;

INSERT INTO tbl_timetabledetails(TimeTableId, SessionId, SessionTitle, ProgramSemesterSubjectId, SubjectTitle, RoomId, LabId , DayTimeSlotId,
LecturerId, DayId , IsActive)

SELECT TimeTableId,SessionId,SessionTitle, ProgramSemesterSubjectId, SubjectTitle, RoomId, LabId , DayTimeSlotId, LecturerId, DayId , IsActive
FROM @TimeTableDetails
SET @Message = 'Routine Created Successfully';
```

Finally, the stored procedure sets the output parameter **Message** to indicate the status of the time table generation process.

SELECTION/CROSSOVER OPERATOR

The code implements the selection and crossover operators of the genetic algorithm to assign practical and non-practical subjects to random time slots in the time table. The algorithm iterates over each practical and non-practical subject and performs the following steps:

1. Retrieves the details of the subject, such as ProgramSemesterID, CrHrs (credit hours), ProgramSemesterSubjectID, SemesterSubjectTitle, Title, RoomTypeID, LecturerID, SessionID, and SessionTitle.

```
-- DECLARE Time Slot Validation Variables
DECLARE @LabTimeSlotNo int = 1;
DECLARE @RoomTimeSlotNo int = 1;

-- DECLARE Time Slot Marks as Empty
DECLARE @RoomBreakDurationNo int = 1;
DECLARE @LabBreakDurationNo int = 1;

-- Assign First Practical Class
DECLARE @OneByOnePracticalSubject int  = 1;
DECLARE @FitnessValueAllPraticalClass int = (SELECT  dbo.CalculateFitnessValue());
-- SELECTION/CROSSOVER OPERATOR
    WHILE @OneByOnePracticalSubject <= @FitnessValueAllPracticalClass
    BEGIN
        DECLARE @SubjectSessionTitle nvarchar(350) = (SELECT TOP 1 SessionTitle FROM @PracticalRandomSubjectTable WHERE RowNo = @OneByOnePracticalSubject);

        DECLARE @SubjectSessionID INT = (SELECT TOP 1 SessionId FROM @PracticalRandomSubjectTable WHERE RowNo = @OneByOnePracticalSubject);
        SET @TimeTableTitle = (SELECT Title FROM @AllSemesters WHERE RowNo = @OneByOneSemester);-- + ' - '+@SessionTitle;

        DECLARE @LabDProgramSemesterID int = (SELECT TOP 1 ProgramSemesterId FROM @PracticalRandomSubjectTable WHERE RowNo = @OneByOnePracticalSubject);

        DECLARE @LabSemesterTitle nvarchar(150) = (SELECT TOP 1 Title FROM @PracticalRandomSubjectTable WHERE RowNo = @OneByOnePracticalSubject);

        DECLARE @LabTimeTableID int = (SELECT TOP 1 TimeTableId FROM @TimeTableHeader WHERE ProgramSemesterId = @LabDProgramSemesterID
        AND SemesterTitle = @LabSemesterTitle)

        DECLARE @LabProgramSemesterSubjectID int = (SELECT TOP 1 ProgramSemesterSubjectId FROM @PracticalRandomSubjectTable WHERE RowNo = @OneByOnePracticalSubject);

        DECLARE @LabSubjectTitle nvarchar(300)=  (SELECT TOP 1 Title + ' ' + SemesterSubjectTitle FROM @PracticalRandomSubjectTable
        WHERE RowNo = @OneByOnePracticalSubject);

        DECLARE @LabRoomTypeID int  =  (SELECT TOP 1 RoomTypeId FROM @PracticalRandomSubjectTable WHERE RowNo = @OneByOnePracticalSubject);

        DECLARE @LabLecturerID int = (SELECT TOP 1 LecturerId FROM @PracticalRandomSubjectTable WHERE RowNo = @OneByOnePracticalSubject);
```

2. Selects a random time slot for the practical subject from the @**ALLLabsSlots** table variable using the TOP 1 clause and the ORDER BY NEWID() statement.

```
DECLARE @LabID int = 0;
DECLARE @LabRoomID int = 0;
DECLARE @LabDayID int = 0;
DECLARE @LabDayTimeSlotID int  = 0;
DECLARE @LabIsActive bit = 1;

    SET @LabID  = (SELECT TOP 1 LabId FROM @ALLLabsSlots WHERE RowNo = @LabTimeSlotNo);

    SET @LabDayTimeSlotID = (SELECT TOP 1 DayTimeSlotId FROM @ALLLabsSlots WHERE RowNo = @LabTimeSlotNo);

    SET @LabDayID = (SELECT TOP 1 DayId FROM @ALLLabsSlots WHERE RowNo = @LabTimeSlotNo);

    IF @LabBreakDurationNo = @LabBreak
        BEGIN
            SET @LabTimeSlotNo = @LabTimeSlotNo + 2;
            SET @LabBreakDurationNo = 1;
        END
        ELSE
        BEGIN
            SET @LabTimeSlotNo = @LabTimeSlotNo + 1;
            SET @LabBreakDurationNo = @LabBreakDurationNo + 1;
        END
```

3. Inserts the practical subject into the @**TimeTableDetails** table variable with the corresponding details, including the time slot.

```
INSERT INTO @TimeTableDetails(TimeTableId,SessionTitle,SessionId, ProgramSemesterSubjectId, SubjectTitle, RoomId,
LabId, DayTimeSlotId, LecturerId, DayId, IsActive)

Values(@LabTimeTableID,@SubjectSessionTitle,@SubjectSessionID,@LabProgramSemesterSubjectID,@LabSubjectTitle,0,@LabID,
@LabDayTimeSlotID,@LabLecturerID,@LabDayID,@LabIsActive);

IF @LabBreakDurationNo = 1
BEGIN
 INSERT INTO @TimeTableDetails(TimeTableId,SessionTitle,SessionId, ProgramSemesterSubjectId, SubjectTitle, RoomId, LabId,
                               DayTimeSlotId, LecturerId, DayId, IsActive)
 Values(@LabTimeTableID,'',0,0,'Break',0,@LabID,@LabDayTimeSlotID,0,@LabDayID,@LabIsActive);
END
```

4. Increments the @**OneByOnePracticalSubject** counter.

```
SET @OneByOnePracticalSubject = @OneByOnePracticalSubject + 1;
END
SET @Message = 'All Practical Time Table Details initialize';
```

The code then proceeds to assign the remaining non-practical subjects in a similar manner. It retrieves the subject details, selects a random time slot from the @**ALLRoomsSlots** table variable, and inserts the subject into the @**TimeTableDetails** table variable. After

assigning all subjects for a particular semester, the @**OneByOneSemester** counter is incremented, and the process repeats for the next semester until all semesters are processed.

Finally, the code sets the output message based on the success or failure of the time table generation and returns the message.

Fitness Function

In the context of a genetic algorithm, the fitness function is used to evaluate the quality of each individual (also called a chromosome or candidate solution) within a population. The fitness function helps determine which individuals are more fit or suitable for solving the given problem, and it plays a crucial role in the selection and evolution process of the genetic algorithm.

```
CREATE OR ALTER FUNCTION CalculateFitnessValue()
RETURNS FLOAT
AS
BEGIN
   DECLARE @FitnessValue FLOAT;
   -- Calculate the fitness value based on your criteria
   -- Consider factors such as room conflicts, lecturer conflicts, etc.
   -- Example: Calculate the fitness value based on room conflicts and lecturer conflict
   DECLARE @RoomConflictCount INT;
   DECLARE @LecturerConflictCount INT = 0;

   -- Calculate the count of room conflicts
   SELECT @RoomConflictCount = COUNT(*)
   FROM tbl_TimeTableDetails TD1
   INNER JOIN tbl_TimeTableDetails TD2 ON TD1.DayTimeSlotId =
TD2.DayTimeSlotId
      AND TD1.DayId = TD2.DayId AND TD1.RoomId = TD2.RoomId
       AND TD1.TimeTableId = TD2.TimeTableId AND TD1.ProgramSemesterSubjectId
< > TD2.ProgramSemesterSubjectId;
```

```
    -- Calculate the count of lecturer conflicts
    SELECT @LecturerConflictCount = COUNT(*)
    FROM tbl_TimeTableDetails TD1
    INNER JOIN tbl_TimeTableDetails TD2 ON TD1.DayTimeSlotId =
TD2.DayTimeSlotId
        AND TD1.DayId = TD2.DayId AND TD1.LecturerId = TD2.LecturerId
        AND TD1.TimeTableId = TD2.TimeTableId AND TD1.ProgramSemesterSubjectId
< > TD2.ProgramSemesterSubjectId;

    -- Calculate the fitness value as a weighted sum of different criteria
    SET @FitnessValue = (0.8 * @RoomConflictCount) + (0.6 *@LecturerConflictCount)

