System Analysis and Design Report

Online Application & Student Management System

Angelo State University

BSc (Hons) Software Engineering
System Analysis and Design

Prepared By: Lovena Ujoodha 2110_22856
Ishwari Veeraragoo 2110_22858

Submission Date: 14 February 2023

Table of Contents

1. Introduction

A student management system also known as student information management system is a software which gives a helping hand for educational institutions to coordinate their students' data securely. It encapsulates features such as grading, attendance tracking and many more. Many educational establishments are implementing this new system to computerize their administrative processes and to enable better communication with their students. Moreover, this concept eases the lives of students' since it provides simple and clear data input methods, accessibility as well as end to end functionality. Additionally, SIS is very eco-friendly as tree do not need to be cut to record paper-based data.

1.1 System Analysis

System analysis is a thorough examination of the numerous tasks carried out by a system and the connections between those tasks both inside and outside the system. It entails investigating and evaluating how a company now generates information by processing data.

Data was gathered during study on the numerous files, decision-making processes, and transactions handled by the current system. The system's effectiveness mostly depends on how precisely the problem is stated, how thoroughly it is examined, and how well the solution chosen is implemented. A good analysis model should offer both the framework for the solution and the methods for comprehending the problem. As a result, it needs to be thoroughly investigated by gathering system data. The suggested system should then be carefully examined considering the requirements.

System analysis can be categorized into four parts.

- ✓ System planning and initial investigation
- ✓ Information Gathering
- ✓ Applying analysis tools for structured analysis
- ✓ Feasibility study
- ✓ Cost/ Benefit analysis.

1.2 System Design

The process of translating requirements into a software representation is known as system design. At first, the depiction shows a comprehensive picture of software. A design representation that is extremely like source code results from further refining. Design is an area of software development where quality is promoted. The only way to accurately translate customer needs into a finished software product or system is through design, which gives us a representation of software that can be evaluated for quality. All subsequent processes in software engineering and maintenance are built upon the foundation provided by system design.

2. Preliminary investigation

2.1 Study of existing system

In the current system we need to keep several records related to the students and applicants want to enter the details of the student, and the applicant manually. Only a teacher or the school authority can view the mark of the student, and they want to input the details of the student. This procedure takes too much time and is costly.

Today in university student details are entered manually. To keep each students' details in a specific and separate record is not an easy task and as a result there can be a risk for manual errors.

1. When the student comes in university.
2. First, he/she takes admission form from reception.
3. Fills it and submits it into office.
4. Filled form is first checked with documents like merit list a details came from university and verified by an official person, if there is any mistake then it is corrected.
5. At the time of submission of it the fees are deposited by the candidate.
6. At the time of submission of admission forms admission ID is assigned to the candidate by the university.
7. Candidate gets the receipt of fees deposition.
8. Enter details of students
9. Enter students' grades
10. Manage programmes and modules
11. Manage cohorts and attendance

Disadvantages of Current System: -

1. Require much manpower i.e., many efforts, much cost and hard to operate and maintain.
2. Since, all the work is done in papers, so it is very hard to locate a particular student record when it is required.
3. It is time consuming.

2.2 The Interview

We have carried out an interview whereby we have asked the interviewee a series of prepared questions so that we can have a better and clear understanding of the university's functionality.

Interview guide:

Interviewee: Mr Thomas Franklin (Owner of the University of Angelo)	**Interviewer**: Ujoodha Lovena Veeraragoo Ishwari
Location / Medium: Through a video meeting on Skype	**Appointment Date**: 05th December 2022 Start Time: 18:30 End Time: 19:30
Main objective: To have a clear understanding about the university's daily operations (administrative and in class)	**Important reminders**: Admission process Management of students' data

Agenda	Duration
Introduction	1 minute
Background on Project	2 minutes
Overview of Interview Topics to be covered Request permission to record	1 minute
What are the services of education your university provide?	5 minutes
How does the university manage students and applicants' records?	10 minutes
What are some of the drawbacks you/ your employees have come across on a daily basis?	5-10 minutes
Have you ever thought of some solutions. Elaborate, if yes	5 minutes
Why do you want us to computerize this system?	5 minutes
What are your overall expectations of the computerized system?	5 minutes
Summary of major points	2-5 minutes
Open questions from interviewee	2-5 minutes
Closing	1 minute

Questions	Answers
1. When was the university established and first accessible to students?	18th November 1928
2. Approximately how many students enroll to your university yearly?	Approximately 5000 students but we get new numbers each year
3. What is the usual the procedure for an admission?	Firstly, the applicant takes a form the reception, fills it and then submit it to the office. The filled form is then checked by a person to see if there are any error and if the applicant is eligible for his/her chosen course.
4. How are the attendance of students recorded?	The attendance is recorded by the lecturer and each individual student have to sign in a hardcopy book.
5. How are the results of students proclaimed?	For each students their marks for a particular module are recorded in a hardcopy book. Once all marks are checked by a university authority, the GPA is calculated and then generate a hardcopy transcript and give it to the respective student.
6. How are the records/ details of the students managed?	The details of students are recorded in a hardcopy book
7. What is the procedure for the students when they pay for their semester fees?	They have to collect a cash deposit voucher from the office and deposit the cash through the bank. However, we now offer internet banking, so they can make the transaction online
8. What happens when you have to make a modification in the record of one student?	We change the details with some correction fluid and a pen. We also have to make sure we have updated the details other places also so as to avoid any confusion
9. What are some of the drawbacks you/ your employees have come across on a daily basis?	Very often we have made some errors while performing data entry during the registration process. We also find that our way of managing students' records is very exhausting and time consuming.
10. Have you ever thought of some solutions, besides computerizing the system Elaborate, if yes	Not really
11. Why do you want us to computerize this system?	Mainly to have a proper control of the university which also includes administrative work. Also, each year the number of students enrolling in our university keeps on increasing and hence to manage all their data is tedious task!
12. What are your overall expectations of the computerized system?	We would like to have a software system where we can manage students' data, results. Also, we want the system accessible to students, lecturer, programme coordinator and module coordinator as well. So that for example our lecturer can post assignments and coursework and students can easily view it. This system management will be very helpful especially during this pandemic situation.

2.3 Proposed System

In our proposed system we have the idea for adding the required details of the students by themselves. So, the overhead of the school authorities and the lecturers is becoming less. Another major con of the system is that it is very simple to modify the details as well as delete with clear instructions given throughout the system. The marks of the student are added in the database and so students can also view the marks for each module whenever they want. The system is also to automate the process carried out in the organization with improved performance and realize the vision of paperless admission. Programme coordinator and module coordinator can update details of modules and programmes whenever they want.

Our proposed system has several advantages: -

- User friendly interface
- Fast access to database
- Less error
- More Storage Capacity
- Search facility
- Look and Feel Environment
- Quick transaction

3. Feasibility Study

The conduction of a feasibility study plays a very crucial role in the proper development of a software system. It helps us to acknowledge the fact that whether the developing software will be feasible or not. Feasibility study is divided into three categories namely, economic feasibility, technical feasibility, and lastly operational feasibility. By going through these three steps, the management will be much more convinced if the project is worth or that it will be just a waste of time.
For our project, the online application and student management system, we will focus more on the technical and economic feasibility study.

3.1 Technical feasibility

According to our project we can firmly say that it is technically feasible and here is why. First and foremost, we will not have the issue of getting the resources required for the development of the software since almost all of them are already available by the client and are in good conditions.
Below are the hardware and system requirements as well as some additional tools that our system will need.

Hardware Requirements:

Processor	Intel P-IV System
RAM	128 MB
Hard disk	20 GB
Ethernet Connection	Wireless Adapter (Wi-Fi); 15 mbps
Processor Speed	Minimum 250 MHz

Software Requirements:

Operating System	Microsoft Windows / Mac OS/ Linux OS
Language	Java/ C++
Database	My SQL

Additional tools:

- HTML
- CSS
- Visual Studio / NetBeans

3.2 Economic feasibility

The economic feasibility relates mainly to the cost involved for the proposed solutions and the potential benefits in terms of tangible and intangible, it can bring to the project. The economic feasibility seems to be very blooming for our system. First, only a reasonable amount of fund is required for its development, since we are using resources which are already available. However, an additional task to be done is a proper environment with a well inspected supervision. This will help to attain maximum usability of the resources. Additionally, after the development of the project, the client will have to have to do proper maintenance of the software and hardware equipment to keep in good condition.

Intangible benefits the organization will encounter:

- More timely information
- Better and modern design interface hence increase satisfaction of users
- Much better initiative (going green)
- Much better organizational planning

One time cost details:

Development Cost	Rs 110,000
Existing Resources (Hardware/ Software)	Rs 30,000
User Training	Rs 4,000
Total-Cost	**Rs 144,000**

Tangible Benefits

Cost reduction or avoidance	Rs 20,000
Increase speed of activity	Rs 200,000
Improve in management planning	Rs 140,000
Total-Cost	**Rs 360,000**

3.3 Operational Feasibility

Operational feasibility refers to what extend does the proposed system, solves the problems that may arise. Our system is a very modern looking and interactive interface with soothing colors, which as a result will make any system user feel more comfortable and easier to work into. Furthermore, the system helps the user to have a much better user experience. We also provide clear and simple instructions on the system to encourage error-free data entry. Throughout our system we have tried to keep minimum mouse clicks to make it have a more professional pace.

4. List of features

Use Case	Actor
Add Student	Administrator
Update Student	Administrator
Delete Student	Administrator
Search Student by Id, Names	Administrator, Lecturer
Add Programme	Programme Coordinator
Update Programme	Programme Coordinator
Delete Programme	Programme Coordinator
Search Programme by Name, Id, Module	Student, Programme Coordinator, Lecturer
Add Module	Module Coordinator
Update Module	Module Coordinator
Delete Module	Module Coordinator
Search Module by Name, Programme	Programme Coordinator, Module Coordinator, Student, Lecturer
Add Cohort	Programme Coordinator
Update Cohort	Programme Coordinator
Delete Cohort	Programme Coordinator
Search Cohort by Id, name, Student Name	Programme Coordinator
Add Lecturer	Module Coordinator
Update Lecturer	Module Coordinator
Delete Lecturer	Module Coordinator
Search Lecturer by Names, Module, Programmes	Student, Module Coordinator, Programme Coordinator
Add Coursework	Lecturer, Module Coordinator
Update Coursework	Lecturer, Module Coordinator
Delete Coursework	Lecturer, Module Coordinator
Search Coursework by date, Description, Lecturer, Student	Module Coordinator, lecturer
Add Resit Coursework	Lecturer, Module Coordinator
Update Resit Coursework	Lecturer, Module Coordinator
Delete Resit Coursework	Lecturer, Module Coordinator
Search Resit Coursework by date, Description, Lecturer, Student	Lecturer, Module Coordinator,
Add Exam	Faculties
Update Exam	Faculties
Delete Exam	Faculties
Search Exam by Date, Module, Student, Programme, Cohort	Students, Module Coordinator
Add Resit Exam	Module Coordinator, Programme Coordinator
Update Resit Exam	Module Coordinator, Programme Coordinator
Delete Resit Exam	Module Coordinator, Programme Coordinator
Search Resit Exam by Date, Module, Student, Programme, Cohort	Module Coordinator, Programme Coordinator, Students
Add Work Placement	Programme Coordinator
Update Work Placement	Programme Coordinator
Delete Work Placement	Programme Coordinator
Search Work Placement	Programme Coordinator
Add Project	Module Coordinator, Lecturer
Update Project	Module Coordinator, Lecturer
Delete Project	Module Coordinator, Lecturer
Search Project	Module Coordinator, Lecturer
Post Coursework	Module Coordinator, Lecturer
Post Exam	Module Coordinator, Lecturer
Post Project	Module Coordinator, Lecturer
Un-post Coursework	Module Coordinator, Lecturer
Un-post Exam	Module Coordinator
Un-post Project	Module Coordinator, Lecturer
Moderate Coursework	Lecturer, Module Coordinator
Moderate Examination	Module Coordinator
Moderate Project	Module Coordinator, Lecturer
Generate Board of Examiners Report	Programme Coordinator
Update Exam after BOE	Module Coordinator
Update Project after BOE	Module Coordinator
Update Coursework after BOE	Module Coordinator
Generate Result	Faculties
Check Result by Student	Faculties
Generate Transcript	Faculties
Add attendance	Lecturer
Update attendance	Lecturer

Delete attendance	Lecturer
Search attendance by students' ID	Lecturer, Module Coordinator
Create Account	Student, Administrator, Lecturer, Programme Coordinator, Module Coordinator, Applicant
Manage registration	Administrator
View admission	Applicant, Administrator
Update admission	Applicant
Manage fees	Administrator
Change account password	Administrator, Students, Applicants, Lecturer, Programme Coordinator, Module Coordinator
Make payments	Students, Applicants
Upload scanned documents	Students, Applicants
Verify admission	Applicant
Merit status tracking	Applicant
Confirm admission	Applicant
Compare qualify scores	Administrator
Merit list	Administrator, Applicant

5. UML Diagrams

5.1 Introduction

An engineering system or product's initial development phase stage is its design. In software development, quality is promoted through design. Only via design will we be able to faithfully transform user needs into a finished software product or system. All subsequent phases involving software engineers and maintenance have their roots in software design. Without design, we run the danger of creating an unstable design that will break down when minor adjustments are made, may be challenging to test, and whose number will not be known until late in the software engineering process.

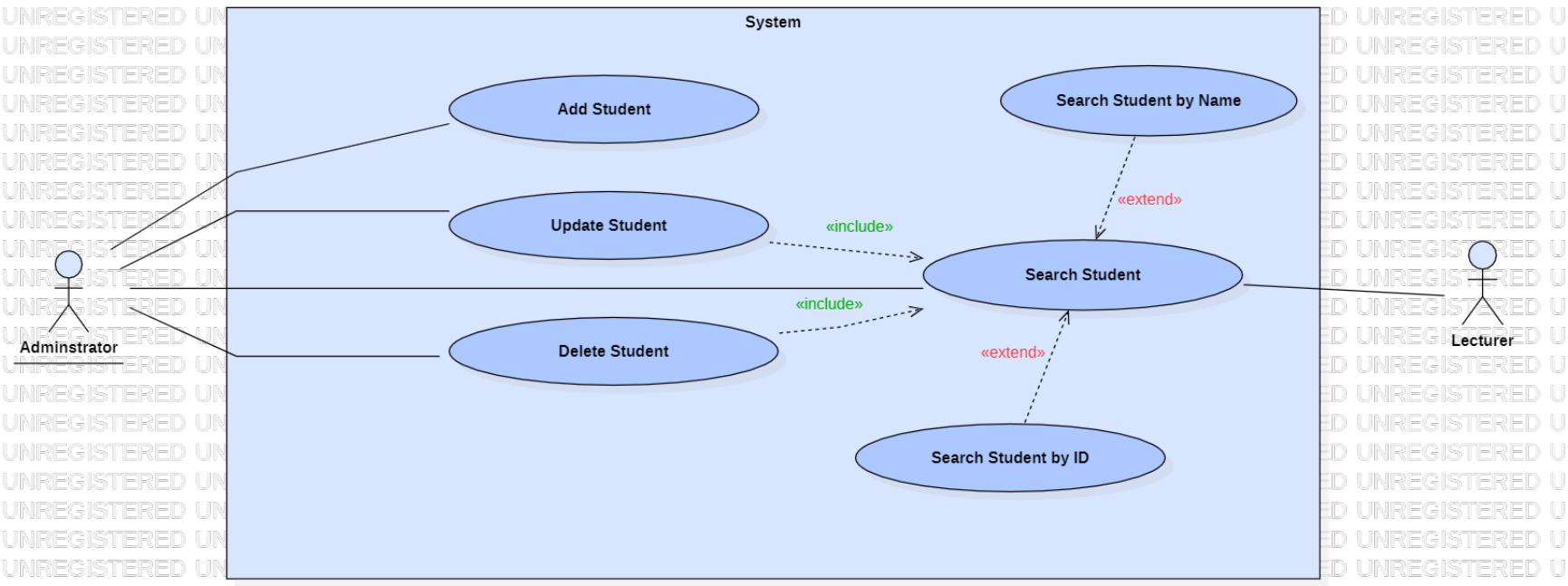
We have created Unified Modelling Language (UML) diagrams using the software requirements specification document from the analysis phase as input to the design phase. UML is dependent on system visual modelling. Visual modelling is the technique of employing a standardized set of graphical elements to represent the information from the model graphically.

The Star UML Diagrammed Software is used to create UML diagrams. When complexity is presented to us visually rather than textually, we tend to be able to comprehend it better. A system's various levels of operation can be demonstrated by creating graphic representations of the system. The interactions between users and the system are subject to modelling.

5.2 Case Diagrams & Use Case Specifications

Use Case Diagram displays the relationship among actors and use cases.

5.2.1 Student



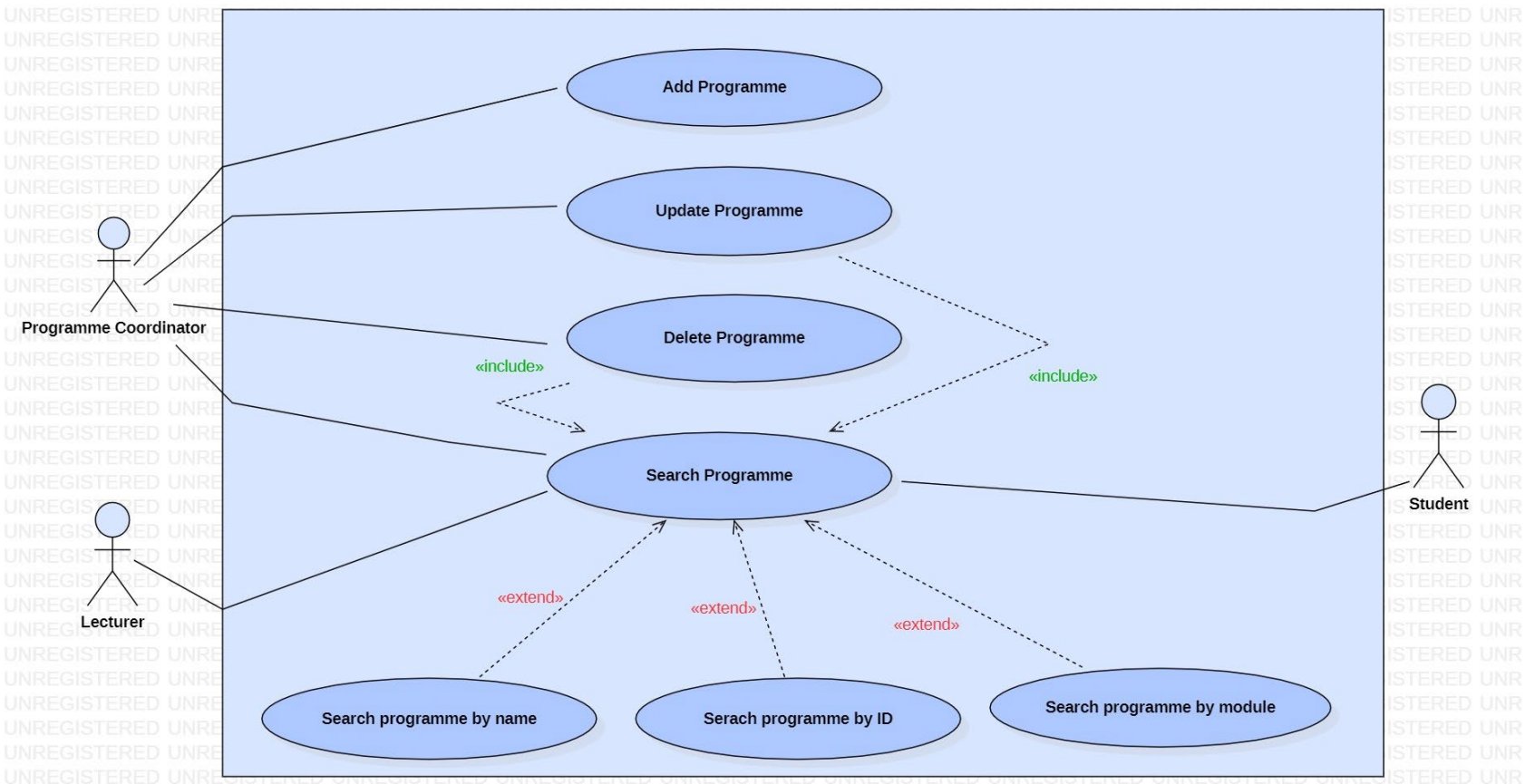
Use case name: Add Student	
Participating actors(s): Admin	
Preconditions: Before the use case starts, the admin must log in the system.	
Flow of events	
Actor Action	**System Response**
1. Enters student's details. 3. Actor confirms the details and save.	2. Validates if data have been typed in the correct format and allow actor to confirm the details. 4. Displays a confirming message that student has been added.
Postconditions: Student has been added successfully.	
Special Requirements: None	

