**New Zealand Diploma in Information Systems**

**HTCS5607 IS Application Project**

**TECHNICAL REPORT TEMPLATE**

**Project Name: Balham College**

|  |  |  |  |
| --- | --- | --- | --- |
| **Project Team Name(s)** | **Student ID** | **Email** | **Phone** |
| ***Project Manager***  *Jason Down* | 1548373 |  |  |

**Client Stakeholders**

|  |  |  |
| --- | --- | --- |
| **Client Stakeholders** | **Full name and title** | **Contact details** |
| **Project Sponsor(s)** | *Lei Song, Simon Dacey* |  |
| ***Balham College*** |  |  |

**DATE OF SUBMISSION**

*dd/mm/yyyy*

Table of Contents

[1. Document Control 4](#_Toc86667014)

[1.1 Version History 4](#_Toc86667015)

[1.2 Contribution to Report sections 4](#_Toc86667016)

[1.3 Glossary 4](#_Toc86667017)

[2. Executive Summary 5](#_Toc86667018)

[3. Introduction 6](#_Toc86667019)

[4. Technology Review 7](#_Toc86667020)

[5. IT Methodology 10](#_Toc86667021)

[6. Project Management 11](#_Toc86667022)

[6.1 Project Management Narrative 11](#_Toc86667023)

[6.2 Project Plan with Milestones 11](#_Toc86667024)

[6.3 Project Governance Responsibilities 11](#_Toc86667025)

[6.4 Project Meetings 11](#_Toc86667026)

[6.5 Project Reports 11](#_Toc86667027)

[6.6 Project Risk and Issue Analysis 11](#_Toc86667028)

[7. Requirements Analysis 12](#_Toc86667029)

[7.1 Introduction 12](#_Toc86667030)

[7.2 Use Case Diagram 12](#_Toc86667031)

[7.3 Business Use Case Narratives (Descriptions) 13](#_Toc86667032)

[7.4 Activity Diagrams 27](#_Toc86667033)

[7.5 Overall Class Diagram 35](#_Toc86667034)

[8. Project Design 36](#_Toc86667035)

[8.1 Introduction 36](#_Toc86667036)

[8.2 Software List 36](#_Toc86667037)

[8.3 Version Control Software 36](#_Toc86667038)

[8.4 Design Use Case Narratives (Descriptions) 37](#_Toc86667039)

[8.5 Sequence Diagrams 50](#_Toc86667040)

[8.6 Deployment Diagram 51](#_Toc86667041)

[8.7 Database Design 52](#_Toc86667042)

[8.8 Annotated User Interface Designs 57](#_Toc86667043)

[8.9 Test Plan 57](#_Toc86667044)

[9. Project Training 58](#_Toc86667045)

[9.1 End User Background and Training Objectives 58](#_Toc86667046)

[9.2 Training Materials 58](#_Toc86667047)

[9.3 Training Deliverables 58](#_Toc86667048)

[9.4 Evaluation 58](#_Toc86667049)

[10. Conclusion & Lessons Learned 59](#_Toc86667050)

[References 60](#_Toc86667051)

[Appendices 62](#_Toc86667052)

# 1. Document Control

## 1.1 Version History

This document has had the following revisions:

| **Version** | **Date** | **Author** | **Description of Change** |
| --- | --- | --- | --- |
| 0.1 | 26/08/2021 | Jason Down | Initial draft |

## 1.2 Contribution to Report sections

| **Project Team Member name** | **Student ID** | **Report Section** |
| --- | --- | --- |
| Jason Down | 1548373 | Everything |

## 1.3 Glossary

To provide clarity, terms and acronyms used in this document are defined as follows:

| **Term / Abbreviation** | **Definition** |
| --- | --- |
| Supervisor | Technical Advisor |
|  |  |

# 2. Executive Summary

# 3. Introduction

Balham College is an educational organization that needs an information system to manage its students, their enrolments, their courses and their programmes. At the current phase of development, only the systems which manage the courses and programmes are required. In this technical report, the planning, initialization, design, development, and launch, along with training and deployment are covered in extensive detail. This report will cover the decision making, and the steps used in the creation of the product. The final product, including all code and testing results, will not be in this document.

# 4. Technology Review

The project that we are tasked with creating will require a user interface and data storage, it will need to be written in a programming language and interact with a database of some kind to manage the courses and programmes of Balham College. The project will be interacted with using a web application (Using HTML, CSS and Bootstrap CSS), meaning it can run on a large selection of devices, and does not require the user to manage or download software. The following section will cover the decision making and reasoning of our chosen language, database, and IDE (more on that later).