    RETURN @FitnessValue;
END;
GO
```

This function calculates a fitness value based on certain criteria, such as room conflicts and lecturer conflicts. The fitness value is returned as a floating-point number.

1. The function is a SQL Server user-defined function.
2. It takes no input parameters and returns a single float value representing the fitness score.
3. It calculates the fitness value based on certain criteria, such as room conflicts and lecturer conflicts.
4. It begins by declaring a variable called @FitnessValue of type FLOAT to hold the calculated fitness value.
5. Two more variables are declared: @RoomConflictCount and @LecturerConflictCount, to store the count of room conflicts and lecturer conflicts, respectively.
6. The count of room conflicts is calculated by joining the tbl_TimeTableDetails table on matching DayTimeSlotId, DayId, RoomId, and TimeTableId but different ProgramSemesterSubjectId values.

7. The count of lecturer conflicts is calculated by joining the tbl_TimeTableDetails table on matching DayTimeSlotId, DayId, LecturerId, and TimeTableId but different ProgramSemesterSubjectId values.
8. The fitness value is calculated as a weighted sum of the room conflict count and the lecturer conflict count, with weights of 0.8 for room conflicts and 0.6 for lecturer conflicts.
9. The calculated fitness value is assigned to the @FitnessValue variable.
10. The function returns the @FitnessValue using the RETURN statement

5.2 Testing

Testing is the process of evaluating and verifying whether the developed software or application works properly or not i.e., whether there is match between the actual results and expected results or not. Testing is carried out during the development of the software.

The primary objective of software testing is to detect and report any discrepancies between the expected and actual behavior of the software. By systematically examining and executing different aspects of the software, testers strive to uncover defects or bugs that may affect its functionality, security, or user experience.

5.2.1 Unit Testing

Unit testing is the part of the testing methodology which includes testing of individual software modules as well as the components that make up the entire software. The purpose is to validate each unit of the software code so that it performs as expected.

Table 5.1 Test case for Register

Test Case ID	Test Case Descriptions	Test Steps	Test Data	Expected Result	Actual Result	Status
TC-R001	Verify the input field with null data.	Click on create account button.	Email: Password: Repeat Password:	Validation message should be displayed" The Email field is required. The Password field is required".	Validation message is displayed" The Email field is required. The Password field is required".	Pass
TC-R002	Verify the input field with valid data.	Enter valid email address and valid password. Click on Create account button.	Email:test@gmail.com Password:Test@1234 Repeat Password:Test@1234	User should be able to create account successfully. Login Dashboard should be visible.	User is able to create account successfully. Login Dashboard is visible.	Pass