Use case name: Update Student	
Participating actors(s): Admin	
Preconditions: Before the use case starts, the admin must log in the system and then search for the student before proceeding with the update	
Flow of events	
Actor Action	**System Response**
1. Enters the student ID and name. 3. Updates the required data and save.	2. Displays the student details. 4. Presents a message confirming update.
Postconditions: Student has been updated successfully	
Special Requirements: None	

Use case name: Delete Student	
Participating actors(s): Admin	
Preconditions: Before the use case starts, the admin must log in the system and then search for the student before proceeding with the deletion	
Flow of events	
Actor Action	**System Response**
1. Enter student ID and name. 3. Selects delete.	2. Retrieves the data, display student's details and allow actor to delete. 4. Displays a confirming delete verification dialog.
Postconditions: Student has been deleted successfully	
Special Requirements: None	

Use case name: Search Student	
Participating actors(s): Admin, Lecturer	
Preconditions: Before the use case starts, admin and lecturer must log in the system.	
Flow of events	
Actor Action	**System Response**
1. Enter student ID / name 3. The use case ends when the actor finds the student	2. Retrieves and displays the data if it exists.
Postconditions: The actors successfully find the respective student they are looking for.	
Special Requirements: Actors can either search student by name or ID at a time	

5.2.2 Programme



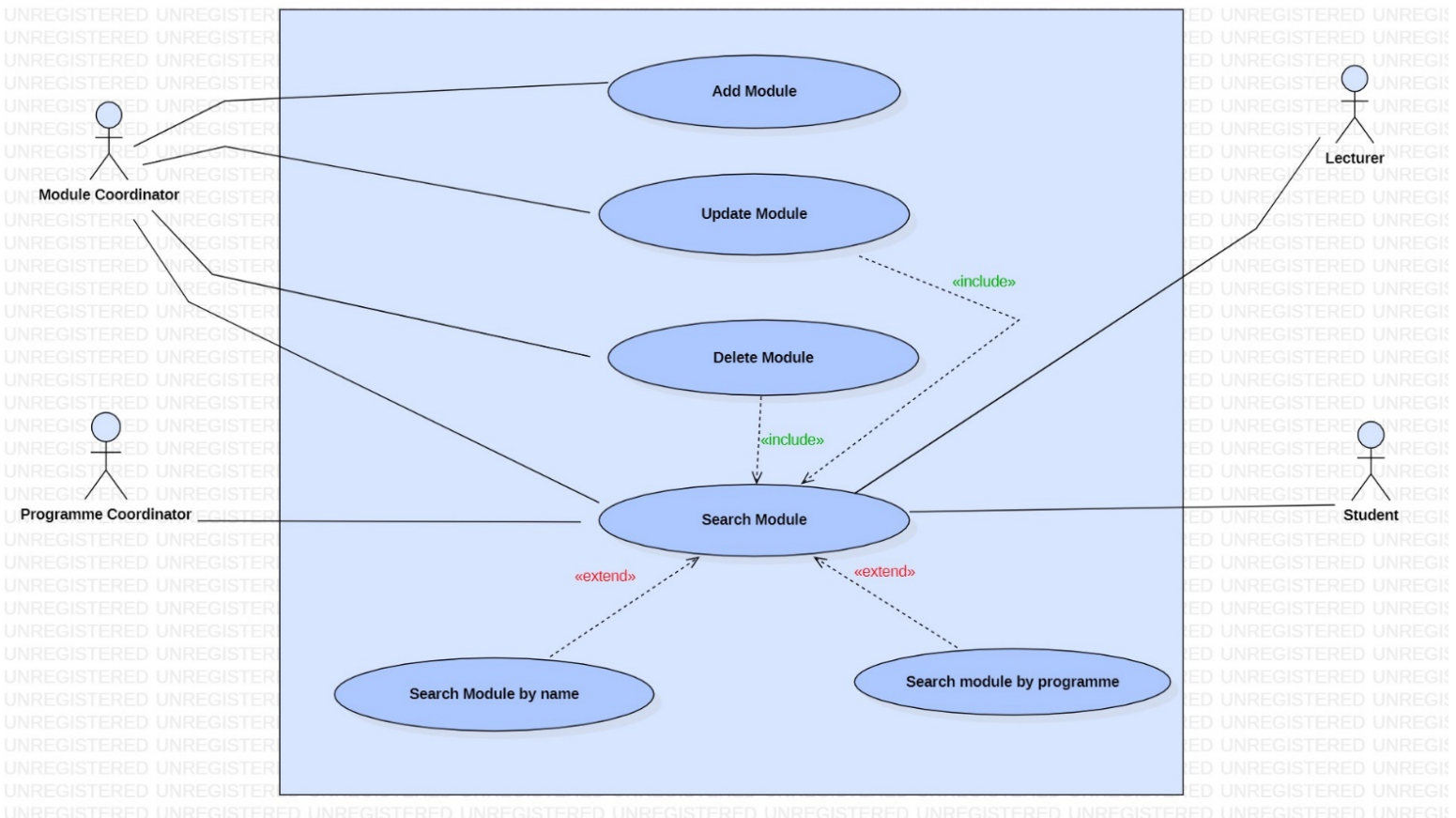
Use case name: Add programme	
Participating actors(s): Programme Coordinator	
Preconditions: Before the use case starts, actor must log in the system	
Flow of events	
Actor Action	**System Response**
1. Enter the programme's details 3. Actor confirms the details and save	2. Validates if data have been typed in the correct format and allow actor to confirm the details. 4. Displays a confirming message that programme has been added.
Postconditions: Programme has been added successfully	
Special Requirements: None	

Use case name: Update programme	
Participating actors(s): Programme Coordinator	
Preconditions: Before the use case starts, actor must log in the system and then search for the programme before proceeding with the update	
Flow of events	
Actor Action	**System Response**
1. Enter the programme's details 3. Allow the actor to update the required data and save.	2. Retrieves and displays the programme's details 4. Displays a confirming message that programme has been updated.
Postconditions: Programme has been updated successfully	
Special Requirements: None	

Use case name: Delete Programme	
Participating actors(s): Programme Coordinator	
Preconditions: Before the use case starts, the actor must log in the system and then actor must search for the programme before proceeding with the deletion	
Flow of events	
Actor Action	**System Response**
1. Enter programme's details. 3. Selects delete.	2. Retrieves the data, display programme's details and allow actor to delete. 4. Displays a confirming delete verification dialog.
Postconditions: Programme has been deleted successfully	
Special Requirements: None	

Use case name: Search Programme	
Participating actors(s): Programme coordinator, lecturer, student	
Preconditions: Before the use case starts, actors must log in the system.	
Flow of events	
Actor Action	**System Response**
1. Enter programme's ID/ name/module 3. The use case ends when the actors find the programme	2. Retrieves and displays the data if it exists.
Postconditions: The actors successfully find the programme they are looking for.	
Special Requirements: Actors can either search programme by name/ID/module at a time	

5.2.3 Module



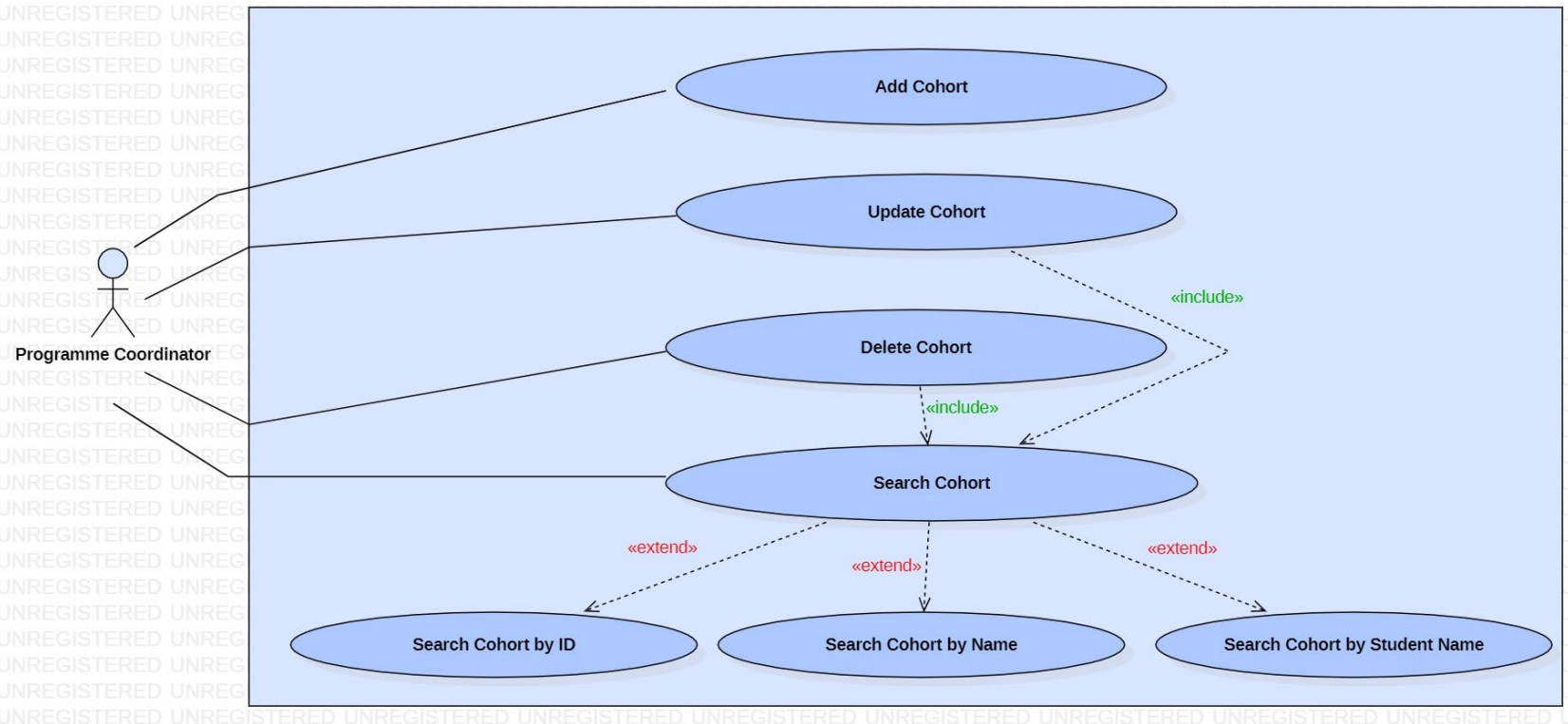
Use case name: Add Module	
Participating actors(s): Module Coordinator	
Preconditions: Before the use case starts, actor must log in the system	
Flow of events	
Actor Action	**System Response**
1. Enter the module's details	2. Validates if data have been typed in the correct format and allow actor to confirm the details.
3. Actor confirms the details and save	4. Displays a confirming message that module has been added.
Postconditions: Programme has been added successfully	
Special Requirements: None	

Use case name: Update module	
Participating actors(s): Module Coordinator	
Preconditions: Before the use case starts, actor must log in the system and then actor must search for the module first before proceeding with the update	
Flow of events	
Actor Action	**System Response**
1. Enter the module's details 3. Allow the actor to update the required data and save.	2. Retrieves and displays the module's details 4. Displays a confirming message that module has been updated.
Postconditions: Programme has been updated successfully	
Special Requirements: None	

Use case name: Delete Module	
Participating actors(s): Module Coordinator	
Preconditions: Before the use case starts, the actor must log in the system and then search for the module before proceeding with the deletion	
Flow of events	
Actor Action	**System Response**
1. Enter module's details. 3. Selects delete.	2. Retrieves the data, display module's details and allow actor to delete. 4. Displays a confirming delete verification dialog.
Postconditions: Module has been deleted successfully	
Special Requirements: none	

Use case name: Search Module	
Participating actors(s): Module Coordinator, Programme Coordinator, Lecturer, Students	
Preconditions: Before the use case starts, actors must log in the system.	
Flow of events	
Actor Action	**System Response**
1. Enter module's name/programme 3. The use case ends when the actors find the module	2. Retrieves and displays the data if it exists.
Postconditions: The actors successfully find the programme they are looking for.	
Special Requirements: Actors can either search the module by its name or programme at a time.	

5.2.4 Cohort



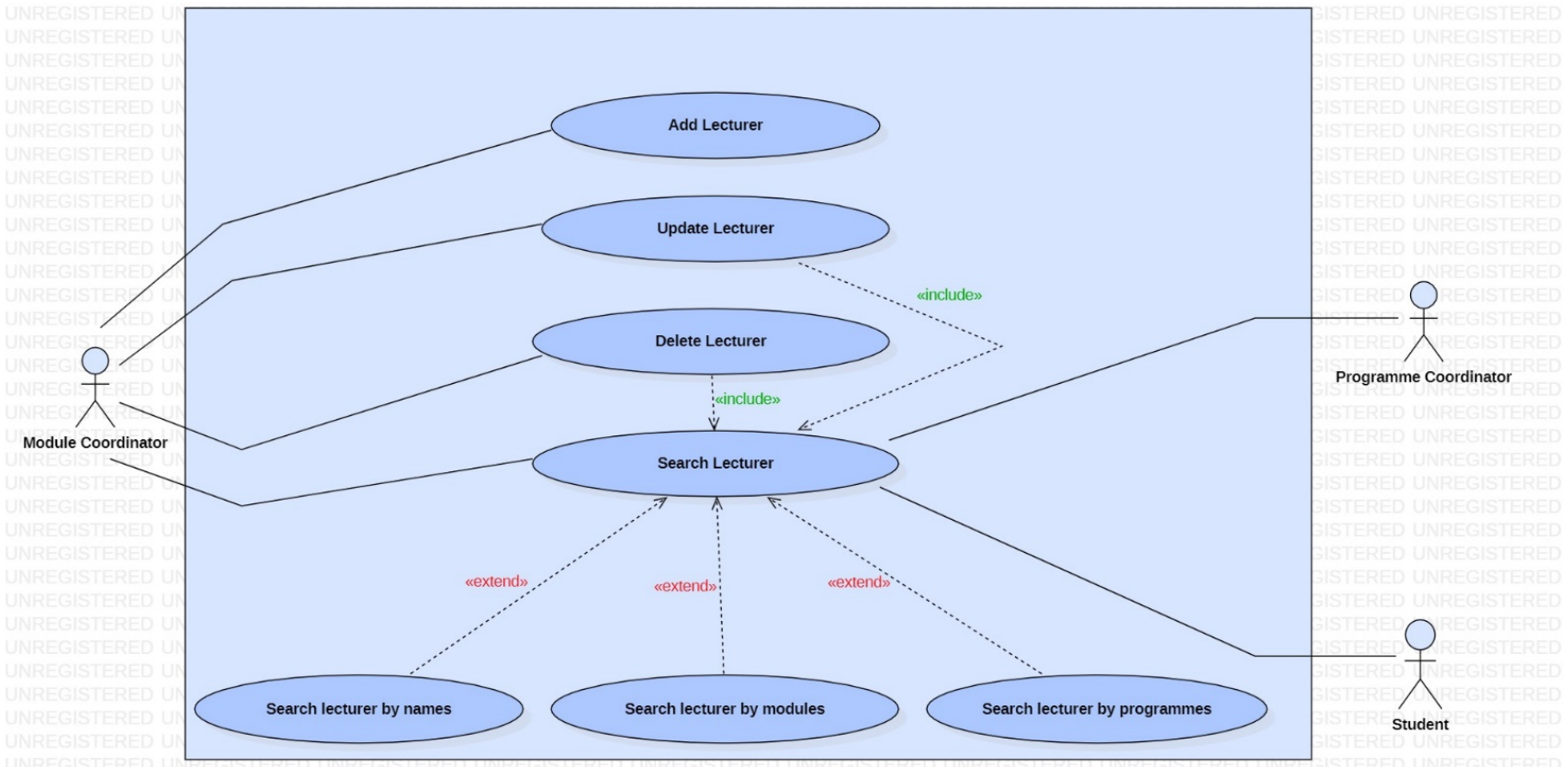
Use case name: Add Cohort	
Participating actors(s): Programme Coordinator	
Preconditions: Before the use case starts, actor must log in the system	
Flow of events	
Actor Action	**System Response**
1. Enter the cohort's details 3. Actor confirms the details and save	2. Validates if data have been typed in the correct format and allow actor to confirm the details. 4. Displays a confirming message that module has been added.
Postconditions: Cohort has been added successfully	
Special Requirements: None	

Use case name: Update Cohort	
Participating actors(s): Programme Coordinator	
Preconditions: Before the use case starts, actor must log in the system and must search for the cohort first and then proceed the update.	
Flow of events	
Actor Action	**System Response**
1. Enter the cohort's details 3. Allow the actor to update the required data and save.	2. Retrieves and displays the cohort's details 4. Displays a confirming message that cohort has been updated.
Postconditions: Cohort has been updated successfully	
Special Requirements: None	

Use case name: Delete Cohort	
Participating actors(s): Programme Coordinator	
Preconditions: Before the use case starts, the actor must log in the system and then search for the module before proceeding with the deletion	
Flow of events	
Actor Action	**System Response**
1. Enter cohort's details. 3. Selects delete.	2. Retrieves the data, display cohort's details and allow actor to delete. 4. Displays a confirming delete verification dialog.
Postconditions: Cohort has been deleted successfully	
Special Requirements: none	

Use case name: Search Cohort	
Participating actors(s): Programme Coordinator	
Preconditions: Before the use case starts, actors must log in the system.	
Flow of events	
Actor Action	**System Response**
1. Enter cohort's name/ID/ Students' name 3. The use case ends when the actors find the cohort	2. Retrieves and displays the data if it exists.
Postconditions: The actors successfully find the cohort they are looking for.	
Special Requirements: Actors can either search the cohort by its name, ID or students' names at a time.	

5.2.5 Lecturer



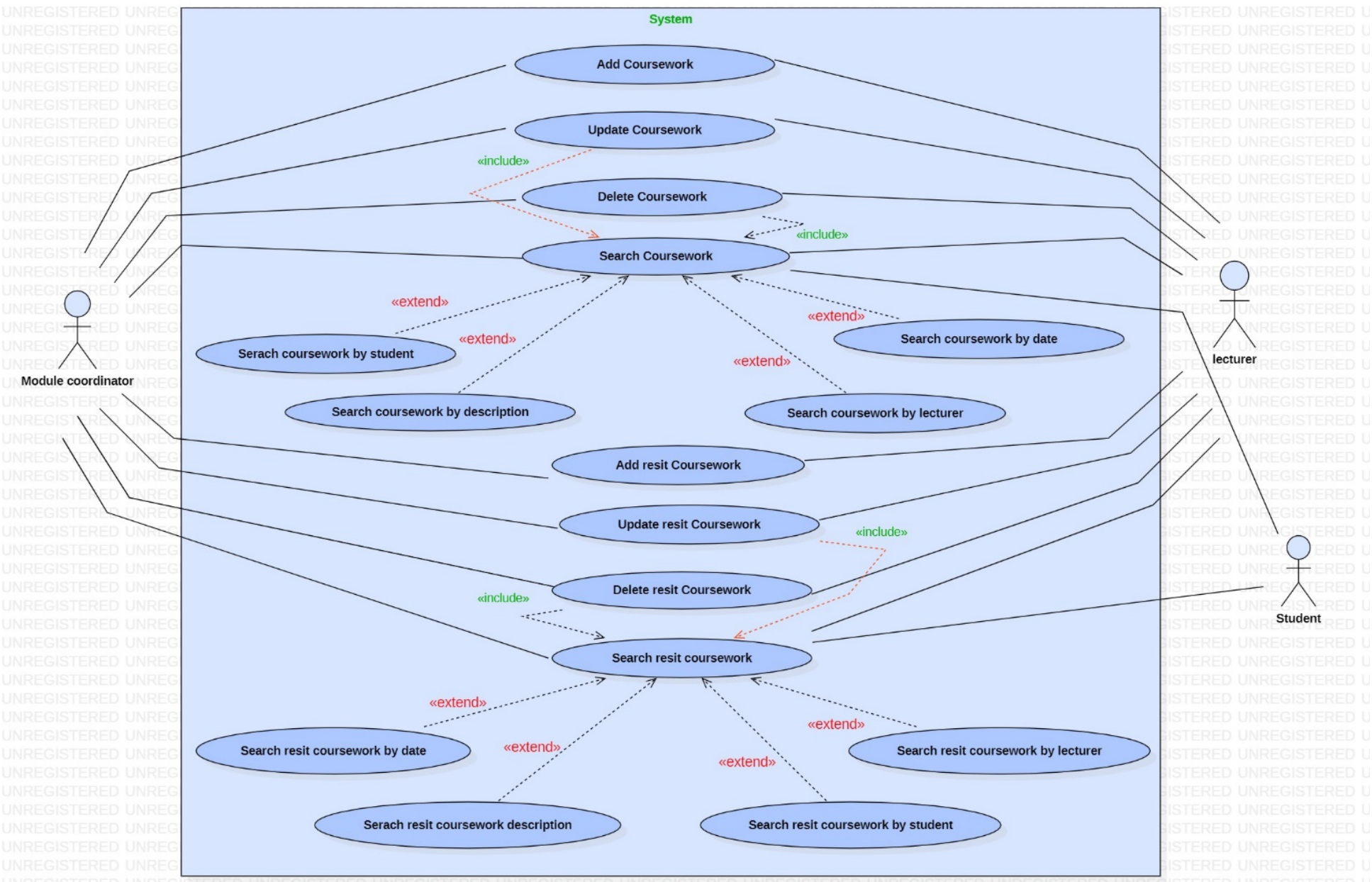
Use case name: Add Lecturer	
Participating actors(s): Module Coordinator	
Preconditions: Before the use case starts, actor must log in the system	
Flow of events	
Actor Action	**System Response**
1. Enter the lecturer's details 3. Actor confirms the details and save	2. Validates if data have been typed in the correct format and allow actor to confirm the details. 4. Displays a confirming message that lecturer has been added.
Postconditions: Lecturer has been added successfully	
Special Requirements: None	

Use case name: Update Lecturer	
Participating actors(s): Module Coordinator	
Preconditions: Before the use case starts, actor must log in the system and must search for the lecturer first and then proceed with the update.	
Flow of events	
Actor Action	**System Response**
1. Enter the lecturer's details 3. Allow the actor to update the required data and save.	2. Retrieves and displays the cohort's details 4. Displays a confirming message that lecturer has been updated.
Postconditions: Lecturer has been updated successfully	
Special Requirements: None	

Use case name: Delete Lecturer	
Participating actors(s): Module Coordinator	
Preconditions: Before the use case starts, the actor must log in the system and then search for the lecturer before proceeding with the deletion	
Flow of events	
Actor Action	**System Response**
1. Enter lecturer's details. 3. Selects delete.	2. Retrieves the data, display lecturer's details and allow actor to delete. 4. Displays a confirming delete verification dialog.
Postconditions: Lecturer has been deleted successfully	
Special Requirements: none	

Use case name: Search Lecturer	
Participating actors(s): Module Coordinator, Programme Coordinator, Student	
Preconditions: Before the use case starts, actors must log in the system.	
Flow of events	
Actor Action	**System Response**
1. Enter lecturer's name/ module/ programmes 3. The use case ends when the actors find the lecturer	2. Retrieves and displays the data if it exists.
Postconditions: The actors successfully find the lecturer they are looking for.	
Special Requirements: Actors can either search the lecturer by his name/module or programme at a time.	

