Helwan University
Faculty of engineering
Computer Department

Student Information System (SIS)

By

Ahmed Gamal Abu al-Hasan Attia
Ahmed Waleed Mohamed Mahmoud
Esraa Abd El-Maksoud Ayman
Hussein Medhat Farouk Ismail
Merna Ashraf Mohamed Nabawy

B.Sc. Graduation Project Submitted to the
Faculty of Engineering at Helwan University
In Fulfilment of the Requirements for the Degree of
BACHELOR OF SCIENCE
In
Computer Engineering

Under the Supervision of
Prof. Dr. Hesham Keshk

Helwan University
Faculty of engineering
Computer Department

Student Information System (SIS)

By

Ahmed Gamal Abu al-Hasan Attia
Ahmed Waleed Mohamed Mahmoud
Esraa Abd El-Maksoud Ayman
Hussein Medhat Farouk Ismail
Merna Ashraf Mohamed Nabawy

Under the Supervision of: Prof. Dr. Hesham Keshk

Acknowledgment

We want to express our sincere gratitude to our supervisor, Professor Hesham Keshk, for his support, insightful comments, helpful information, practical advice, and unceasing ideas that have always helped us tremendously in our research.

We are also grateful to all the Teaching staff in the Computer Engineering Dep., for their consistent support and assistance. It was great sharing premises with all of you during the last five years.

Abstract

The Student Information System (SIS) is an online platform designed to streamline the course registration process in universities and colleges. By providing a solution, the SIS project aims to enhance efficiency, accuracy, and user experience associated with scheduling, registration, and administrative tasks.

Students benefit from the SIS system through its user-friendly interface, which simplifies the course registration process. The platform enables students to browse available courses, access information such as basic requirements, and enroll in their preferred courses smoothly. The SIS system also provides students with tailored lecture schedules and exam schedules, facilitating effective management of the course burden and study time.

Professors have experienced increased management efficiency with the implementation of the SIS platform. This implementation ensures that teachers have accurate and up-to-date information about their students, allowing them to focus more on teaching and academic activities.

Administrators gain significant advantages through the Student Information System (SIS), thanks to the management toolkit available. These tools simplify course admission management, generate course schedules, and process student records. Officials can efficiently process students' applications. By automating the scheduling process, SIS improves resource allocation and facilitates the balanced distribution of classes throughout the week.

Table of contents

Table of figures

Chapter 1

Introduction

1.1 Introduction

In today's digital age, the efficient management of student information and academic processes is crucial for universities and colleges. The traditional methods of manual paperwork and cumbersome processes often result in delays, errors, and frustration for both students and administrative staff. Recognizing the need for a streamlined and user-friendly system, the Student Information System (SIS) project was initiated.

The SIS project is an innovative web-based platform developed to modernize and optimize the course registration process, offering a solution tailored to the needs of universities and colleges. This platform aims to simplify and enhance the workflow associated with scheduling and registration, catering to the needs of students, professors, and administrators alike.

For students, the SIS system offers a user-friendly interface that facilitates seamless course registration. With just a few clicks, students can browse through available courses, select their desired ones, and secure their spots in classes. The system provides real-time updates on course availability, allowing students to make informed decisions and avoid scheduling conflicts. Furthermore, students can access their personalized schedules and view exam timetables, enabling them to effectively manage their academic commitments.

The SIS system also provides professors with lists of students who have attended their own courses. Professors can choose the right time for them to teach their courses and enter the results for all students easily and review them before sending it to administrators. This simplified approach allows professors to focus on teaching and academic activities, rather than dealing with administrative burdens.

Administrators also benefit greatly from the SIS platform. The system provides advanced tools for the development of lecture schedules, considering different factors such as the availability of the chapter, the availability of the professor and their preferences. By automating the scheduling process, universities can improve resource allocation, reduce scheduling conflicts, and ensure a balanced distribution of classrooms throughout the week. In addition, the SIS system provides effective course acceptance management, allowing officials to process student applications, review eligibility criteria, and allocate seats based on available core resources and conditions. These posts reduce administrative workload, enhance transparency, and ensure a fair and streamlined acceptance process. It also provides a management of students' grades and the possibility of reviewing them before announcing them and tracking the progress of students.

The SIS project aims to improve the overall efficiency and effectiveness of the course registration process in universities and colleges. By implementing this web-based platform, educational institutions can eliminate manual paperwork, reduce errors, and provide a seamless experience for students, professors, and administrators. The SIS system centralizes student information, streamlines processes, and promotes effective communication, ultimately creating an enriched learning environment for all stakeholders involved.

1.2 Problem Statements

- Problem Statement 1:

  Traditional subject registration processes in college are time-consuming and burdensome, leading to an increasing workload.

In many colleges, the process of subject registration for students is still done using traditional methods, such as filling out paper forms or manually selecting courses. This approach often involves long queues, paperwork, and administrative tasks that consume a significant amount of time and resources. As a result, both students and staff members experience an increasing workload, which can lead to inefficiencies and delays in the registration process.

The time-consuming nature of traditional subject registration processes can have several negative consequences. First, it can be frustrating and stressful for students. Students may have to wait in long lines, struggle to get the courses they need, or face difficulties in coordinating their schedules. This can result in delayed graduation, wasted time, and increased stress levels.

Additionally, the administrative burden on college staff members increases with the manual processing of registration forms and managing student requests and inquiries. This can lead to errors, miscommunications, and delays in processing registrations, further adding to the workload of staff members, and hindering the efficiency of the overall registration system.

- Problem Statement 2:

  Existing timetable generator software often fails to meet the required specifications, and the source code is not editable due to its complexity and lack of clean coding practices, leading to an inefficient process that requires significant time and effort.

  Generating a timetable that accommodates the scheduling needs and preferences of students and faculty members can be a complex task. Many colleges rely on timetable generator software to automate this process. However, existing software solutions

often fall short in meeting the required specifications and can be challenging to modify or customize due to their complex and poorly structured source code.

When the timetable generator software fails to meet the required specifications, it can result in a range of issues. For example, the generated timetable may have clashes between classes, inadequate time slots for certain courses, or scheduling conflicts for faculty members. These issues can disrupt the smooth functioning of the college, impact students' ability to enroll in their desired courses, and create logistical challenges for faculty members.

The lack of editable source code further compounds the problem. If the software does not adhere to clean coding practices or lacks proper documentation, it becomes difficult for developers or administrators to modify or enhance the software to suit specific institutional requirements. This limitation restricts the ability to improve the software's functionality, adapt it to changing needs, or integrate it with other systems used by the college.

1.3 Objectives

Objective 1: Improve subject registration processes in college to reduce the time and workload for both students and staff members.

- Develop an online registration system that allows students to select and enroll in courses conveniently and efficiently.
- Implement automation and digital tools to streamline the registration process and minimize manual administrative tasks.
- Enhance the user experience by providing a user-friendly interface, clear instructions, and real-time availability updates for course selection.

- Reduce waiting times and queues by optimizing the registration system's performance and capacity.

Objective 2: Enhance the efficiency and customization capabilities of timetable generator software for better scheduling in colleges.

- Develop timetable generator software with editable source code, enabling customization and adaptation to specific institutional requirements.
- Improve the existing timetable generator algorithms to optimize the scheduling process and reduce conflicts and clashes between classes.
- Incorporate clean coding practices and proper documentation to ensure the software's maintainability and ease of future enhancements.

By achieving these objectives, colleges can streamline their subject registration processes, reduce the workload on students and staff, and enhance the overall efficiency and effectiveness of the scheduling system. This would ultimately contribute to a better educational experience for students and a more streamlined administrative process for the college.

1.4 Literature review

Extensive research has been performed in the field of automated timetable construction. The problem was first formulated in 1965 [1] and was proven to be NP-complete in 1995 [2].

- FET (free open source) : FET is an open-source free timetabling software in C++ that is developed by Liviu Lalescu and Volker Dirr [3]. The first release of the current algorithm was on November 5, 2009, and is still updated frequently. It started in 2002 with a genetic algorithm, but this was computationally slow. For

this reason, they created another algorithm which they called Recursive Swapping FET supports many different languages, such as Arabic, English, and Greek. It also supports many different constraints. The hardness of a constraint can be set between 0% and 100%, where 100% means a hard constraint. Bauteu et Al. [4] did a case study using FET and the results showed that the execution time is low (max 30 seconds in the demonstrated cases) compared to other algorithms. The main factor on the execution time is the (combination of) constraints used.

- Unitime (free) : Unitime is a web-based automated timetabling application with a constraints solver written in Java [5]. It is based on the iterative forward search algorithm. Unitime was developed by staff from several faculties from North America and Europe. The development started in the year 2001. It uses a MySQL database to store. data and works as a server-client package. The constraints are based on penalties, where -4 means strongly preferred and +4 is strongly discouraged. A penalty of 1 represent a constraint that should always be true. The run time of Unitime differs simple timetables take around 10 seconds, where larger and more difficult timetables take up to 15 minutes [6].

Chapter 2

Analysis and requirements

2.1 Analysis

To arrive at the specific requirements of our system, we wrote below, we did some steps to collect ideas and data about the proposed project, which are:

- We did research about some systems serving the same goal as we mentioned in the literature review.
- We compared different ways to make the schedule and chose the best possible way.
- We had a lot of discussions with our supervisor professor about the topic.

2.2 System requirements

2.2.1 Functional requirement

REQ - 1	All students are directed to the login page when using the web application system. Administrators and professors are directed to another login page. From this page, the student can log in or register as a new user. From this page, the admin or the professor can log in.
REQ - 2	Role management means specifying the role of every user in the system. Admin generates a lecture timetable and exam timetable for the semester, Add, Edit, and Delete users. Academic advisor users have access to an interface for Admit/Add the student registration, generating transcripts, or Viewing students' courses.
REQ - 3	Register courses: Students can choose any course to join

REQ - 4	View of the table of lectures: Students can see his exam schedule and lectures.
REQ - 5	View of results and statistics: Students can follow his academic progress
REQ - 6	Upload the result: Professor uploads all the results and send them to the admin and can save and complete the results later
REQ - 7	Admit/Add the student registration: Academic Advisors can accept or change of students' preferences according to his perception Academic Advisors can accept or change of students' preferences according to his perception.
REQ - 8	Generate transcript: Academic Advisors can generate a detailed transcript for each student.
REQ - 9	Academic advisor can view students’ courses.
REQ - 10	The admin can generate lectures or exams timetable using the data saved in the system
REQ - 11	The admin can upload courses for any regulation in the system
REQ - 12	The admin can review the results then edit or admit those results
REQ - 13	The admin can control the registration process (open the registration for students or close it), also he can close some courses for the current registration and re-open them again
REQ - 14	The admin can control all the users in the system

2.2.2 Non-functional requirement

REQ-15	Security is paramount in a student information system, employing robust measures like encryption, authentication, and access control to safeguard sensitive data from unauthorized access or breaches, ensuring the privacy and integrity of student information. Regular security audits and updates help to identify and address vulnerabilities, providing a safe and secure environment for data management and student confidentiality.
REQ-16	Availability ensures that the site runs constantly in all browsers. The web application could be running at every time.
REQ-17	In a student information system ensures that the software is designed and implemented in a way that allows for easy updates, bug fixes, and modifications as needed, ensuring smooth operation and adaptability to changing requirements. By prioritizing maintainability, the system can be efficiently managed, reducing downtime and enhancing overall user experience for students, administrators, and educators.
REQ-18	Queries upon the database shall be performed in less than 5 seconds. The system shall be fast and responsive. The system shall be able to be reused for each new semester.
REQ-19	A user shall be able to log into the system at any instance of time.

2.3 Functional Requirements Specification

2.3.1 System Stakeholders

- Students
- Instructors
- College committee
- College

2.3.2 Actors and goals

- Administrator
    - The administrator will have overall control over the system.
    - He has the privilege to access the entire system and distribute. the permissions to a group of users.
    - Add all courses for different regulations in the system.
    - Admit students results.
    - Generate lectures/exams timetables.
    - Manage database, access, and modify it.
- Academic advisor
    - The academic advisor can admit the students' registered courses.
    - He can print a transcript for a student.
    - He can view all the student details (GPA, hours, courses)

- Professor
    - Upload all the results and send them to the admin and can save and complete the results later.
- Student
    - Choose any course to join.
    - See his exam schedule and lectures.
    - Follow his academic progress.

2.3.3 Use cases Description.

- Registration

  Actor: student.

  Brief Description: This use case describes how new users register to the System.

  Basic Flow: This use case starts when a new actor wishes to register to the System.

  1. The user, if new, requests to register himself with the system

  2. The system asks for the personal details of the user(name, email, password).

  3. User enters the personal details and submits the registration form.

- Login

  Actor: Academic advisor /admin/student/professor

  Brief Description: This use case describes how a user logs into the System.

  Basic Flow: This use case starts when an actor wishes to log into the System:

  1. The system requests that the actor enter his/her email and password.

  2. The actor enters his/her email and password.

  3. The system validates the entered email and password and logs the actor into the system.

  Alternative Flows: Invalid email / Password

  If in the Basic Flow the actor enters an invalid password and/or password, the system displays an error message. The actor can choose to either return to the beginning of the Basic Flow or cancel the login, at which point the use case ends.

- Upload results

  Actor: professor

  Brief Description: This use case describes how a professor uploads the results.

  Basic Flow: This use case starts when an actor wishes to upload students results:

  1. The actor opens the upload results page.

2. The actor chooses the course that he wants to upload its result.

3. The system gets all the students' information and a place to fill in the degrees.

4. The actor fills in the degrees and presses send to admin or save.

5. The system validates the entered degrees and sends a notification message to the actor

Alternative Flows: The results exceed the maximum degrees.

If in the Basic Flow the actor enters an invalid degree. The system displays an error message. The actor then should fix this degree.

- Admit results.

  Actor: Admin

  Brief Description: This use case describes how an actor admits the results.

  Basic Flow: This use case starts when an actor wishes to admit the students results.

  1. The actor opens the admit results page.

  2. The system displays all the results details.

  3. The actor chooses to admit or drop or increase the degrees in the result.

  Alternative Flows: The actor can see the students that need from 4 to 1 degree for success in each course.

- Generate lecture/exam timetable.

  Actor: Admin

  Brief Description: This use case describes how an actor generates lectures/exams timetable.

  Basic Flow: This use case starts when an actor wishes to generate lectures/exams timetable.

  1. The actor opens the generation page.

  2. The actor clicks the generate button.

  3. The System displays a confirm message “timetable is being generated

- Course Registration

  Actor: Student

  Brief Description: This use case describes how to enroll a student for any open course.

  Basic Flow: This use case begins when the actor wants to register for any course.

  1. The system presents all available courses.
  2. The actor adds a course.
  3. The system sends the courses that have been registered to the Academic advisor to admit them.

  Alternative Flows: choose closed courses/less than the minimum hour per term.

If in Basic Flow, the actor chooses invalid courses, the system displays an error message. The actor can choose the author course.

- Courses registration admission

  Actor: Academic advisor

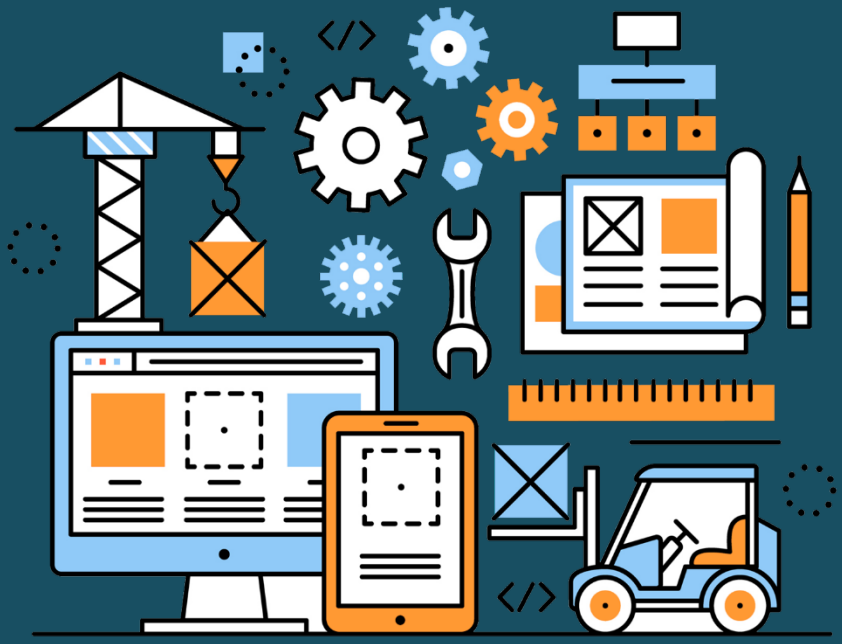
  Brief Description: This use case describes how the academic advisor admits the course registration for each student.

  Basic Flow: This use case starts when an actor wants to admit the course registration.

  1. The system shows all information about the courses that the student registered it.
  2. The actor presses the admit button.
  3. The system admits the courses and changes the course's status to 'studying'.

  Alternative Flows: Add/Drop courses.

  If in the Basic Flow, the actor enters the Drop or Add button, the system displays a new page. The actor can choose any course and add or delete it.