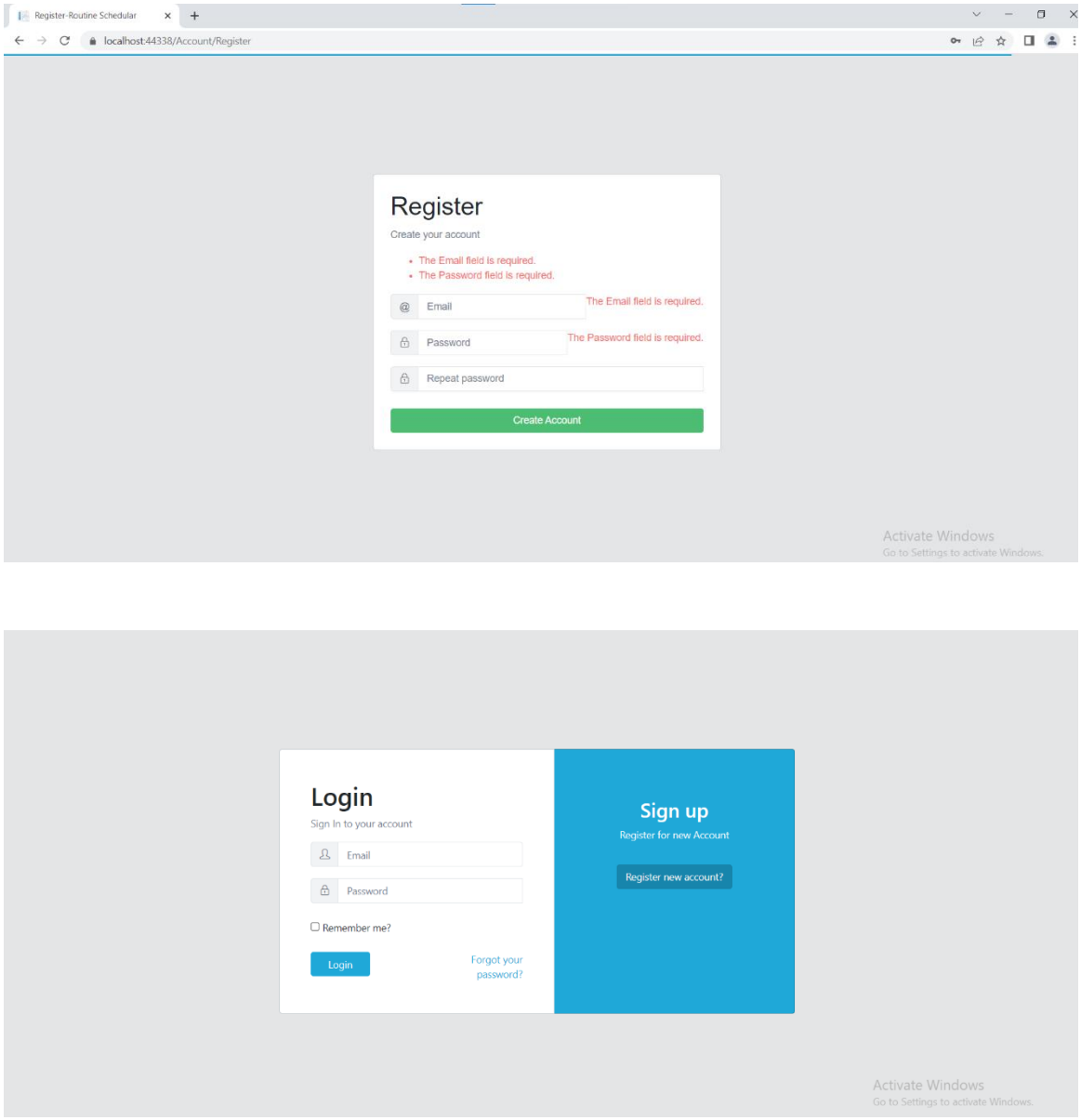
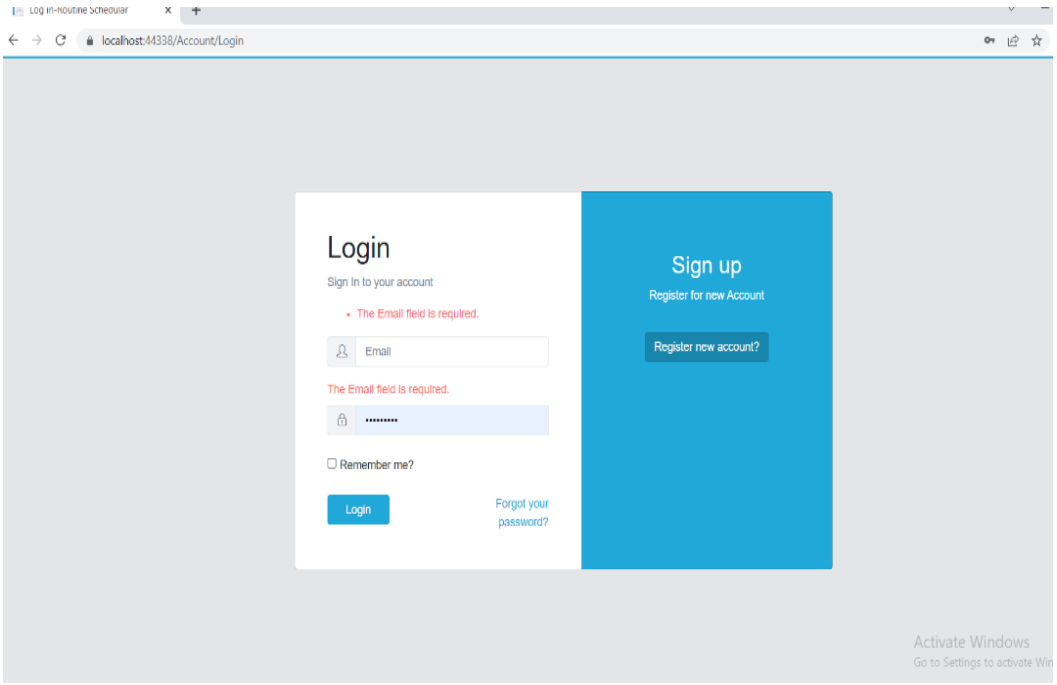


Figure 5.14 Test Case for Register

Table 5.2 Test case for Login

Test Case ID	Test Case Descriptions	Test Steps	Test Data	Expected Result	Actual Result	Status
TC-L001	Verify the input field with null data.	Click on Login Button.	Email: Password:	Validation message should be displayed" The Email field is required. The Password field is required".	Validation message is displayed" The Email field is required. The Password field is required".	Pass
TC-L002	Verify the input field with invalid data.	Enter invalid email address and invalid password. Click on Login button.	Email:Test@gmail.com Password:test@1234	Validation message should be displayed "Invalid Login Attempt".	Validation message is displayed "Invalid Login Attempt".	Pass
TC-L003	Verify the input field with valid data.	Enter valid email address and valid password.	Email:shirishmainali2@gmail.com Password:Test@1234	User should be able to login and dashboard should be displayed.	User is able to login and dashboard is displayed.	Pass

		Click on Login button.				



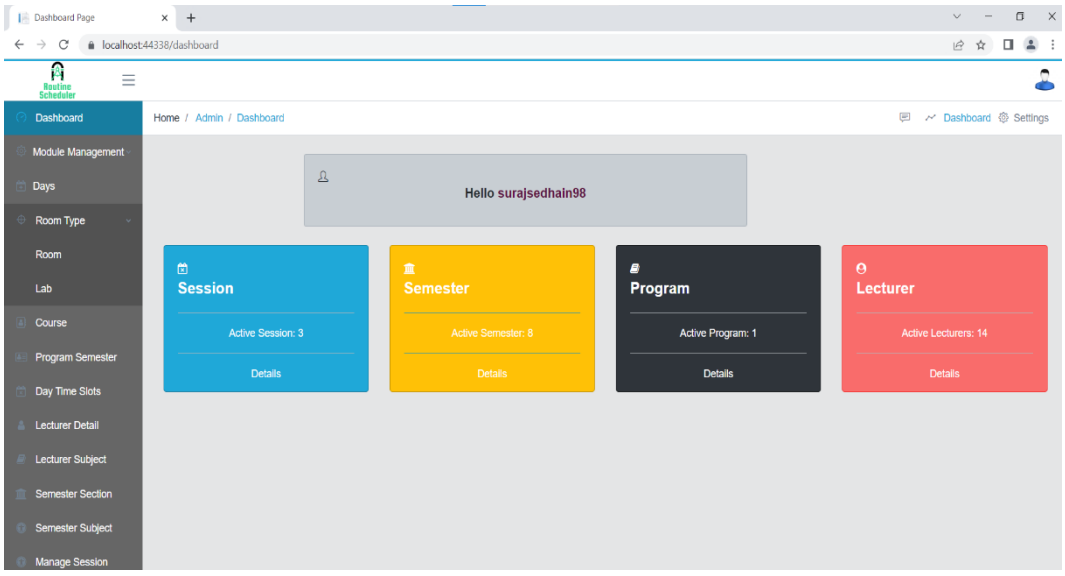


Figure 5.15 Test Case for Login

5.2.2 System Testing

System testing is a process of verifying that a software system meets the specified requirements and works as intended. It evaluates the system as a whole and ensures its correct functioning.

Test Case ID	Test Case Descriptions	Test Steps	Test Data	Expected Result	Actual Result	Status
TC-S001	Verify the functionality of Add Day button.	Click on Add day Button.		Add button should be clicked.	Add button is clicked.	Pass
TC-S002	Verify the Select dropdown.	Click on Select dropdown.		Dropdown should be displayed and day should be selected.	Dropdown is displayed and day is selected.	Pass
TC-S003	Verify when checkbox is checked.	Click on checkbox. Click on Submit button.		Checkbox should be checked.	Checkbox is checked	Pass
TC-S004	Verify functionality of edit button.	Click on edit button.		Edit button should be clicked and be able to edit the day.	Edit button is clicked and is able to edit the day.	Pass
TC-S005	Verify the functionality of Add Room button.	Click on Add Room Button.		Add button should be clicked.	Add button is clicked.	Pass
TC-S006	Verify the input text		Enter room:	Validation message should be displayed	Validation message is	Pass

	field with null data.		Capacity:	“the room number field is required” “The room capacity field is required”.	displayed “the room number field is required” “The room capacity field is required”.	

Table 5.3 Test case for system testing

TC-S007	Verify when checkbox is checked.	Click on checkbox. Click on Submit button.		Checkbox should be checked.	Checkbox is checked	Pass
TC-S008	Verify functionality of Add Lab button.	Click on Add Lab button.	Lab number: Lab 1 Capacity:24	Add lab button should be clicked and be able to add the lab.	Add lab button is clicked and able to add lab.	Pass