5.2.6 Coursework



Use case name: Add Coursework	
Participating actors(s): Module Coordinator, lecturer	
Preconditions: Before the use case starts, actors must log in the system	
Flow of events	
Actor Action	**System Response**
1. Enter the coursework's details 3. Actors confirm the details and save	2. Validates if data have been typed in the correct format and allows actor to confirm the details. 4. Displays a confirming message that coursework has been added.
Postconditions: Coursework has been added successfully	
Special Requirements: None	

Use case name: Update Coursework	
Participating actors(s): Module Coordinator, Lecturer	
Preconditions: Before the use case starts, actors must log in the system and must search for the coursework first and then proceed with the update.	
Flow of events	
Actor Action	**System Response**
1. Enter the coursework's details 3. Allow the actors to update the required data and save.	2. Retrieves and displays the coursework's details 4. Displays a confirming message that lecturer has been updated.
Postconditions: Coursework has been updated successfully	
Special Requirements: None	

Use case name: Delete Coursework	
Participating actors(s): Module Coordinator, Lecturer	
Preconditions: Before the use case starts, the actors must log in the system and then search for the coursework before proceeding with the deletion	
Flow of events	
Actor Action	**System Response**
1. Enter coursework's details. 3. Selects delete.	2. Retrieves the data, display coursework's details and allow actor to delete. 4. Displays a confirming delete verification dialog.
Postconditions: Coursework has been deleted successfully	
Special Requirements: none	

Use case name: Search Coursework	
Participating actors(s): Module Coordinator, Lecturer	
Preconditions: Before the use case starts, actors must log in the system.	
Flow of events	
Actor Action	**System Response**
1. Enter coursework's details 3. The use case ends when the actors find the coursework	2. Retrieves and displays the data if it exists.
Postconditions: The actors successfully find the coursework they are looking for.	
Special Requirements: Actors can either search coursework by student/description/ lecturer/date at a time.	

Use case name: Add Re-sit Coursework	
Participating actors(s): Module Coordinator, lecturer	
Preconditions: Before the use case starts, actors must log in the system	
Flow of events	
Actor Action	**System Response**
1. Enter the re-sit coursework's details 3. Actors confirm the details and save	2. Validates if data have been typed in the correct format and allows actor to confirm the details. 4. Displays a confirming message that re-sit coursework has been added.
Postconditions: Resit coursework has been added successfully	
Special Requirements: None	

Use case name: Update re-sit coursework	
Participating actors(s): Module Coordinator, Lecturer	
Preconditions: Before the use case starts, the actor must log in the system and then search for the re-sit before proceeding with the update.	
Flow of events	
Actor Action	**System Response**
1. Enter the re-sit coursework's details 3. Allow the actors to update the required data and save.	2. Retrieves and displays the re-sit coursework's details 4. Displays a confirming message that the re-sit coursework has been updated.
Postconditions: Resit coursework has been updated successfully	
Special Requirements: None	

Use case name: Delete Resit coursework	
Participating actors(s): Module Coordinator, Lecturer	
Preconditions: Before the use case starts, the actor must log in the system and then search for the re-sit coursework before proceeding with the deletion	
Flow of events	
Actor Action	**System Response**
1. Enter re-sit coursework's details. 3. Selects delete.	2. Retrieves the data, display the re-sit coursework's details, and allow actor to delete. 4. Displays a confirming delete verification dialog.
Postconditions: Resit coursework has been deleted successfully	
Special Requirements: none	

Use case name: Search re-sit coursework	
Participating actors(s): Module Coordinator, Lecturer	
Preconditions: Before the use case starts, actors must log in the system.	
Flow of events	
Actor Action	**System Response**
1. Enter coursework's details 3. The use case ends when the actors find the re-sit coursework	2. Retrieves and displays the data if it exists.
Postconditions: The actors successfully find the coursework they are looking for.	
Special Requirements: Actors can either search re-sit coursework by date/description/student/lecturer at a time.	

5.2.7 Exam

Use case name: Add Exam	
Participating actors(s): Faculties	
Preconditions: Before the use case starts, actors must log in the system	
Flow of events	
Actor Action	**System Response**
1. Enter the exam's details 3. Actors confirm the details and save	2. Validates if data have been typed in the correct format and allows actor to confirm the details. 4. Displays a confirming message that exam has been added.
Postconditions: Exam has been added successfully	
Special Requirements: None	

Use case name: Update Exam	
Participating actors(s): Faculties	
Preconditions: Before the use case starts, actors must log in the system and must search for the exam first and then proceed with the update.	
Flow of events	
Actor Action	**System Response**
1. Enter the exam's details 3. Allow the actors to update the required data and save.	2. Retrieves and displays the exam's details 4. Displays a confirming message that lecturer has been updated.
Postconditions: Exam has been updated successfully	
Special Requirements: None	

Use case name: Delete Exam	
Participating actors(s): Faculties	
Preconditions: Before the use case starts, the actors must log in the system and then search for the exam before proceeding with the deletion	
Flow of events	
Actor Action	**System Response**
1. Enter exam's details. 3. Selects delete.	2. Retrieves the data, exam's details and allow actor to delete. 4. Displays a confirming delete verification dialog.
Postconditions: Exam has been deleted successfully	
Special Requirements: none	

Use case name: Search Exam	
Participating actors(s): Module Coordinator, Student	
Preconditions: Before the use case starts, actors must log in the system.	
Flow of events	
Actor Action	**System Response**
1. Enter exam's details 3. The use case ends when the actors find the exam	2. Retrieves and displays the data if it exists.
Postconditions: The actors successfully find the exam they are looking for.	
Special Requirements: Actors can either Search Exam by date, module, student, programme, cohort at a time	

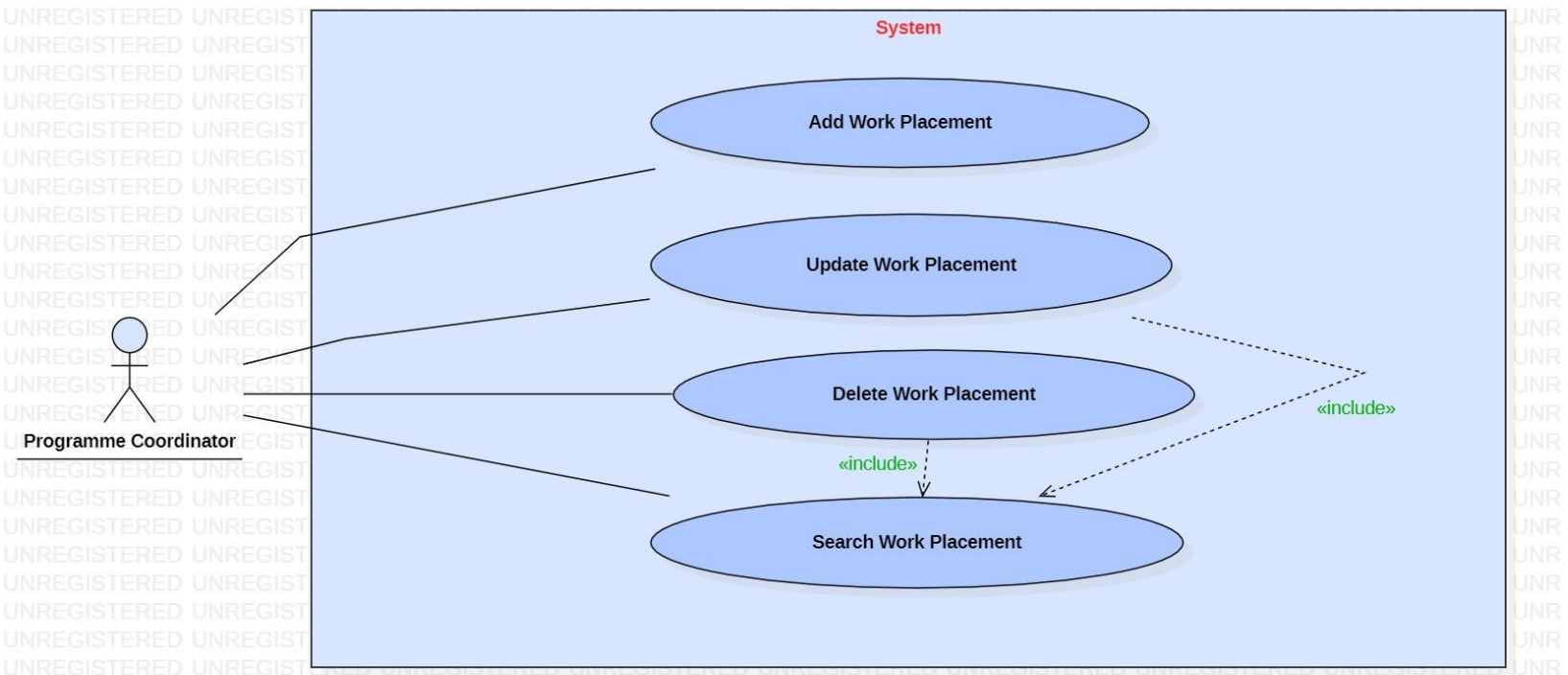
Use case name: Add Re-sit Exam	
Participating actors(s): Module Coordinator, Programme Coordinator	
Preconditions: Before the use case starts, actors must log in the system	
Flow of events	
Actor Action	**System Response**
1. Enter the re-sit exam's details 3. Actors confirm the details and save	2. Validates if data have been typed in the correct format and allows actor to confirm the details. 4. Displays a confirming message that re-sit exam has been added.
Postconditions: Re-sit exam has been added successfully	
Special Requirements: None	

Use case name: Update re-sit Exam	
Participating actors(s): Module Coordinator, Programme Coordinator	
Preconditions: Before the use case starts, actors must log in the system and must search for the re-sit exam first and then proceed with the update.	
Flow of events	
Actor Action	**System Response**
1. Enter the re-sit exam's details 3. Allow the actors to update the required data and save.	2. Retrieves and displays the re-sit exam's details 4. Displays a confirming message that the re-sit exam has been updated.
Postconditions: Re-sit exam has been updated successfully	
Special Requirements: None	

Use case name: Delete Resit Exam	
Participating actors(s): Module Coordinator, Programme Coordinator	
Preconditions: Before the use case starts, the actor must log in the system and then search for the re-sit exam before proceeding with the deletion	
Flow of events	
Actor Action	**System Response**
1. Enter re-sit exam s details. 3. Selects delete.	2. Retrieves the data, display the re-sit exam's details, and allow actor to delete. 4. Displays a confirming delete verification dialog.
Postconditions: Resit exam has been deleted successfully	
Special Requirements: none	

Use case name: Search re-sit Exam	
Participating actors(s): Module Coordinator, Programme Coordinator, Student	
Preconditions: Before the use case starts, actors must log in the system.	
Flow of events	
Actor Action	**System Response**
1. Enter exam's details 3. The use case ends when the actors find the re-sit exam	2. Retrieves and displays the data if it exists.
Postconditions: The actors successfully find the exam they are looking for.	
Special Requirements: Actors can either search re-sit exam by date, module, student, programme, cohort at a time.	

5.2.8 Work Placement



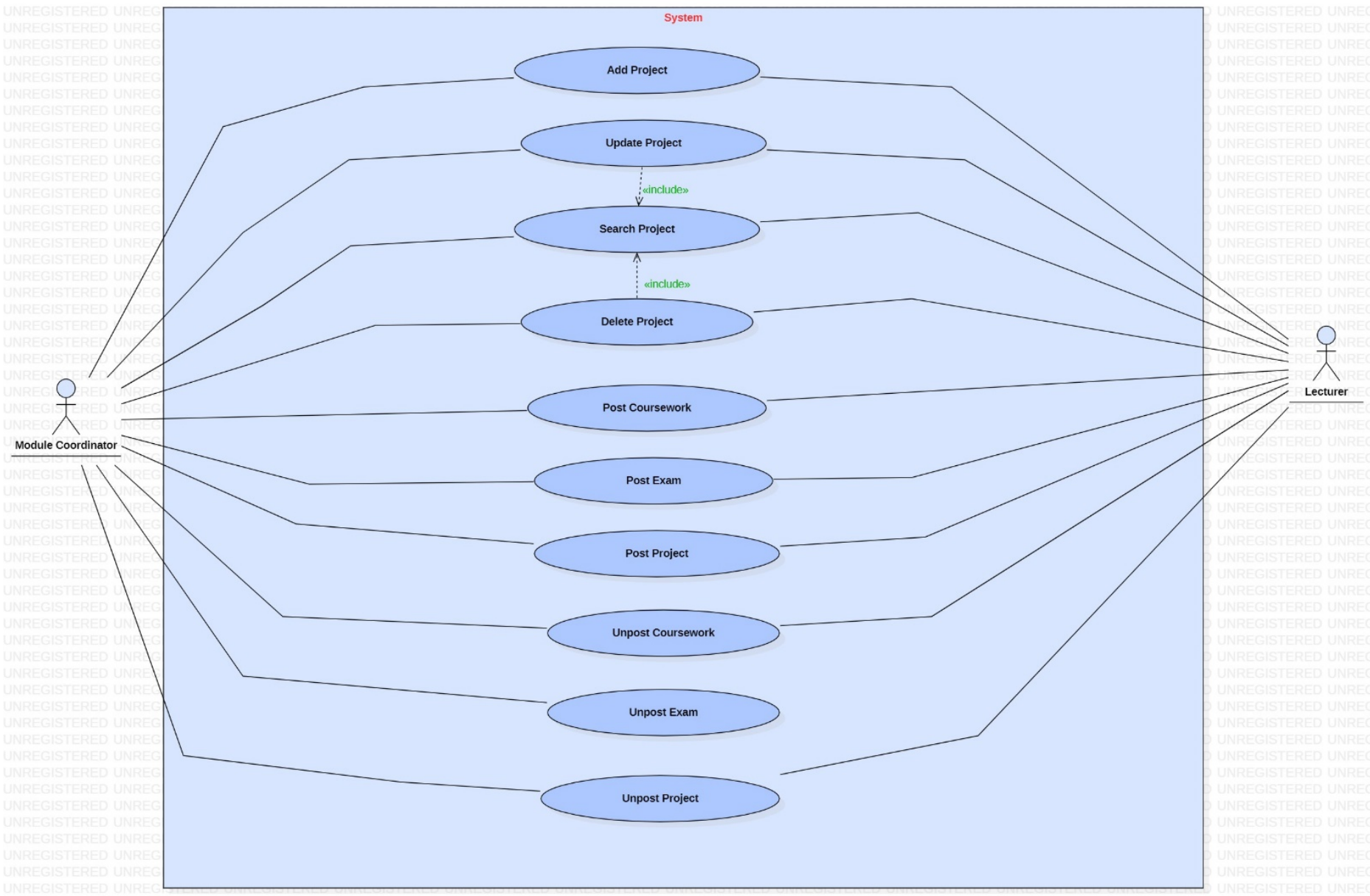
Use case name: Add Work placement	
Participating actors(s): Programme Coordinator	
Preconditions: Before the use case starts, actor must log in the system	
Flow of events	
Actor Action	**System Response**
1. Enter the work placement's details 3. Actor confirms the details and save	2. Validates if data have been typed in the correct format and allow actor to confirm the details. 4. Displays a confirming message that work placement has been added.
Postconditions: Work placement has been added successfully	
Special Requirements: None	

Use case name: Update Work placement	
Participating actors(s): Programme Coordinator	
Preconditions: Before the use case starts, actor must log in the system and must search for the work placement first and then proceed with the update.	
Flow of events	
Actor Action	**System Response**
1. Enter the work placement's details 3. Allow the actor to update the required data and save.	2. Retrieves and displays the cohort's details 4. Displays a confirming message that work placement has been updated.
Postconditions: Work placement has been updated successfully	
Special Requirements: None	

Use case name: Delete Work placement	
Participating actors(s): Programme Coordinator	
Preconditions: Before the use case starts, the actor must log in the system and then search for the work placement before proceeding with the deletion	
Flow of events	
Actor Action	**System Response**
1. Enter work placement's details. 3. Selects delete.	2. Retrieves the data, display work placement's details and allow actor to delete. 4. Displays a confirming delete verification dialog.
Postconditions: Work placement has been deleted successfully	
Special Requirements: none	

Use case name: Search Work placement	
Participating actors(s): Programme Coordinator	
Preconditions: Before the use case starts, actors must log in the system.	
Flow of events	
Actor Action	**System Response**
1. Enter work placement 3. The use case ends when the actors find the work placement	2. Retrieves and displays the data if it exists.
Postconditions: The actors successfully find the work placement they are looking for.	
Special Requirements: None	

5.2.9 Project



Use case name: Add Project	
Participating actors(s): Module Coordinator, Lecturer	
Preconditions: Before the use case starts, actor must log in the system	
Flow of events	
Actor Action	**System Response**
1. Enter the project's details 3. Actor confirms the details and save	2. Validates if data have been typed in the correct format and allow actor to confirm the details. 4. Displays a confirming message that project has been added.
Postconditions: Project has been added successfully	
Special Requirements: None	

Use case name: Update Project	
Participating actors(s): Module Coordinator, Lecturer	
Preconditions: Before the use case starts, actor must log in the system and must search for the project first and then proceed with the update.	
Flow of events	
Actor Action	**System Response**
1. Enter the project's details 3. Allow the actor to update the required data and save.	2. Retrieves and displays the project's details 4. Displays a confirming message that project has been updated.
Postconditions: Project has been updated successfully	
Special Requirements: None	

Use case name: Delete Project	
Participating actors(s): Module Coordinator, Lecturer	
Preconditions: Before the use case starts, the actor must log in the system and then search for the project before proceeding with the deletion	
Flow of events	
Actor Action	**System Response**
1. Enter project's details. 3. Selects delete.	2. Retrieves the data, display project s details and allow actor to delete. 4. Displays a confirming delete verification dialog.
Postconditions: Project has been deleted successfully	
Special Requirements: none	