Chapter 3

Software design

3.1 Entity relationship diagram

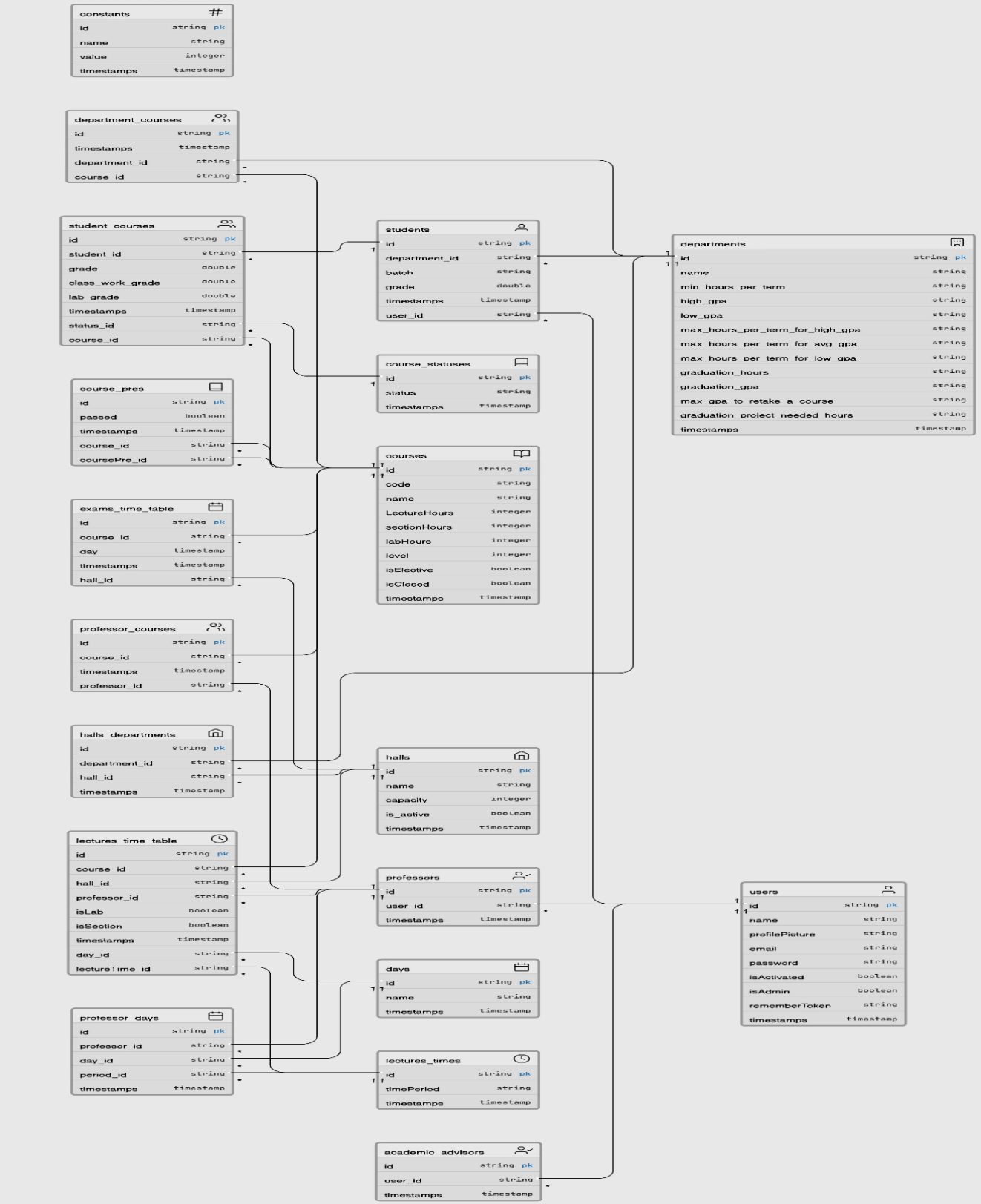


Figure 1: entity relationship diagram

3.2 System Sequence Diagram

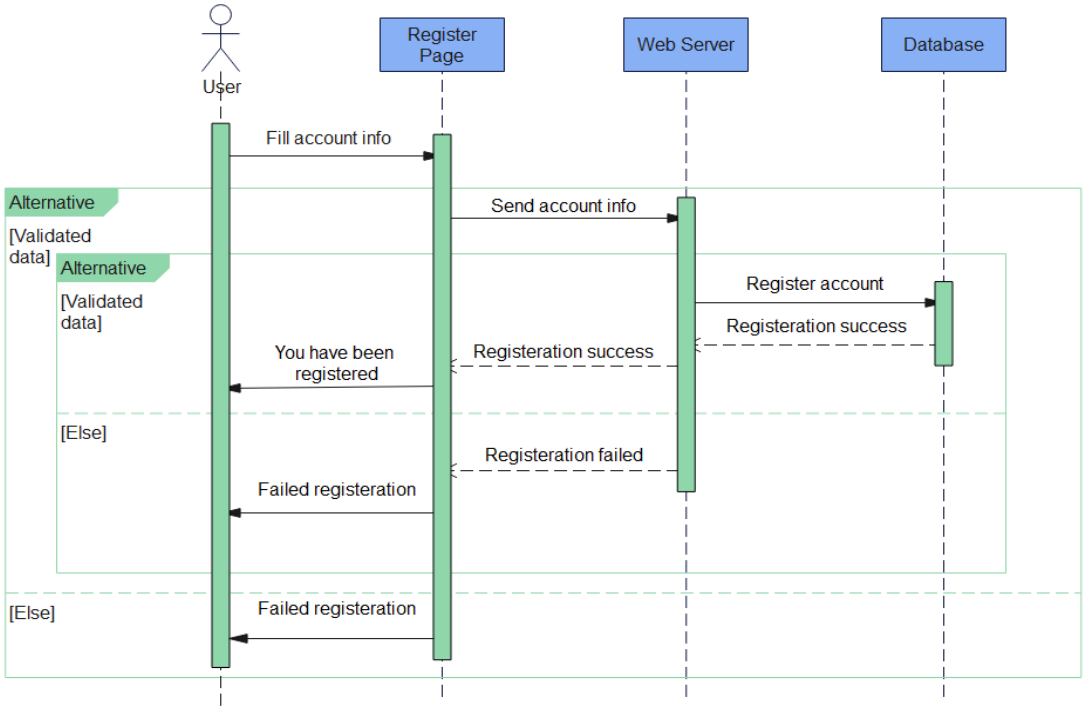


Figure 2: User register sequence diagram

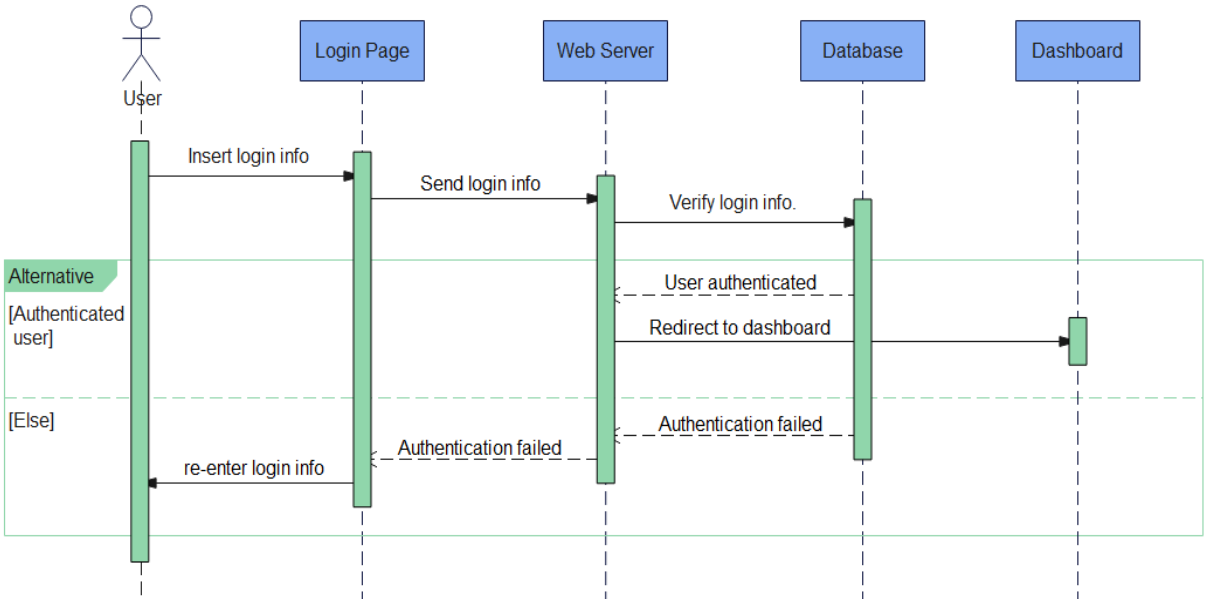


Figure 3: User login sequence diagram

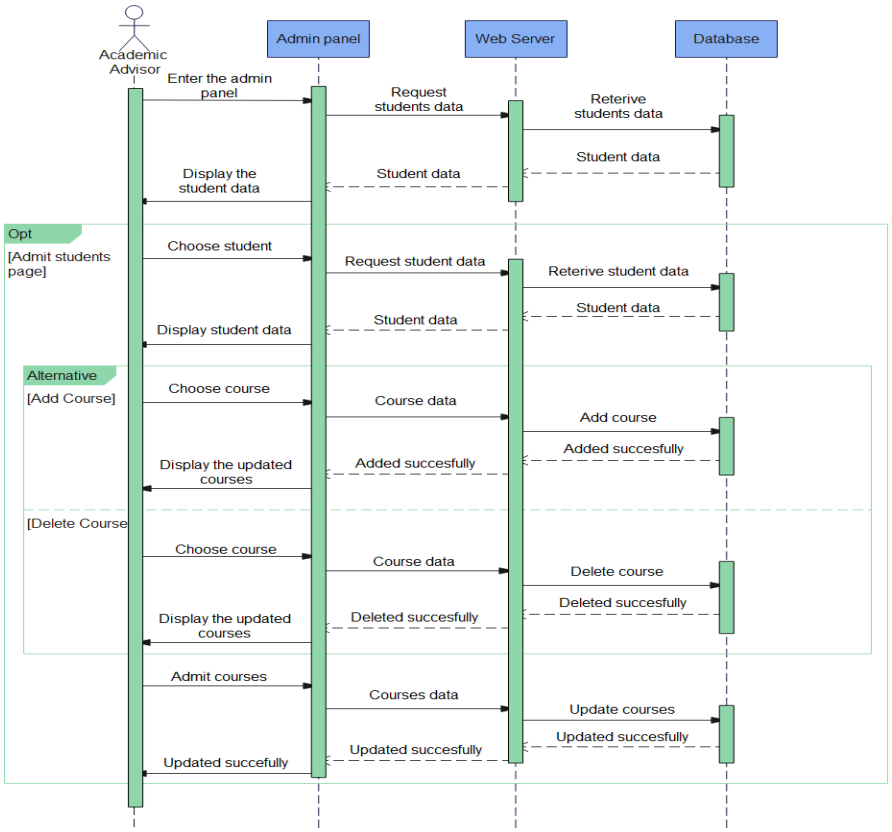


Figure 4: Academic advisor admits students' courses.

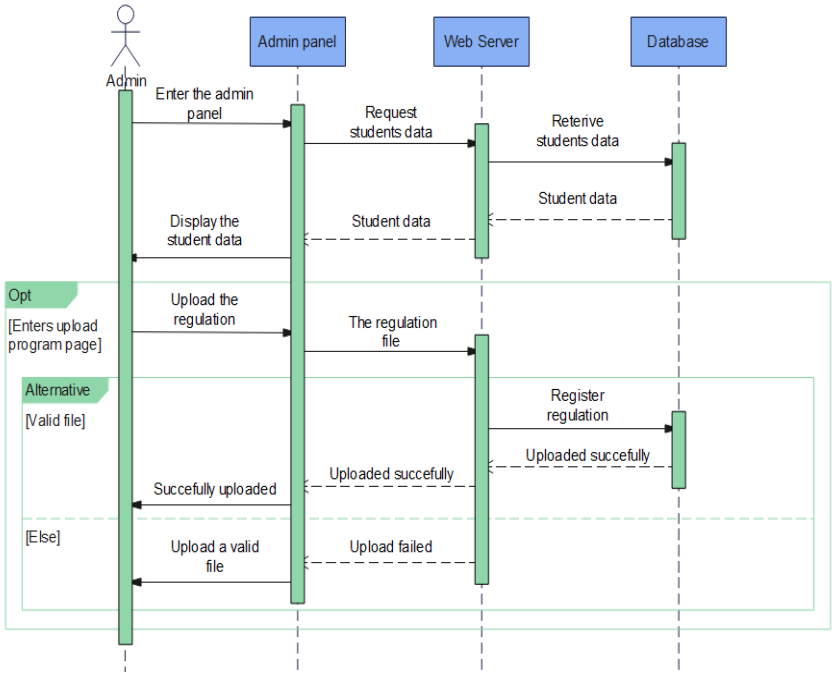


Figure 5: Admin upload the regulation course.

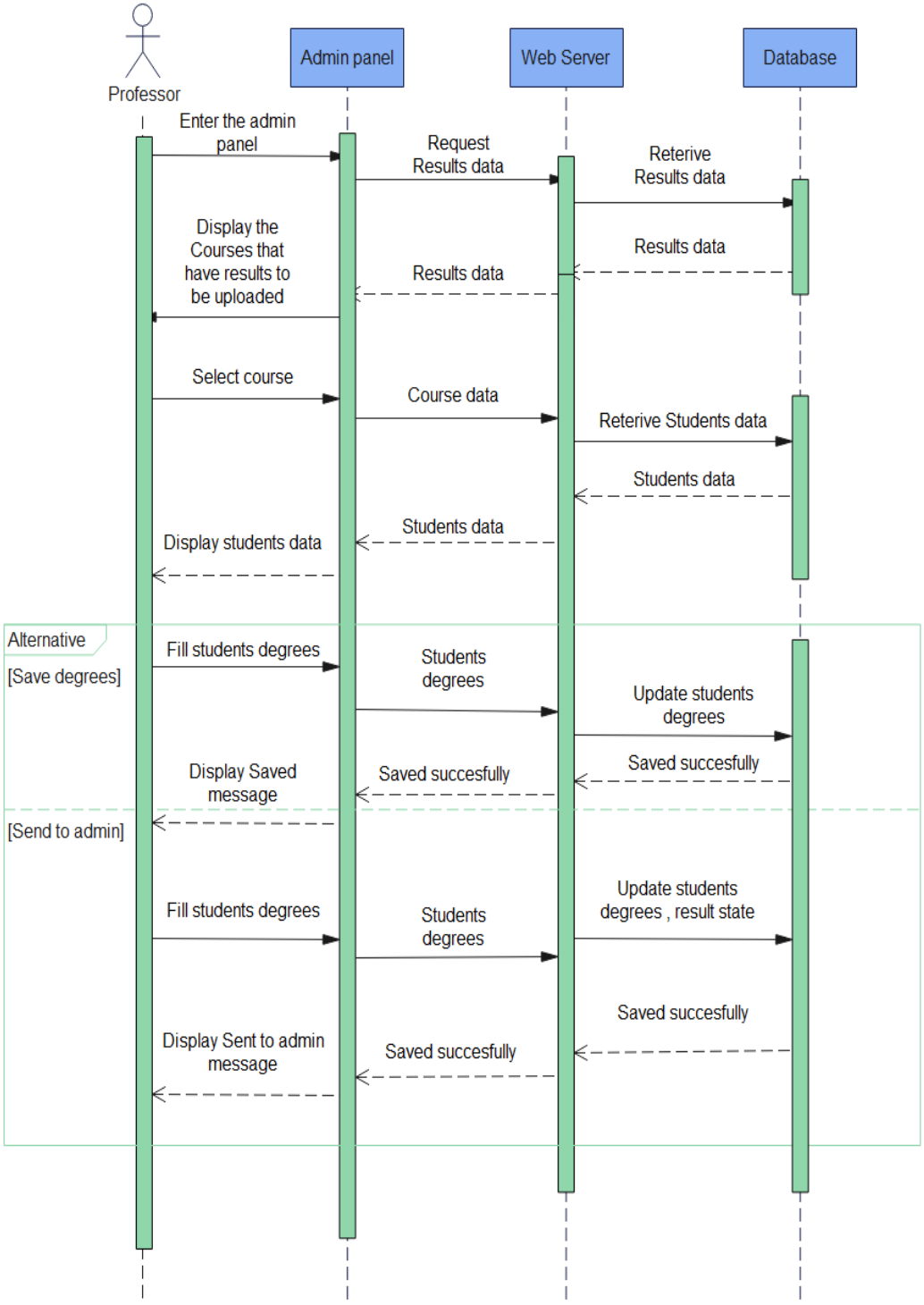


Figure 6: Professor uploads students results.

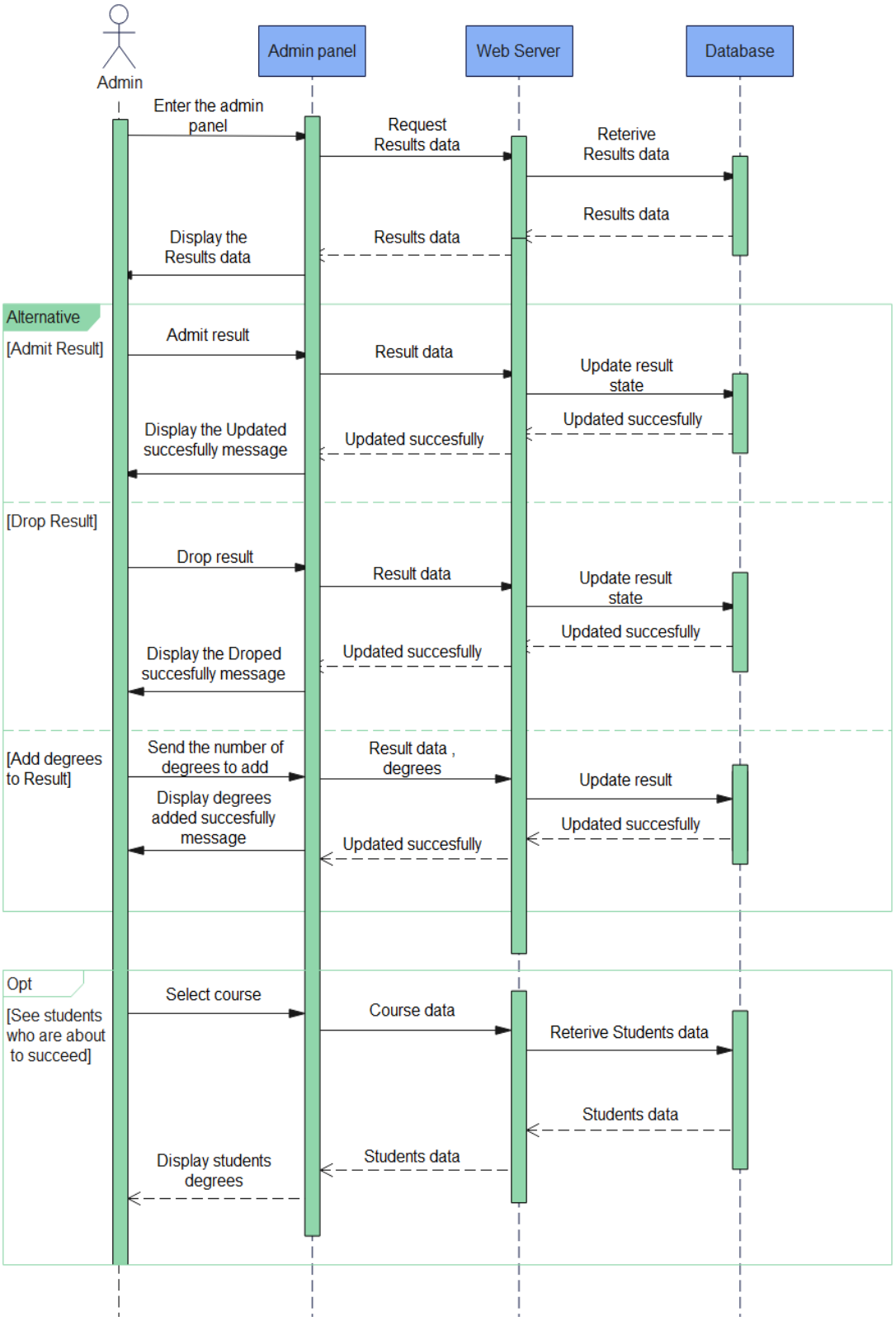


Figure 7: Admin admits students' results.