TC-S009	Verify the functionality of Register Course button.	Click on Register Course button		Register button should be clicked.	Register button is clicked.	Pass
TC-S010	Verify select dropdown and input text field.	Select the dropdown and enter the subject title.	Enter Subject: Digital Logic Credit hours:3 Room type: Theory	Course should be added.	Course is added.	Pass

Table 5.4 Test Cases for Routing Generate

Test Case ID	Test Case Descriptions	Test Steps	Test Data	Expected Result	Actual Result	Status
TC-S001	Generating Routing (Semester Wise)	First, Select the date that you want to generate report for semester. And click on 'Generate Timetable'. Now click on 'Show Semester Wise' button.		Report should be Generated successfully	Report is Generated successfully	Pass
TC-S002	Generating Routing (Teacher Wise)	Select the date interval and click on 'Generate Timetable'. Now click on 'Show Teacher Wise' button.		Report should be Generated successfully	Report is Generated successfully	Pass

SEMESTER WISE ROUTINE

(2023-06-11-2023-06-11)BSC.CSIT 1ST SEMESTER (SEC A)						
Time	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday
06:00 AM-07:00 AM	Sudan Prajapati (Introduction To Information Technology) \ Room 300	Sudan Prajapati (Introduction To Information Technology) \ Room 400	Sudan Prajapati (Introduction To Information Technology) \ Room 300	Sudan Prajapati (Introduction To Information Technology) \ Room 400	Sudan Prajapati (Introduction To Information Technology) \ Room 300	Sudan Prajapati (Introduction To Information Technology) \ Room 400
07:00 AM-08:00 AM	Sharad Maharjan (C Programming) \ Room 401	Sharad Maharjan (C Programming) \ Room 301	Sharad Maharjan (C Programming) \ Room 401	Sharad Maharjan (C Programming) \ Room 301	Sharad Maharjan (C Programming) \ Room 401	Sharad Maharjan (C Programming) \ Room 301
09:00 AM-10:00 AM	Prakash Shrestha (Mathematics I) \ Room 302	Prakash Shrestha (Mathematics I) \ Room 402	Prakash Shrestha (Mathematics I) \ Room 302	Prakash Shrestha (Mathematics I) \ Room 402	Prakash Shrestha (Mathematics I) \ Room 302	Prakash Shrestha (Mathematics I) \ Room 402
10:00 AM-11:00 AM	Sudesh singh karki(Physics) \ Room 403	Sudesh singh karki(Physics) \ Room 303	Sudesh singh karki(Physics) \ Room 403	Sudesh singh karki(Physics) \ Room 303	Sudesh singh karki(Physics) \ Room 403	Sudesh singh karki(Physics) \ Room 303

Generate Time Table | Back To Dashboard | Export to CSV

Teacher Wise Routine

Search:

SHARAD MAHARJAN						
Time	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday
10:00 AM-11:00 AM	(2023-06-11-2023-06-11)BSC.CSIT 1st Semester (Sec B) Sharad Maharjan (C Programming) \ Room 403	(2023-06-11-2023-06-11)BSC.CSIT 1st Semester (Sec B) Sharad Maharjan (C Programming) \ Room 303	(2023-06-11-2023-06-11)BSC.CSIT 1st Semester (Sec B) Sharad Maharjan (C Programming) \ Room 403	(2023-06-11-2023-06-11)BSC.CSIT 1st Semester (Sec B) Sharad Maharjan (C Programming) \ Room 303	(2023-06-11-2023-06-11)BSC.CSIT 1st Semester (Sec B) Sharad Maharjan (C Programming) \ Room 403	(2023-06-11-2023-06-11)BSC.CSIT 1st Semester (Sec B) Sharad Maharjan (C Programming) \ Room 303