Use case name: Search Project	
Participating actors(s): Module Coordinator, Lecturer	
Preconditions: Before the use case starts, actors must log in the system.	
Flow of events	
Actor Action	**System Response**
1. Enter project's details 3. The use case ends when the actors find the project	2. Retrieves and displays the data if it exists.
Postconditions: The actors successfully find the project they are looking for.	
Special Requirements: None	

Use case name: Post coursework	
Participating actors(s): Module Coordinator, Lecturer	
Preconditions: Before the use case starts, actors must log in the system.	
Flow of events	
Actor Action	**System Response**
1. Enter coursework's details 3. Actors confirm the details and save	2. Dialog pops up to confirm the posting of the coursework 4. Displays a confirming message that coursework has been posted.
Postconditions: The actors successfully post the coursework	
Special Requirements: None	

Use case name: Post exam	
Participating actors(s): Module Coordinator, Lecturer	
Preconditions: Before the use case starts, actors must log in the system.	
Flow of events	
Actor Action	**System Response**
1. Input exam's details 3. Actors confirm the details and save.	2. Pop of a dialog box to confirm the posting of the exam 4. Displays a confirming message that exam has been posted.
Postconditions: The actors successfully post the exam	
Special Requirements: None	

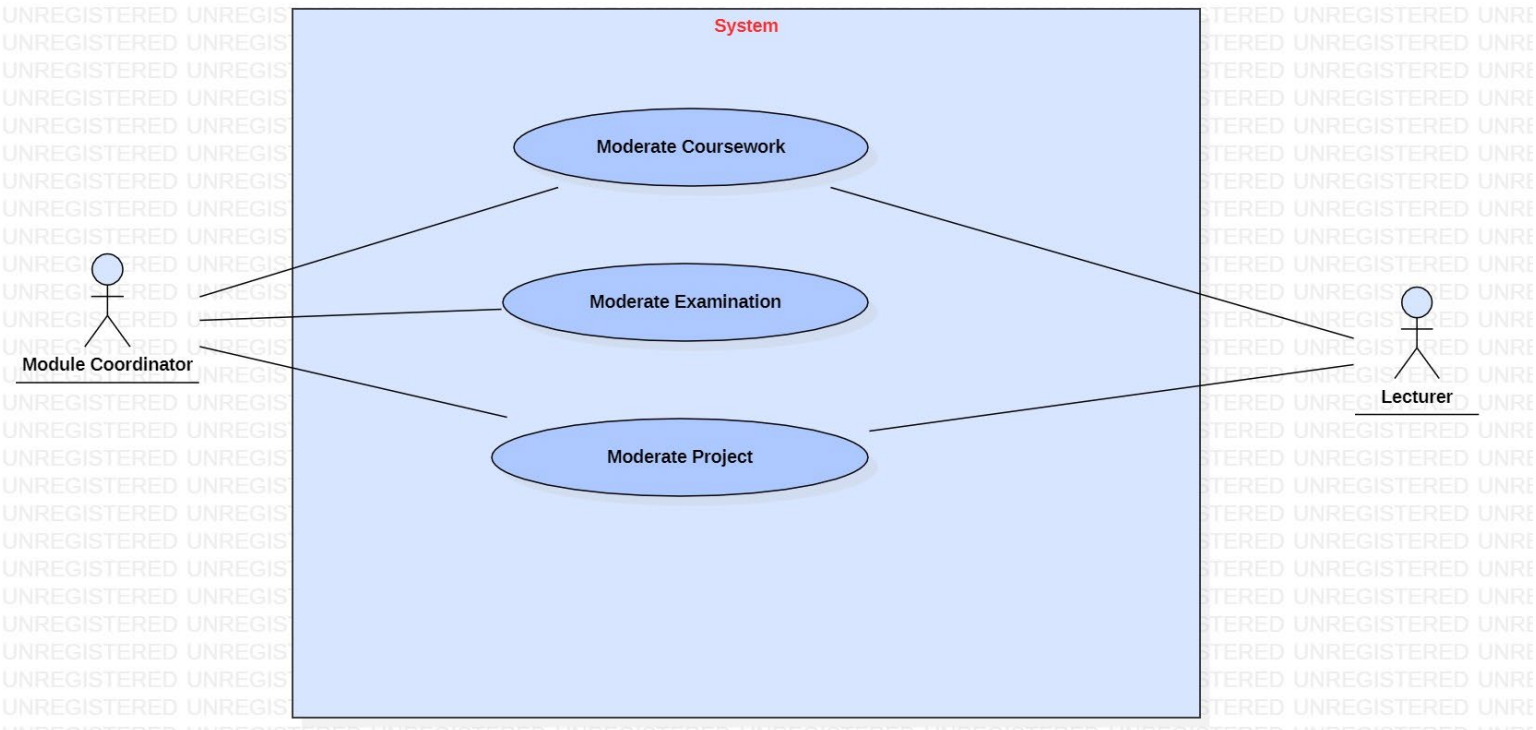
Use case name: Post project	
Participating actors(s): Module Coordinator, Lecturer	
Preconditions: Before the use case starts, actors must log in the system.	
Flow of events	
Actor Action	**System Response**
1. Input project's details 3. Actors confirm the details and save	2. Pop up of the dialog box to confirm the posting of the project 4. Displays a confirming message that project has been posted.
Postconditions: The actors successfully post the project	
Special Requirements: None	

Use case name: Un-post coursework	
Participating actors(s): Module Coordinator, Lecturer	
Preconditions: Before the use case starts, actors must log in the system.	
Flow of events	
Actor Action	**System Response**
1. Enter coursework details 3. Actor confirm the details	2. Retrieves and displays the data if it exists and shows a dialog box to confirm the un-posting of the coursework 4. Displays a message that coursework has been un-posted.
Postconditions: The actors successfully un-post the coursework	
Special Requirements: None	

Use case name: Un-post exam	
Participating actors(s): Module Coordinator	
Preconditions: Before the use case starts, actors must log in the system.	
Flow of events	
Actor Action	**System Response**
1. Enter exam's details 3. Actor confirm the details	2. Retrieves and displays the data if it exists and shows a dialog box to confirm the un-posting of the exam 4. Displays a message that exam has been un-posted.
Postconditions: The actors successfully un-post the exam	
Special Requirements: None	

Use case name: Un-post project	
Participating actors(s): Module Coordinator, Lecturer	
Preconditions: Before the use case starts, actors must log in the system.	
Flow of events	
Actor Action	**System Response**
1. Enter project's details	2. Retrieves and displays the data if it exists and shows a dialog box to confirm the un-posting of the project
3. Actor confirm the details	4. Displays a message that project has been un-posted.
Postconditions: The actors successfully un-post the project	
Special Requirements: None	

5.2.10 Moderation

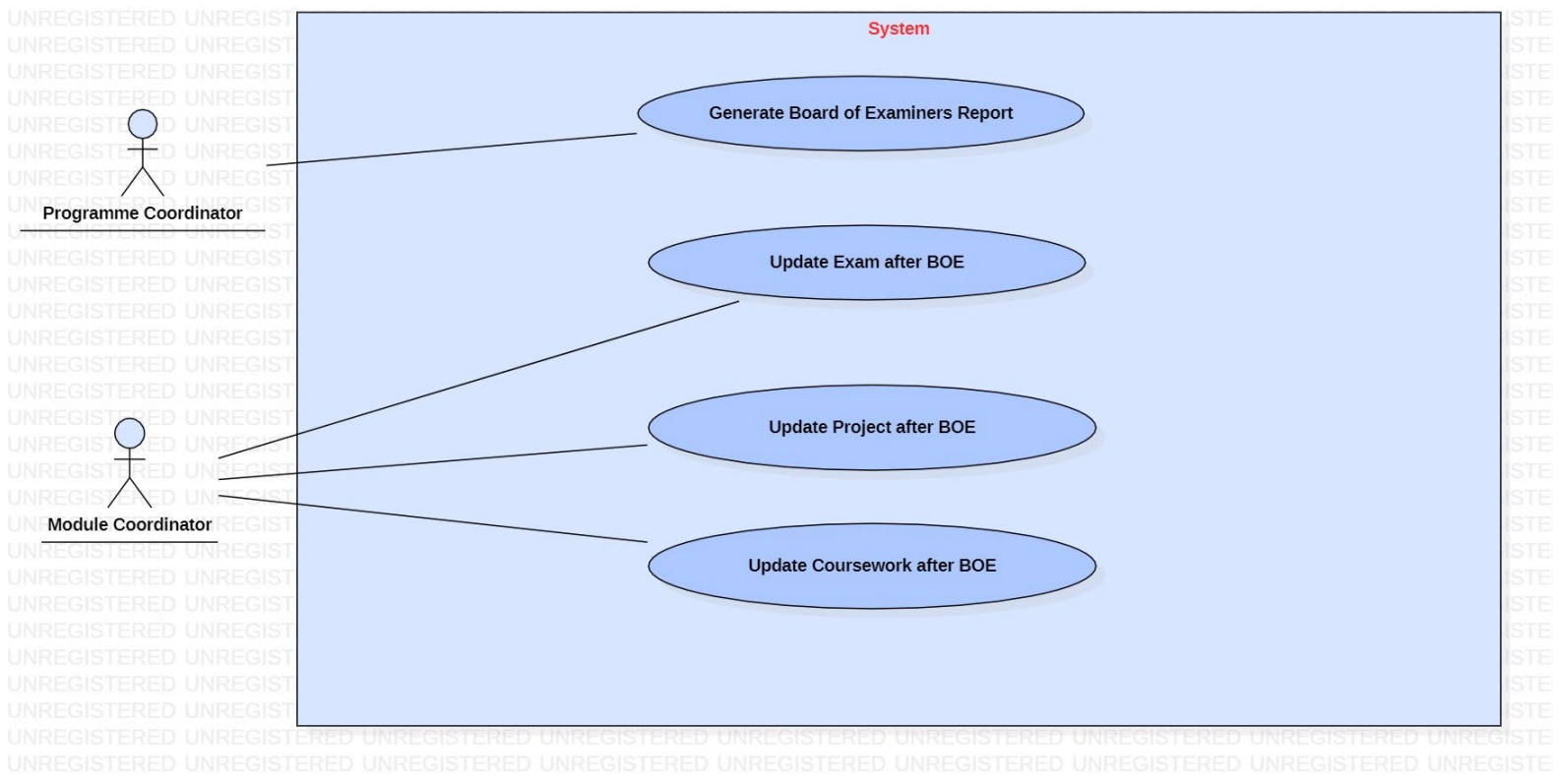


Use case name: Moderate Coursework	
Participating actors(s): Module Coordinator, Lecturer	
Preconditions: Before the use case starts, actor must log in the system	
Flow of events	
Actor Action	**System Response**
1. Enter details to moderate coursework	2. Validates if data have been typed in the correct format and allow actor to confirm the details.
3. Actor confirms the details and save	4. Displays a confirming message that coursework has been moderated.
Postconditions: Coursework has been moderated successfully	
Special Requirements: Retrieve coursework details to moderate	

Use case name: Moderate Examination	
Participating actors(s): Module Coordinator	
Preconditions: Before the use case starts, actor must log in the system	
Flow of events	
Actor Action	**System Response**
1. Enter details to moderate examination	2. Validates if data have been typed in the correct format and allow actor to confirm the details.
3. Actor confirms the details and save	4. Displays a confirming message that examination has been moderated.
Postconditions: Examination has been moderated successfully	
Special Requirements: Retrieve exam details to moderate	

Use case name: Moderate Project	
Participating actors(s): Module Coordinator, Lecturer	
Preconditions: Before the use case starts, actor must log in the system	
Flow of events	
Actor Action	**System Response**
1. Enter details to moderate project	2. Validates if data have been typed in the correct format and allow actor to confirm the details.
3. Actor confirms the details and save	4. Displays a confirming message that project has been moderated.
Postconditions: Project has been moderated successfully	
Special Requirements: Retrieve project details to moderate	

5.2.11 Board of Examiners



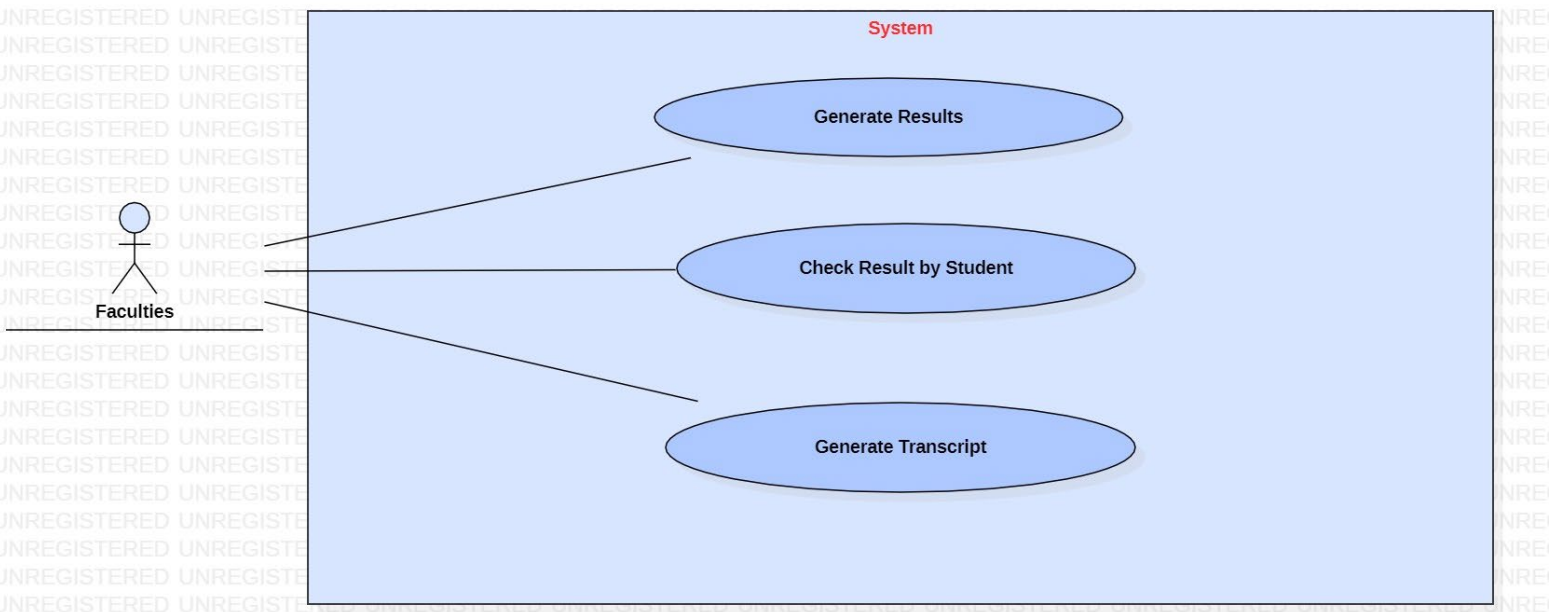
Use case name: Generate Board of Examiners report	
Participating actors(s): Programme Coordinator	
Preconditions: Before the use case starts, actor must log in the system	
Flow of events	
Actor Action	**System Response**
1. Enter the details of Board of Examiners report 3. Actor confirms the details and save	2. Validates if data have been typed in the correct format and allow actor to confirm the details. 4. Displays a confirming message that report of Board of Examiners has been generated
Postconditions: Board of Examiners has been generated successfully	
Special Requirements: None	

Use case name: Update Exam after BOE	
Participating actors(s): Module Coordinator	
Preconditions: Before the use case starts, actor must log in the system and must search for the exam first and then proceed with the update.	
Flow of events	
Actor Action	**System Response**
1. Enter the exam's details as per BOE 3. Allow the actor to update the required data and save.	2. Retrieves and displays the exam's details as per BOE 4. Displays a confirming message that exam has been updated.
Postconditions: Exam after BOE has been updated successfully	
Special Requirements: None	

Use case name: Update Project after BOE	
Participating actors(s): Module Coordinator	
Preconditions: Before the use case starts, actor must log in the system and must search for the project first and then proceed with the update.	
Flow of events	
Actor Action	**System Response**
1. Enter the project's details as per BOE 3. Allow the actor to update the required data and save.	2. Retrieves and displays the project's details as per BOE 4. Displays a confirming message that project has been updated.
Postconditions: Project after BOE has been updated successfully	
Special Requirements: None	

Use case name: Update Coursework after BOE	
Participating actors(s): Module Coordinator	
Preconditions: Before the use case starts, actor must log in the system and must search for the exam first and then proceed with the update.	
Flow of events	
Actor Action	**System Response**
1. Enter the coursework's details as per BOE 3. Allow the actor to update the required data and save.	2. Retrieves and displays the coursework's details as per BOE 4. Displays a confirming message that coursework has been updated.
Postconditions: Coursework after BOE has been updated successfully	
Special Requirements: None	

5.2.12 Generate

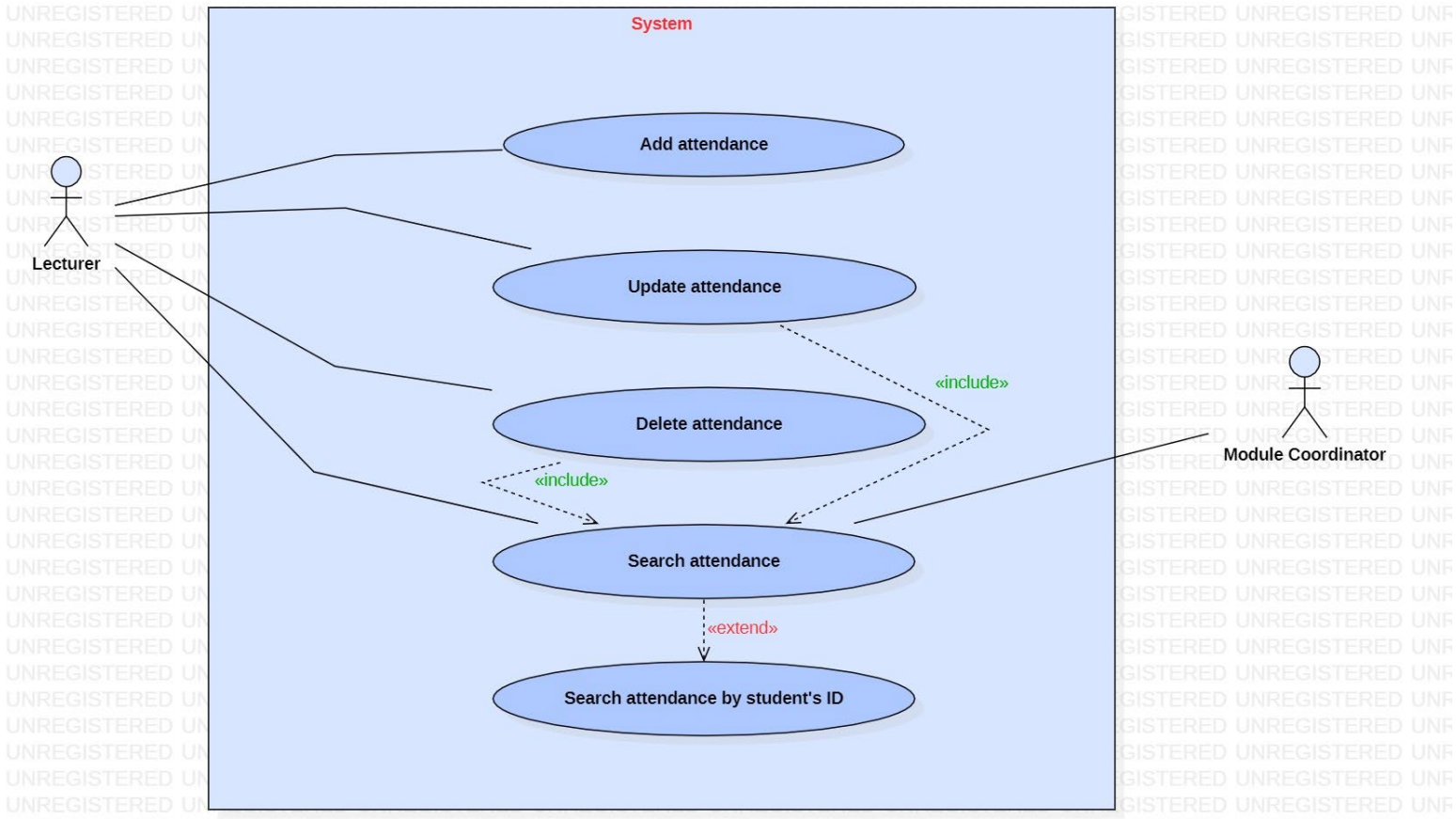


Use case name: Generate results	
Participating actors(s): Faculties	
Preconditions: Before the use case starts, actor must log in the system	
Flow of events	
Actor Action	**System Response**
1. Enter the results of students 3. Actor confirms the details and save	2. Validates if data have been typed in the correct format and allow actor to confirm the details. 4. Displays a confirming message that results has been generated
Postconditions: Results has been generated successfully	
Special Requirements: None	