The first piece of technology we will choose and the most important, is the programming language. Not only can the programming language we choose determine the difficulty to code the program, but it can also determine the overall functionality of the software, such as what database and IDE we can use. Some of the most popular languages we can choose from include C#, JavaScript, Python, Java, and Go.

C# is an object oriented, class-based programming language, commonly paired with the .NET framework, a set of libraries that can be run on many types of information systems, such as IOT devices, watches, phones and many operating systems. C# and .NET are developed by Microsoft (Microsoft, 2021). C# however is very complicated to use, is rather outdated, and for our application, its speed is not necessary.

JavaScript was “initially developed to be used as a client-side scripting language for dynamic web pages” but is now used on server-side applications using runtimes such as Node.js, (Kanchev, 2021). JavaScript is scalable, fast and popular but is harder to master than other languages. For our use case, we don’t need our program to be scalable, and don’t need the speed.

Python is an object oriented, high-level programming language designed for rapid development and ease of use (Python, 2021). it is very beginner friendly, has an extensive list of libraries from web development (Flask, Django), database interfacing (SQLite, MySQL) to game creation (PyGame, PyKyra) and machine learning (Keras, PyTorch, Tensorflow), (FutureLearn, 2021). Python suffers from performance, but its pros outweigh its cons in our project.

Go is a programming language designed to replace languages like C++ with its speed and simplicity. Go is mostly used in server-side applications (Boyd, 2021); however, it is rather new, and many issues one may have learning it won’t have been solved, resulting in longer development times. The languages ties to C++ and Java also make it much more complicated, meaning it would be far more difficult to master.

The 2 languages that seem the most relevant for the type of application we are making (a web application) are JavaScript (using some sort of server-side engine) and Python (using an http server library like Flask). Due to the simplicity of Python, and its wide application, we will choose it for our project, along with the Flask library for web development.

The IDE or Integrated Development Environment is the next utility we must choose for our project. An IDE is essentially a text editor, though designed for developers to write better code faster, with all the tools they need (source code editor, build automation, debugging) in one place. Some integrated development environments are designed to support many different languages, while others are tailored to individual languages (RedHat, 2019).

For our selected programming language (Python) there is a handful of integrated development environments to choose from: PyCharm, Visual Studio, Sublime, or the Python IDE. PyCharm is an IDE developed by JetBrains. IntelliJ and many other IDEs are created by JetBrains; PyCharm is a Python IDE tailored to the language specifically, it has code completion, error highlighting and many other features such as integration with popular frameworks (Flask, Django) (JetBrains, 2021). PyCharm would be ideal as it is tailored for Python, and its error highlighting and code completion will greatly assist in development speed.

Visual Studio is another IDE, popular for its .NET forum creator and its vast support for many languages such as C++, C#, JavaScript and many others. Visual Studio has very powerful code completion, though its interface is very complicated, and its capabilities far exceed our requirements, making it very sluggish in terms of performance (Microsoft, 2021). These cons make Visual Studio very hard to master.

Sublime text is more of a text editor with syntax highlighting rather than an IDE. It features code completion, albeit only containing previously declared variables and functions. It features very primitive error highlighting, and while it has some integrations with Python (console output, build shortcuts) it is not ideal for a large project such as a web application. (Tutorials Point, 2021)

The Python IDE otherwise known as ‘IDLE’ is a Python specific integrated development environment that comes with the default installation of Python. It features the same syntax highlighting of any IDE, console output and error messages and is entirely written in Python, using the Tkinter library (Python, 2021). The IDE has a very outdated appearance, no code completion and weak error highlighting. For the requirements of our IT project, PyCharm IDE is a good fit for our requirements. It’s simple, easy to learn interface, in conjunction with its efficient code completion and error highlighting make it perfect for our project.

Databases have many implementations. Databases supported by Python that suite our application consist of SQL implementations. SQL or ‘Structured Query Language’ is a standard for relational databases. Microsoft Access is a local database management system that only functions on Windows, though since we are using a web application this technically isn’t important. The databases graphical user interface (GUI) makes it very easy to create tables and their relations, however Access is rather outdated, and its limitation to Windows operating systems, as well as it being part of the Office suite of applications makes it costly and hard to deploy in the future (Tsai, D, 2021).

SQLite is a small and reliable local database engine already included in most devices such as mobile phones, it is bundled in the default installation of Python, and its very easy to learn (SQLite, 2021). SQLite is not proficient in handling multiple users, meaning it is not suitable for large websites, however in our instance of a web application used to manage a school, it is highly unlikely that our application will fall into the hands of a wider audience.