07:00 AM-08:00 AM	(2023-06-11-2023-06-11)BSC.CSIT 1st Semester (Sec A) Sharad Maharjan (C Programming) \ Room 401	(2023-06-11-2023-06-11)BSC.CSIT 1st Semester (Sec A) Sharad Maharjan (C Programming) \ Room 301	(2023-06-11-2023-06-11)BSC.CSIT 1st Semester (Sec A) Sharad Maharjan (C Programming) \ Room 401	(2023-06-11-2023-06-11)BSC.CSIT 1st Semester (Sec A) Sharad Maharjan (C Programming) \ Room 301	(2023-06-11-2023-06-11)BSC.CSIT 1st Semester (Sec A) Sharad Maharjan (C Programming) \ Room 401	(2023-06-11-2023-06-11)BSC.CSIT 1st Semester (Sec A) Sharad Maharjan (C Programming) \ Room 301

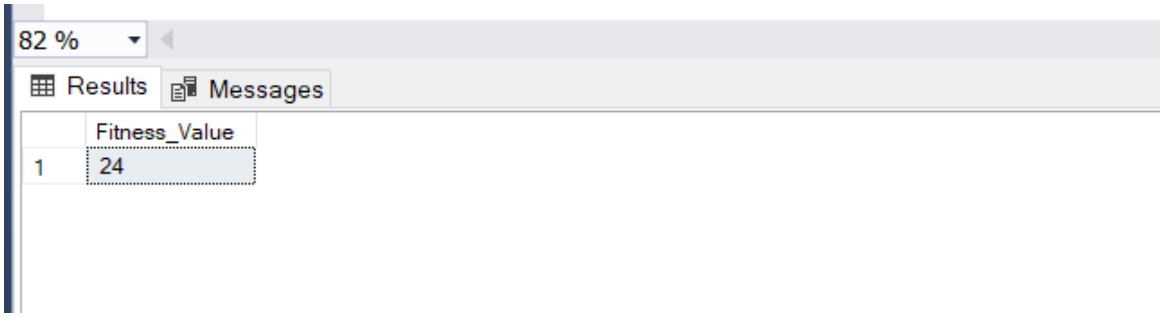
Figure 5.16 System Testing

5.3. Result Analysis

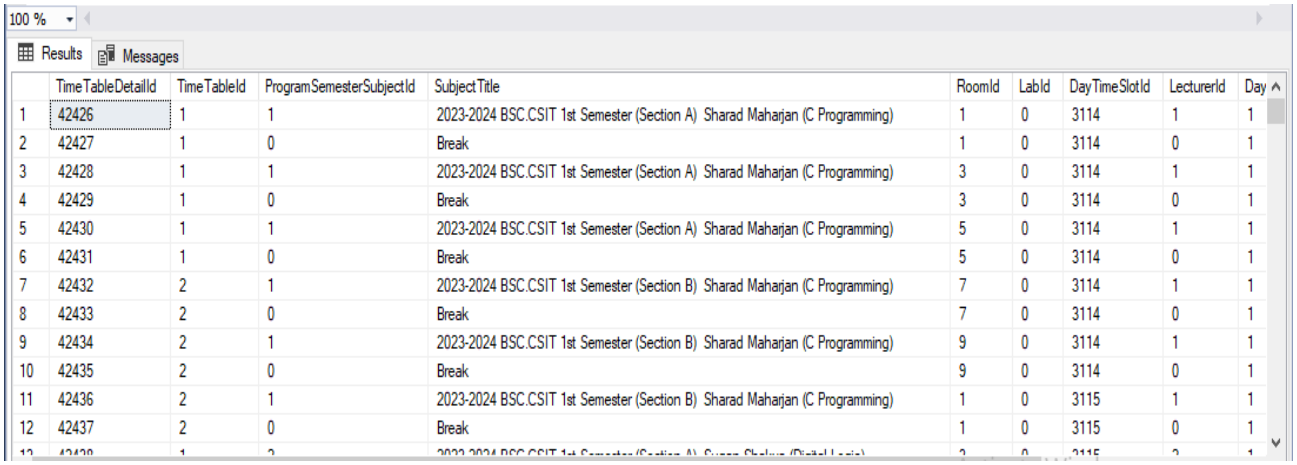
During the demonstration of project, the success in generating routine schedules for both semesters and teachers is shown. System was tested, and the results confirmed that the system was able to perform its functions as intended. While the project was able to meet its goals, there is still room for improvement where system will be provided with the features to enhance the system.

Fitness Value

Calculated fitness value of system is



Output of the system



	TimeTableDetailId	TimeTableId	ProgramSemesterSubjectId	SubjectTitle	RoomId	LabId	DayTimeSlotId	LecturerId	Day
1	42426	1	1	2023-2024 BSC.CSIT 1st Semester (Section A) Sharad Maharjan (C Programming)	1	0	3114	1	1
2	42427	1	0	Break	1	0	3114	0	1
3	42428	1	1	2023-2024 BSC.CSIT 1st Semester (Section A) Sharad Maharjan (C Programming)	3	0	3114	1	1
4	42429	1	0	Break	3	0	3114	0	1
5	42430	1	1	2023-2024 BSC.CSIT 1st Semester (Section A) Sharad Maharjan (C Programming)	5	0	3114	1	1
6	42431	1	0	Break	5	0	3114	0	1
7	42432	2	1	2023-2024 BSC.CSIT 1st Semester (Section B) Sharad Maharjan (C Programming)	7	0	3114	1	1
8	42433	2	0	Break	7	0	3114	0	1
9	42434	2	1	2023-2024 BSC.CSIT 1st Semester (Section B) Sharad Maharjan (C Programming)	9	0	3114	1	1
10	42435	2	0	Break	9	0	3114	0	1
11	42436	2	1	2023-2024 BSC.CSIT 1st Semester (Section B) Sharad Maharjan (C Programming)	1	0	3115	1	1
12	42437	2	0	Break	1	0	3115	0	1

CHAPTER 6
CONCLUSION AND FUTURE RECOMMENDATIONS

6.1. Conclusion

A routine scheduler is a Web Application designed to automate the creation of schedules or routines for Colleges. The software is designed to generate a schedule that meets specific requirements such as time slots, teacher availability, and class distribution. In conclusion, this project has demonstrated that routine creation can be generated with an optimal solution, which has the potential to save time and improve efficiency. The system can generate a routine based on the data inserted, such as the semester and teacher preferences.

In conclusion, a routine scheduler is a powerful tool that can save time and improve efficiency in many colleges. The software is flexible, adaptable, and can quickly generate a new schedule if there are any changes to the input data. The routine scheduler reduces the chances of errors and optimizes the schedule to avoid conflicts and other inefficiencies.

6.2. Future Recommendations