3.3 Use case diagram.

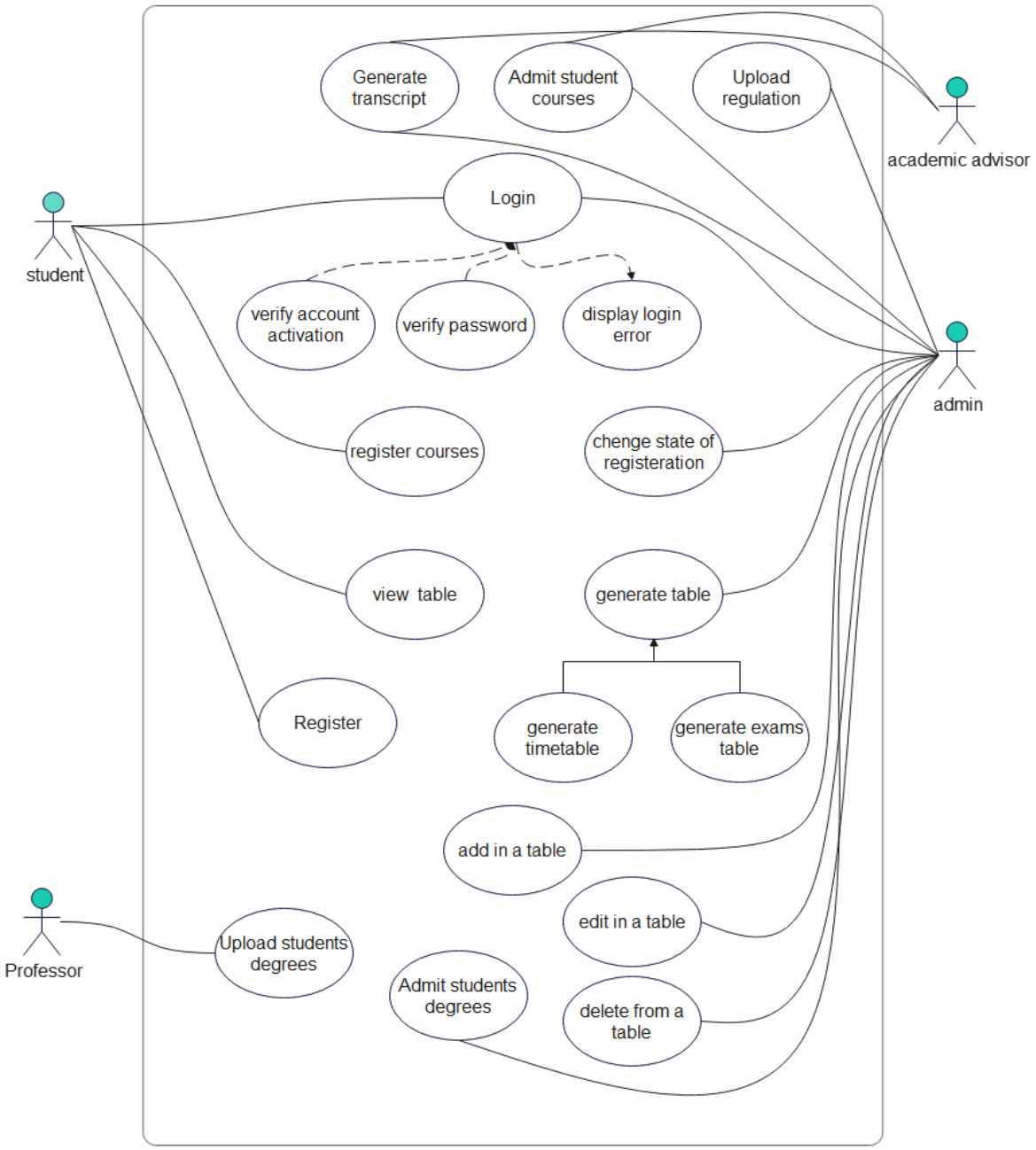


Figure 8: System use case diagram.

3.4 Block diagram.

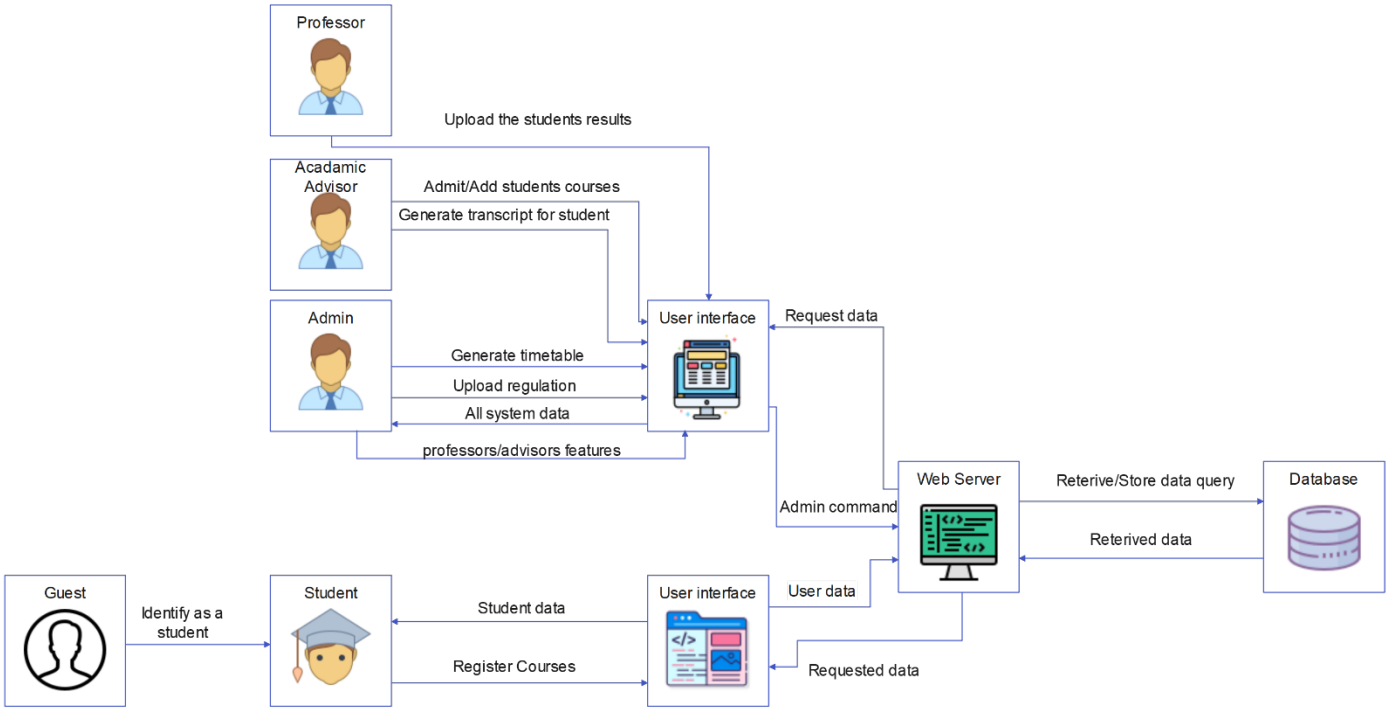
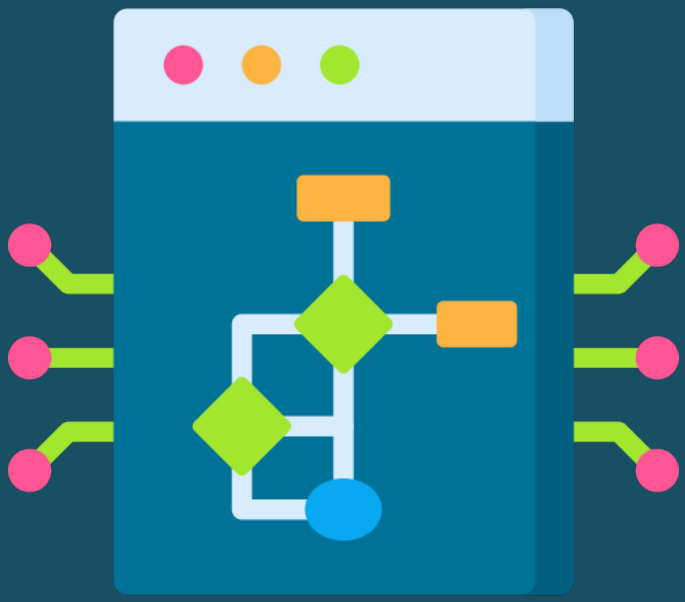


Figure 9: System block diagram



Chapter 4

GENETIC ALGORITHMS

4.1 Fundamentals of Genetic Algorithm (GA)

A genetic algorithm (GA) is a powerful problem-solving programming technique. It is in a category of evolutionary algorithms which is a subset of evolutionary computation in artificial intelligence [7]. It was developed in 1960 by Professor John Holland of the University of Michigan. His book, *Adaptation in Natural and Artificial Systems* pioneered genetic algorithm (GA) research in the 1970s [8]. This technique was inspired by the Darwinian theory of natural evolution, which states that the organisms in the world multiply in geometric proportions leading to a struggle for existence due mainly to limited space and food. In this struggle the fittest will survive. The fittest are those with favorable variations, the accumulation of which lead to the evolution of species. The chances for the survival of organisms with injurious variations are rather slim. Thus, evolution is a process of natural selection [9]. Each cell of an organism of a species has a definite number of genetic structures called chromosomes consisting of linearly arranged units called genes (blocks of DNA) carrying genetic information for one or many characters.

A complete set of genetic material, namely all the chromosomes is called a genome. Man for example has 23 pairs of chromosomes. A group of genes in a genome is called the genotype. The expression of a genotype is referred to as the phenotype. Each gene has a definite position in the chromosome called the locus. A given gene can be in several states called alleles which express characters such as the eye color of the organism.34 When organisms reproduce.

Genetic algorithms closely mimic the biological model of chromosomes and

genes with each chromosome representing an individual organism and genes forming components of a solution that is to be used with a genetic algorithm. Thus, a chromosome represents a data structure. As in nature, new chromosomes are generated by mixing genetic material by means of crossover and mutation. Crossover is equivalent to mating in the biological process. New information is introduced to the population by way of mutation. Thus, it is the chromosome (individual) that changes (evolves) by altering the order and the makeup of its genes. It is noteworthy that a finite number of genes are used in the process.

4.2 How Genetic Algorithm Works

In a genetic algorithm, a population of chromosomes consisting of a given random collection of genes is initiated according to the following steps (Figure 10).

1. Generating an initial population of chromosomes.
2. Evaluating the suitability of each chromosome (individual) that forms the population.
3. Selecting the chromosomes for mating based on the above results.
4. Producing offspring by mating (cross over) the selected chromosomes.
5. Mutating genes randomly.
6. Repeating steps 3-5 until a new population is generated.
7. Ending the algorithm when the best solution is obtained.

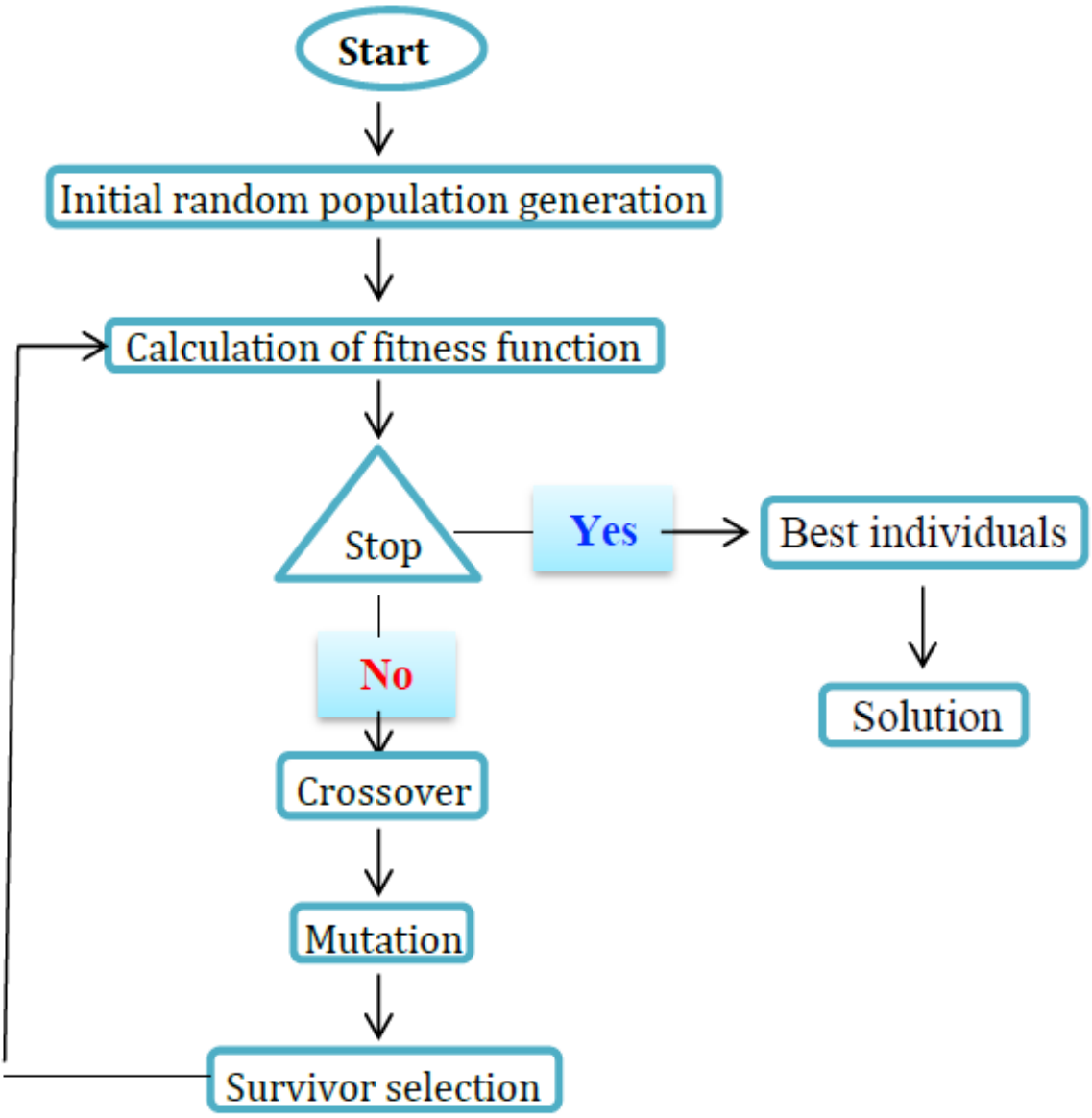


Figure 10: How genetic algorithms work

Three basic operations are evident from the above procedure. They are the selection, crossover, and mutation. The technique provides more than one solution for complex problems such as the "NP-hard" problems which probably cannot be solved exactly in polynomial time. Such problems increase at a much greater rate, described by the factorial operator (n!). An example of an NP-hard problem is the traveling salesman problem.

Polynomial problems solved by present day computers involve several steps to arrive at a solution. Each problem is bound by a polynomial, a mathematical expression of exponents and expressions. The above steps of genetic algorithm are elaborated as follows.

4.3 Initial population of chromosomes

A genetic algorithm creates an initial population of chromosomes with each chromosome (organism) representing a complete solution to a given problem. The genes which constitute the chromosomes are initialized to random values. Based on a function specific to a given problem, each chromosome is evaluated for its "fitness" which defines the quality of the solution.

4.4 Suitability of chromosomes to mate

In a genetic algorithm, creation of an improved population is the aim of mating the most suitable chromosomes in a population. It results in the production of offspring (chromosomes) which join the existing population of chromosomes to

mating population in subsequent generations. For the genetic algorithm to function effectively a "fitness function" in the form of a numeric score must be designed to evaluate the solutions (chromosomes). In nature there is no assignment of a score the -- organisms just die or survive.

As an example of designing a numeric score to evolve the word "run" as described in the literature, a population consisting of three members namely, "son", "rug" and "cat" can be considered [6]. The word "rug" has two correct features as against the other two and has the highest number of points. The word "son" has one feature against that of "mat" with none of the characters having a score amounting to zero. Thus, the "fitness" in this example is attributed to the number of correct features as depicted in **following table**.

DNA	Fitness
son	1
rug	2
mat	0

4.5 Selection to the mating pool.

With the above example it is obvious that only two members are fit to be parents to be placed in a mating pool as shown by the fitness calculations made on the members of the population. One of the approaches called the *elitist* method can be used to select the members with highest scores are allowed to make offspring for the next generation. If there are only two members among thousands available as parents, the variety of the offspring will be very much limited, which could stunt the

process of evolution. The obvious conclusion therefore is to make a large mating pool to obtain better results. However, this will not render optimal solutions as the top scoring individuals have the same chance of being selected as the ones toward the middle.

4.5.1 Roulette Wheel Selection

A method popularly known as the “wheel of fortune” or the “roulette wheel” based on probability seems to provide a better solution in selecting members to the mating pool. A simple example to illustrate the method is to consider a five-member population having the following fitness scores **in the following table**.

DNA	Fitness
P	4
Q	3
R	0.7
S	1.3
T	1

The total fitness is 4 + 3 + 0.7 + 1.3 + 1 = 10

The normalized fitness scores and the percentage values are given **in the following table**.

DNA	Fitness	Normalized Fitness	Expressed as a Percentage
P	4	0.4	40%
Q	3	0.3	30%
R	0.7	0.07	7%
S	1.3	0.13	13%
T	1	0.1	10%

4.5.2 Selection based on the wheel of fortune.

Parent	Probability
P	40%
Q	30%
R	7%
S	13%
T	10%

Figure 11: Selection based on wheel of fortune.

The selection chance for members **P** - **T** are **P**> **Q**> **S**> **T**>**R (Figure 11)**

This procedure is considered satisfactory because it ensures the reproduction of the highest scoring individuals without eliminating the ones with lowest scores providing an opportunity to pass the information to the next generation. This could be explained by trying to evolve the phrase, “to do or not to do” using the available individuals P-R.

P: to do or not

to be Q: to do

or not to go

R: xxxxxxxxxxxxxxxxdo

The chromosomes P and Q have most of the features of the required phrase having the highest scores as against R with lower score. However, the individuals P and Q lack the correct features for the end of the phrase. R on the other hand contains the necessary genetic data for the end of the phrase. As can be seen, elements P and Q are clearly the most fit and would have the highest score. But neither contains the correct characters for the end of the phrase. Element R, even though it would receive a very low score, happens to have the genetic data for the end of the phrase. Whilst P and Q are selected to produce the majority of offspring of the next generation it is necessary to have some opportunity for R to take part in reproduction to obtain a perfect solution for the present problem.

4.5.3 Tournament selection

This method also selects individuals for the mating pool depending on their fitness function. It is a frequently used method in GA due to its simplicity and efficiency. The method involves a random selection of individuals from a bigger population. They will compete for a place in the next generation population.

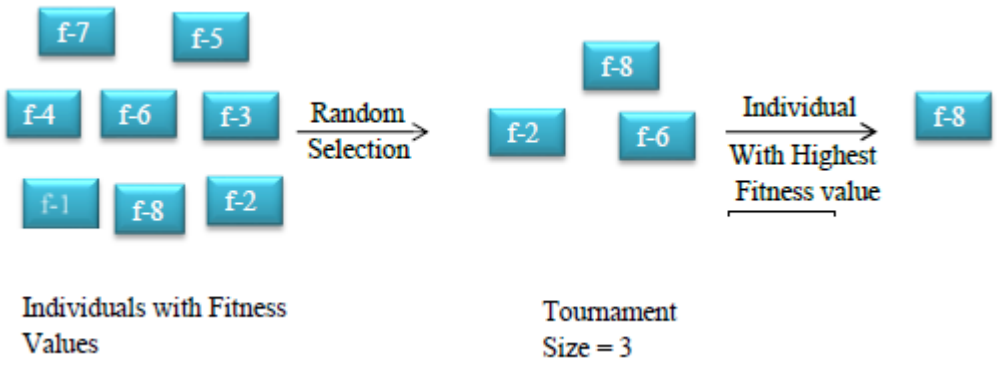


Figure 12: Tournament mechanism

4.5.4 Stochastic Tournament selection

In stochastic tournament selection, the individual selected by the Roulette Wheel is taken to the Tournament.