Use case name: Check results by students	
Participating actors(s): Faculties	
Preconditions: Before the use case starts, actor must log in the system	
Flow of events	
Actor Action	**System Response**
1. Enter student's details 3. The use case ends when the actors find the results of students.	2. Retrieves and displays the data if it exists.
Postconditions: The actors successfully find the results they are looking for.	
Special Requirements: Actors can search the results by student's details at a time	

Use case name: Generate Transcript	
Participating actors(s): Faculties	
Preconditions: Before the use case starts, actor must log in the system	
Flow of events	
Actor Action	**System Response**
1. Enter the transcript's details 3. Actor confirms the details and save	2. Validates if data have been typed in the correct format and allow actor to confirm the details. 4. Displays a confirming message that Transcript has been generated
Postconditions: Transcript has been generated successfully	
Special Requirements: None	

5.2.13 Attendance



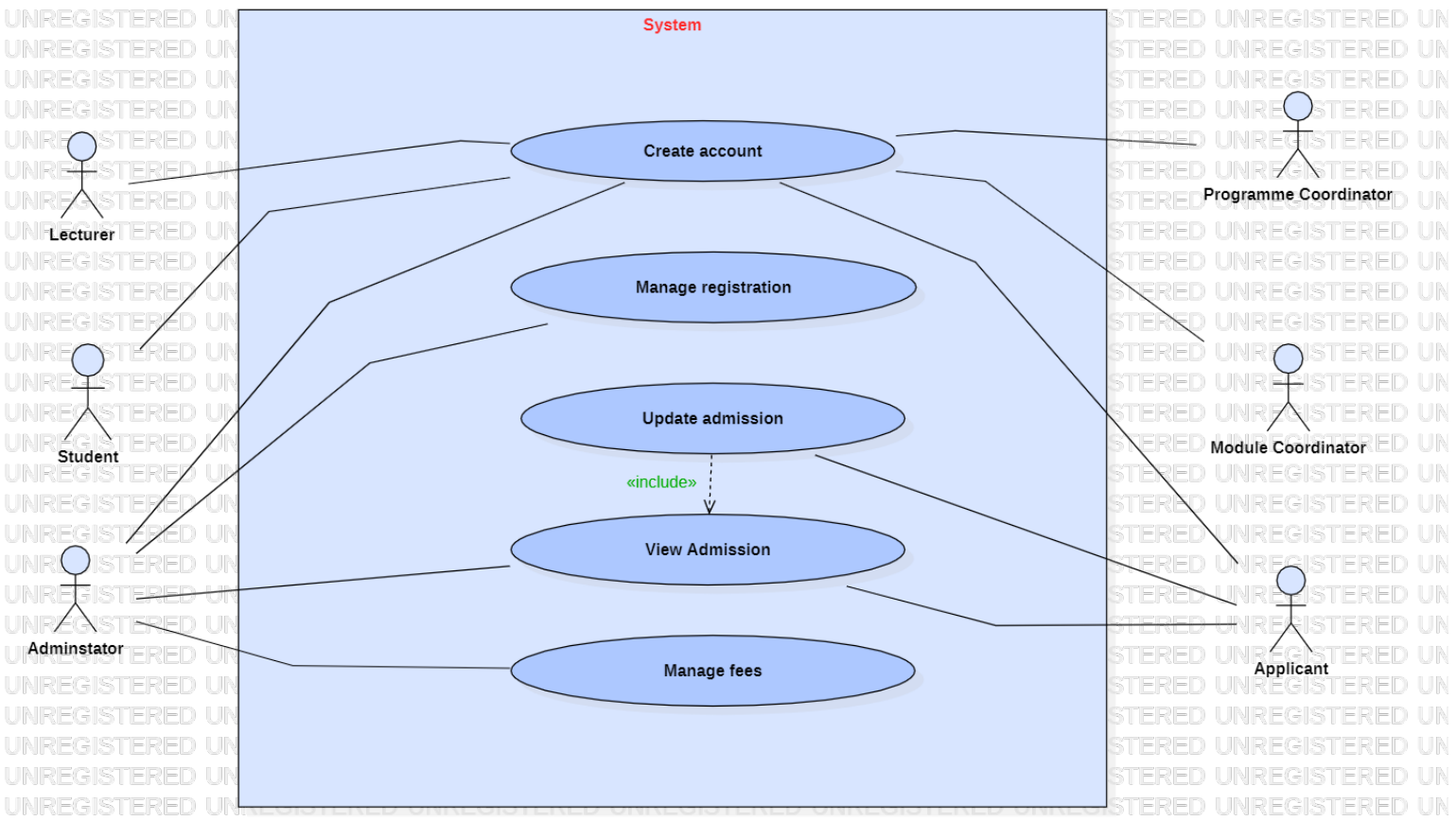
Use case name: Add attendance	
Participating actors(s): Lecturer	
Preconditions: Before the use case starts, actor must log in the system	
Flow of events	
Actor Action	**System Response**
1. Enter the attendance's details 3. Actor confirms the details and save	2. Validates if data have been typed in the correct format and allow actor to confirm the details. 4. Displays a confirming message that attendance has been added.
Postconditions: Attendance has been added successfully	
Special Requirements: None	

Use case name: Update attendance	
Participating actors(s): Lecturer	
Preconditions: Before the use case starts, actor must log in the system and then search for the attendance before proceeding with the update	
Flow of events	
Actor Action	**System Response**
1. Enter the attendance's details 3. Allow the actor to update the required data and save.	2. Retrieves and displays the attendance's details 4. Displays a confirming message that attendance has been updated.
Postconditions: Attendance has been updated successfully	
Special Requirements: None	

Use case name: Delete attendance	
Participating actors(s): Lecturer	
Preconditions: Before the use case starts, the actor must log in the system and then actor must search for the attendance before proceeding with the deletion	
Flow of events	
Actor Action	**System Response**
1. Enter attendance's details. 3. Selects delete.	2. Retrieves the data, display attendance's details and allow actor to delete. 4. Displays a confirming delete verification dialog.
Postconditions: Attendance has been deleted successfully	
Special Requirements: None	

Use case name: Search attendance	
Participating actors(s): Module coordinator, lecturer	
Preconditions: Before the use case starts, actors must log in the system.	
Flow of events	
Actor Action	**System Response**
1. Enter students' ID 3. The use case ends when the actors find the attendance.	2. Retrieves and displays the data if it exists.
Postconditions: The actors successfully find the attendance they are looking for.	
Special Requirements: Actors can search attendance by students' ID at a time	

5.2.14 Account



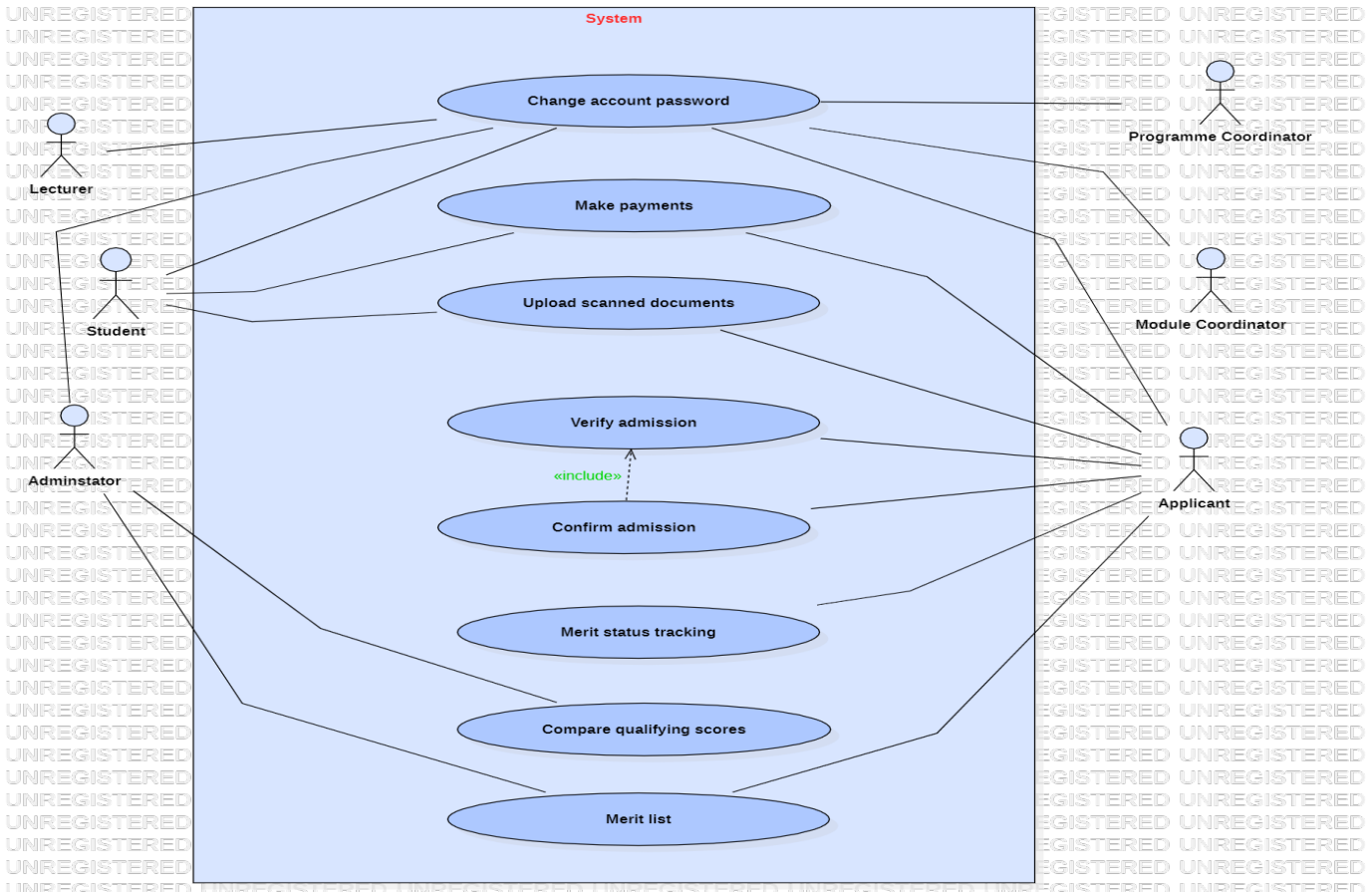
Use case name: Create Account	
Participating actors(s): Lecturer, Student, Administrator, Programme Coordinator, Module Coordinator, Applicant	
Preconditions: None	
Flow of events	
Actor Action	**System Response**
1. Enter the details for creation of the account 3. Actor confirms the details and save	2. Validates if data have been typed in the correct format and allow actor to confirm the details. 4. Displays a confirming message that account has been created
Postconditions: Account has been created successfully	
Special Requirements: None	

Use case name: Manage registration	
Participating actors(s): Administrator	
Preconditions: Before the use case starts, actor must log in the system	
Flow of events	
Actor Action	**System Response**
1. Enter registration's details 3. The use case ends when registration has been managed	2. Retrieves and displays the registration details
Postconditions: Registration has been managed successfully.	
Special Requirements: None	

Use case name: Update admission	
Participating actors(s): Applicant	
Preconditions: Before the use case starts, actors must log in the system and must view the admission first before proceeding with the update.	
Flow of events	
Actor Action	**System Response**
1. Enter admission's details 3. Allow the actor to update the required data and save.	2. Retrieves and displays the data if it exists. 4. Displays a confirming message that the admission has been updated
Postconditions: The admission has been updated successfully	
Special Requirements: None	

Use case name: View admission	
Participating actors(s): Applicant, Administrator	
Preconditions: Before the use case starts, actors must log in the system and must view the admission first before proceeding with the update.	
Flow of events	
Actor Action	**System Response**
1. Enter admission's details 3. The use case ends when the actor has viewed the admission	2. Retrieves and displays the data if it exists.
Postconditions: None	
Special Requirements: None	

5.2.15 Admission



Use case name: Manage fees	
Participating actors(s): Administrator	
Preconditions: Before the use case starts, actors must log in the system	
Flow of events	
Actor Action	**System Response**
1.Enter admission's details 3. Actor confirms and save	2.System shows if fees have been received and allow actor to confirm
Postconditions: Fees has been managed successfully	
Special Requirements: None	

Use case name: Change account password	
Participating actors(s): Lecturer, Student, Administrator, Programme Coordinator, Module Coordinator, Applicant	
Preconditions: Actors must log in the system first	
Flow of events	
Actor Action	**System Response**
1.Sign in and enter account's details (with current account password) 3. Allow the actor to change the account password and save	2. Retrieves and display data 4. Displays a confirming message that account's password has been changed
Postconditions: Password of the account has been changed successfully	
Special Requirements: None	

Use case name: Make payments	
Participating actors(s): Students, Applicants	
Preconditions: Before the use case starts, actors must log in the system	
Flow of events	
Actor Action	**System Response**
1.Enter payment's details 3. Actor select the mode of payment and confirm the transaction	2.Display the mode of payment 4. Shows a confirming message that payment has been done
Postconditions: Payment of fees has been successfully made	
Special Requirements: None	

Use case name: Upload scanned documents	
Participating actors(s): Applicants, students	
Preconditions: Actors must log in the system first	
Flow of events	
Actor Action	**System Response**
1.Select the required scanned document and upload it. 3. Actor confirms and save.	2. A pop of the dialog box to confirm the uploading of the documents 3. displays a confirming message that documents have been uploaded
Postconditions: Documents have been uploaded successfully	
Special Requirements: None	

Use case name: Verify admission	
Participating actors(s): Applicant	
Preconditions: Before the use case starts, actors must log in the system	
Flow of events	
Actor Action	**System Response**
1. Enter admission's details 3. The use case ends once actor has verified the admission	2. Retrieves and displays the data if it exists.
Postconditions: None	
Special Requirements: None	

Use case name: Confirm admission	
Participating actors(s): Applicant	
Preconditions: Before the use case starts, actors must log in the system and must verify the admission first before proceeding with the confirmation	
Flow of events	
Actor Action	**System Response**
1. Enter admission's details 3. Actor verifies and the confirm the admission	2. Retrieves and displays the data if it exists and allow actor to confirm 4. Displays a message that admission has been confirmed
Postconditions: The admission has been confirmed successfully	
Special Requirements: None	

Use case name: Merit status tracking	
Participating actors(s): Applicants	
Preconditions: Actor must log in the system first	
Flow of events	
Actor Action	**System Response**
1.Enter tracking details 3.Use case ends once actor viewed the status of the tracking	2. Displays the status of the tracking merit
Postconditions: None	
Special Requirements: None	

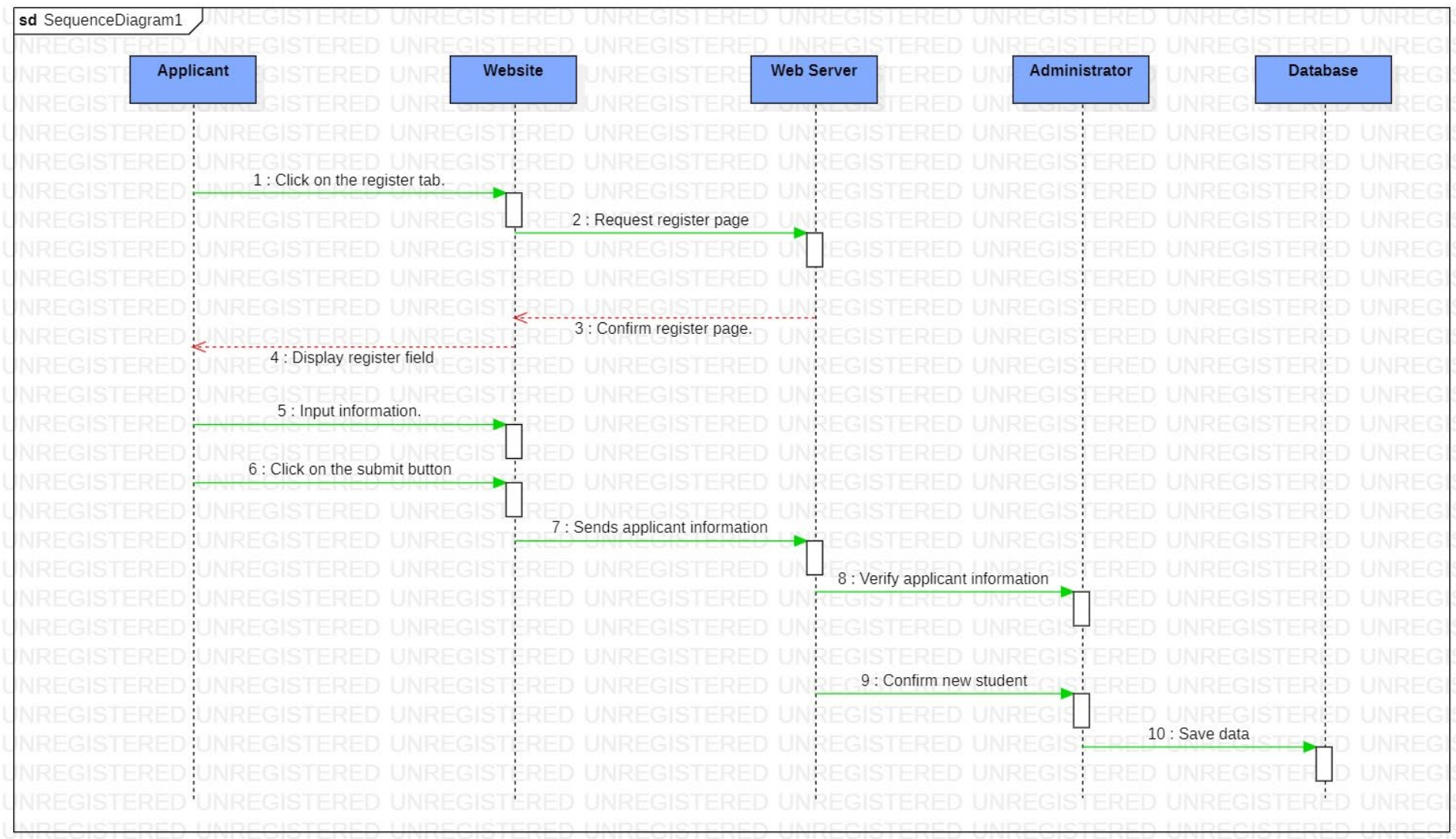
Use case name: Compare qualifying scores	
Participating actors(s): Administrator	
Preconditions: Actor must log in the system first	
Flow of events	
Actor Action	**System Response**
1.Enter the admission details 3.Actor then compares the qualifying scores	2. retrieves and display the admission results.
Postconditions: None	
Special Requirements: None	

Use case name: Merit List	
Participating actors(s): Administrator, Applicant	
Preconditions: Actors must log in the system first	
Flow of events	
Actor Action	**System Response**
1.Enter exam/ admission details. 3.Actors can view the merit list with the high achiever student / applicant at the top.	2. retrieves and display the admission results.
Postconditions: None	
Special Requirements: None	

5.3 Sequence Diagram

Sequence Diagram displays the time sequence of the objects participating in the interaction. This consists of the vertical dimension (time) and horizontal dimension (different objects).

In the diagram below, the applicant is the actor.