Future recommendations for this project could include expanding the system's functionality to include exam routine creation for colleges. This would involve gathering data on the dates and times of exams for each course, and using that data to generate an optimal exam routine.

Overall, these enhancements would make the system even more useful for educational institutions, providing a comprehensive solution for routine creation and scheduling.

It is possible to develop a mobile application for the system. Additionally, the system can be enhanced to effectively schedule routine for part-time teachers.

REFERENCES

[1] [Online]. (2022) *gizmoa.com*. Available at: https://gizmoa.com/college-schedule-maker/ [Accessed: 2023].

[2] [Online]. Available: https://timetablemaker.com/ [Accessed:2023].

[3] [Online].Available: https://teachworks.com/features/calendar?fbclid=IwAR2_GQyc9d1ex0iOnFQCSlsbtp2S6xCARaJKX-WtNwIThYdKMge-yVpU_xA [Accessed Dec 2022].

[4] M. Doulaty, M. R. FeiziDerakhshi, and M. Abdi, "Timetabling: A State-of-the-Art Evolutionary Approach", International Journal of Machine Learning and Computing,Vol. 3, No. 3, June 2013.

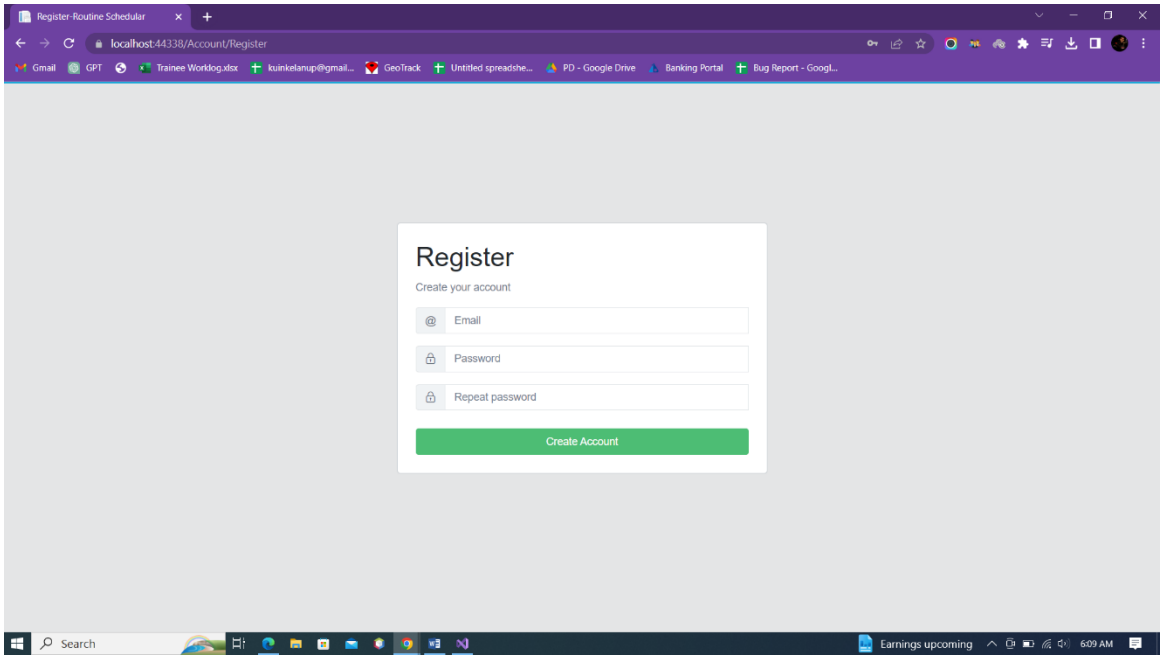
[5] Cite Seerx , Optimisation of Active Rule Agents using a Genetic Algorithm approach, Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.56.2599

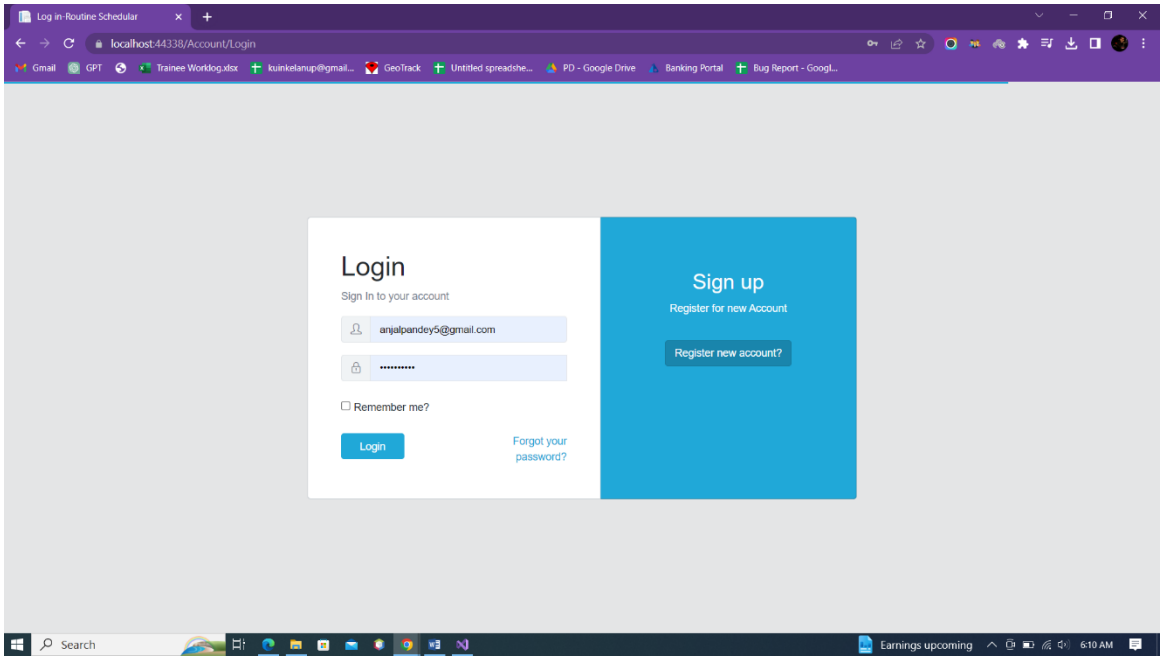
[6] Rawat, B. *et al.* (2022) *A Comparative Review Between Various Selection Techniques In Genetic Algorithm For Finding Optimal Solutions* [Preprint]. Available at: https://www.ijcseonline.org/pdf_paper_view.php?paper_id=5522&3-IJCSE-08999.pdf (Accessed: 2023).

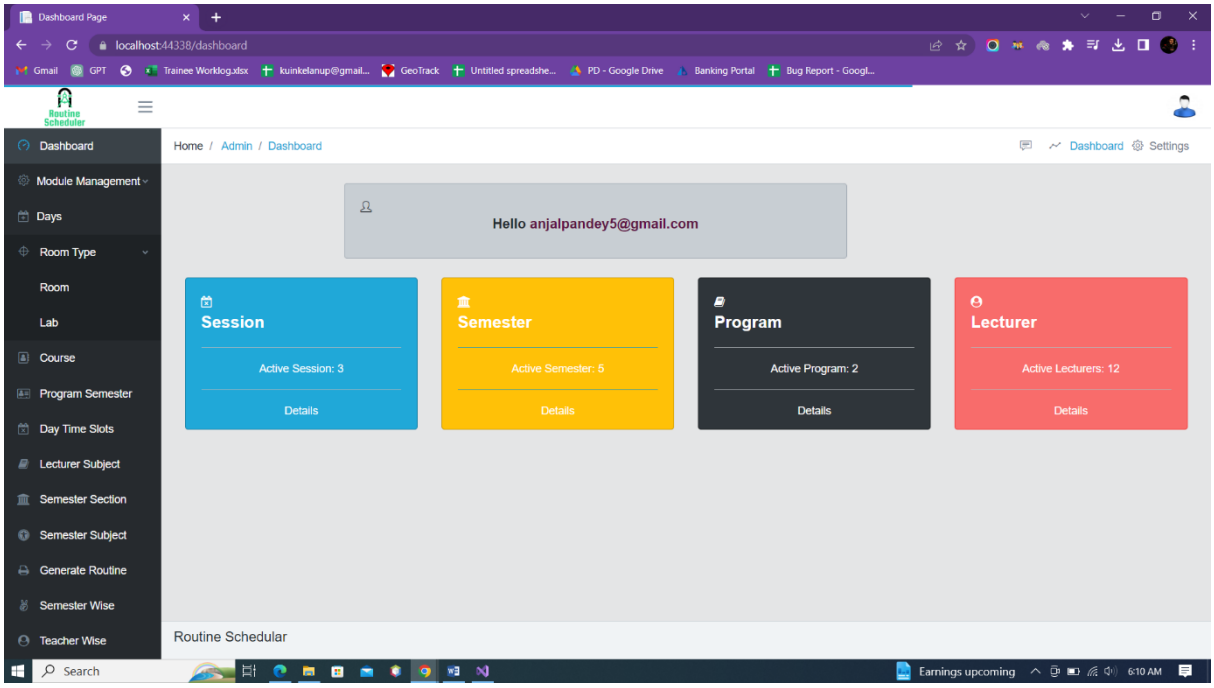
[7] Anuja Chowdhary, Priyanka Kakde, Shruti Dhoke, Sonali Ingle,Rupal Rushiya, Dinesh Gawande, 'Time table Generation System‖, International Journal of Computer Science and Mobile Computing, Vol.3 Issue.2, February- 2014.

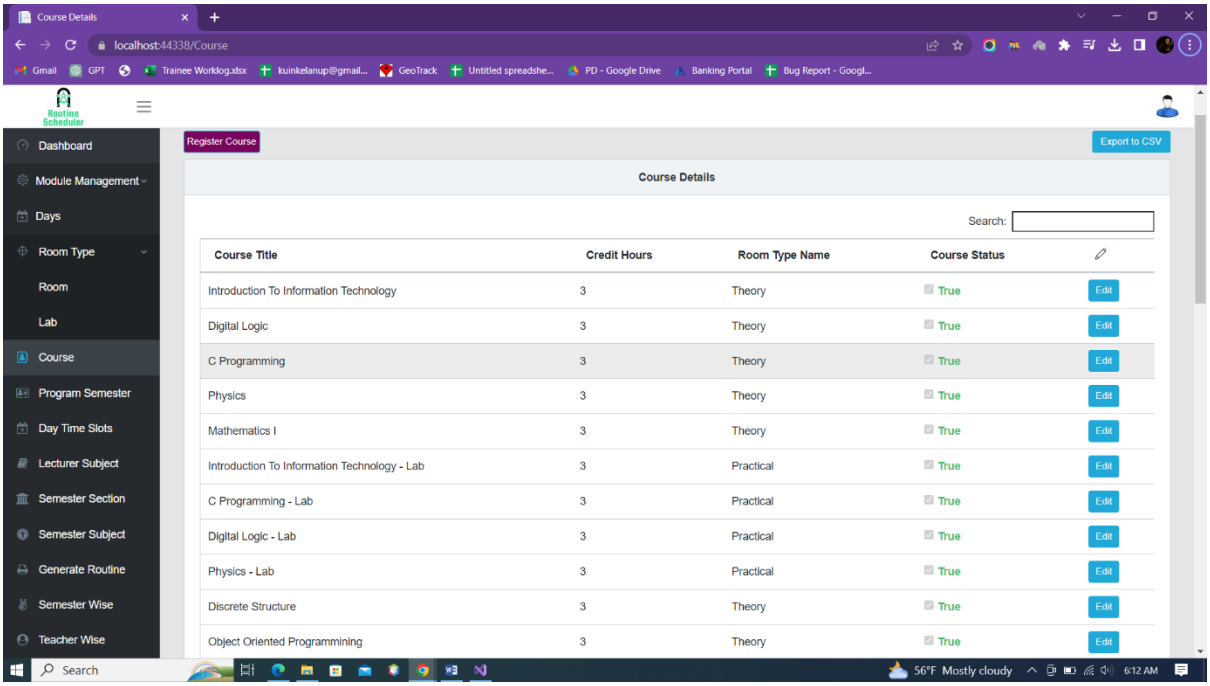
APPENDICES

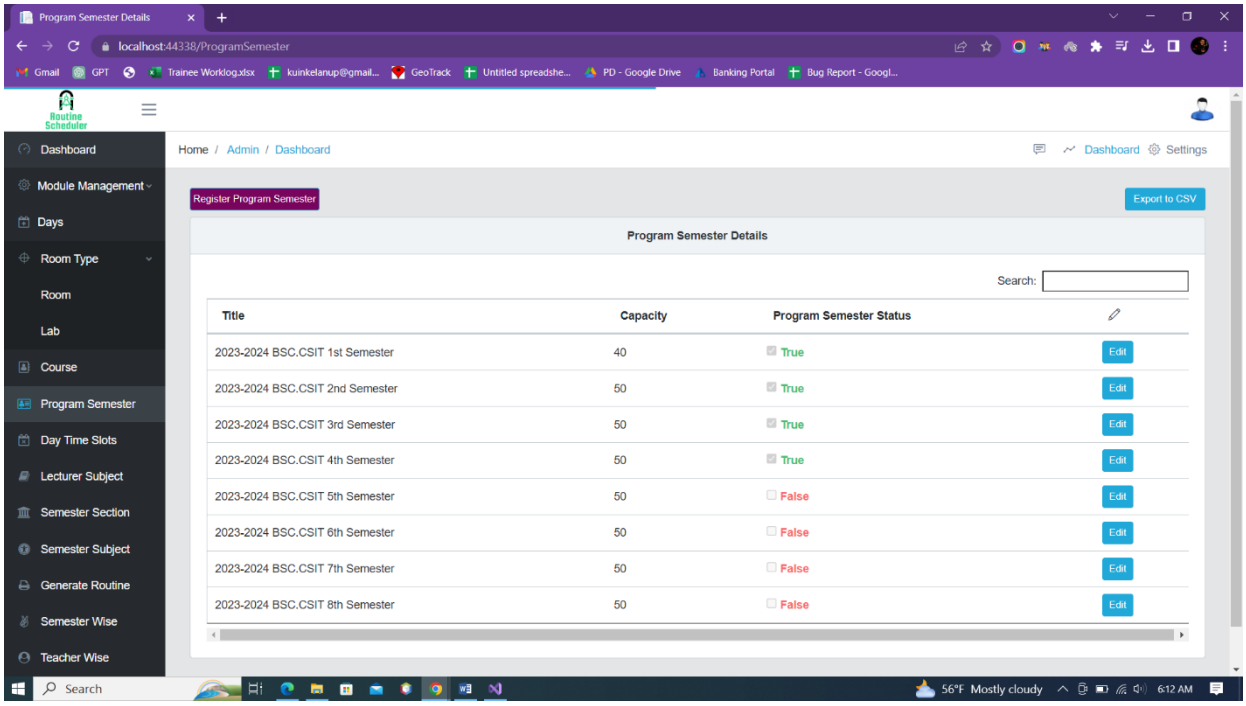
Appendix-1

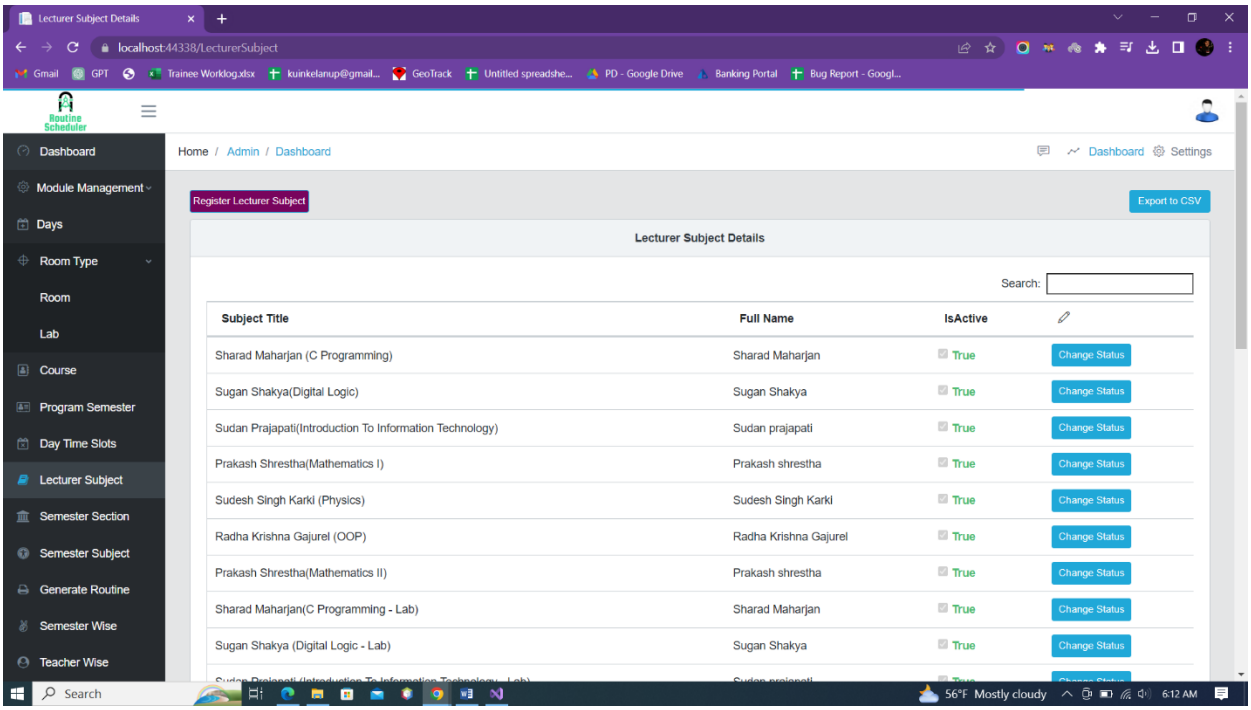


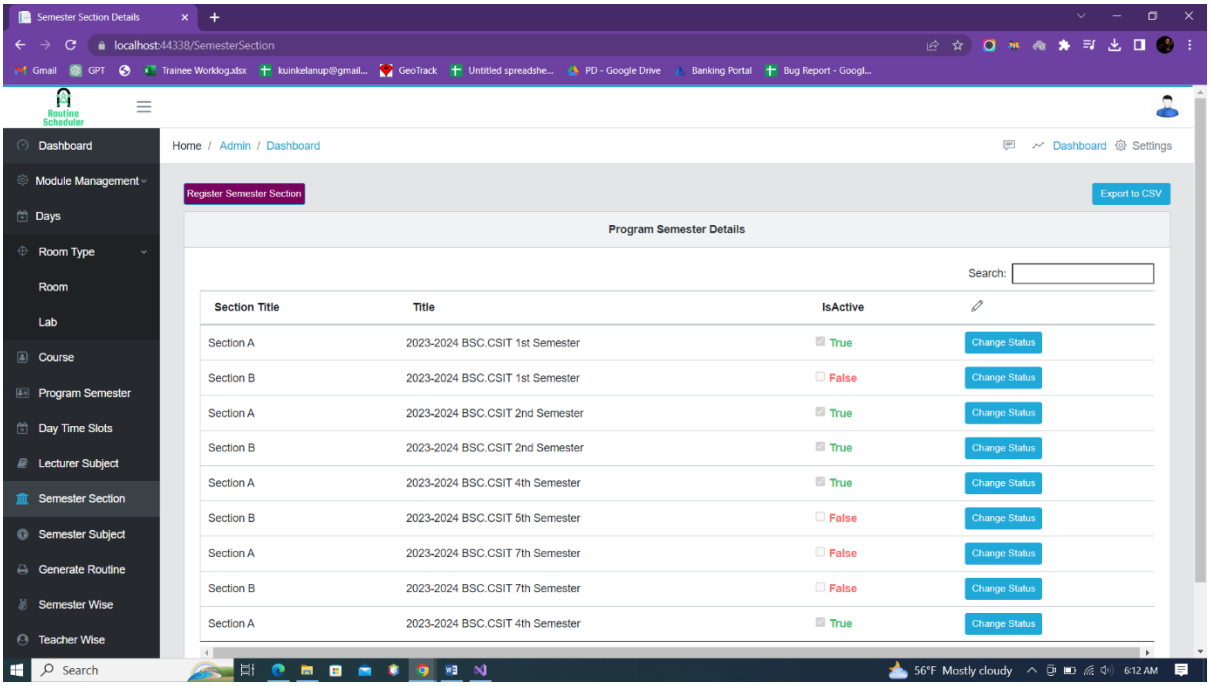


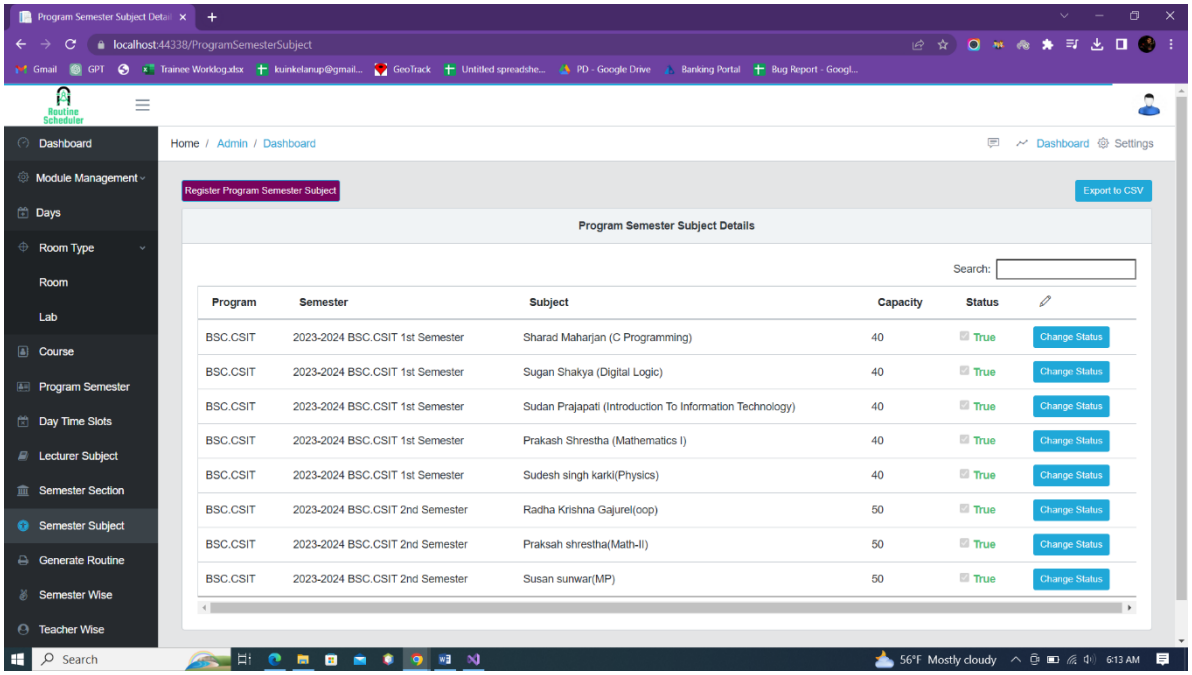

Dashboard Page
localhost:44338/dashboard
Routine Scheduler
Dashboard
Module Management
Days
Room Type
Room
Lab
Course
Program Semester
Day Time Slots
Lecturer Subject
Semester Section
Semester Subject
Generate Routine
Semester Wise
Teacher Wise
Home / Admin / Dashboard
Dashboard
Settings
Hello anjalpandey5@gmail.com
Session
Active Session: 3
Details
Semester
Active Semester: 5
Details
Program
Active Program: 2
Details
Lecturer
Active Lecturers: 12
Details
Routine Schedular
Search
Earnings upcoming
6:10 AM

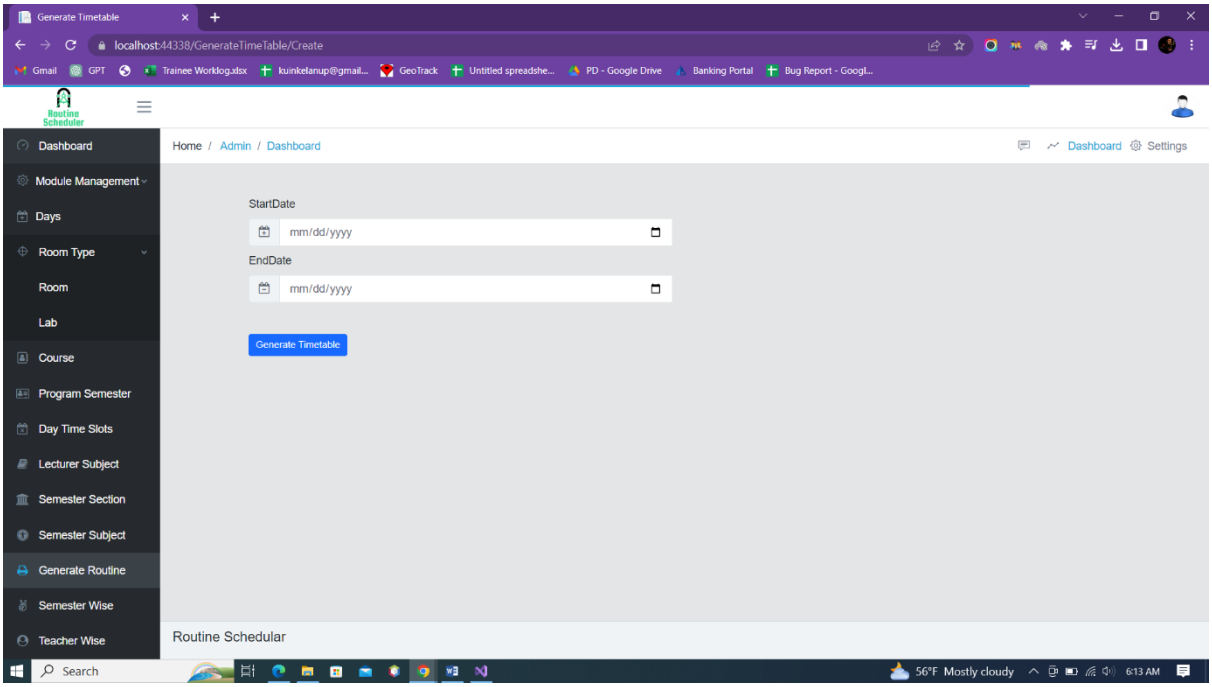

Course Details
localhost:44338/Course
Routine Scheduler
Dashboard
Module Management
Days
Room Type
Room
Lab
Course
Program Semester
Day Time Slots
Lecturer Subject
Semester Section
Semester Subject
Generate Routine
Semester Wise
Teacher Wise
Register Course
Export to CSV
Course Details
Search:
Course Title
Credit Hours
Room Type Name
Course Status
Introduction To Information Technology 3 Theory True Edit