Scaling, Rank, Generational, Steady-state, and Hierarchical are other selection methods used [10].

The convergence rate together with the number of generations required to yield the optimal solution defines the performance of the genetic algorithm [11].

The Roulette Wheel method is associated with high time complexity as well as faster. convergence with the selection of certain parents. The tournament method, on the other hand, exhibits less time complexity and therefore is faster [12]. In this study, tournament selection method is used for its simplicity with regards to coding and

efficiency towards parallel and nonparallel constructions [11],[13]. In addition, this method leaves room for selection pressure adjustments which has an impact on its usage [13],[14]. Selection of individuals with higher fitness values for the next generation is controlled by selection pressure which is described as the extent to which the better individuals of the population are favored. The tournament size can be increased or decreased to obtain a desired selection pressure. A higher tournament size (higher selective pressure) will prevent some individuals taking part in a tournament whereas some others may not be selected at all. This brings about loss of genetic diversity resulting in premature convergence restricting the results to a local optimum. [9] Genetic diversity in essence, is the maintenance of a diverse solution population which makes sure that the solution space is searched adequately. A too small selection pressure (lower tournament size) on the other hand, may take a longer time to converge to an optimum. Thus, a proper selection pressure which ensures genetic diversity is required for the genetic algorithm to converge to a global optimum [11].

4.6 Methods of change in reproduction.

4.6.1 Cross-over

The characteristics of the suitable individuals selected must be altered to improve their fitness to the next generation. This is achieved by means of cross-over and mutation. Crossover simulates the recombination of chromosomes in sexual reproduction which involves two parents. To explain the phenomenon the following two parent phrases (individuals) from the mating pool can be considered.

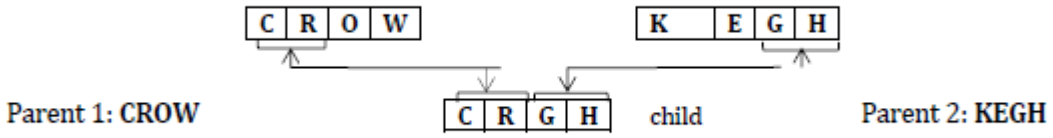


Figure 13: Cross-over

A child phrase can be made in a 50/50 manner by taking two features from each parent as shown in **Figure 13.** This is a one-point crossover.

Picking up a random point instead of the 50/50 selection is preferred as it increases variety for the next generation as indicated in **Figures 14 and 15**.

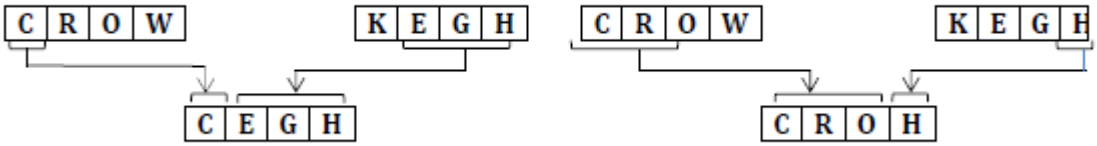


Figure 14: Selecting a random point.

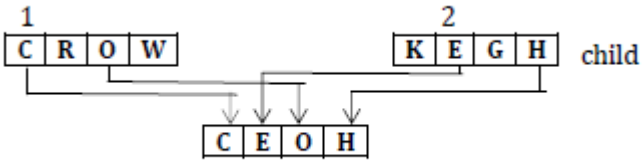


Figure 15: KEOH, CEOW, CROH, CEOW are some of the possible combinations.

4.6.2 Mutation.

Additional variety to the offspring DNA which was created during the crossover can. be introduced by mutation as shown in **Figure VIII**. It is a process of picking up a new random character. A mutation rate of 1% indicates that each character in the phrase resulted from crossover, there is a 1% chance that it will mutate. A mutation may produce a desirable or undesirable character. Natural selection will decide upon the fate of the mutated chromosome.

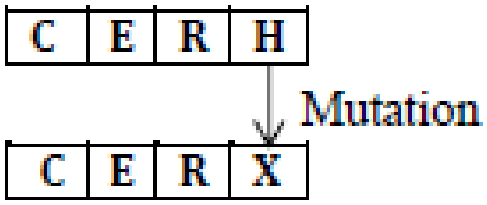


Figure 16: Mutation

The processes of selecting parents and reproduction are repeated until desirable results are obtained.

4.6.3 Methods of representation.

Several methods are available to encode potential solutions to a given problem in a form that the computer can process the data [10]. In one method binary strings in sequences of 1's and 0's are used to encode solutions with each digit representing some aspect of the solution. A similar method is to encode solutions in integers and decimal numbers with each position representing some feature of the solution allowing precision and more complexity than using only the binary numbers [15]. In a third method a string of letters is used to represent an individual where each letter is designated to some aspect of the problem.

All three methods clearly define the changes which the selected individuals undergo during the process. The changes include the flipping of a 0 to 1, changing a letter to another or adding or subtracting a randomly selected amount to a number.

Another approach is called genetic programming where programs are represented as branching data structures (**Figure 17**) [16] and random changes are brought about by changing the value at a specific node in the tree. Changes can also be made by replacing one subtree with another.

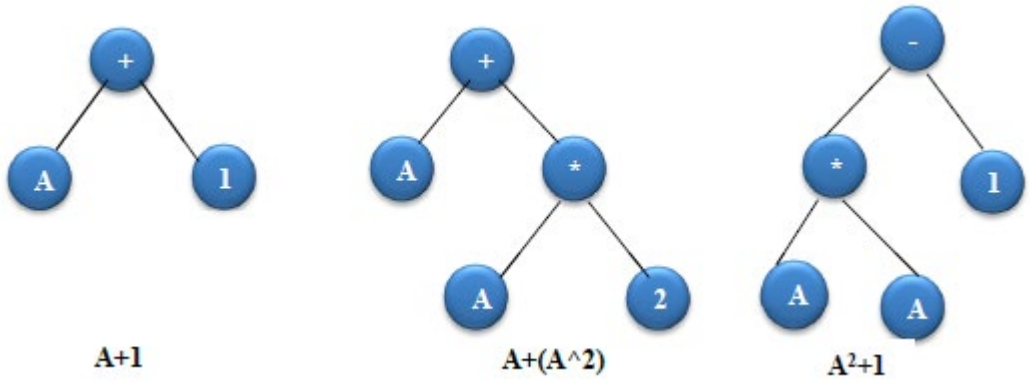


Figure 17: Genetic representation

4.7 Advantages of genetic algorithm

As compared to genetic algorithms, other optimization algorithms look for solutions in a serial manner (in one direction) in the search field at a given time. The disadvantage of a serial search is that if the solution obtained is not favorable, the work carried out thus far must be abandoned and a new search must be started. Genetic algorithms, which have multiple offspring on the other hand, can look for solutions in many directions at a given time, abandoning the paths which lead to suboptimal solutions. Thus, the opportunity of finding a favorable solution in a genetic algorithm is high during each run. Further, as

per Holland's schema theorem, the fitness schemata which are above average grow while the ones below average decay. As such, parallelism in addition, indirectly evaluates many different schemata to identify individuals of highest fitness values. Thus, although genetic algorithms seem to evaluate a small group, they evaluate a very large group of individuals in a reasonable amount of time. As an example, an 8-digit binary string designated as, ******** could be considered. * Is either 0 or 1. All these 8-digit strings form a search space. 01101010 string, for example with a fitness of 24, becomes not only a member of this search space, but it also becomes a member of the spaces,
0******* ,01****** ,0****** 0, 0*1*1*1* ,10*01**0 Genetic algorithms

will sample to evaluate the fitness of the string in each of these spaces to which the string, 01101010 belongs.

Another advantage of a genetic algorithm (GA) is its capability to handle

several parameters simultaneously. Usually many of the real-world problems can only be described in terms of multiple objectives. As such, they cannot be expressed in the form of a single value for minimization and maximization purposes. A parameter can only be improved at the expense of another. Due to parallelism however, a GA can generate many solutions with one individual optimizing one parameter and another optimizing another. This allows one to select a desired solution for use.

4.8 Disadvantages of genetic algorithm

There are disadvantages to genetic algorithms. The representation for the problem must be clearly defined. The language employed to describe solutions should be able to bear random changes so that no fatal errors occur. Binary, integer, or real-valued list of numbers is usually used to define individuals.

Choosing a fitness function is difficult. It should be carefully written to obtain the desired solution. An ill-defined fitness function could end up in not getting any solution or it might result in solving the wrong problem [17]. In addition to getting a good fitness function, it is also necessary to pay attention when selecting such factors as the number of individuals (population), crossover and mutation rates. Less solution space created by a small population leads to inaccurate results.

Premature convergence is another problem associated with a GA, especially when small populations are involved. This results from an individual (more fit) who is relatively capable of reproducing abundantly diminishes the

population diversity too soon leading to convergence to a local optimum. In such an event, finding the global optimum becomes impossible as there could not be a search in the search field due to the individual representation. However, there are methods that can be adopted to overcome this problem.

Analytical techniques are generally recommended to solve analytical problems instead of GA methods, as the former take less computational effort and time

Chapter 5

System implementation

5.1 Student view

- Login

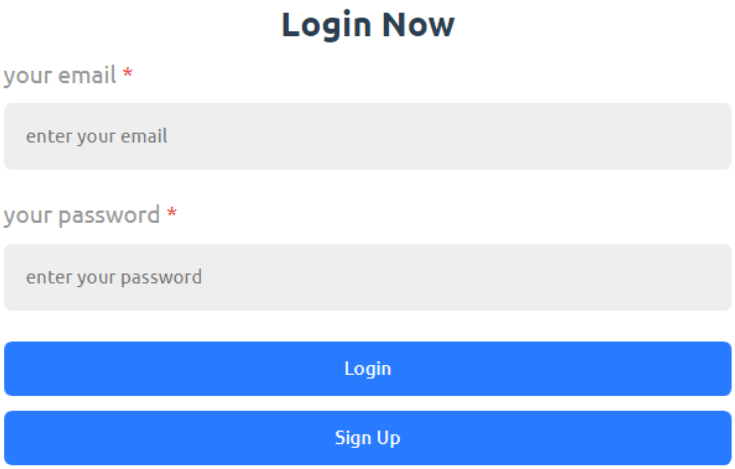


Figure 18: Login page

- Register

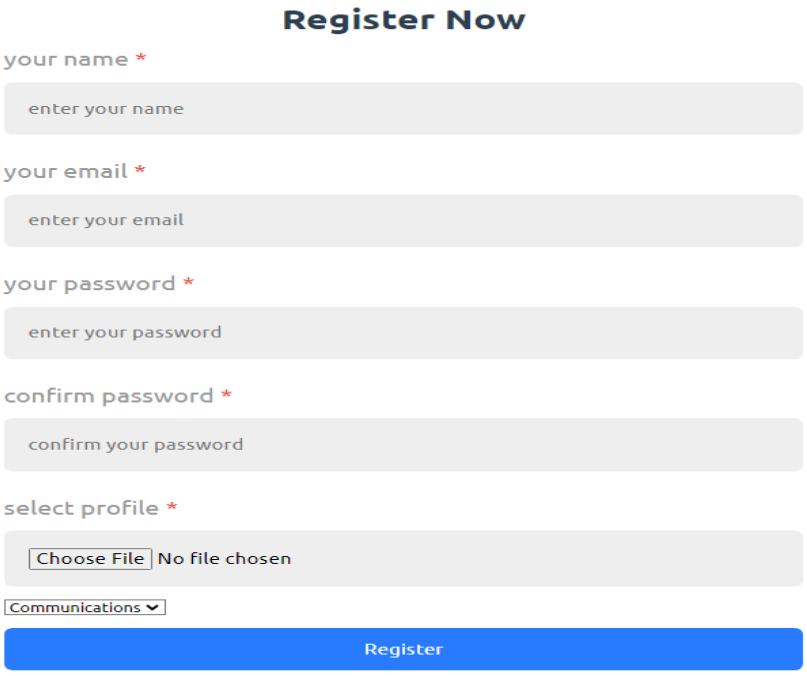


Figure 19: Register page

- Student home page

In this page the student can view his current courses , statistics about his finished hours , the results of the most recent finished courses

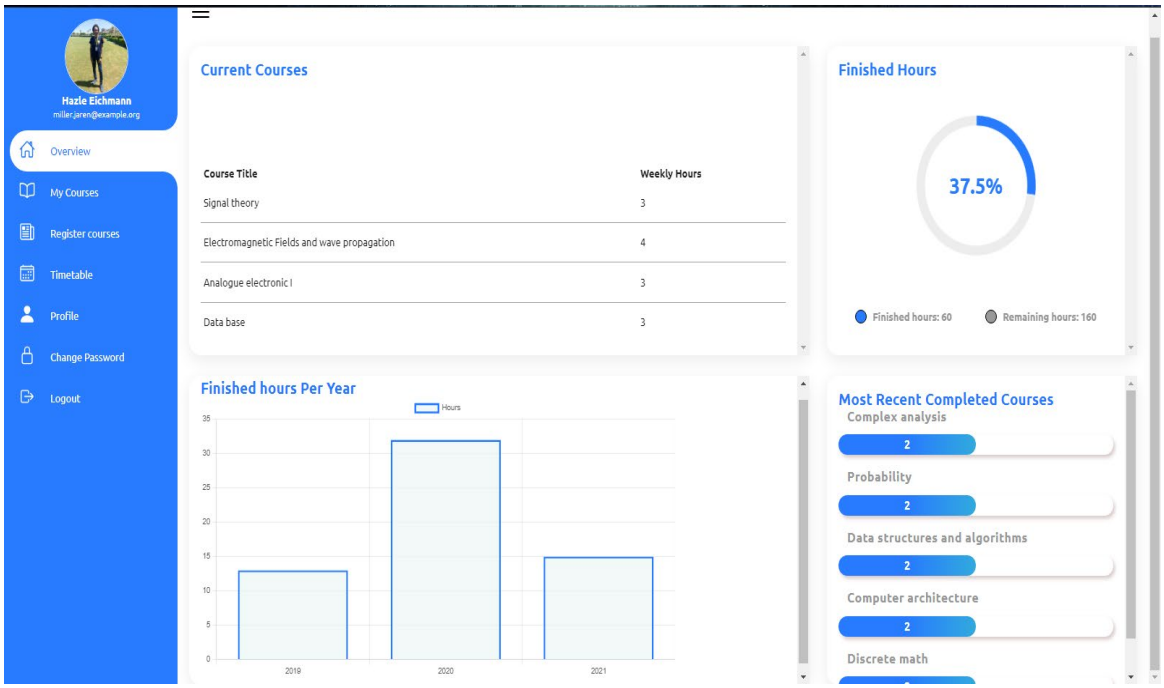


Figure 20: Student home page

- Finished courses.

  In this page the student can view all his previous courses results

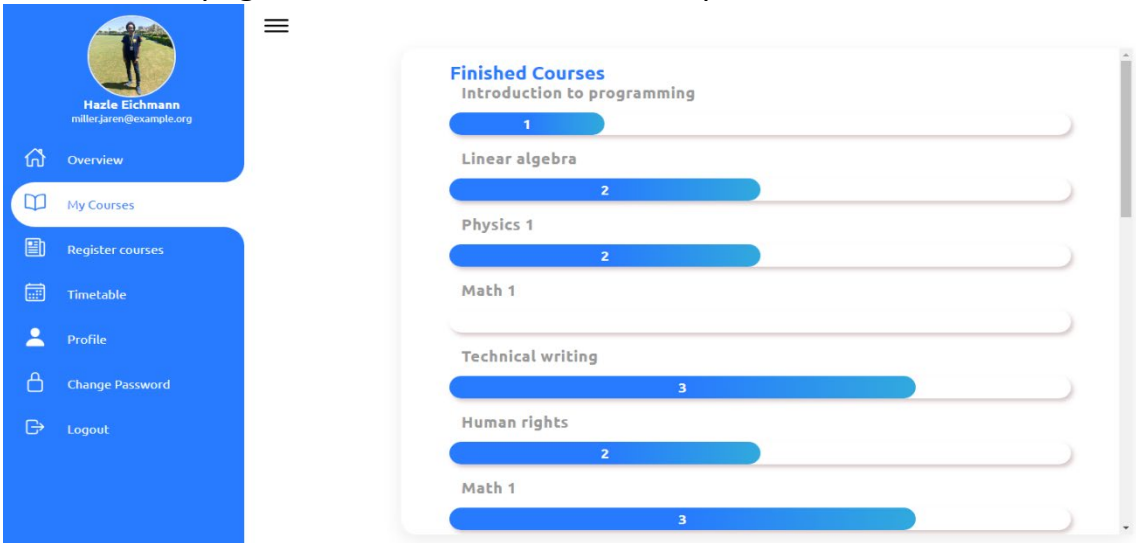


Figure 21: Finished courses

- Student profile

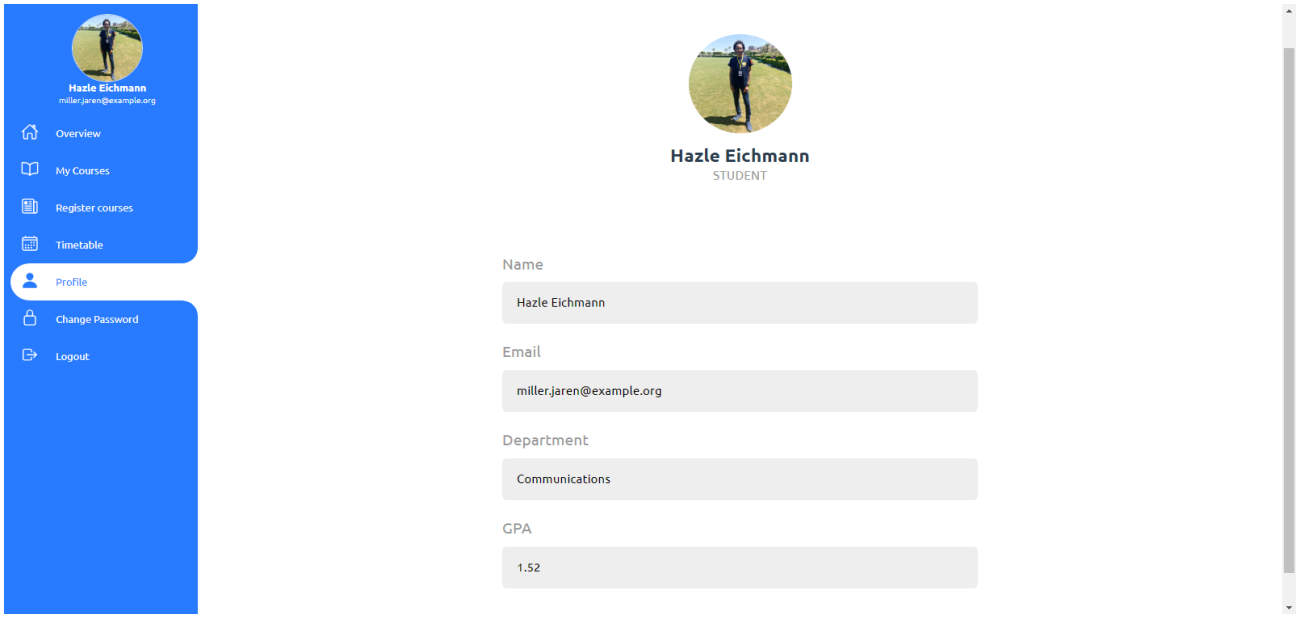


Figure 22: Student profile

- Register courses.