6. Database Design

6.1 Table Student

Attributes	Data Type	Field Size	Constraints
StudentID	Varchar	14	Primary key
StudentName	Varchar	50	Not null
StudentDOB	Date	7	Not null
StudentEmail	Varchar	50	Not null
StudentAddress	Varchar	100	Not null
StudentGender	Varchar	15	Not null
Cohort	Varchar	10	Foreign key
AttendanceID	Varchar	15	Foreign key
ProgrammeID	Varchar	10	Foreign key
ModuleID	Varchar	10	Foreign key
CourseworkID	Varchar	15	Foreign key
ResitCourseworkID	Varchar	15	Foreign key
ExamID	Varchar	15	Foreign key
ResitExamID	Varchar	15	Foreign key
ProjectID	Varchar	15	Foreign key
ResitProjectID	Varchar	15	Foreign key
PaymentID	Varchar	15	Foreign key
WorkPlacementID	Varchar	15	Foreign key
AccountID	Varchar	10	Foreign key
Result	Varchar	250	Foreign key

6.2 Table Programme

Attributes	Data Type	Field Size	Constraints
ProgrammeID	Varchar	10	Primary key
ProgrammeName	Varchar	100	Not null
ProgrammeDuration	Varchar	10	Not null
ProgrammeCoordinatorID	Varchar	14	Foreign key
LecturerID	Varchar	14	Foreign key
ModuleID	Varchar	10	Foreign key
StudentID	Varchar	14	Foreign key
Cohort	Varchar	10	Foreign key
WorkPlacementID	Varchar	15	Foreign key

6.3 Table Programme Coordinator

Attributes	Data Type	Field Size	Constraints
ProgrammeCoordinatorID	Varchar	14	Primary key
ProgrammeCoordinatorName	Varchar	50	Not null
ProgrammeCoordinatorAddress	Varchar	100	Not null
ProgrammeCoordinatorDOB	Date	7	Not null
ProgrammeCoordinatorGender	Varchar	15	Not null
ProgrammeCoordinatorMobileNumber	Number	12	Not null
ProgrammeCoordinatorEmail	Varchar	50	Not null
AccountID	Varchar	10	Foreign key
ProgrammeID	Varchar	10	Foreign key

6.4 Table Module

Attributes	Data Type	Field Size	Constraints
ModuleID	Varchar	10	Primary key
ModuleName	Varchar	100	Not null
ModuleDuration	Varchar	10	Not null
ModuleCoordinatorID	Varchar	14	Foreign key
LecturerID	Varchar	14	Foreign key
StudentID	Varchar	14	Foreign key
Cohort	Varchar	10	Foreign key
ProgrammeID	Varchar	10	Foreign key
ProgrammeCoordinatorID	Varchar	14	Foreign key
ModuleGPA	Nvarchar	5	Not null
CourseworkID	Varchar	15	Foreign key
ResitCourseworkID	Varchar	15	Foreign key
ExamID	Varchar	15	Foreign key
ResitExamID	Varchar	15	Foreign key
ProjectID	Varchar	15	Foreign key
ResitProjectID	Varchar	15	Foreign key

6.5 Table Module Coordinator

Attributes	Data Type	Field Size	Constraints
ModuleCoordinatorID	Varchar	14	Primary key
ModuleCoordinatorName	Varchar	50	Not null
ModuleCoordinatorAddress	Varchar	100	Not null
ModuleCoordinatorDOB	Date	7	Not null
ModuleCoordinatorGender	Varchar	15	Not null
ModuleCoordinatorEmail	Varchar	50	Not null
ModuleCoordinaorMobileNumber	Number	12	Not null
ModuleID	Varchar	10	Foreign key
ProgrammeID	Varchar	10	Foreign key
AccountID	Varchar	10	Foreign key

6.6 Table Cohort

Attributes	Data Type	Field Size	Constraints
Cohort	Varchar	10	Primary key
YearStart	Number	4	Not null
YearEnd	Number	4	Not null
StudentID	Varchar	14	Foreign key
ProgrammeID	Varchar	10	Foreign key
ModuleID	Varchar	10	Foreign key

6.7 Table Lecturer

Attributes	Data Type	Field Size	Constraints
LecturerID	Varchar	14	Primary key
LecturerName	Varchar	50	Not null
LecturerGender	Varchar	15	Not null
LecturerDOB	Date	7	Not null
LecturerAddress	Varchar	100	Not null
LecturerMobileNumber	Number	12	Not null
LecturerEmail	Varchar	50	Not null
AttendanceID	Varchar	15	Foreign key
ProgrammeCoordinatorID	Varchar	14	Foreign key
ModuleCoordinatorID	Varchar	14	Foreign key
CourseworkID	Varchar	15	Foreign key
ResitCourseworkID	Varchar	15	Foreign key
ProjectID	Varchar	15	Foreign key
ResitProjectID	Varchar	15	Foreign key
AccountID	Varchar	10	Foreign key

6.8 Table Coursework

Attributes	Data Type	Field Size	Constraints
CourseworkID	Varchar	15	Primary key
CourseworkName	Varchar	100	Not null
CourseworkStartline	Varchar	15	Not null
CourseworkDeadline	Varchar	15	Not null
CourseworkGrade	Char	1	Not null
LecturerID	Varchar	14	Foreign key
StudentID	Varchar	14	Foreign key
Cohort	Varchar	10	Foreign key
ModuleID	Varchar	10	Foreign key
ProgrammeID	Varchar	10	Foreign key
ModuleCoordinatorID	Varchar	14	Foreign key
ModeratedCourseworkReview	Varchar	250	Null

6.9 Table Re-sit Coursework

Attributes	Data Type	Field Size	Constraints
ResitCourseworkID	Varchar	15	Primary key
ResitCourseworkName	Varchar	100	Not null
ResitCourseworkStartline	Varchar	15	Not null
ResitCourseworkDeadline	Varchar	15	Not null
ResitCourseworkGrade	Char	1	Null
StudentID	Varchar	14	Foreign key
Cohort	Varchar	10	Foreign key
LecturerID	Varchar	14	Foreign key
ModuleID	Varchar	10	Foreign key
ModuleCoordinatorID	Varchar	14	Foreign key
ProgrammeID	Varchar	10	Foreign key

6.10 Table Exam

Attributes	Data Type	Field Size	Constraints
ExamID	Varchar	15	Primary key
ExamName	Varchar	100	Not null
ExamDuration	Varchar	10	Not null
ExamStartDate	Date	7	Not null
ExamEndDate	Date	7	Not null
ProgrammeID	Varchar	10	Foreign key
ModuleID	Varchar	10	Foreign key
ModuleCoordinatorID	Varchar	14	Foreign key
ProgrammeCoordinatorID	Varchar	14	Foreign key
Cohort	Varchar	10	Foreign key
StudentID	Varchar	14	Foreign key
ExamGrade	Char	1	Not null
ModeratedExaminationReview	Varchar	250	Null

6.11 Table Re-sit Exam

Attributes	Data Type	Field Size	Constraints
ResitExamID	Varchar	15	Primary key
ResitExamName	Varchar	100	Not null
ResitExamDuration	Varchar	10	Not null
ResitExamStartDate	Date	7	Not null
ResitExamEndDate	Date	7	Not null
Cohort	Varchar	10	Foreign key
StudentID	Varchar	14	Foreign key
ProgrammeID	Varchar	10	Foreign key
ProgrammeCoordinatorID	Varchar	14	Foreign key
ModuleID	Varchar	10	Foreign key
ModuleCoordinatorID	Varchar	14	Foreign key
ResitExamGrade	Char	1	Null

6.12 Table Project

Attributes	Data Type	Field Size	Constraints
ProjectID	Varchar	15	Primary key
ProjectName	Varchar	100	Not null
ProjectStartline	Varchar	15	Not null
ProjectDeadline	Varchar	15	Not null
ProgrammeID	Varchar	10	Foreign key
ModuleID	Varchar	10	Foreign key
ModuleCoordinatorID	Varchar	14	Foreign key
LecturerID	Varchar	14	Foreign key
Cohort	Varchar	10	Foreign key
StudentID	Varchar	14	Foreign key
ProjectGrade	Char	1	Not null
ModeratedProjectReview	Varchar	250	Null

6.13 Table Re-sit Project

Attributes	Data Type	Field Size	Constraints
ResitProjectID	Varchar	15	Primary key
ResitProjectName	Varchar	100	Not null
ResitProjectStartline	Varchar	15	Not null
ResitProjectDeadline	Varchar	15	Not null
ProgrammeID	Varchar	10	Foreign key
ModuleID	Varchar	10	Foreign key
ModuleCoordinatorID	Varchar	14	Foreign key
LecturerID	Varchar	14	Foreign key
Cohort	Varchar	10	Foreign key
StudentID	Varchar	14	Foreign key
ResitProjectGrade	Char	1	Null

6.14 Table Work Placement

Attributes	Data Type	Field Size	Constraints
WorkPlacementID	Varchar	15	Primary key
WorkPlacementName	Varchar	100	Not null
WorkPlacementAddress	Varchar	100	Not null
WorkPlacementStartingDate	Date	7	Not null
WorkPlacementEndingDate	Date	7	Not null
ProgrammeID	Varchar	10	Foreign key
ProgrammeCoordinatorID	Varchar	14	Foreign key
Cohort	Varchar	10	Foreign key
StudentID	Varchar	14	Foreign key
WorkPlacementReview	Varchar	250	Null

6.15 Table Board of Examiners

Attributes	Data Type	Field Size	Constraints
BOE_ID	Varchar	10	Primary key
ProgrammeID	Varchar	10	Foreign key
ProgrammeCoordinatorID	Varchar	14	Foreign key
ModuleID	Varchar	10	Foreign key
ModuleCoordinatorID	Varchar	14	Foreign key

6.16 Table Result

Attributes	Data Type	Field Size	Constraints
Result	Varchar	250	Primary key
Cohort	Varchar	10	Foreign key
StudentID	Varchar	14	Foreign key
ProgrammeID	Varchar	10	Foreign key
CourseworkID	Varchar	15	Foreign key
ResitCourseworkID	Varchar	15	Foreign key
ExamID	Varchar	15	Foreign key
ResitExamID	Varchar	15	Foreign key
ProjectID	Varchar	15	Foreign key
ResitProjectID	Varchar	15	Foreign key
TranscriptName	Varchar	250	Not null

6.17 Table Attendance

Attributes	Data Type	Field Size	Constraints
AttendanceID	Varchar	15	Primary key
Cohort	Varchar	10	Foreign key
NumOfStudents	Number	3	Not null
StudentID	Varchar	14	Foreign key
Absent/Present	Varchar	7	Not null
WeekNo	Number	2	Not null
ProgrammeID	Varchar	10	Foreign key
ModuleID	Varchar	10	Foreign key
ModuleCoordinatorID	Varchar	14	Foreign key
LecturerID	Varchar	14	Foreign key
Remarks	Varchar	250	Null

6.18 Table Account

Attributes	Data Type	Field Size	Constraints
AccountID	Varchar	10	Primary key
AccountName	Varchar	50	Not null
StudentID	Varchar	14	Foreign key
ProgrammeCoordinatorID	Varchar	14	Foreign key
ModuleCoordinatorID	Varchar	14	Foreign key
LecturerID	Varchar	14	Foreign key
AdmissionID	Varchar	10	Foreign key

6.19 Table Applicant

Attributes	Data Type	Field Size	Constraints
ApplicantID	Varchar	14	Primary key
ApplicantName	Varchar	50	Not null
ApplicantDOB	Date	7	Not null
ApplicantGender	Varchar	15	Not null
ApplicantEmail	Varchar	100	Not null
ApplicantContactNo	Number	12	Not null
ApplicantAddress	Varchar	100	Not null
ApplicantDocs	Varchar	50	Not null
ApplicantStatus	Varchar	50	Not null
PaymentID	Varchar	15	Foreign key

6.20 Table Admission

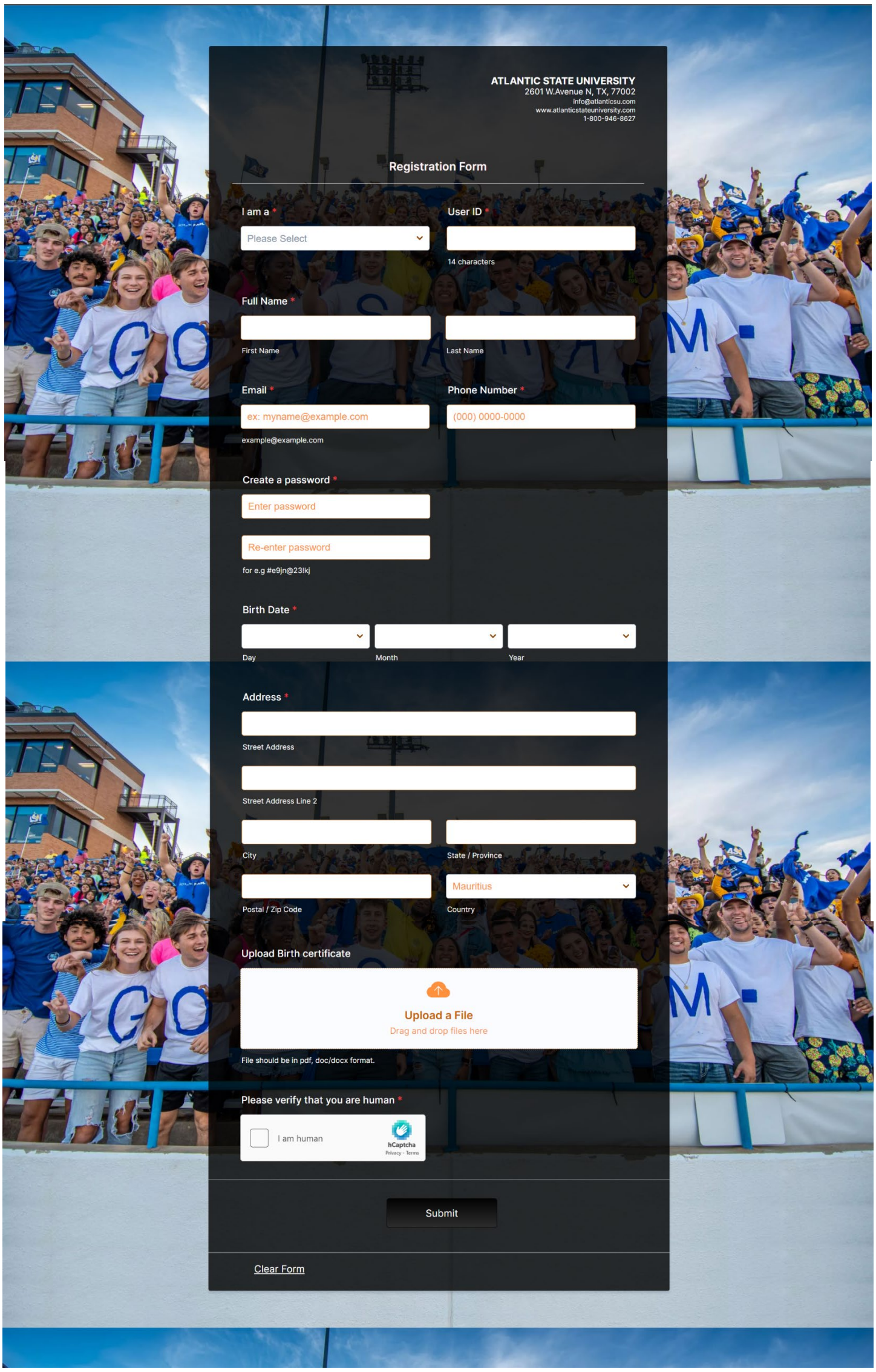
Attributes	Data Type	Field Size	Constraints
AdmissionID	Varchar	10	Primary key
AdmissionDate	Date	7	Not null
NoOfDocScanned	Number	2	Not null
AccountID	Varchar	10	Foreign key
ApplicantID	Varchar	14	Foreign key
AdmissionStatus	Varchar	50	Not null
MeritStatusTracking	Varchar	50	Not null
QualifyingScores	Number	2	Not null
MeritList	Varchar	250	Not null

6.21 Table Payment

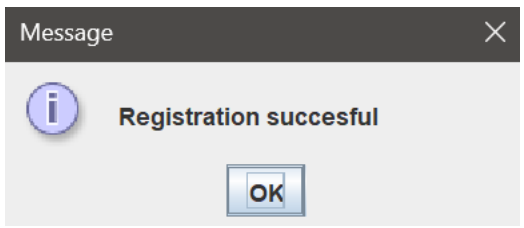
Attributes	Data Type	Field Size	Constraints
PaymentID	Varchar	15	Primary key
PaymentDate	Date	7	Not null
StudentID	Varchar	14	Foreign key
Cohort	Varchar	10	Foreign key
AccountID	Varchar	10	Foreign key
ApplicantID	Varchar	14	Foreign key
PaymentMethod	Varchar	20	Not null
FeesPaid	Number	6	Not null
FeesDue	Number	6	Not null

7. Input and output Designs

7.1 Registration

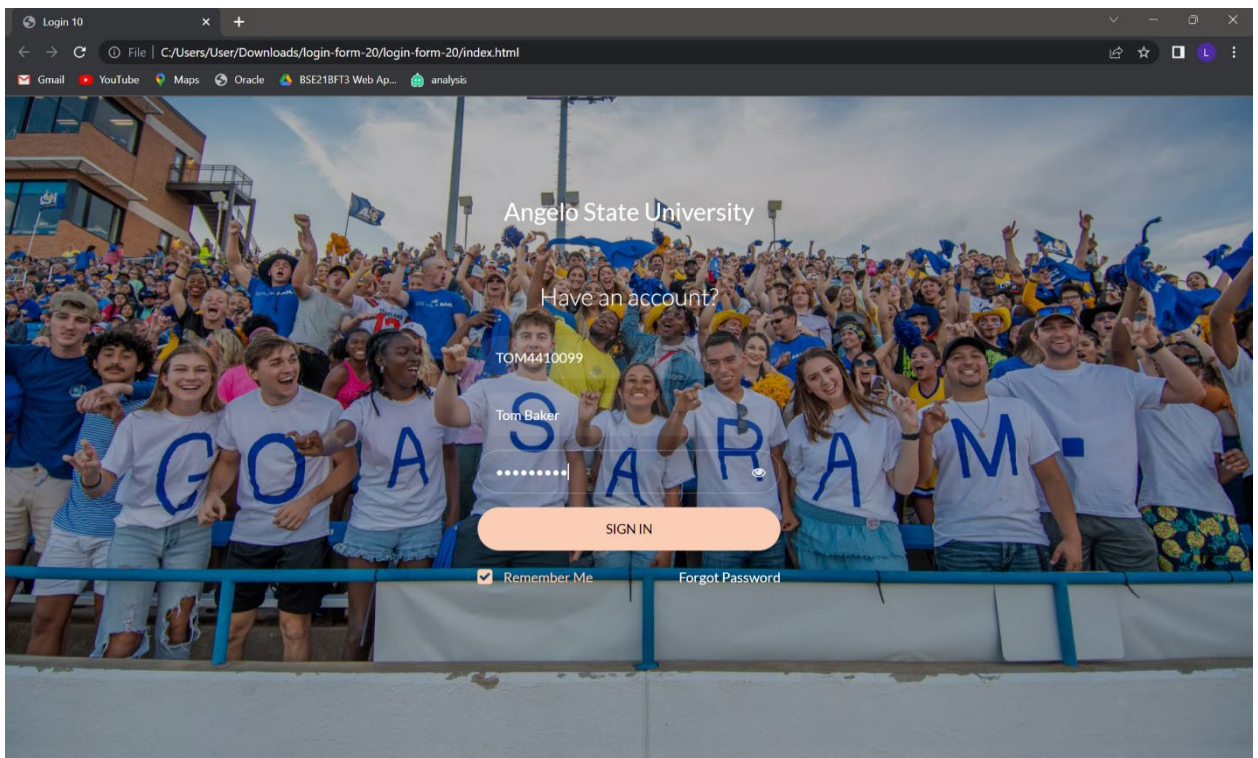


Description: Users such as Lecturer, Student, Administrator, Programme Coordinator, Module Coordinator, Applicant can create and account and register themselves.

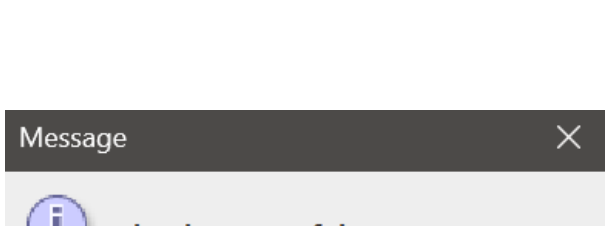


Description: User was able to register successfully.

7.2 Login



Description: Users log into their account using their account ID.