MySQL is a widely popular SQL implementation that is multithreaded compared to SQLite, it also runs on its own external server. This approach means that MySQL can be used for many users, which is not necessary for our product, and its external location makes it safer as the project is distributed (Talend, 2021). Due to Python already having SQLite built in, its simplicity, wide use and intuitive learning, it is a good choice for our IT project. Having our web application server, and database be on the same server also makes deployment much easier because we can distribute and run the server as one contained package.

A similar application to our project that we can gather positive inspiration from, and negative traits to avoid is Moodle. Moodle is an open-source education-based learning management system designed for educators to “deliver learning materials online” (Titus, 2021). Despite our application being a far simpler form based database editor; its place in an educational environment, and functioning as a server based web application makes it relevant to Moodle. According to (Budram, 2021), the application excels in being a “central repository for course material”, while our application is about tying objects together rather than providing materials for students, we can let this review influence our application by keeping everything in one place. (Budram, 2021) also states that Moodle is complicated for non tech users, resulting in many issues often requiring external support for resolution. For our application, having everything in one place can make it more convenient, however we must also reduce making the interface over complicated and tough to learn.

# 5. IT Methodology

# 6. Project Management

## 6.1 Project Management Narrative

*Details with evidence how the development of the project followed the selected systems development lifecycle*

## 6.2 Project Plan with Milestones

*Include an overall plan here and attach a detailed GANTT chart to the appendices*

## 6.3 Project Governance Responsibilities

*Explain who was responsible for project management and quality assurance, and explain how these tasks were carried out*

## 6.4 Project Meetings

*Include a schedule of your meetings (date, duration, participants, and type) and attach the minutes of each meeting to the appendices*

## 6.5 Project Reports

*Discuss the project status reports and attach your project status reports to the appendices*