  In this page the students choose the courses he want to register for this semester

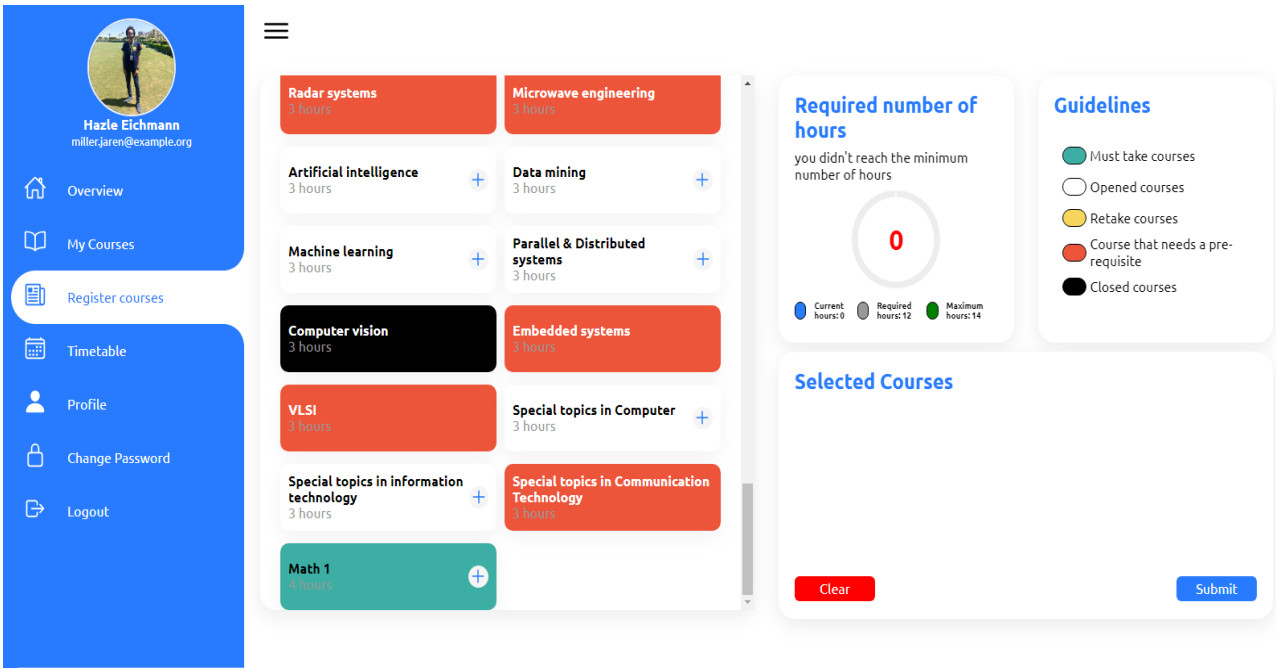


Figure 23: Register courses

- Lectures timetable

  In this page the student can view his own lectures timetable

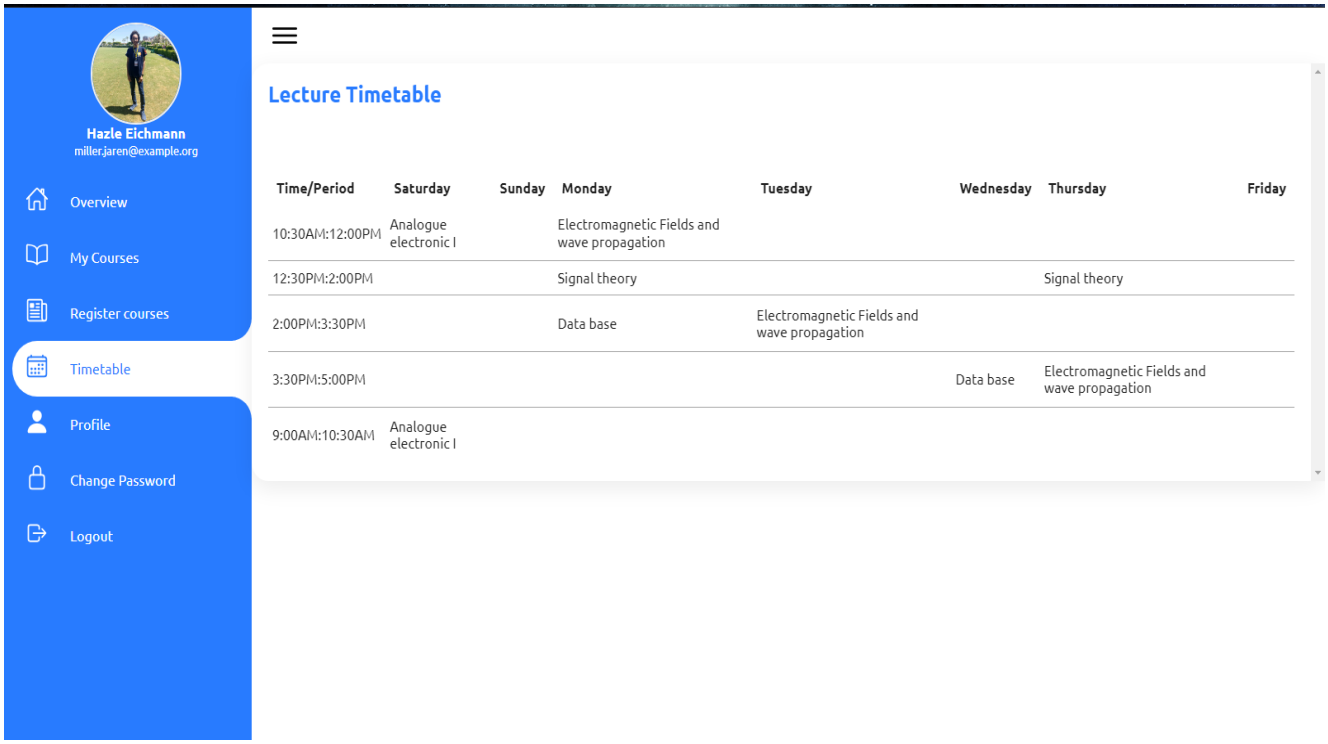


Figure 24: Lectures timetable

- Change password

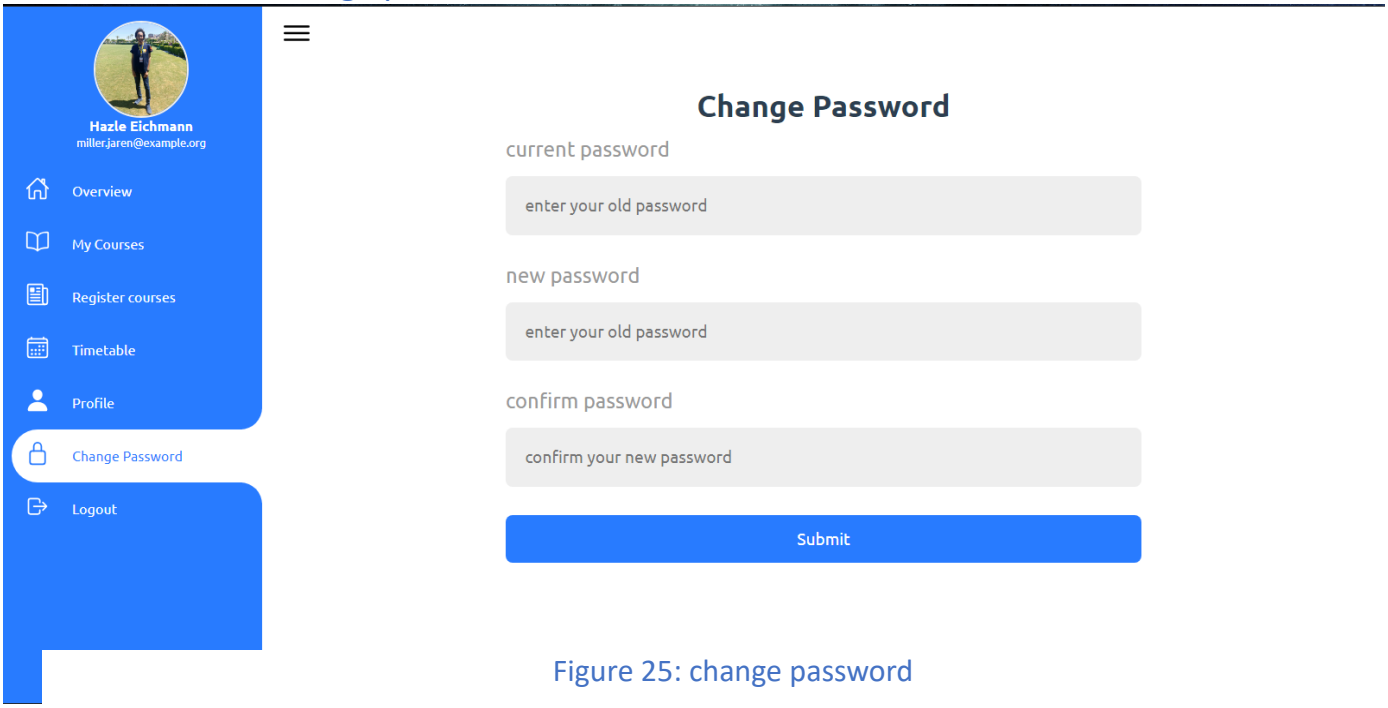


Figure 25: change password

5.2 Administrators view.

Before we see the administrators' view, we'll talk about the exams/lectures timetable generation model implementation.

5.2.1 Generation model implementation

Creating efficient and conflict-free timetables for lectures and exams is a complex task faced by educational institutions worldwide. To optimize this process, many institutions have turned to genetic algorithms (GAs), a powerful evolutionary computation technique. We'll explore the implementation of a genetic algorithm approach for generating lecture and exam timetables, with a focus on crossover and mutation operations.

- Crossover Operation:

The crossover operation in our implementation involves replacing the positions of two subjects in the timetable. This operation aims to combine the favorable characteristics of two parent timetables to generate offspring timetables. The crossover point is chosen randomly, and the subjects located at that point in both parents are swapped. The resulting offspring inherits a mix of lecture and exam slots from each parent, potentially leading to better timetables.

Day	Room-0	Room-1	Room-2	Room-3	Room-4	Room-5	Room-6	Room-7	Room-8	Room-9
1	-	-	ECC52	-	-	-	-	-	-	-
2	-	-	-	-	ECC39	-	-	-	ECC07	-
3	-	-	-	MECH16	-	HUM305	-	-	-	-
4	-	-	-	-	-	-	-	-	ECC25	ECC35
5	-	-	-	-	ECC23	-	-	-	ECC53	-
6	-	-	-	-	-	-	MECH24	-	ECC34	-
7	-	-	-	-	-	-	-	-	-	GEN33
8	-	-	-	GEN26	-	-	-	-	-	-
9	-	-	ECC40	GEN05	-	-	-	-	-	-
10	-	-	-	ECC47	-	-	-	ECC20	-	-
11	ECC45	-	-	-	-	-	ECC32	-	-	-
12	-	GEN44	-	-	ECC48	-	-	-	-	-
13	ECC37	-	-	-	-	-	-	GEN41	-	-
14	-	GEN04	ECC31	-	-	-	-	-	-	ECC413
15	ECC49	ECC50	-	-	-	-	ECC18	-	GEN11	MECH14

Figure 26: Crossover initial

Day	Room-0	Room-1	Room-2	Room-3	Room-4	Room-5	Room-6	Room-7	Room-8	Room-9
1	-	-	ECC52	-	-	-	-	-	-	-
2	-	-	-	-	ECC39	-	-	-	ECC07	-
3	-	-	-	MECH16	-	HUM305	-	-	-	-
4	-	-	-	-	-	-	-	-	ECC25	ECC35
5	-	-	-	-	ECC23	-	-	-	ECC53	-
6	-	-	-	-	-	-	MECH24	-	ECC34	-
7	-	-	-	-	-	-	-	-	-	GEN33
8	-	-	-	GEN26	-	-	-	-	-	-
9	-	-	ECC40	GEN05	-	-	-	-	-	-
10	-	-	-	ECC47	-	-	-	ECC20	-	-
11	ECC45	-	-	-	-	-	ECC32	-	-	-
12	-	GEN44	-	-	ECC48	-	-	-	-	-
13	ECC37	-	-	-	-	-	-	GEN41	-	-
14	-	GEN04	ECC31	-	-	-	-	-	-	ECC413
15	ECC49	ECC50	-	-	-	-	ECC18	-	GEN11	MECH14

Figure 27: Crossover operation

Day	Room-0	Room-1	Room-2	Room-3	Room-4	Room-5	Room-6	Room-7	Room-8	Room-9
1	-	-	ECC52	-	-	-	-	-	-	-
2	-	-	-	-	ECC39	-	-	-	ECC07	-
3	-	-	-	GEN44	-	HUM305	-	-	-	-
4	-	-	-	-	-	-	-	-	ECC25	ECC35
5	-	-	-	-	ECC23	-	-	-	ECC53	-
6	-	-	-	-	-	-	MECH24	-	ECC34	-
7	-	-	-	-	-	-	-	-	-	GEN33
8	-	-	-	GEN26	-	-	-	-	-	-
9	-	-	ECC40	GEN05	-	-	-	-	-	-
10	-	-	-	ECC47	-	-	-	ECC20	-	-
11	ECC45	-	-	-	-	-	ECC32	-	-	-
12	-	MECH16	-	-	ECC48	-	-	-	-	-
13	ECC37	-	-	-	-	-	-	GEN41	-	-
14	-	GEN04	ECC31	-	-	-	-	-	-	ECC413
15	ECC49	ECC50	-	-	-	-	ECC18	-	GEN11	MECH14

Figure 28: Crossover final

- Mutation Operation:

The mutation operation introduces randomness into the population by changing the location of a random subject to an empty slot. This operation helps explore new regions of the solution space, potentially leading to better solutions. In our implementation, a subject is selected randomly, and its current location is replaced with an empty slot in the timetable. This allows for the possibility of improving the overall timetable by repositioning individual subjects.

Day	Room-0	Room-1	Room-2	Room-3	Room-4	Room-5	Room-6	Room-7	Room-8	Room-9
1	-	-	ECC52	-	-	-	-	-	-	-
2	-	-	-	-	ECC39	-	-	-	ECC07	-
3	-	-	-	MECH16	-	HUM305	-	-	-	-
4	-	-	-	-	-	-	-	-	ECC25	ECC35
5	-	-	-	-	ECC23	-	-	-	ECC53	-
6	-	-	-	-	-	-	MECH24	-	ECC34	-
7	-	-	-	-	-	-	-	-	-	GEN33
8	-	-	-	GEN26	-	-	-	-	-	-
9	-	-	ECC40	GEN05	-	-	-	-	-	-
10	-	-	-	ECC47	-	-	-	ECC20	-	-
11	ECC45	-	-	-	-	-	ECC32	-	-	-
12	-	GEN44	-	-	ECC48	-	-	-	-	-
13	ECC37	-	-	-	-	-	-	GEN41	-	-
14	-	GEN04	ECC31	-	-	-	-	-	-	ECC413
15	ECC49	ECC50	-	-	-	-	ECC18	-	GEN11	MECH14

Figure 29: Mutation initial

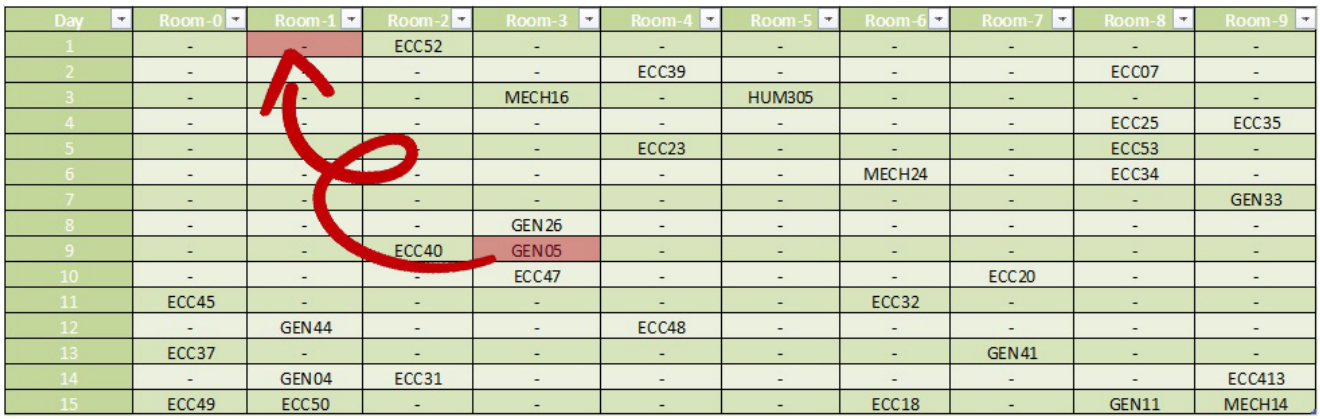


Figure 30: Mutation operation

Day	Room-0	Room-1	Room-2	Room-3	Room-4	Room-5	Room-6	Room-7	Room-8	Room-9
1	-	GEN05	ECC52	-	-	-	-	-	-	-
2	-	-	-	-	ECC39	-	-	-	ECC07	-
3	-	-	-	MECH16	-	HUM305	-	-	-	-
4	-	-	-	-	-	-	-	-	ECC25	ECC35
5	-	-	-	-	ECC23	-	-	-	ECC53	-
6	-	-	-	-	-	-	MECH24	-	ECC34	-
7	-	-	-	-	-	-	-	-	-	GEN33
8	-	-	-	GEN26	-	-	-	-	-	-
9	-	-	ECC40	-	-	-	-	-	-	-
10	-	-	-	ECC47	-	-	-	ECC20	-	-
11	ECC45	-	-	-	-	-	ECC32	-	-	-
12	-	GEN44	-	-	ECC48	-	-	-	-	-
13	ECC37	-	-	-	-	-	-	GEN41	-	-
14	-	GEN04	ECC31	-	-	-	-	-	-	ECC413
15	ECC49	ECC50	-	-	-	-	ECC18	-	GEN11	MECH14

Figure 31: Mutation final

5.2.2 Professor View

- Students results upload.

  Here the professor can choose a course from his courses to upload the students results for it.