Digital Logic 3 Theory True Edit
C Programming 3 Theory True Edit
Physics 3 Theory True Edit
Mathematics I 3 Theory True Edit
Introduction To Information Technology - Lab 3 Practical True Edit
C Programming - Lab 3 Practical True Edit
Digital Logic - Lab 3 Practical True Edit
Physics - Lab 3 Practical True Edit
Discrete Structure 3 Theory True Edit
Object Oriented Programming 3 Theory True Edit
Search
56°F Mostly cloudy
6:12 AM

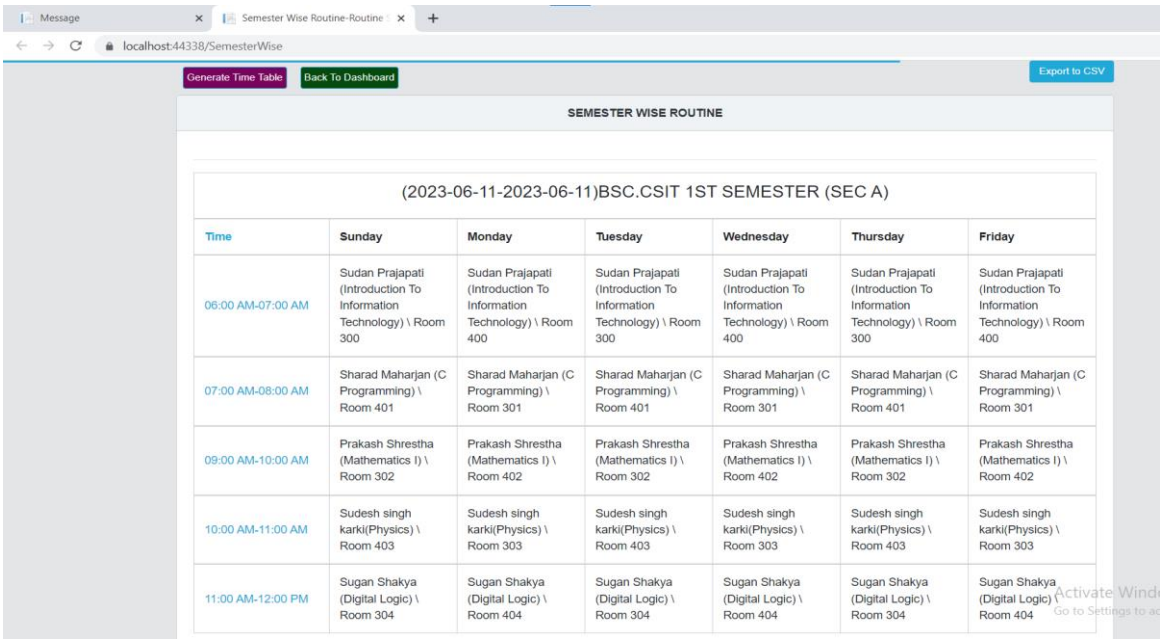

Program Semester Details
localhost:44338/ProgramSemester
Routine Scheduler
Dashboard
Module Management
Days
Room Type
Room
Lab
Course
Program Semester
Day Time Slots
Lecturer Subject
Semester Section
Semester Subject
Generate Routine
Semester Wise
Teacher Wise
Home / Admin / Dashboard
Dashboard Settings
Register Program Semester
Export to CSV
Program Semester Details
Search:
Title
Capacity
Program Semester Status
2023-2024 BSC.CSIT 1st Semester 40 True Edit
2023-2024 BSC.CSIT 2nd Semester 50 True Edit
2023-2024 BSC.CSIT 3rd Semester 50 True Edit
2023-2024 BSC.CSIT 4th Semester 50 True Edit
2023-2024 BSC.CSIT 5th Semester 50 False Edit
2023-2024 BSC.CSIT 6th Semester 50 False Edit
2023-2024 BSC.CSIT 7th Semester 50 False Edit
2023-2024 BSC.CSIT 8th Semester 50 False Edit
Search
56°F Mostly cloudy
6:12 AM


Lecturer Subject Details
localhost:44338/LecturerSubject
Routine Scheduler
Dashboard
Module Management
Days
Room Type
Room
Lab
Course
Program Semester
Day Time Slots
Lecturer Subject
Semester Section
Semester Subject
Generate Routine
Semester Wise
Teacher Wise
Home / Admin / Dashboard
Dashboard Settings
Register Lecturer Subject
Export to CSV
Lecturer Subject Details
Search:
Subject Title
Full Name
IsActive
Sharad Maharjan (C Programming) Sharad Maharjan True Change Status
Sugan Shakya(Digital Logic) Sugan Shakya True Change Status
Sudan Prajapati(Introduction To Information Technology) Sudan prajapati True Change Status
Prakash Shrestha(Mathematics I) Prakash shrestha True Change Status
Sudesh Singh Karki (Physics) Sudesh Singh Karki True Change Status
Radha Krishna Gajurel (OOP) Radha Krishna Gajurel True Change Status
Prakash Shrestha(Mathematics II) Prakash shrestha True Change Status
Sharad Maharjan(C Programming - Lab) Sharad Maharjan True Change Status
Sugan Shakya (Digital Logic - Lab) Sugan Shakya True Change Status
Search
56°F Mostly cloudy
6:12 AM


Semester Section Details
localhost:44338/SemesterSection
Routine Scheduler
Dashboard
Module Management
Days
Room Type
Room
Lab
Course
Program Semester
Day Time Slots
Lecturer Subject
Semester Section
Semester Subject
Generate Routine
Semester Wise
Teacher Wise
Home / Admin / Dashboard
Dashboard Settings
Register Semester Section
Export to CSV
Program Semester Details
Search:
Section Title
Title
IsActive
Section A
2023-2024 BSC.CSIT 1st Semester
True
Change Status
Section B
2023-2024 BSC.CSIT 1st Semester
False
Change Status
Section A
2023-2024 BSC.CSIT 2nd Semester
True
Change Status
Section B
2023-2024 BSC.CSIT 2nd Semester
True
Change Status
Section A
2023-2024 BSC.CSIT 4th Semester
True
Change Status
Section B