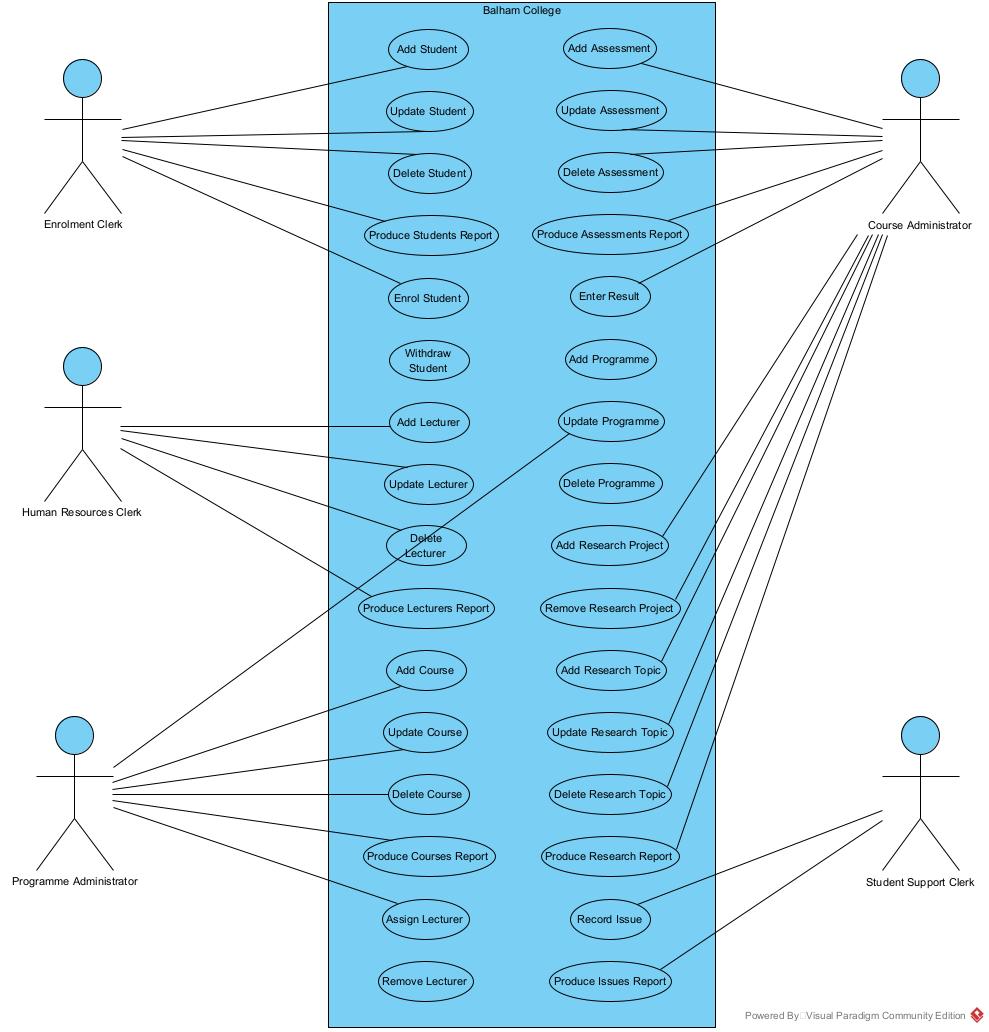
## 6.6 Project Risk and Issue Analysis

*Discuss project risks and issues and attach your project risk and issue register to the appendices*

# 7. Requirements Analysis

## 7.1 Introduction

## 7.2 Use Case Diagram



## 7.3 Business Use Case Narratives (Descriptions)

|  |  |  |
| --- | --- | --- |
| **USE CASE NAME:** | Add Course | **USE CASE TYPE** |
| **USE CASE ID:** | 11 | **Business Requirements: 🗹** |
| **PRIORITY:** | High |  |
| **PRIMARY BUSINESS ACTOR:** | Programme Administrator | |
| **OTHER PARTICIPATING ACTORS:** | None | |
| **DESCRIPTION:** | This use case enables the programme administrator to add a course’s details. | |
| **PRE-CONDITIONS:** | The programme administrator has logged onto the system | |
| **TYPICAL COURSE** | 1. The programme administrator selects the “Add Course” function.  2. The system displays the “Add Course” form with all fields blank.  3. The programme administrator enters the course’s details, (course name, credits fee and status).  4. The system confirms that the details are entered in correctly.  5. The system displays a list of programmes (programme name and level).  6. The programme administrator selects a programme.  7. The programme administrator selects “Add Course”.  8. The system saves the course’s details (course id, course name, credits, status, and programme).  9. The system displays the “Course added successfully” message.  10. The system displays the “Add another course?” prompt.  11. The programme administrator elects to end the use case.  12. The system closes the form. | |
| **OF EVENTS:** |
|  |
|  |
|  |
|  |
| **ALTERNATE COURSES:** | 5a1. The system identifies incorrect fields have been entered.  5a2. The system prompts the programme administrator with an error.  5a3. The system returns to step 2. | |
| 7a1. The programme administrator cancels the operation.  7a2. The system closes the forum. | |
|  | 11a1. The programme administrator elects to add another course.  11a2. The system goes to step 2. | |
| **POST CONDITIONS:** | None | |
| **ASSUMPTIONS:** | None | |

|  |  |  |
| --- | --- | --- |
| **USE CASE NAME:** | Update Course | **USE CASE TYPE** |
| **USE CASE ID:** | 12 | **Business Requirements: 🗹** |
| **PRIORITY:** | High |  |
| **PRIMARY BUSINESS ACTOR:** | Programme Administrator | |
| **OTHER PARTICIPATING ACTORS:** | None | |
| **DESCRIPTION:** | This use case enables the programme administrator to update a course’s details. | |
| **PRE-CONDITIONS:** | The programme administrator has logged onto the system. | |
| **TYPICAL COURSE** | 1. The programme administrator selects the “Update Course” function.  2. The system displays the “Update Course” form and displays a list of all the courses (course id and course name).  3. The programme administrator selects the course that has details that needs updating.  4. The system displays the course’s details (course id, course name, credits, status, fee, and programme name).  5. The programme administrator updates the relevant details (name, credits, fee, and status only).  6. The system validates the entries in the fields and prompts for confirmation to change the courses details.  7. The programme administrator confirms the change of details.  8. The system saves the course’s details.  9. The system displays the “Course updated successfully” message.  10. The system displays the “Update another course?” prompt.  11. The programme administrator elects to end the use case.  12. The system closes the form. | |
| **OF EVENTS:** |
|  |
|  |
|  |
|  |
| **ALTERNATE COURSES:** | 3a1. The programme administrator elects to cancel the operation.  3a2. The system closes the form. | |
| 6a1. The system identifies missing or incorrect fields.  6a2. The system prompts for completion of the entry  6a3. The system returns to step 4. | |
|  | 7a1. The programme administrator cancels the changes.  7a2. The system closes the form. | |
|  | 11a1. The programme administrator elects to update another course.  11a2. The system returns to step 2. | |
| **POST CONDITIONS:** | None | |
| **ASSUMPTIONS:** | None | |