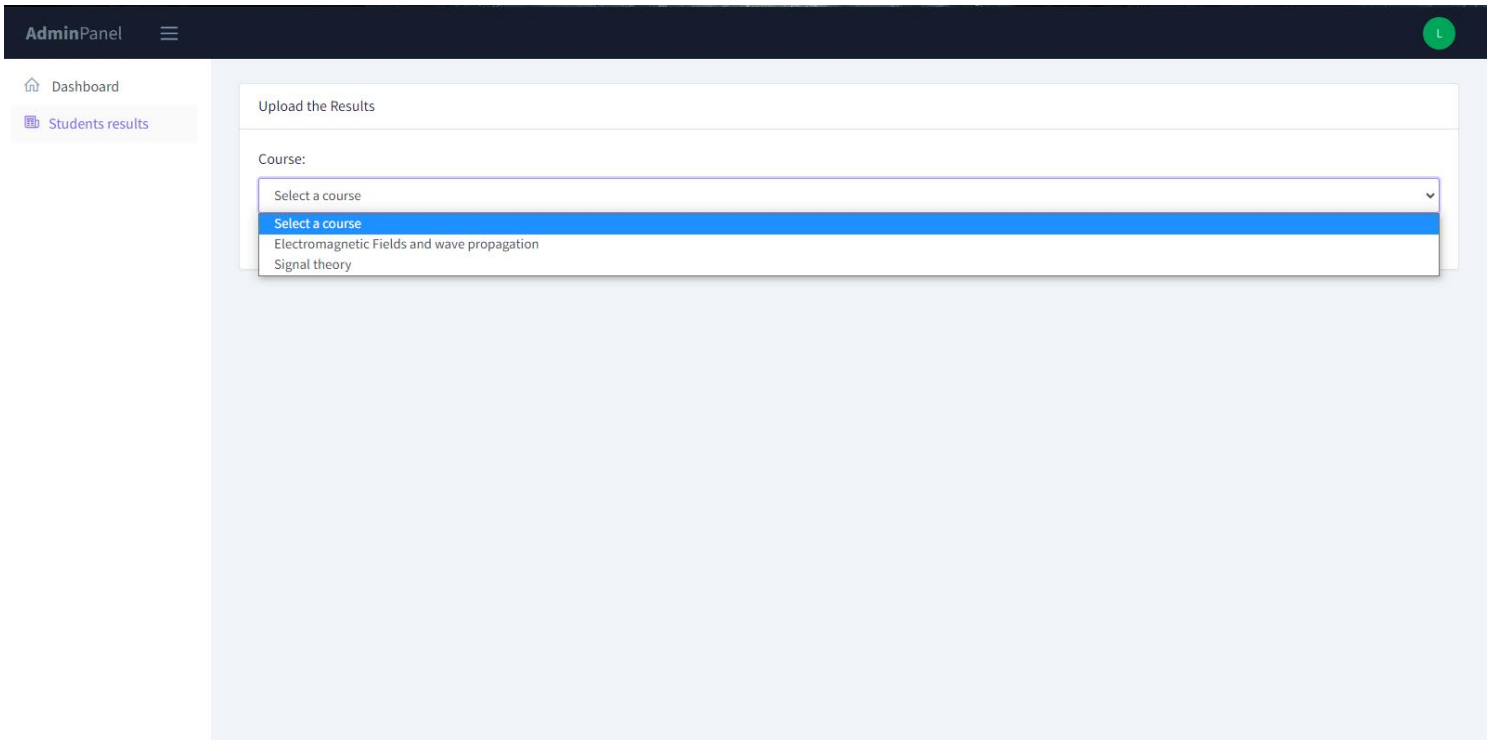


Figure 32: Professor student result upload page

5.2.3 Academic Advisor View

- Students page.

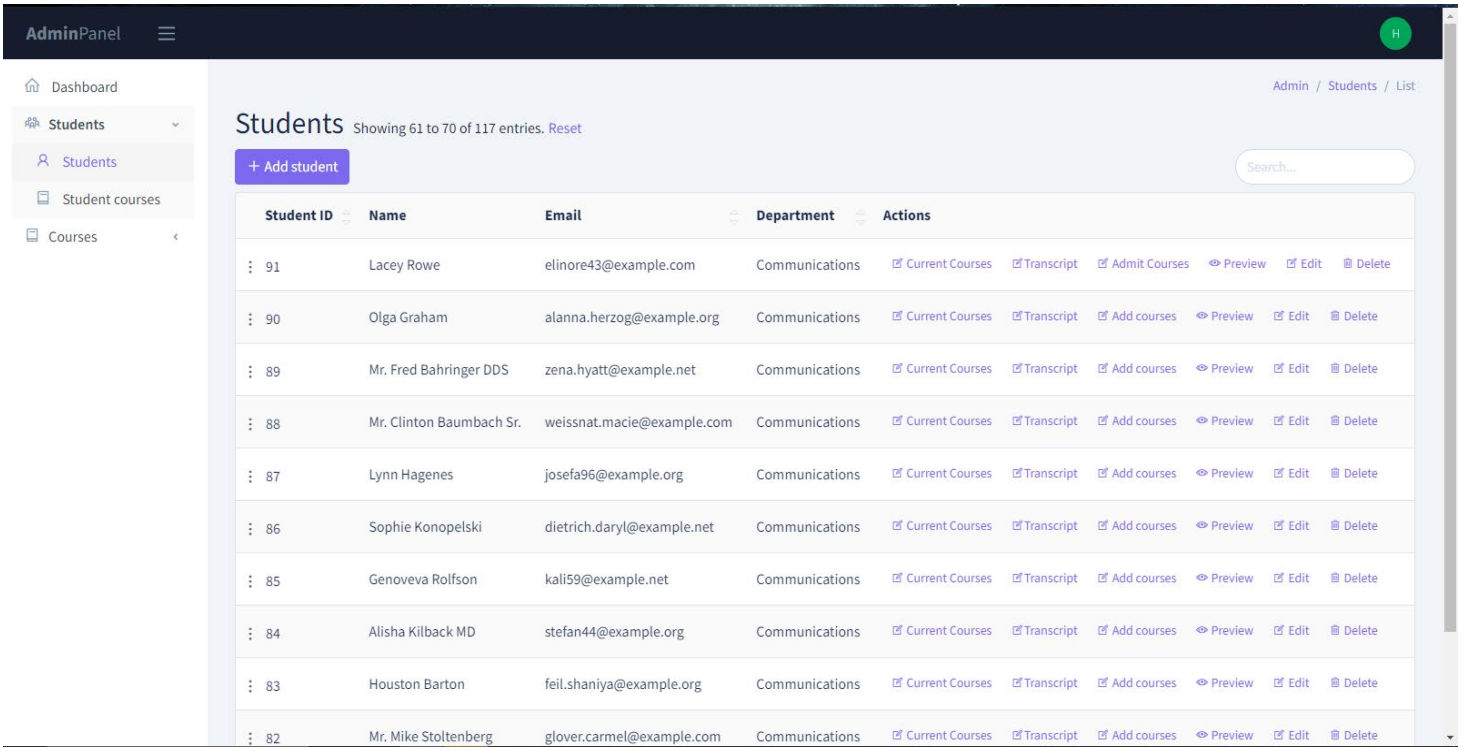


Figure 33: Academic advisor student's page

- Students' current courses

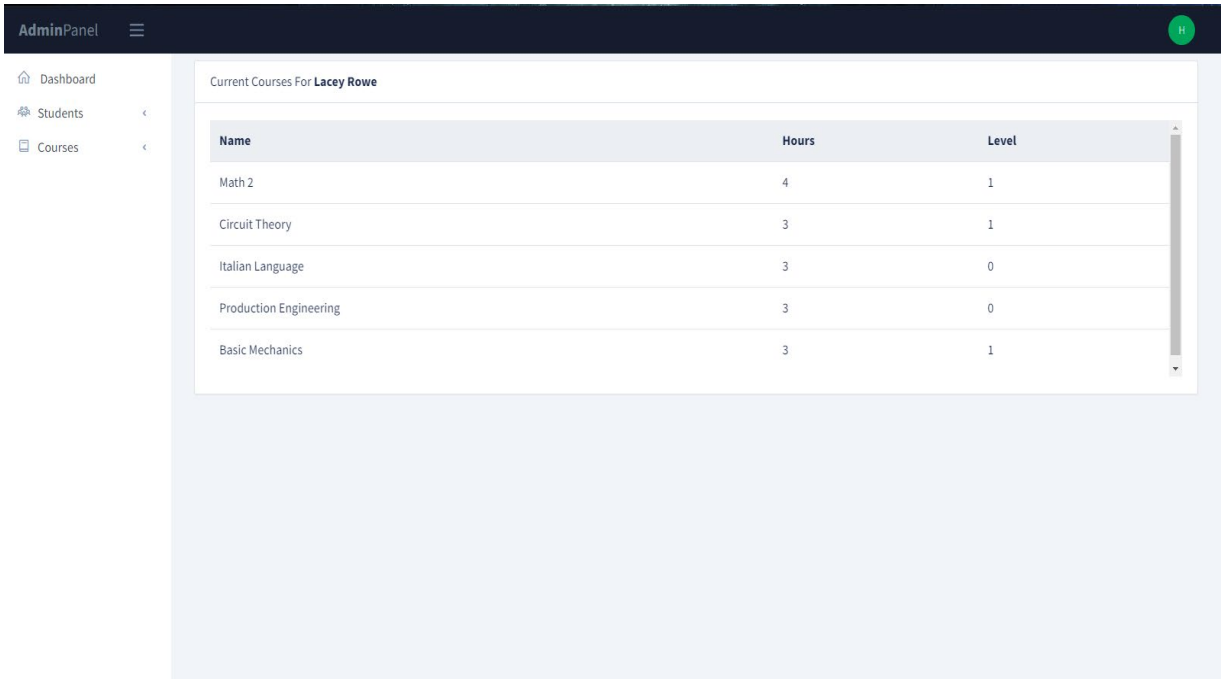


Figure 34: Academic advisor student current courses page

- Student transcript

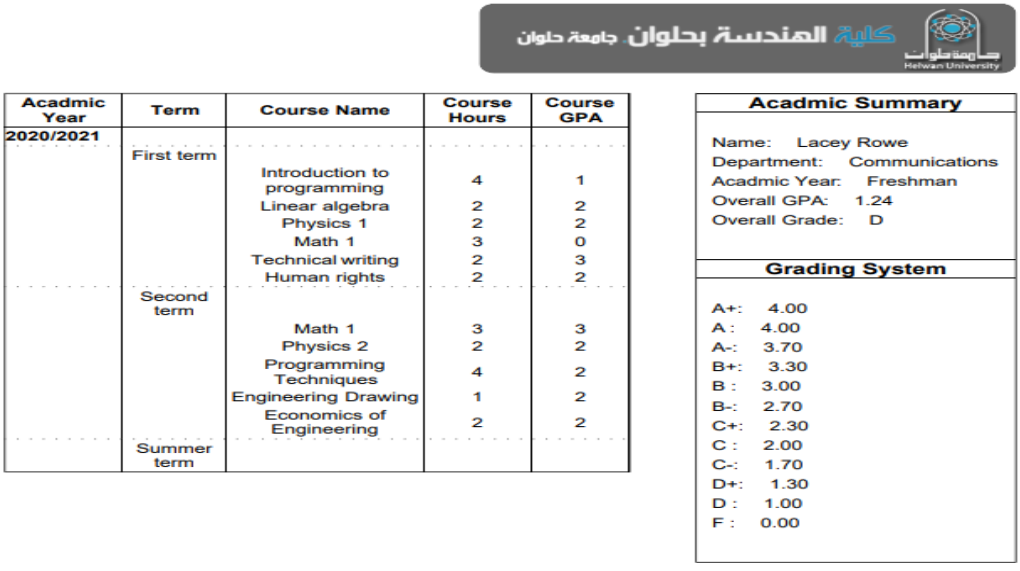


Figure 35: Student transcript

- Admit student registered courses.

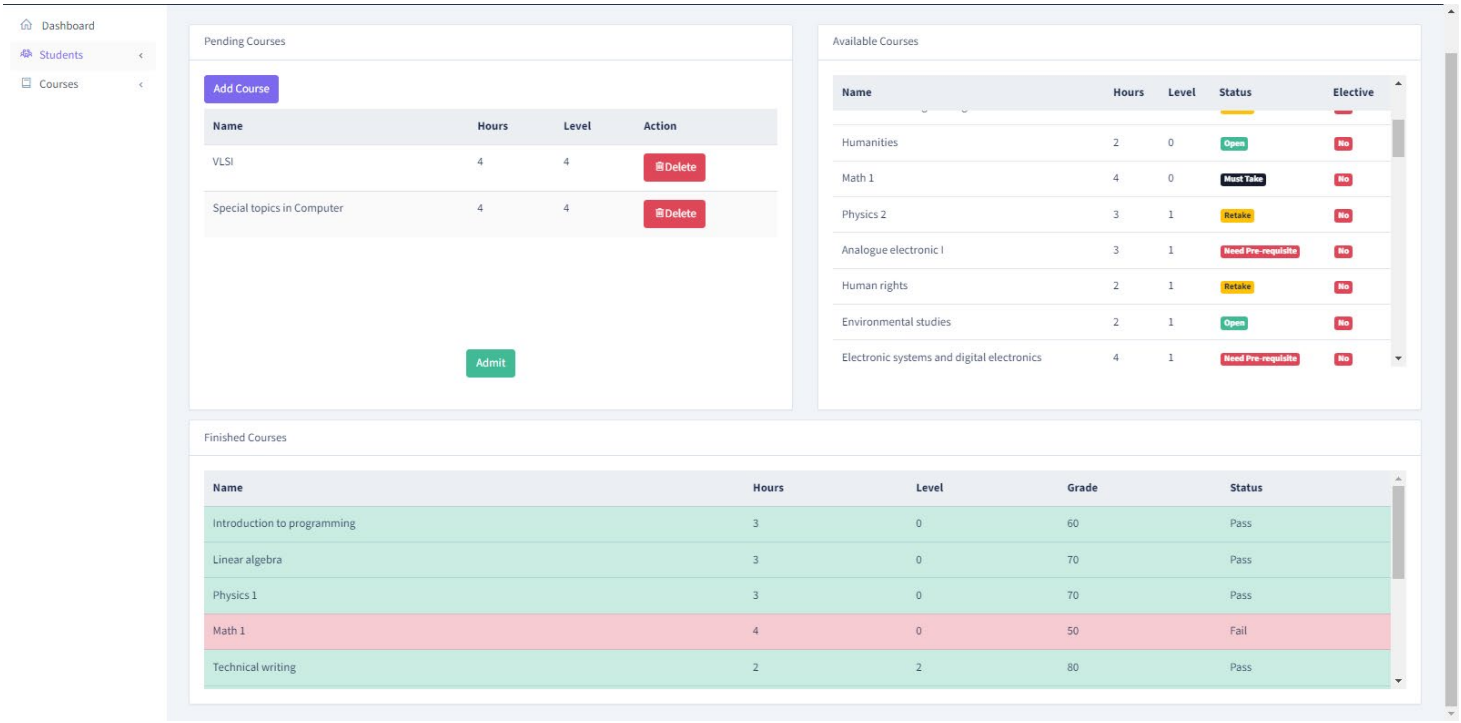


Figure 36: Academic advisor students course admission page

- Add student course.

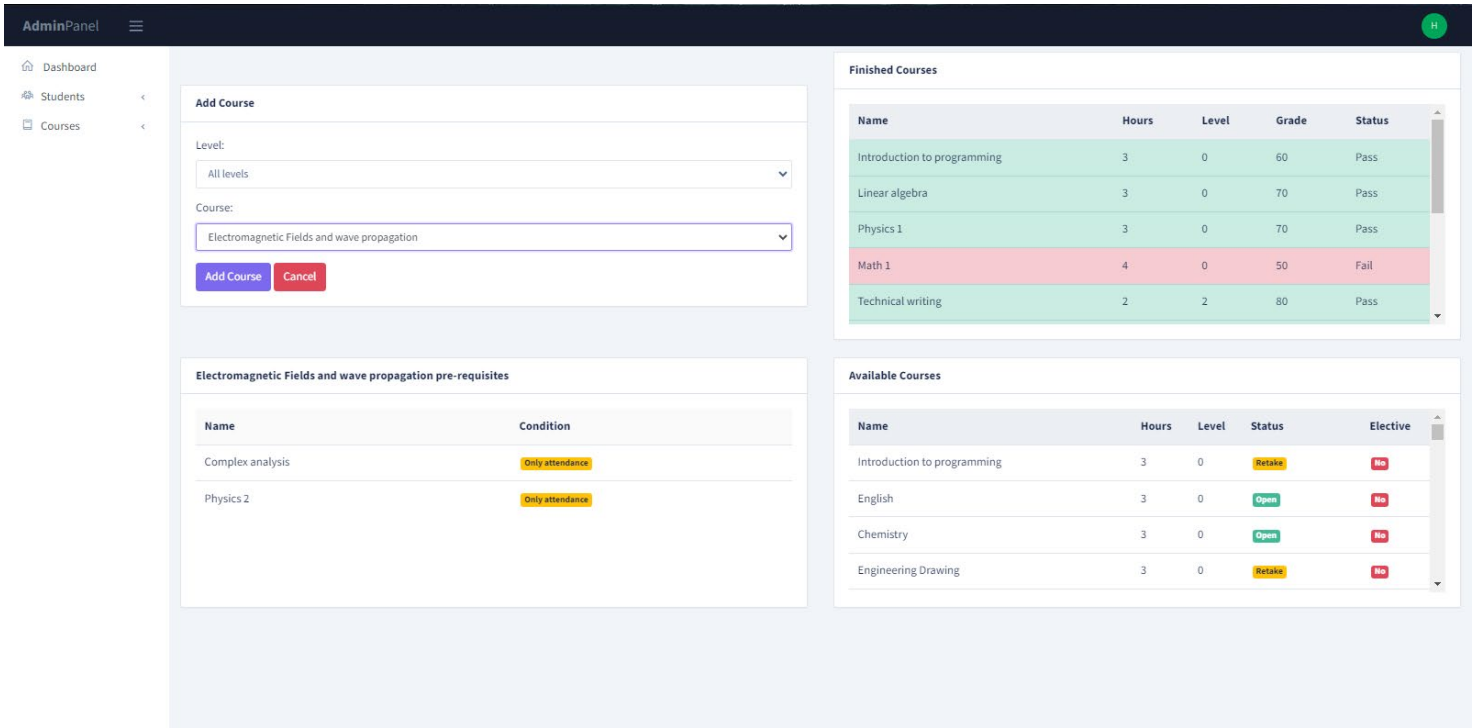


Figure 37: Academic advisor students course addition page

5.2.4 Admin View

- Admin Dashboard.