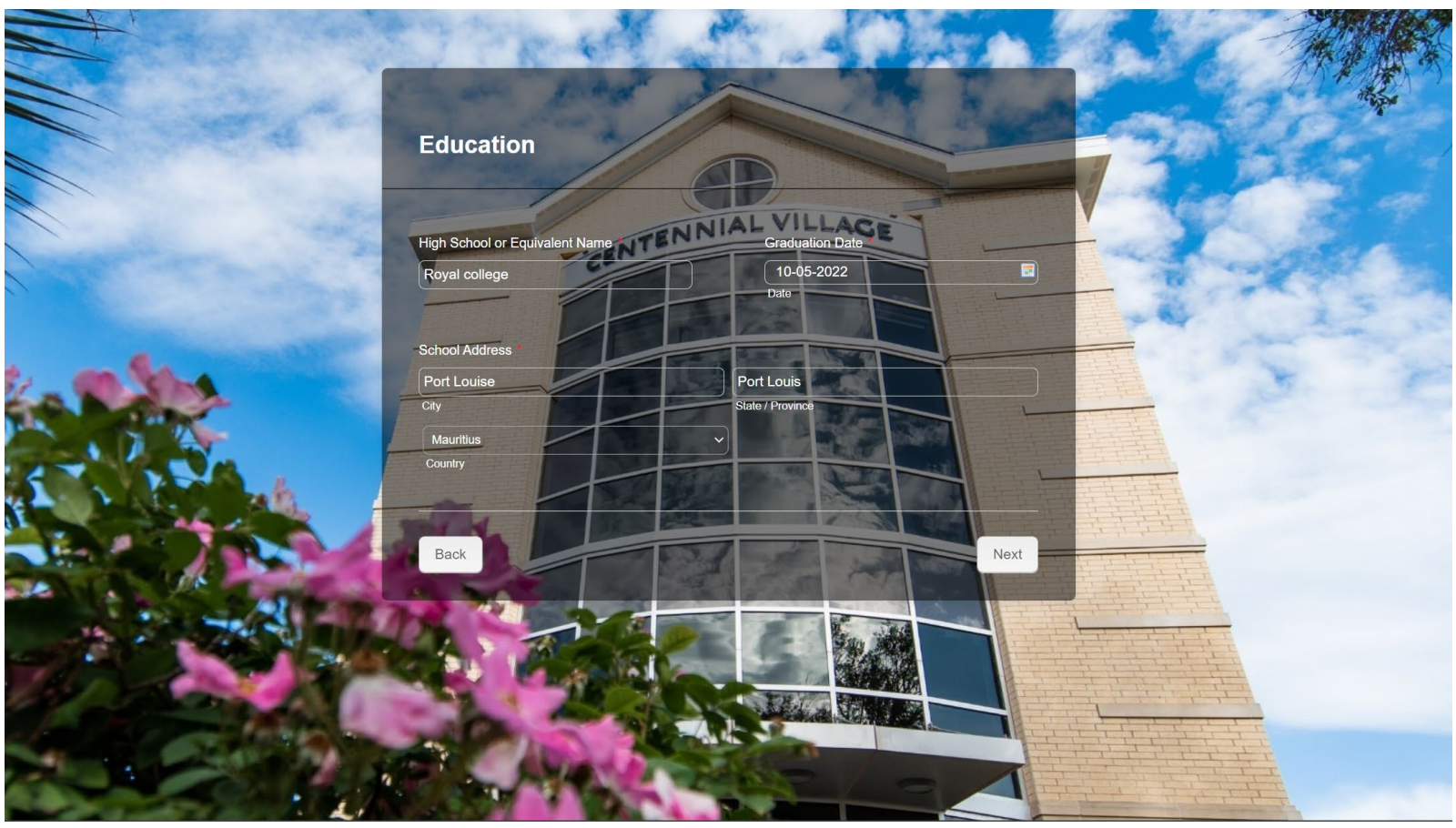


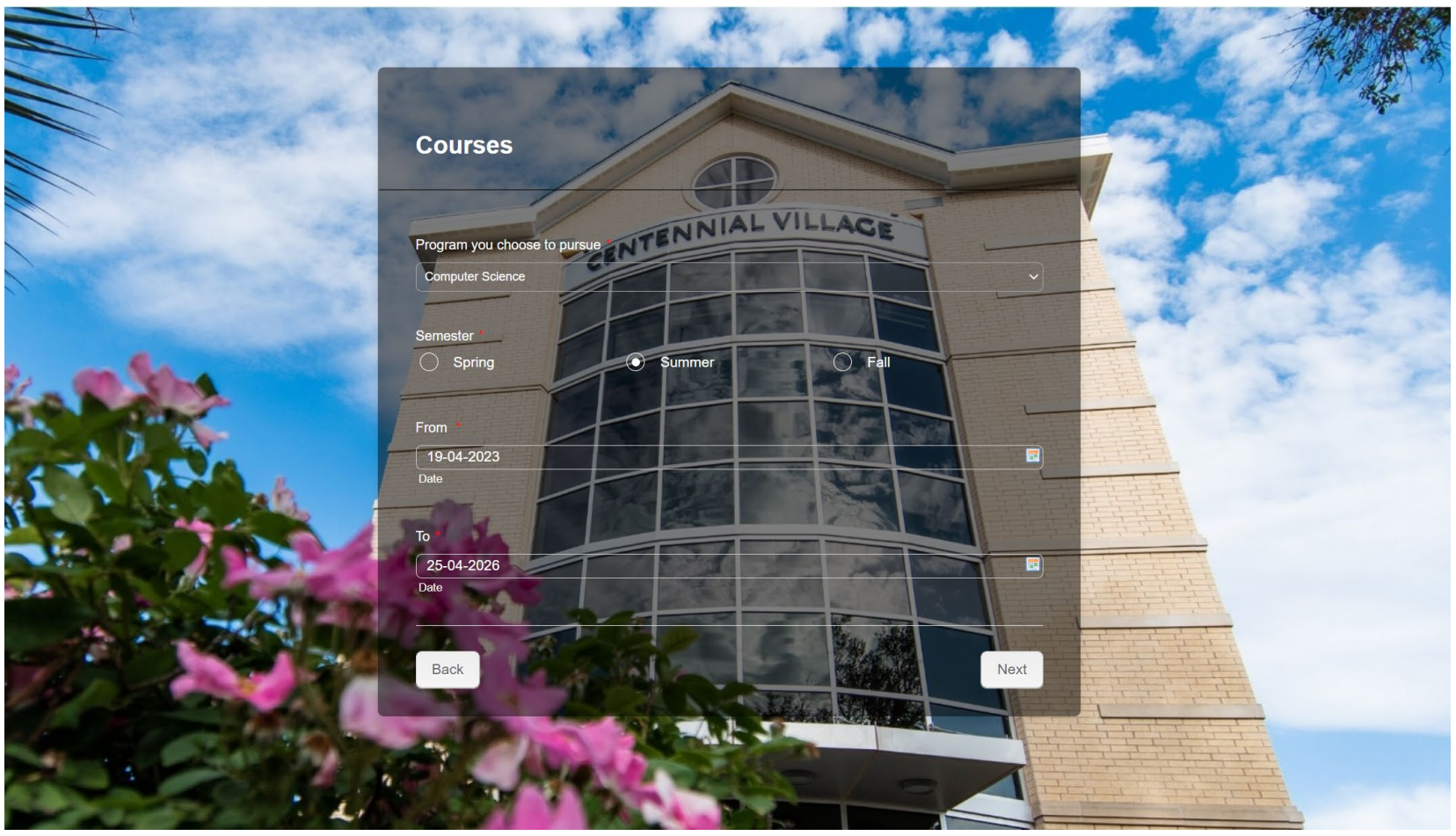
Description: User was able to log into their account successfully.

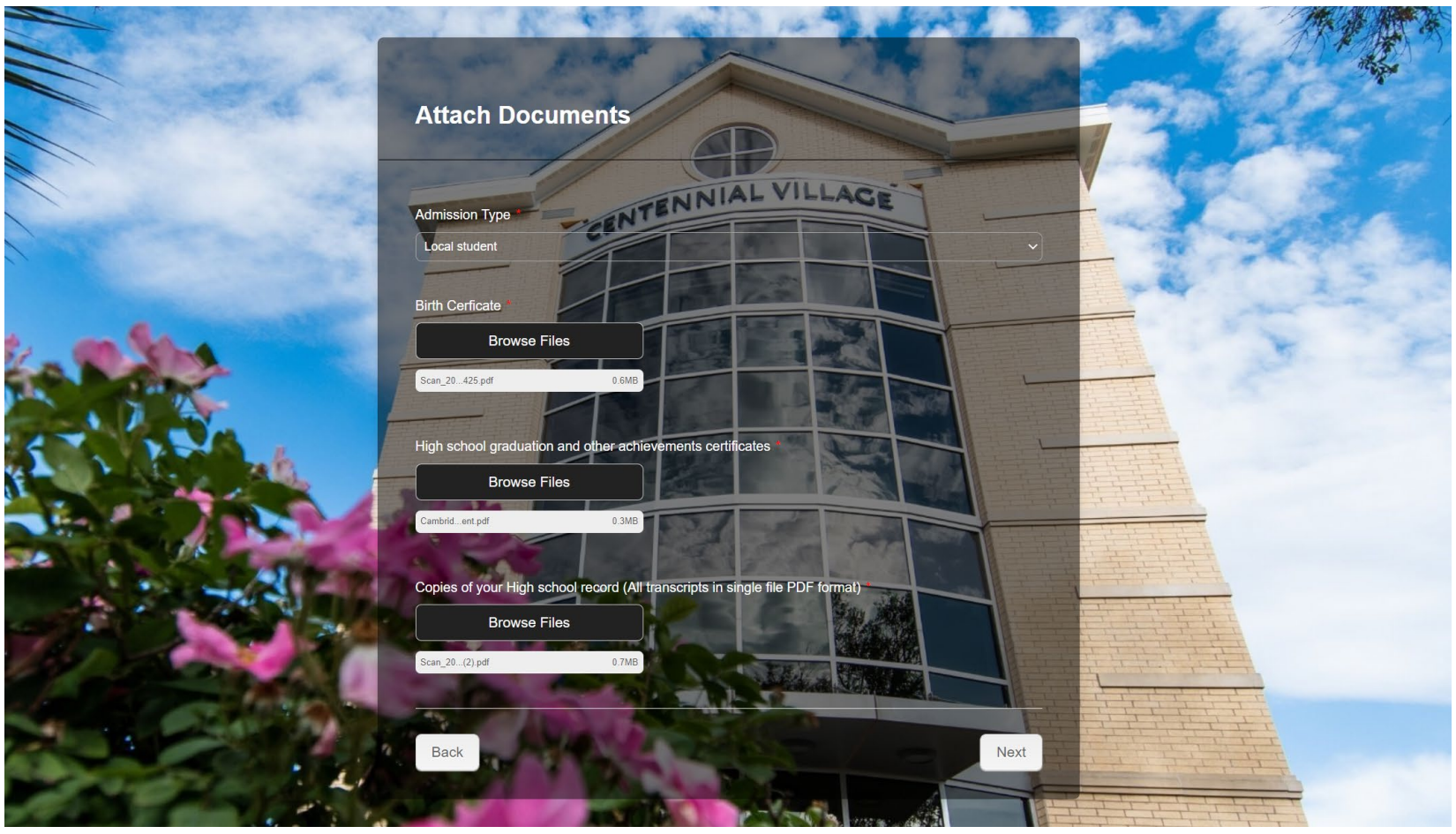
View Login form

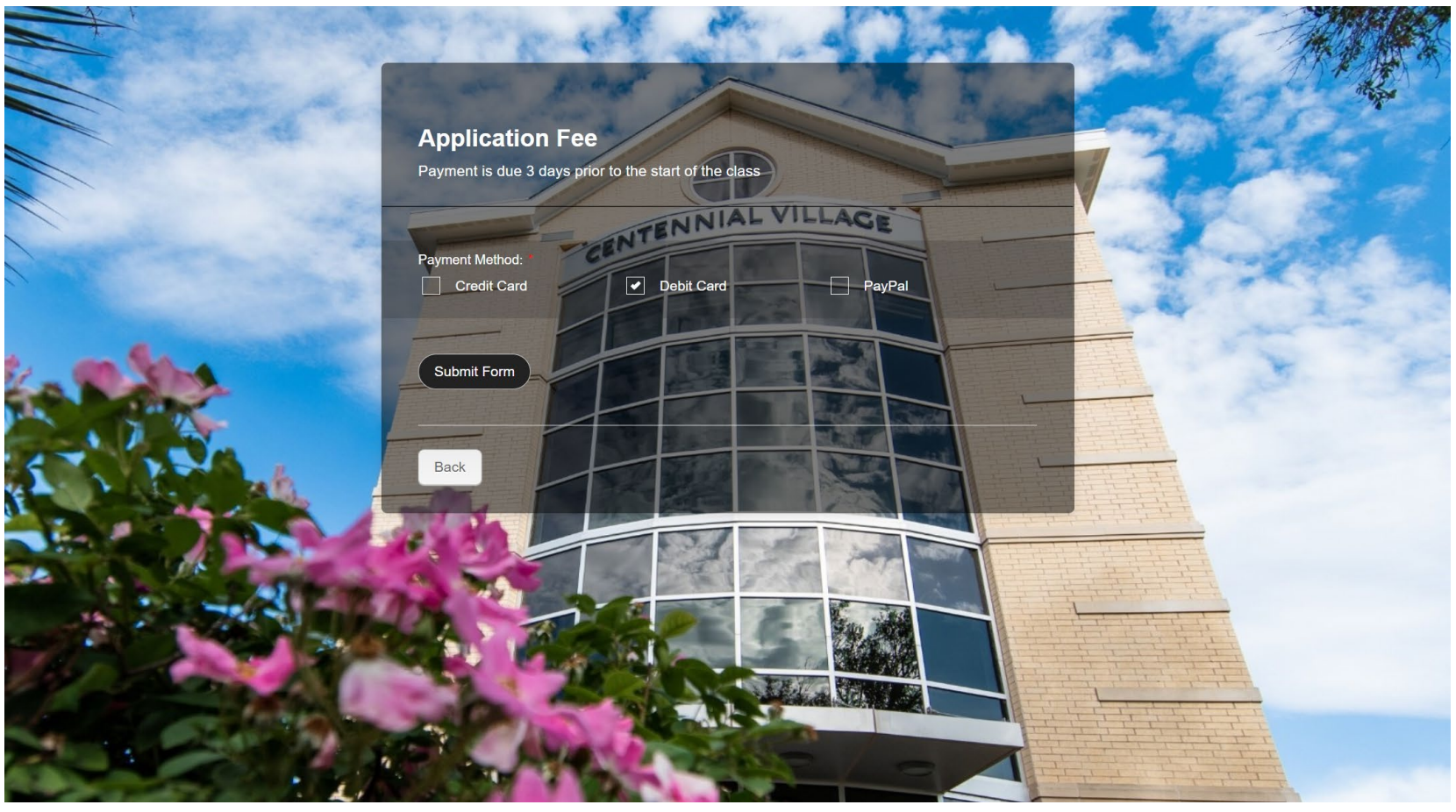
Description: Details can be added, updated, and deleted.

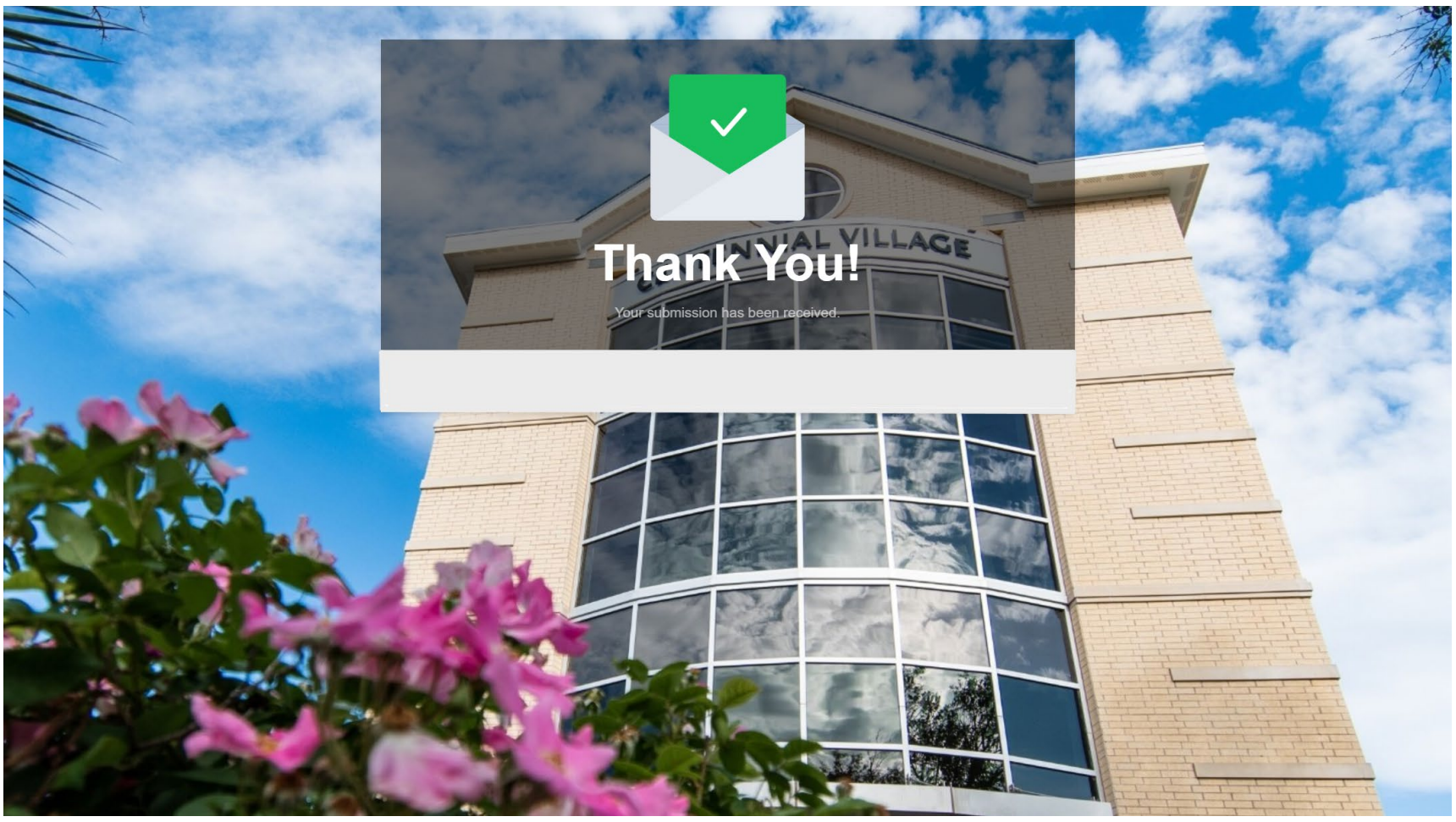
7.3 Admission



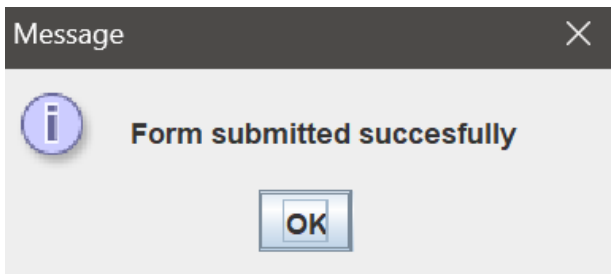

Courses
Program you choose to pursue *
Computer Science
Semester *
Spring
Summer
Fall
From *
19-04-2023
Date
To *
25-04-2026
Date
Back
Next


Attach Documents
Admission Type *
Local student
Birth Cerficate *
Browse Files
Scan_20...425.pdf
0.6MB
High school graduation and other achievements certificates *
Browse Files
Cambrid...ent.pdf
0.3MB
Copies of your High school record (All transcripts in single file PDF format) *
Browse Files
Scan_20...(2).pdf
0.7MB
Back
Next

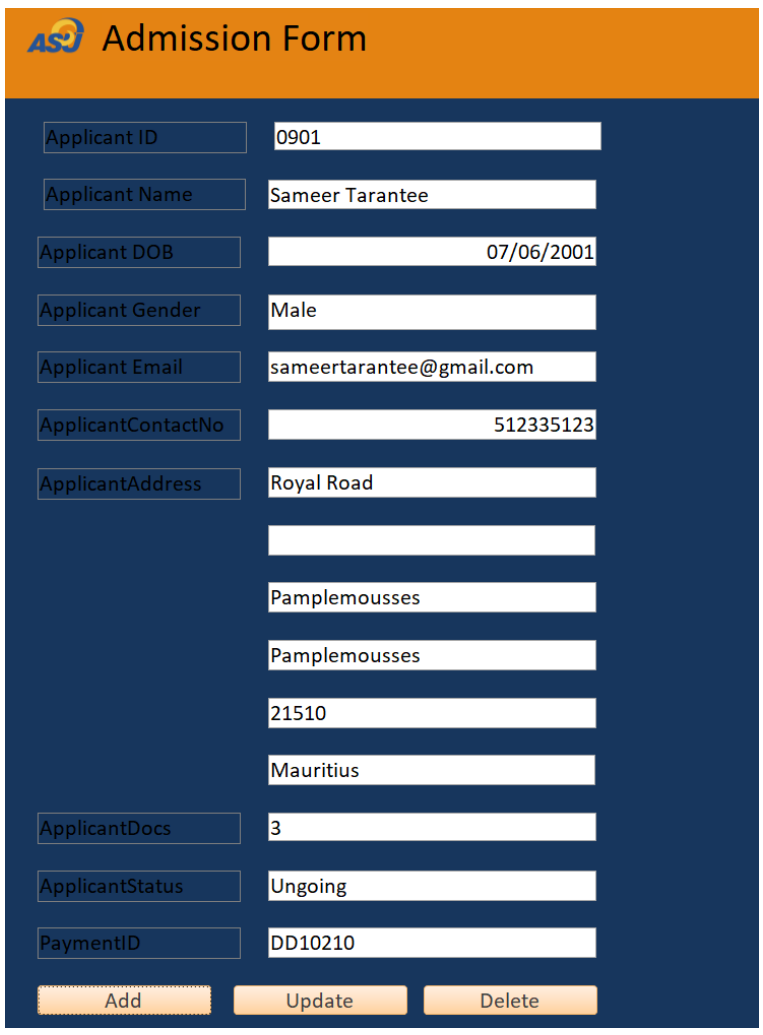

Application Fee
Payment is due 3 days prior to the start of the class
Payment Method: *
Credit Card
Debit Card
PayPal
Submit Form
Back



Description: Applicants fill out admission form and submit.