|  |  |  |
| --- | --- | --- |
| **USE CASE NAME:** | Delete Course | **USE CASE TYPE** |
| **USE CASE ID:** | 13 | **Business Requirements: 🗹** |
| **PRIORITY:** | High |  |
| **PRIMARY BUSINESS ACTOR:** | Programme Administrator | |
| **OTHER PARTICIPATING ACTORS:** | None | |
| **DESCRIPTION:** | This use case enables the programme administrator to delete a course’s details. | |
| **PRE-CONDITIONS:** | The programme administrator has logged onto the system | |
| **TYPICAL COURSE** | 1. The programme administrator selects the “Delete Course” function.  2. The system displays the “Delete Course” form and displays a list of all the courses (course id and course name) that have no lecturers, assessments, or enrolments assigned to them.  3. The programme administrator selects the course that requires deleting.  4. The system displays the course’s details (course id, course name, credits, and status).  5. The programme administrator selects “Delete Course”  6. The system deletes the course.  7. The system displays the “Course deleted successfully” message.  8. The system displays the “Delete another course?” prompt.  9. The programme administrator elects to delete another course.  10. The system returns to step 2. | |
| **OF EVENTS:** |
|  |
|  |
|  |
|  |
| **ALTERNATE COURSES:** | 3a1. The programme administrator elects to cancel the operation  3a2. The system closes the form. | |
| 5a1. The programme administrator elects to cancel the operation.  5a2. The system closes the form. | |
|  | 9a1. The programme administrator elects to end the use case.  9a2. The system closes the form. | |
| **POST CONDITIONS:** | None | |
| **ASSUMPTIONS:** | None | |

|  |  |  |
| --- | --- | --- |
| **USE CASE NAME:** | Produce Courses Report | **USE CASE TYPE** |
| **USE CASE ID:** | 14 | **Business Requirements: 🗹** |
| **PRIORITY:** | High |  |
| **PRIMARY BUSINESS ACTOR:** | Programme Administrator | |
| **OTHER PARTICIPATING ACTORS:** | None | |
| **DESCRIPTION:** | This use case enables the programme administrator to produce the courses report. | |
| **PRE-CONDITIONS:** | The programme administrator has logged onto the system | |
| **TYPICAL COURSE** | 1. The programme administrator selects the “Courses Report” function.  2. The system displays the “Courses Report” form.  3. The programme administrator selects the option to generate the report.  4. The system gets the details (course ID, course name, credits, status, fee, and programme name) of each course.  5. The system counts the number of enrolments for each course.  6. The system counts the number of assessments for each course.  7. The system then generates the courses report (course ID, course name, credits, status, and programme name) with the course sorted by name.  8. The system concludes the use case by closing the form. | |
| **OF EVENTS:** |
|  |
|  |
|  |
|  |
| **ALTERNATE COURSES:** | 3a1. The programme administrator elects to cancel the operation.  3a2. The system closes the form. | |
| **POST CONDITIONS:** | None | |
| **ASSUMPTIONS:** | None | |

|  |  |  |
| --- | --- | --- |
| **USE CASE NAME:** | Enter Result | **USE CASE TYPE** |
| **USE CASE ID:** | 21 | **Business Requirements: 🗹** |
| **PRIORITY:** | High |  |
| **PRIMARY BUSINESS ACTOR:** | Course Administrator | |
| **OTHER PARTICIPATING ACTORS:** | None | |
| **DESCRIPTION:** | This use case enables the course administrator to enter a result for a student against an assessment. | |
| **PRE-CONDITIONS:** | The course administrator has logged onto the system. | |
| **TYPICAL COURSE** | 1. The course administrator selects the “Enter Result” function.  2. The system displays the “Enter Result” form and displays a list of all assessments (assessment id, assessment number, assessment name and course name).  3. The course administrator selects the assessment that is to be assigned.  4. The system displays the assessment’s details (assessment number, assessment name, maximum mark, and course name) and then displays a list of all enrolments (student name, year, semester, and status) for the course of the selected assessment.  5. The course administrator selects an enrolment and enters a mark and a result date.  6. The system checks if a result already belongs to the assessment.  7. The system saves the result (assessment, enrolment, result date, and mark).  8. The system displays the “Result entered successfully” message.  9. The system displays the “Enter another result?” prompt.  10. The course administrator elects to end the use case.  11. The system closes the form. | |
| **OF EVENTS:** |
|  |
|  |
|  |
|  |
| **ALTERNATE COURSES:** | 3a1. The course administrator elects to cancel the operation  3a2. The system closes the form. | |
|  | 5a1. The course administrator elects to cancel the operation.  5a2. The system closes the form. | |
|  | 6a1. Upon a result already belonging to the assessment and enrolment, an error message is displayed.  6a2. The system returns to step 4. | |
|  | 10a1. The course administrator elects to enter another result.  11a2. The system returns to step 2. | |
| **POST CONDITIONS:** | None | |
| **ASSUMPTIONS:** | None | |