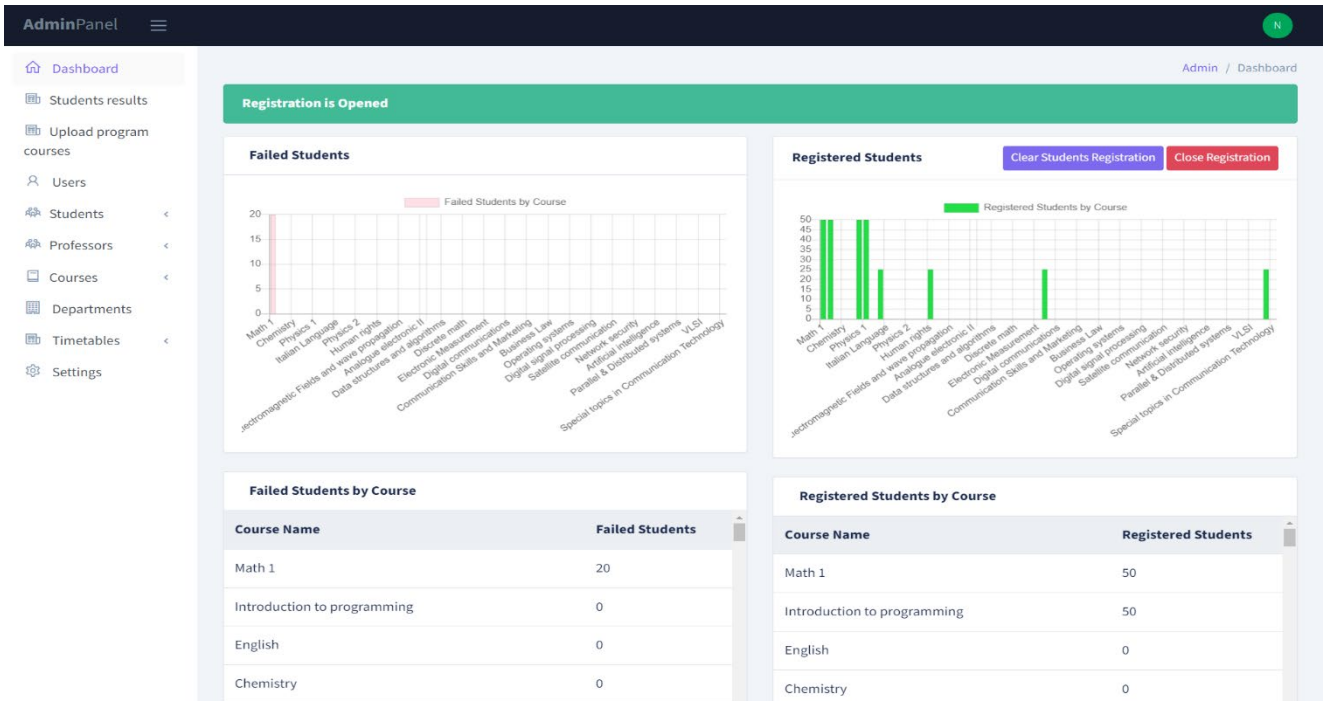


Figure 38: Admin dashboard

- Add/Edit Department with its regulation.

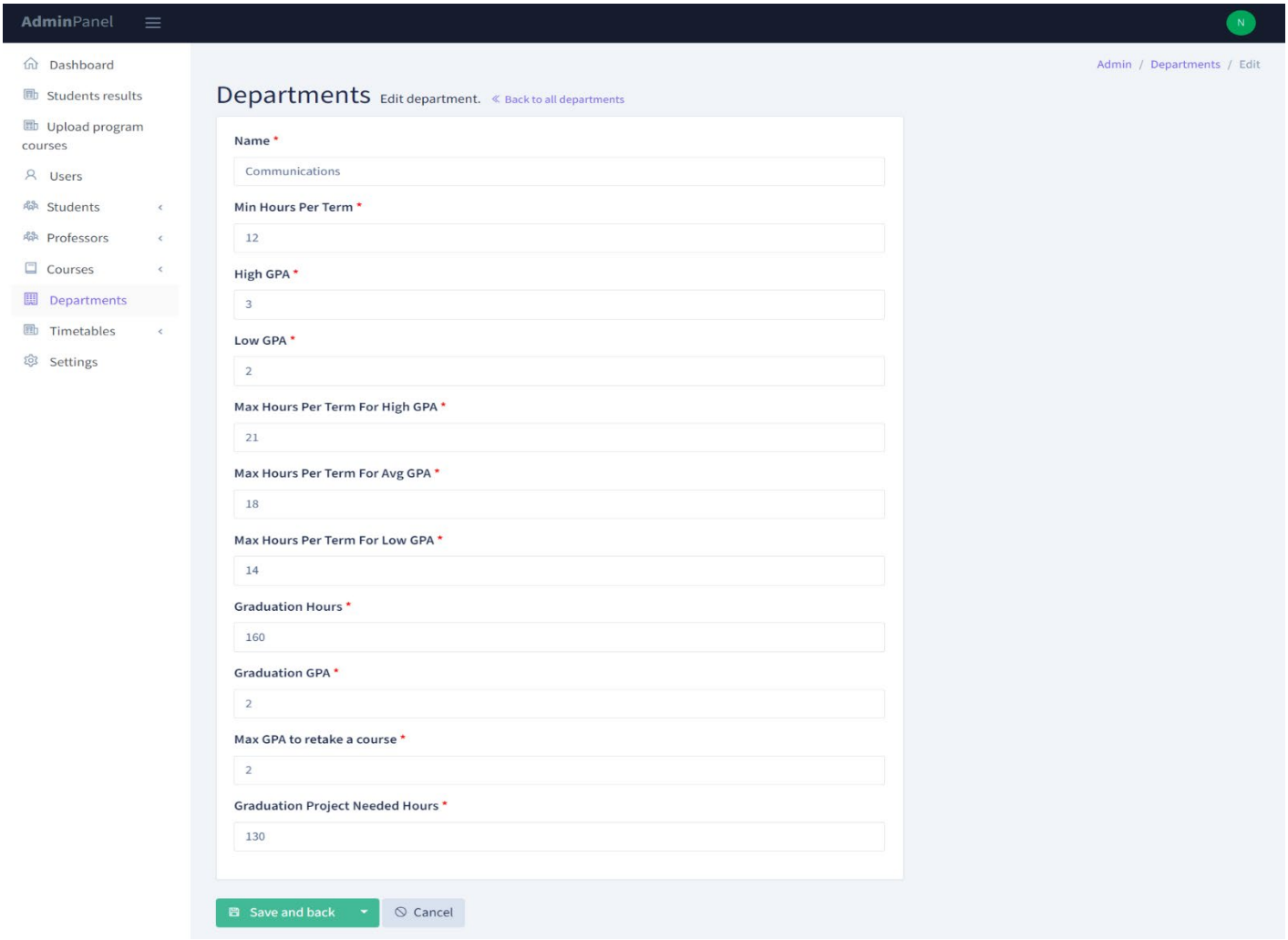


Figure 39: Admin department add/edit.

- Exam timetable generation

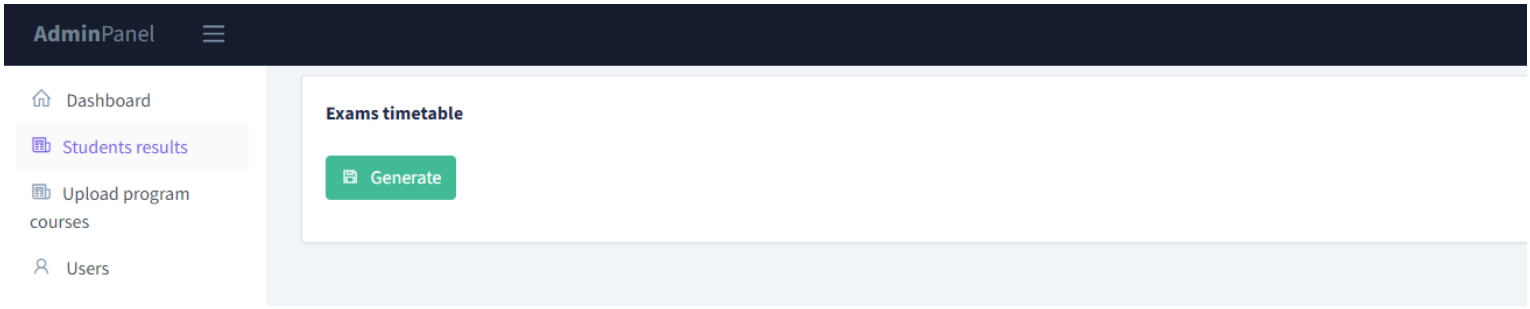


Figure 40: Exam timetable generation page

- Lecture timetable generation

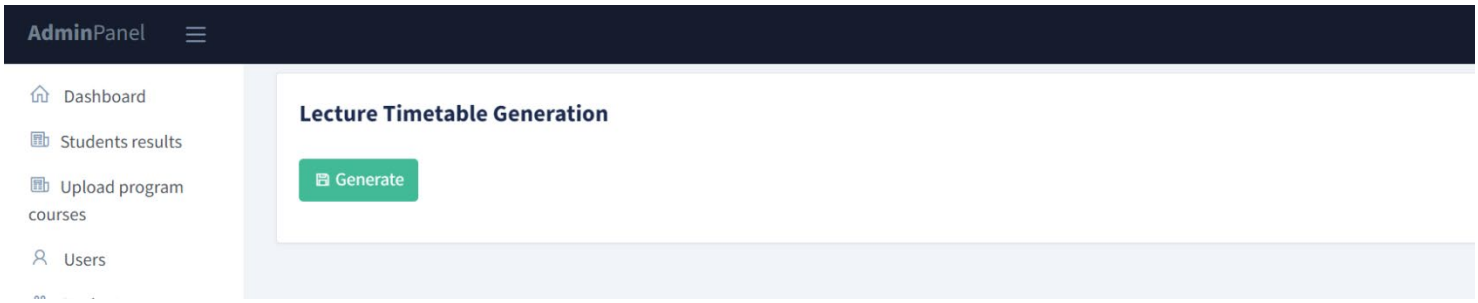


Figure 41: Lecture timetable generation page

- Timetable admission

Figure 42: Timetables admision page

- Upload regulation courses

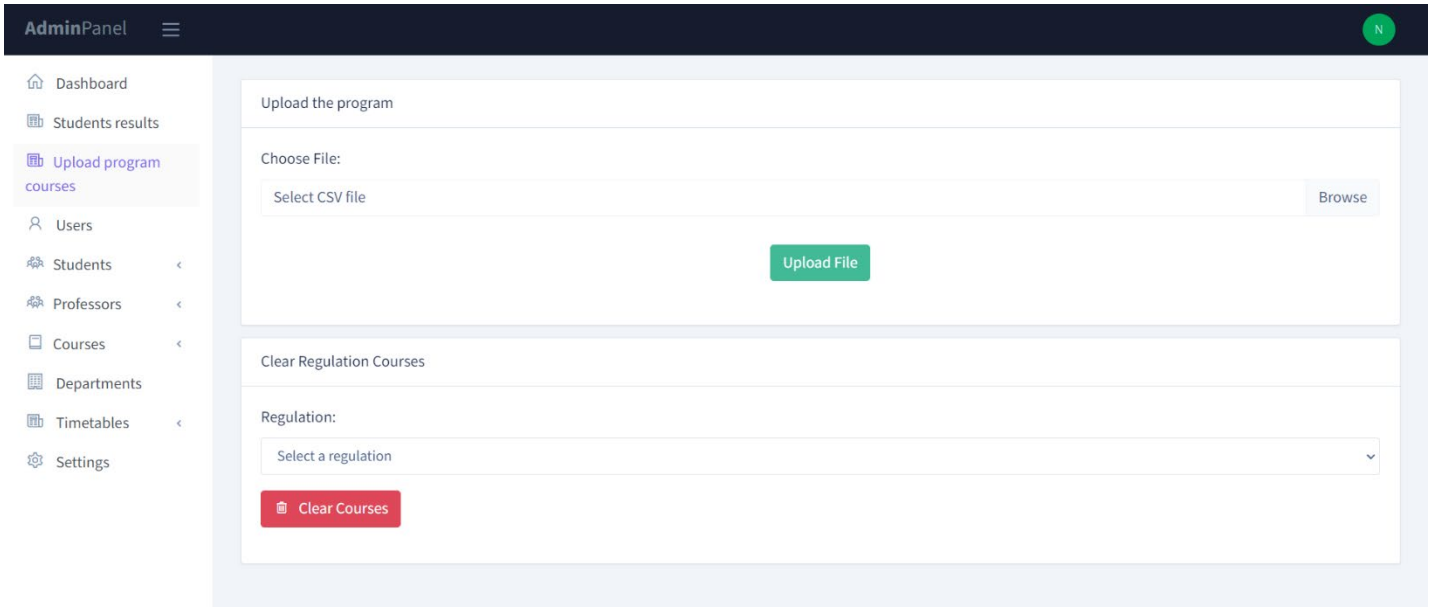


Figure 43: Upload regulation courses page

	A	B	C	D	E	F	G	H	I
1	course code	name	LectureHours	level	isElective	labHours	sectionHours	Departments	Pre-requists
2	GEN0801	Math 1	3	0	0	0	2	Communications	
3	GEN0802	Introduction to programming	2	0	0	2	0	Communications	
4	GEN0803	English	3	0	0	0	0	Communications	
5	GEN0804	Chemistry	3	0	0	0	0	Communications	
6	MEC0805	Engineering Drawing	1	0	0	0	4	Communications	
7	GEN0806	Linear algebra	2	0	0	0	2	Communications	
8	GEN0807	Physics 1	2	0	0	0	2	Communications	
9	MEC0808	Economics of Engineering	2	0	0	0	2	Communications	
10	GEN0809	Humanities	2	0	0	0	0	Communications	
11	GEN0810	Italian Language	3	0	0	0	0	Communications	
12	MEC0811	Production Engineering	2	0	0	0	2	Communications	
13	GEN1801	Math 2	3	1	0	0	2	Communications	GEN0801,GEN0806(A)
14	GEN1802	Physics 2	2	1	0	0	2	Communications	GEN0807
15	CIE1803	Analogue electronic I	2	1	0	2	0	Communications	GEN1802(A),POW1804(A)
16	POW1804	Circuit Theory	3	1	0	0	1	Communications	GEN0801(A),GEN0807(A)
17	GEN1805	Human rights	2	1	0	0	0	Communications	
18	GEN1806	Environmental studies	2	1	0	0	0	Communications	
19	CIE1807	Electronic systems and digital electronics	3	1	0	2	0	Communications	GEN1802(A),POW1804(A)
20	CIE1808	Electromagnetic Fields and wave propagation	3	1	0	0	2	Communications	GEN2804(A),GEN1802(A)
21	GEN1809	Basic Mechanics	2	1	0	0	2	Communications	GEN0801(A),GEN0807(A)
22	CIE1810	Programming Techniques	2	1	0	2	0	Communications	GEN0802(A)
23	CIE1811	Analogue electronic II	2	1	0	2	0	Communications	CIE1803(A)
24	GEN2801	Probability	2	2	0	0	2	Communications	GEN1801
25	CIE2802	Signal theory	2	2	0	0	2	Communications	GEN2801,GEN2804,POW1804(A)
26	CIE2803	Data structures and algorithms	2	2	0	2	0	Communications	GEN0802,CIE1810(A)
27	GEN2804	Complex analysis	2	2	0	0	2	Communications	GEN1801

Figure 44: Regulation courses CSV file

- Students' results admission.

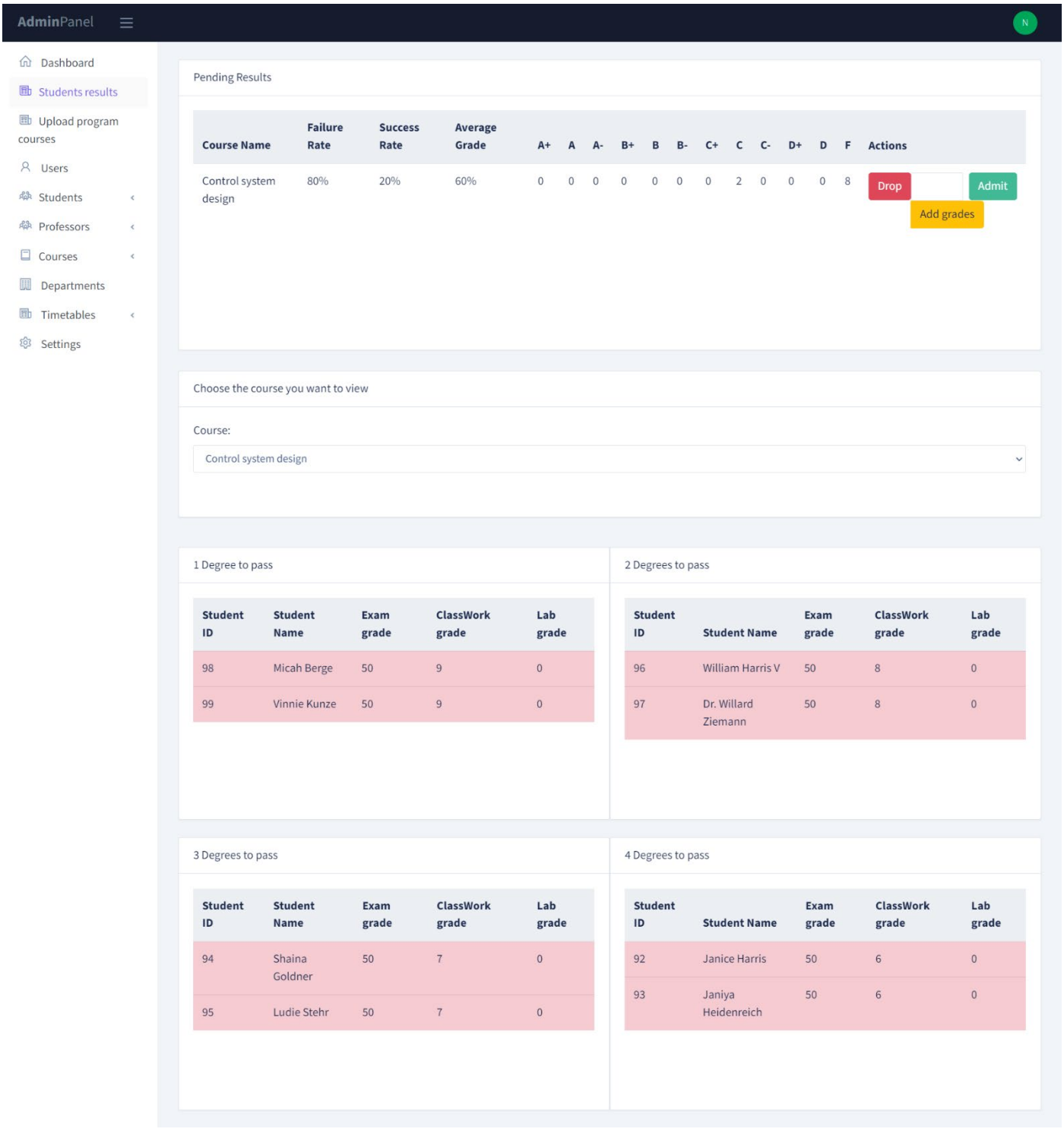


Figure 45: Admin students' results admission page

- Adding Professor available times

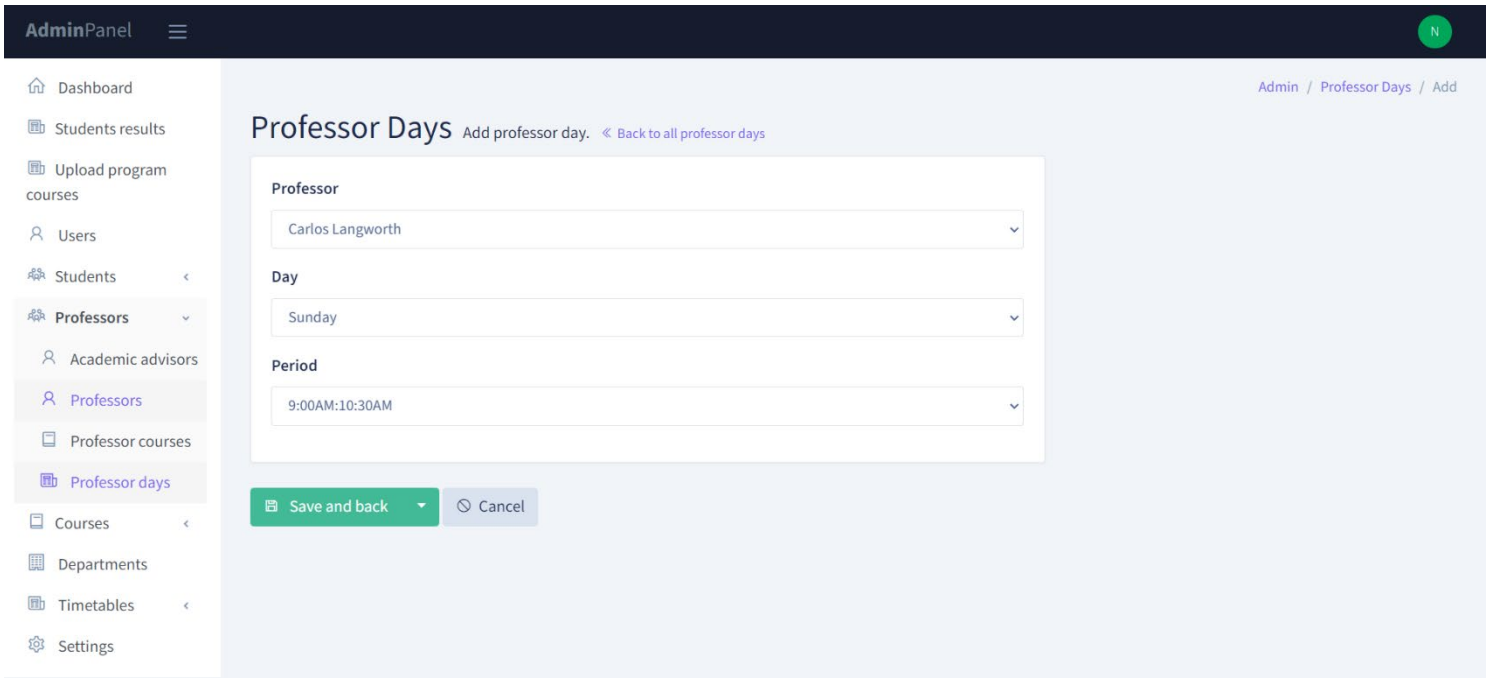


Figure 46: Professor available time addition page

5.2.5 Course registration algorithm

Course registration plays a crucial role in the academic journey of students. To streamline the process and ensure that students enroll in the appropriate courses, an efficient course registration algorithm is essential. This article presents a step-by-step approach to implementing a course registration algorithm, covering various scenarios such as retakes, closed courses, prerequisites, and determining course importance using the breadth-first search (BFS) algorithm.