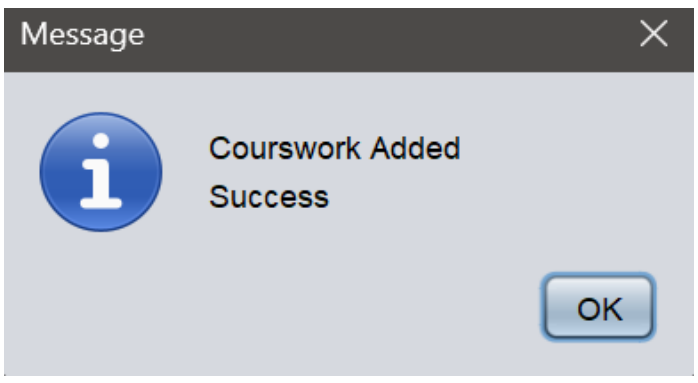
Description: Admission form submitted successfully.



Description: Application form details can be added, updated, and deleted.

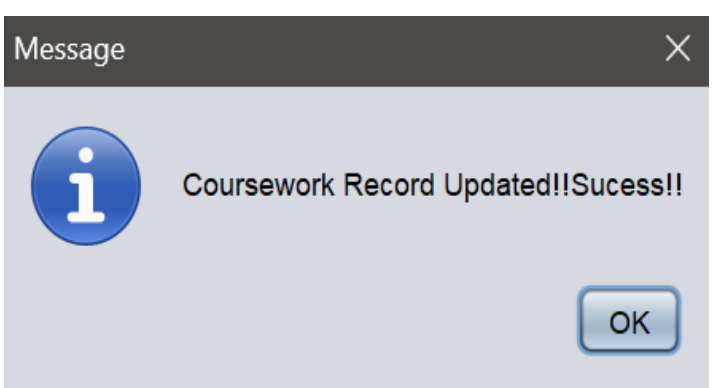
7.4 Coursework

Description: We have entered coursework details. Records can be updated, deleted, and searched.



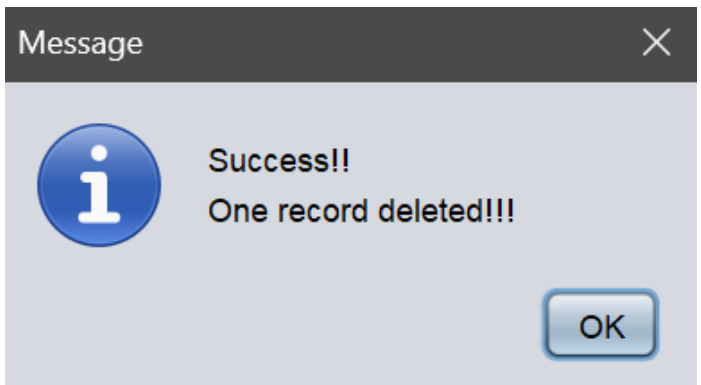
Add coursework

Description: Coursework has been added successfully.



Update coursework

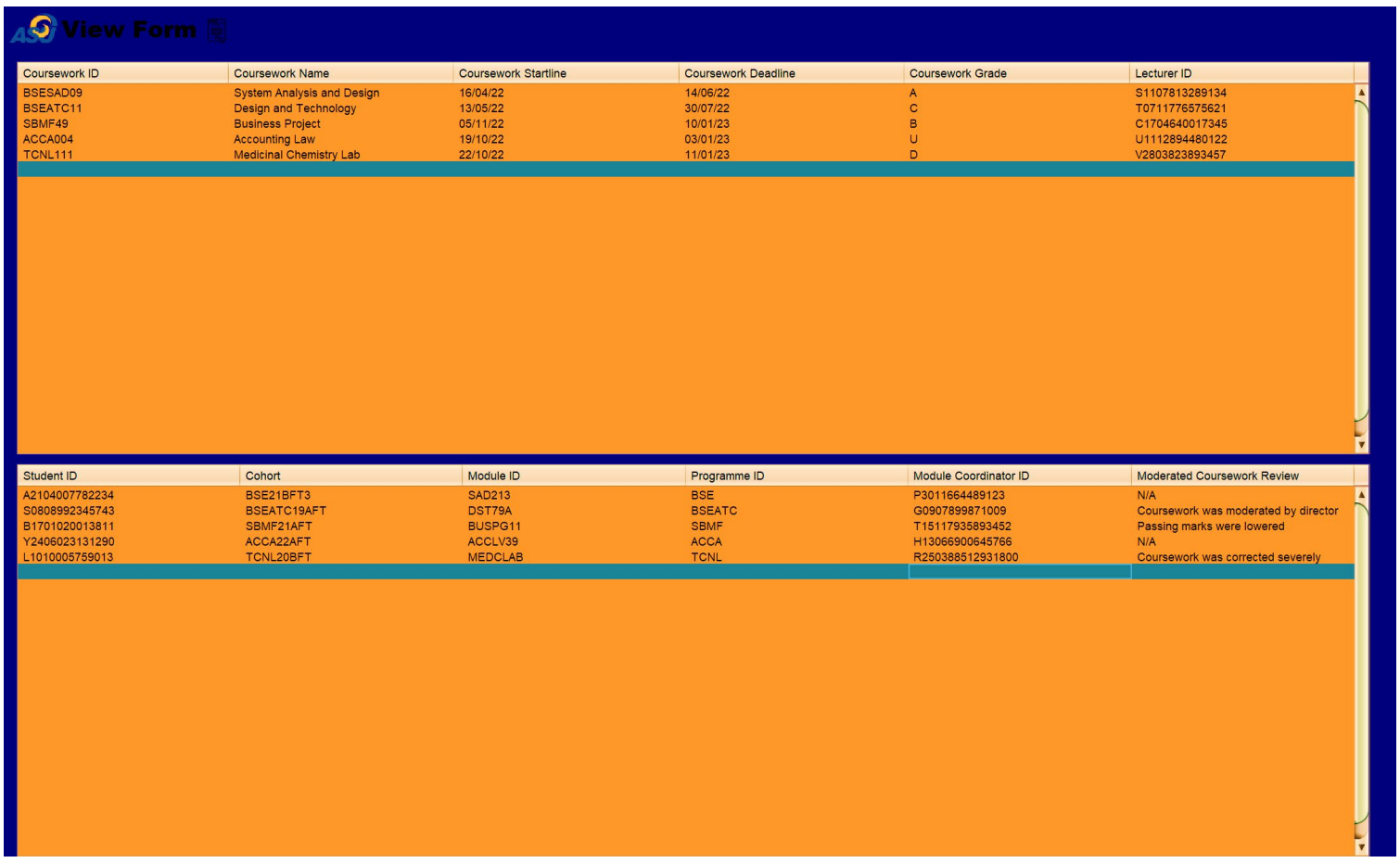
Description: Coursework has been updated successfully.



Delete coursework

Description: Coursework has been deleted successfully.

View coursework



Description: We can view the complete list we have entered of coursework details.

8. Testing

Software testing is a very essential aspect that forms part in the software development cycle. The testing demonstrate the Quality of the software developed and reflects the ultimate view of specification, design, and coding. Software testing is known as the process through which one can inspect anomalies in the software program. Testing is a set of activities that work towards the integration of an entire computer-based system.

Testing Objectives:

1. Efficient testing helps preventing defects and that helps in provided an error-free application
2. To evaluate work products such as requirements, user stories, design, and code
3. To find failures and defects. Defects should be primarily identified exceedingly early in the test cycle
4. To check whether all mentioned requirements have been fulfilled
5.To make informed decisions, especially regarding the level of quality of the test object
6. To validate whether the test object is completed and works as the users and other stakeholders expect
7. To reduce the level of risk of inadequate software quality
8. To build confidence in the level of quality of the test object

8.1 White Box Testing

White-box testing techniques analyse the internal architecture, the data structures used, the internal design, the code structure, and the behaviour of the software, in addition to functionality like black-box testing.

Software development can be tested at the system, integration, and unit levels. Verifying that an application's working flow is one of White box testing's fundamental objectives. It is comparing a sequence of specified inputs to desired or expected outputs to identify bugs when a particular input does not provide the desired outcome.

8.2 Black Box Testing

Black box testing entails evaluating a system without being aware of how it operates within. A tester enters data and analyze the output produced by the system being tested. This allows for the identification of the system's response time, usability difficulties, and reliability concerns as well as how the system reacts to expected and unexpected user activities.

Since it tests a system from the start until the end, black box testing is said to be a redoubtable testing method. A tester can imitate user action to check whether the system fulfills its promises, much as end users "don't care" how a system is programmed or designed and expect to get a suitable response to their requests. A black box test checks all pertinent subsystems along the route, including UI/UX, the web server or application server, the database, dependencies, and integrated systems.

8.3 Unit Testing

Individual software components or components are tested as part of a type of software testing known as unit testing. The goal is to confirm that each piece of software code operates as intended. Developers perform unit testing while creating an application (the coding phase). Unit tests isolate a specific piece of code and confirm its accuracy. A singular function, method, procedure, module, or object might be considered a unit.

Unit testing is the first level of testing carried out prior to integration testing in the SDLC, STLC, and V Model. Unit testing is a type of White Box testing that is often carried out by the developer. However, in the real world, QA engineers also perform unit testing because of time constraints or developers' resistance to testing.

8.4 Integration Testing

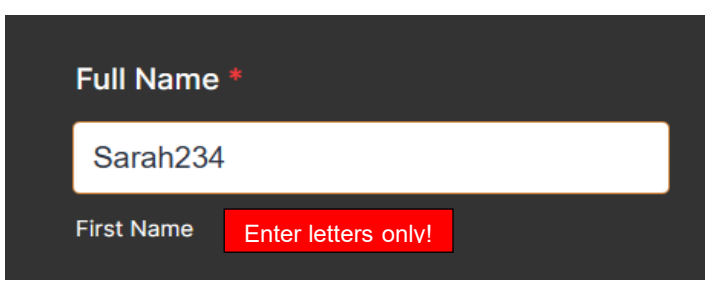
Software components are logically connected and tested as a unit in a type of testing called integration testing. Multiple software modules created by various programmers make up a typical software project. This level of testing's objective is to find issues with how various software modules interact when they are combined.

The main goal of integration testing is to examine how effectively these modules communicate data. The name "I & T" (Integration and Testing), "String Testing," and occasionally "Thread Testing" are also used to describe it.

9. Test Cases

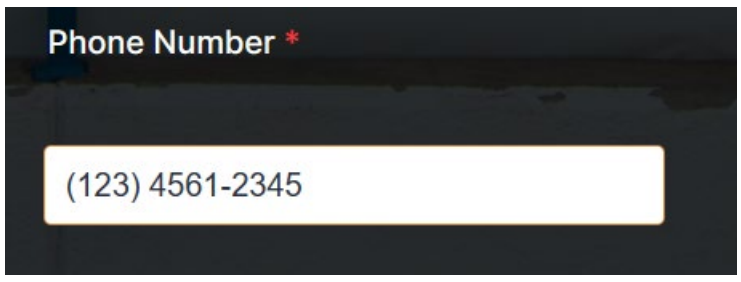
9.1 Registration form:

Test Case: First Name		
Test Values	**Result**	**Displayed error message**
Sarah	Correct	-
Sarah234	Incorrect	Enter letters only
Sarah/*	Incorrect	Enter letters only

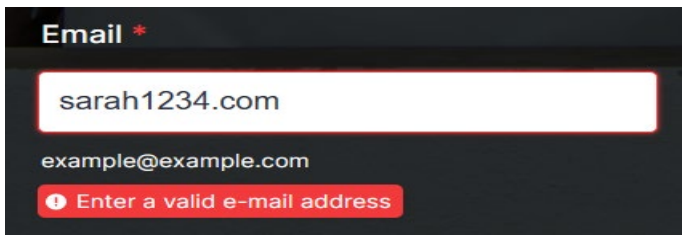


Test Case: Last Name		
Test Values	**Result**	**Displayed error message**
Jones	Correct	-
Jones234	Incorrect	Enter letters only
Jones/*	Incorrect	Enter letters only

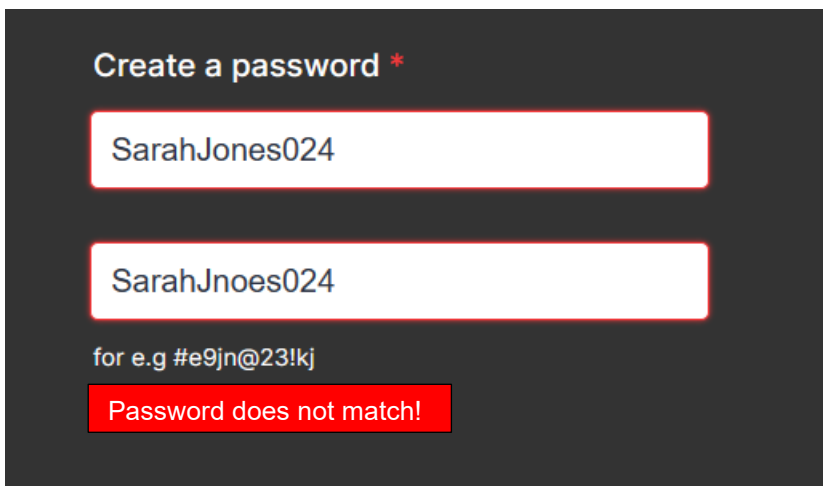
Test Case: Phone Number		
Test Values	**Result**	**Displayed error message**
(123) 4561-2344	Correct	-
(230)4561-23ab	Incorrect	System does not display anything when letters are added



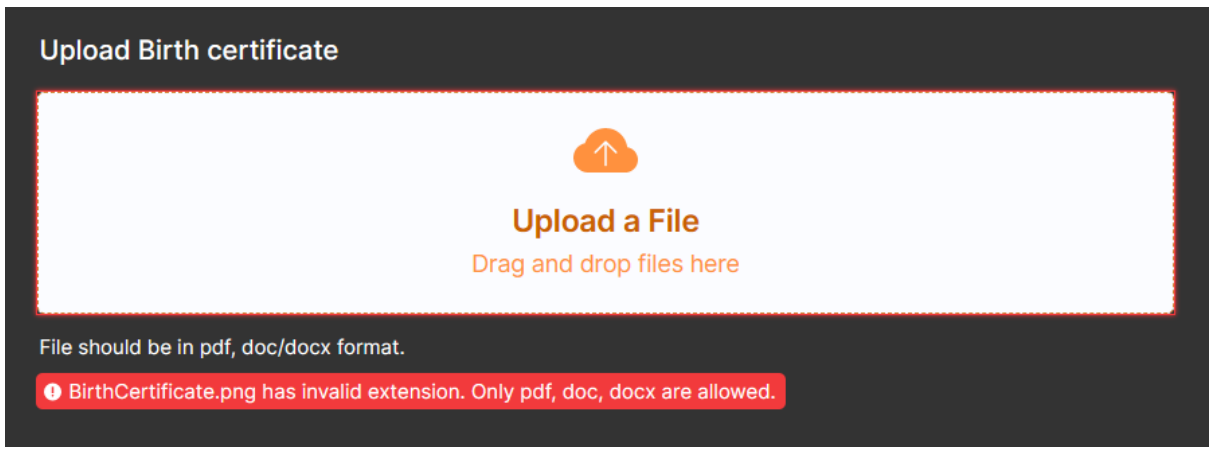
Test Case: Email		
Test Values	**Result**	**Displayed error message**
SarahJones@gmail.com	Correct	-
Sarah1234.com	Incorrect	Enter a valid email address
@gmail.comsarah	Incorrect	Enter a valid email address



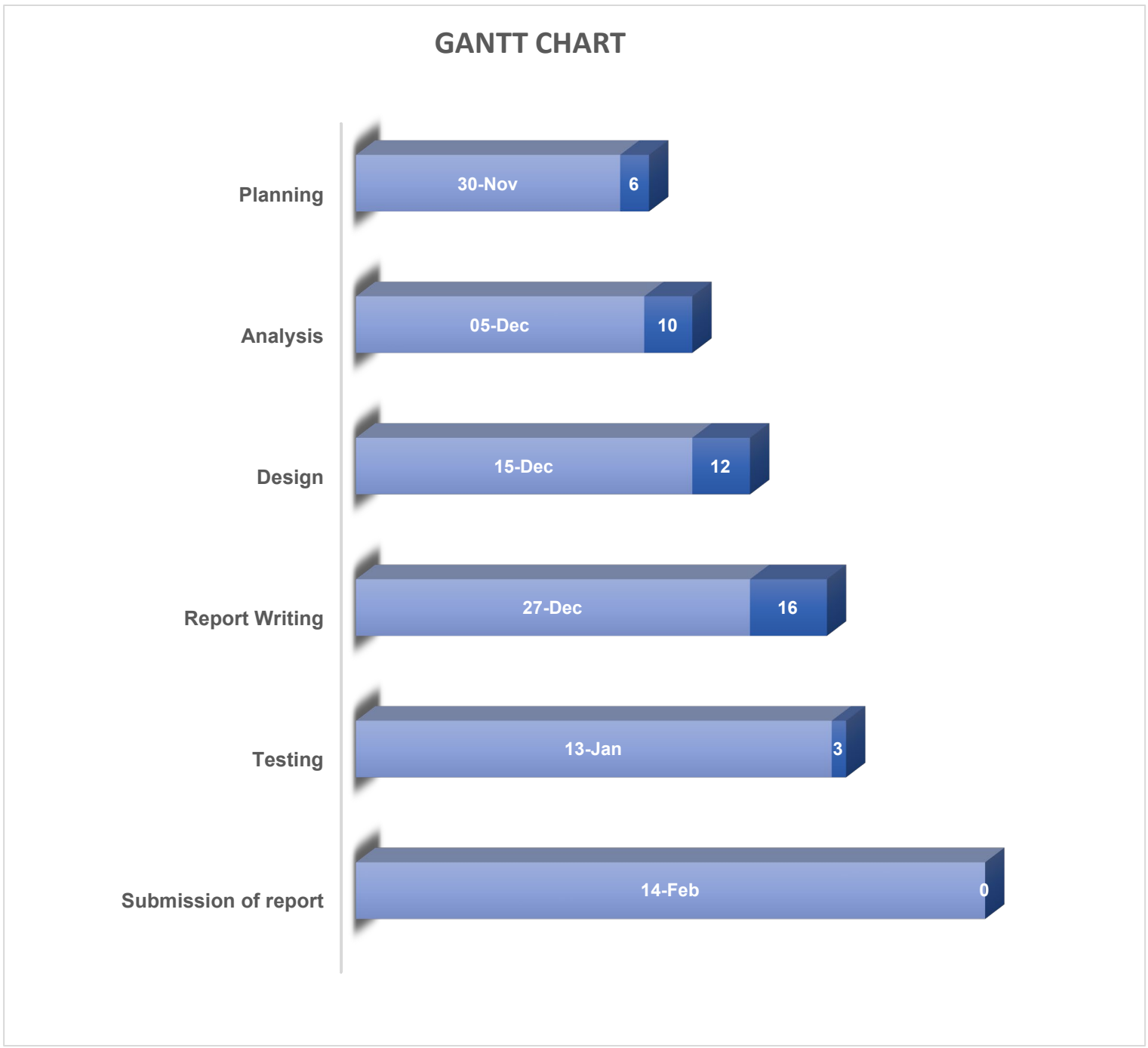
Test Case: Password		
Test Values	**Result**	**Displayed error message**
Enter Password: SarahJones024	Correct	-
Re-enter Password: SarahJones024	Correct	-
Enter Password: SarahJones024	Incorrect	-
Re-enter Password: SarahJnoes024	Incorrect	Password does not match!



Test Case: Upload Birth Certificate		
Test Values	**Result**	**Displayed error message**
File with pdf, doc extension	Correct	-
File with png extension	Incorrect	File has an invalid extension Only pdf, doc, docx are allowed



10. Gantt Chart



11. Conclusion

To summarize everything, the idea behind this management system consists of mainly to reduce the exhaustive burden of admission and administrative paperwork. This new concept makes the jobs of all the system users much easier and also a little enjoyable compared to the old manual system. An effective and a proper student management system is so fundamental for the success of tertiary educational establishments. Nowadays, with the increasing number of students enrolments, most of the institutions tend to stick to the student management system to improve operational efficiency and encourage students' overall experience throughout their journeys!