2023-2024 BSC.CSIT 5th Semester
False
Change Status
Section A
2023-2024 BSC.CSIT 7th Semester
False
Change Status
Section B
2023-2024 BSC.CSIT 7th Semester
False
Change Status
Section A
2023-2024 BSC.CSIT 4th Semester
True
Change Status
Search
56°F Mostly cloudy
6:12 AM


localhost:44338/ProgramSemesterSubject
Routine Scheduler
Dashboard
Module Management
Days
Room Type
Room
Lab
Course
Program Semester
Day Time Slots
Lecturer Subject
Semester Section
Semester Subject
Generate Routine
Semester Wise
Teacher Wise
Home / Admin / Dashboard
Dashboard Settings
Register Program Semester Subject
Export to CSV
Program Semester Subject Details
Search:
Program
Semester
Subject
Capacity
Status
BSC.CSIT
2023-2024 BSC.CSIT 1st Semester
Sharad Maharjan (C Programming)
40
True
Change Status
BSC.CSIT
2023-2024 BSC.CSIT 1st Semester
Sugan Shakya (Digital Logic)
40
True
Change Status
BSC.CSIT
2023-2024 BSC.CSIT 1st Semester
Sudan Prajapati (Introduction To Information Technology)
40
True
Change Status
BSC.CSIT
2023-2024 BSC.CSIT 1st Semester
Prakash Shrestha (Mathematics I)
40
True
Change Status
BSC.CSIT
2023-2024 BSC.CSIT 1st Semester
Sudesh singh karki(Physics)
40
True
Change Status
BSC.CSIT
2023-2024 BSC.CSIT 2nd Semester
Radha Krishna Gajurel(oop)
50
True
Change Status
BSC.CSIT
2023-2024 BSC.CSIT 2nd Semester
Praksah shrestha(Math-II)
50
True
Change Status
BSC.CSIT
2023-2024 BSC.CSIT 2nd Semester
Susan sunwar(MP)
50
True
Change Status
Search
56°F Mostly cloudy
6:13 AM





(2023-06-11-2023-06-11)BSC.CSIT 1ST SEMESTER (SEC A)

Time	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday
06:00 AM-07:00 AM	Sudan Prajapati (Introduction To Information Technology) \ Room 300	Sudan Prajapati (Introduction To Information Technology) \ Room 400	Sudan Prajapati (Introduction To Information Technology) \ Room 300	Sudan Prajapati (Introduction To Information Technology) \ Room 400	Sudan Prajapati (Introduction To Information Technology) \ Room 300	Sudan Prajapati (Introduction To Information Technology) \ Room 400
07:00 AM-08:00 AM	Sharad Maharjan (C Programming) \ Room 401	Sharad Maharjan (C Programming) \ Room 301	Sharad Maharjan (C Programming) \ Room 401	Sharad Maharjan (C Programming) \ Room 301	Sharad Maharjan (C Programming) \ Room 401	Sharad Maharjan (C Programming) \ Room 301
09:00 AM-10:00 AM	Prakash Shrestha (Mathematics I) \ Room 302	Prakash Shrestha (Mathematics I) \ Room 402	Prakash Shrestha (Mathematics I) \ Room 302	Prakash Shrestha (Mathematics I) \ Room 402	Prakash Shrestha (Mathematics I) \ Room 302	Prakash Shrestha (Mathematics I) \ Room 402
10:00 AM-11:00 AM	Sudesh singh karki(Physics) \ Room 403	Sudesh singh karki(Physics) \ Room 303	Sudesh singh karki(Physics) \ Room 403	Sudesh singh karki(Physics) \ Room 303	Sudesh singh karki(Physics) \ Room 403	Sudesh singh karki(Physics) \ Room 303
11:00 AM-12:00 PM	Sugan Shakya (Digital Logic) \ Room 304	Sugan Shakya (Digital Logic) \ Room 404	Sugan Shakya (Digital Logic) \ Room 304	Sugan Shakya (Digital Logic) \ Room 404	Sugan Shakya (Digital Logic) \ Room 304	Sugan Shakya (Digital Logic) \ Room 404