- **Algorithm Steps:**
  - **Retrieve Department Courses from Database:**

The algorithm begins by retrieving the list of available courses for a specific department from the database. This ensures that students have access to the most up-to-date course offerings.

  - **Mark Courses as Retake:**

Check the student's academic history to identify courses that have been previously taken. Mark these courses as "retaken"

  - **Mark Courses as Closed:**

Review the availability of each course in real-time. If a course is no longer offered, mark it as "closed" to inform students that they cannot register for that course.

  - **Mark Courses Needing Prerequisites:**

Examine the prerequisites for each course and compare them with the courses the student has completed. If a student has not fulfilled the necessary prerequisites, mark those courses as "need-pre-requisite" to indicate that the student must complete the prerequisites before registering.

  - **Using BFS Algorithm, Determine Course Importance:**

Implement the BFS algorithm to determine the Importance of each course. Start with a course and explore its direct dependencies and prerequisites. Keep track of the number of courses that directly or indirectly rely on each course. Compare this count with the number stored in the database for that course. If the count is equal to or greater than the stored number, mark the course as "must-take" to indicate that it is a crucial course for the student's progression. Repeat this process for all courses in the department.

- **Mark Remaining Courses as Open:**

After considering retakes, closed courses, prerequisites, and must-take courses, any remaining courses are marked as "open." These courses are available for students to register.

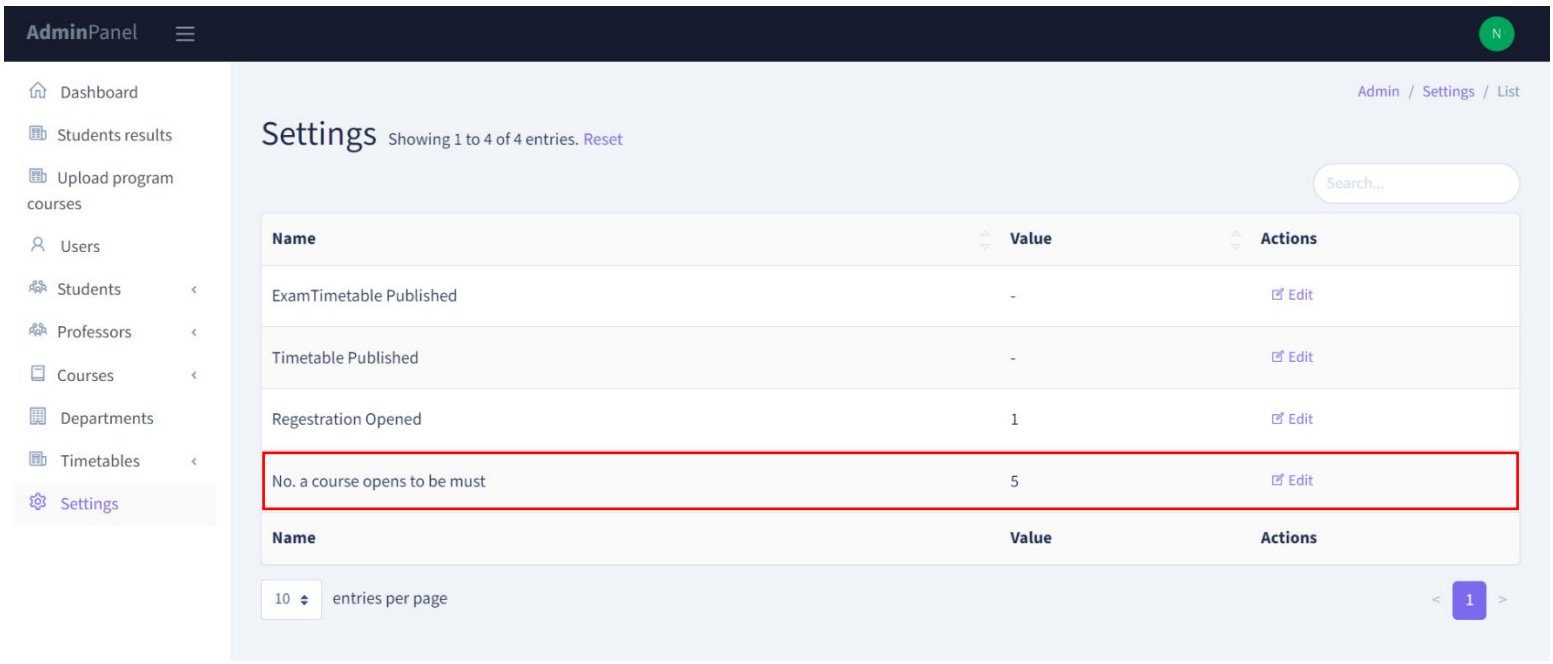


Figure 47: The number of courses to be a must in admin page.

Chapter 6

Testing

6.1 Unit testing

A Type of testing used to test separate functions and components black box testing included equivalence partitioning and determining set boundaries.

6.1.1 Test Cases:

- **Unit Testing of uploading regulation courses:**

Test Case Number	Input data (file)	Expected output
TC_#1	Valid file extension	Upload successfully
TC_#2	Invalid file extension	Error message
TC_#3	Invalid data inside the csv file	Error message and returns
TC_#4	empty	Error message

- **Unit Testing of fill students' results:**

Test Case Number	Input data	Expected output
TC_#1	Valid degrees	Upload successfully
TC_#2	Out of range degrees	Error message
TC_#3	Empty	Error message

- **Unit Testing of filling the system data:**

  Equivalence Classes:

  EC for Name:

  EC1: empty -> invalid (0 Character)
  EC2: numbers -> invalid
  EC3: alphanumeric -> valid (6 Character)
  EC4: alphabetic -> valid (6 Character)
  EC5: string invalid length -> invalid (Not 6 characters)

  EC for Values:

  EC1: empty -> invalid (0 Character)
  EC2: positive numbers -> valid
  EC3: alphanumeric -> invalid
  EC4: alphabetic -> invalid
  EC5: Symbols -> invalid
  EC6: negative numbers -> invalid

Test Case Number	Input data	Expected output
TC_#1	Valid name Valid values	Added successfully
TC_#2	Valid name Invalid values	Error message
TC_#3	Invalid name Valid values	Error message
TC_#4	Invalid name Invalid values	Error message

- **Unit Testing of Sign in:**

Equivalence Classes:

EC for Email:

EC1: empty -> Invalid (0 Character)
EC2: numbers -> Invalid
EC3: alphanumeric -> Invalid
EC4: email structure -> valid (1 ->50 Character)
EC5: email structure invalid length -> invalid (>50 Character)
EC6: Symbols -> invalid

EC for Password:

EC1: empty -> invalid
EC2: alphanumeric but not correct -> invalid.
EC3: Correct Password -> valid.

Test Case Number	**Email**	**Password**	**Output**
TC_#1	Empty	Empty	Error
TC_#2		Incorrect Alphanumeric	Error
TC_#3		Correct Password	Error
TC_#4	Numbers	Empty	Error
TC_#5		Incorrect Alphanumeric	Error
TC_#6		Correct Password	Error
TC_#7	Alphanumeric	Empty	Error
TC_#8		Incorrect Alphanumeric	Error
TC_#9		Correct Password	Error
TC_#10	Email structure (Correct)	Empty	Error
TC_#11		Incorrect Alphanumeric	Error
TC_#12		Correct Password	**Login**

TC_#13	Email structure invalid length	Empty	Error
TC_#14		Incorrect Alphanumeric	Error
TC_#15		Correct Password	Error
TC_#16	Symbols	Empty	Error
TC_#17		Incorrect Alphanumeric	Error
TC_#18		Correct Password	Error

- **Unit Testing of Sign up:**

EC for Name:
EC1: empty -> Invalid (0 Character)
EC2: numbers -> Invalid
EC3: alphanumeric -> Invalid
EC4: Symbols -> Invalid
EC5: Alphabetic -> Valid

EC for Profile picture:
EC1: image file -> valid
EC2: non-image file -> Invalid
EC3: empty -> Invalid

Test Cases Considering Name, Profile picture are Valid:

Test Case Number	**Email**	**Password**	**Output**
TC_#1	Empty	Empty	Error
TC_#2		Numbers	Error
TC_#3		Alphanumeric	Error
TC_#4		Symbols	Error
TC_#5	Numbers	Empty	Error
TC_#6		Numbers	Error
TC_#7		Alphanumeric	Error
TC_#8		Symbols	Error

TC_#9	Alphanumeric	Empty	Error
TC_#10		Numbers	Error
TC_#11		Alphanumeric	Error
TC_#12		Symbols	Error
TC_#13	Email structure (Correct)	Empty	Error
TC_#14		Numbers	Error
TC_#15		Alphanumeric	Register
TC_#16		Symbols	Error
TC_#17	Email structure invalid length	Empty	Error
TC_#18		Numbers	Error
TC_#19		Alphanumeric	Error
TC_#20		Symbols	Error
TC_#21	Symbols	Empty	Error
TC_#22		Numbers	Error
TC_#23		Alphanumeric	Error
TC_#24		Symbols	Error

Chapter 7

Tools and Technologies

7.1 Programming languages

- **PHP**

  PHP is a general-purpose scripting language geared towards web development. It was originally created by Danish-Canadian programmer Rasmus Lerdorf in 1994. The PHP reference implementation is now produced by The PHP Group.

Figure 48: PHP logo

- **Python**

  Python is an interpreted high-level general-purpose programming language. Python's design philosophy emphasizes. code readability with its notable use of significant indentation.

Figure 49: Python logo

- **JavaScript**

  JavaScript is a versatile programming language that powers dynamic and interactive web applications, allowing developers to create engaging user experiences. With its widespread adoption and robust ecosystem, JavaScript continues to be a cornerstone of modern web development.

Figure 50: JavaScript logo

- **SQL**

  SQL is a domain-specific language used in programming and designed for managing data held in a relational database management system, or for stream processing in a relational data stream management system.

Figure 51: SQL logo

7.2 Frameworks & Technologies

- **Laravel**

  Laravel is a popular PHP framework known for its elegant syntax and powerful features, making web development a breeze for developers. With its robust ecosystem and built-in tools, Laravel enables rapid application development, allowing developers to focus on creating innovative and scalable solutions.

Figure 52: Laravel logo

- **Backpack**

  Laravel Backpack is a powerful administration panel for Laravel applications, providing a quick and easy way to build custom backends. With its intuitive interface and extensive features, it streamlines development tasks and enhances productivity for developers.

Figure 53: Laravel backpack logo

- **React**

  React is a powerful JavaScript library for building user interfaces, enabling developers to create interactive and dynamic web applications with ease. By utilizing a component-based approach, React promotes reusability and modularity, making it a popular choice for creating scalable and efficient front-end applications.

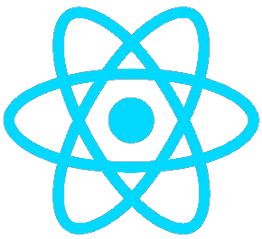


Figure 54: React logo.

- **MySQL**

  MySQL is a popular open-source relational database management system (RDBMS) known for its robustness and scalability, offering efficient data storage and retrieval capabilities.

Figure 55: MySQL logo

7.3 Tools

- **Visual Studio Code**

  Visual Studio Code is a popular source code editor developed by Microsoft, offering a versatile and customizable platform for developers. With its extensive plugin ecosystem and intuitive interface, it empowers programmers to efficiently write, debug, and collaborate on a wide range of programming languages and projects.

Figure 56: Visual studio code logo

- **Juypter**

  Jupiter, also known as Jupyter, is a popular open-source interactive computing environment for Python."
  "With its rich ecosystem of extensions and support for various programming languages, Jupiter enables data exploration, analysis, and visualization in an interactive and collaborative manner.

Figure 57: Jupyter logo

- **XAMPP**

  XAMPP is a powerful web development solution that combines Apache, MySQL, PHP, and Perl into a single easy-to-use package, providing a convenient way to set up a local web server environment.

Figure 58: XAMPP logo

- **Postman**

  Postman is a powerful API testing and development tool that simplifies the process of building, documenting, and testing APIs. It offers a user-friendly interface and robust features, making it a popular choice among developers worldwide.

Figure 59: POSTMAN logo

Chapter 8

Conclusion and future improvements

8.1 Conclusion

The Student Information System (SIS) project is a transformative platform that revolutionizes course registration in educational institutions. With its user-friendly interface, the SIS system streamlines academic management, simplifies registration, and automates administrative tasks.

Students benefit from a seamless registration experience, real-time course updates, personalized schedules, and exam timetables. Professors enjoy automated administrative processes, access to up-to-date student information, and effective communication tools. Administrators benefit from simplified course admission management, optimized scheduling, streamlined administrative tasks, and valuable reporting and analysis capabilities.

By centralizing student information and promoting effective communication, the SIS system creates an enriched learning environment. It empowers students to take ownership of their academic journey, enables professors to focus on teaching, and equips administrators with valuable insights for data-driven decision-making.

In conclusion, the SIS project transforms traditional academic management practices, making the course registration process efficient, effective, and user-friendly. It empowers students, professors, and administrators, fostering a harmonious and technologically advanced educational ecosystem.

8.2 Future work

In terms of future work, there are several key areas for improvement and enhancement in the Student Information System (SIS) project. Firstly, the admin panel can be further improved by adding more functionality. This can involve expanding the administrative toolkit to include additional features such as advanced reporting capabilities, analytics dashboards, and tools for managing student records and academic resources. By enhancing the admin panel, administrators will have more comprehensive and efficient control over the system, enabling them to make data-driven decisions and streamline their administrative tasks.

Another area of future work involves incorporating teaching assistants into the timetable generation model. Currently, the system focuses on generating schedules based on course availability and professor preferences. By extending the model to include teaching assistants, the system can allocate suitable time slots for their involvement in specific courses. This enhancement will not only optimize resource allocation but also ensure a balanced distribution of workload among teaching staff. It will enhance the overall teaching and learning experience by leveraging the expertise of teaching assistants effectively.

Moreover, system security is of paramount importance for safeguarding student information and maintaining the integrity of the SIS project. Future work should focus on improving the system security measures by implementing robust authentication and authorization mechanisms, encrypted data storage, and regular security audits. Additionally, integrating multi-factor authentication and monitoring tools for detecting and responding to potential security breaches will enhance the overall system resilience and protect sensitive data from unauthorized access.

Chapter 9

References

References

[1] TomášMüller. “Real-life examination timetabling”. In: Journal of Scheduling 19.3 (June 2016), pp. 257–270. ISSN: 1099-1425. DOI: 10.1007/s10951-014-0391-z.

[2] TimB. Cooper and Jeffrey H. Kingston. “The Complexity of Timetable Construction Problems”. In: 1153 (Mar. 1995)

[3] L. Lalescu. FET - Free Timetabling Software. Retrieved on May 9, 2018.

[4] Andrei Bautu and Elena Bautu. “PRACTICAL ASPECTS ON AUTOMATIC GENERATION OF UNIVERSITY TIMETABLES–A CASE STUDY”. In: (2015).

[5] UniTime. UniTime | University Timetabling. Retrieved on May 9, 2018.

[6] Z. Michalewicz, Genetic Algorithms+Data Structures = Evolution programs, second edition, Springer-Verlag Heidenberg New York, 15 (1996). (ISBN 3-540-60676-9).

[7] Rosen, Kenneth H. (2012). Discrete Mathematics and Its Applications (7th ed.). New York: McGraw-Hill. p. 119. ISBN 978-0-07-338309-5.

[8] D. Shiffman, The Nature of Code, Chapter 9, 2012, Creative Commons, 444 Castro Street, Suite 900, Mountain View, California 94041, USA. ISBN-13: 978-0985930806.

[9] Z. Michalewicz, Genetic Algorithms+Data Structures = Evolution programs, second edition, Springer-Verlag Heidenberg New York, 15 (1996). (ISBN 3-540-60676-9).

[10] A. Marczyk, Genetic Algorithms and Evolutionary Computation, (2004).

[11] N. M. Razali, John Geraghty, Genetic Algorithm Performance with Different Selection Strategies in Solving TSP, Proceedings of the World Congress on Engineering, Vol II, WCE 2011, July 6 - 8, (2011), London, U.K.

[12] A. Jain, S. V. Chande, International Journal on Computational Science & Applications (IJCSA) 5(6), (2015).

[13] B. L. Miller, D. E. Goldberg, Genetic Algorithms, Tournament Selection, and the Effects of Noise, Complex Systems 9, 193- 212 (1995).

[14] H. Xie, M. Zhang, Tuning Selection Pressure in Tournament Selection, Technical Report Series, School of Engineering and Computer Science, Victoria University of Wellington, New Zealand (2009).

[15] P. J. Fleming, R. C. Purshouse (2002), Control Evolutionary Practice, 10: 1223-1241.

[16] J. Koza, M. Keane, M. Streeter, W. Mydlowec, J. Yu, G. Lanza, Genetic Programming IV: Routine Human-Competitive Machine Intelligence, Kluwer Academic Publishers, 35, (2003).

[17] D. Graham-Rowe, "Radio Emerges from the Electronic Soup.", New Scientist, 175, 19, (2002).