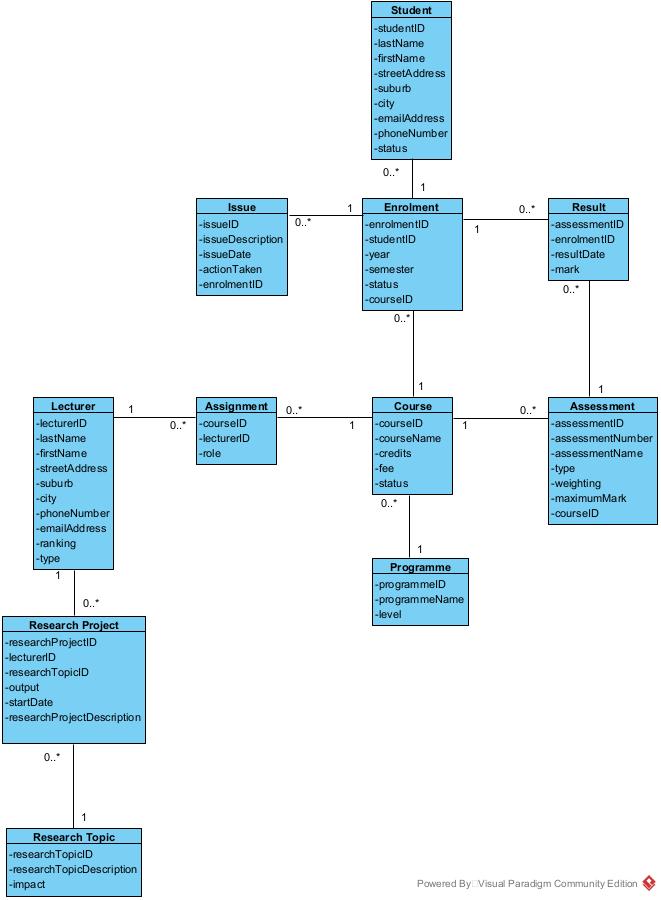
|  |  |  |
| --- | --- | --- |
| **USE CASE NAME:** | Add Programme | **USE CASE TYPE** |
| **USE CASE ID:** | 22 | **Business Requirements: 🗹** |
| **PRIORITY:** | High |  |
| **PRIMARY BUSINESS ACTOR:** | Programme Administrator | |
| **OTHER PARTICIPATING ACTORS:** | None | |
| **DESCRIPTION:** | This use case enables the programme administrator to add a programme’s details. | |
| **PRE-CONDITIONS:** | The programme administrator has logged onto the system. | |
| **TYPICAL COURSE** | 1 .The programme administrator selects the “Add Programme” function.  2. The system displays the “Add Programme” form with all fields blank.  3. The programme administrator enters the programme’s details (programme id, programme name and level.  4. The system confirms that the details are filled in correctly.  5. The programme administrator elects to add the programme.  6. The system saves the programme’s details (programme id, programme name, and level).  7. The system displays the “Programme added successfully” message.  8. The system displays the “Add another programme?” prompt.  9. The programme administrator elects to end the use case  10. The system closes the form. | |
| **OF EVENTS:** |
|  |
|  |
|  |
|  |
| **ALTERNATE COURSES:** | 5a1. The system detects incorrect fields  5a2. The system displays an error message  5a3. The system returns to step 2.  5b1. The course administrator elects to cancel the operation.  5b2. The system closes the form. | |
|  | 9a1. The programme administrator elects to add another programme.  9a2. The system returns to step 2. | |
| **POST CONDITIONS:** | None | |
| **ASSUMPTIONS:** | None | |

|  |  |  |
| --- | --- | --- |
| **USE CASE NAME:** | Update Programme | **USE CASE TYPE** |
| **USE CASE ID:** | 23 | **Business Requirements: 🗹** |
| **PRIORITY:** | High |  |
| **PRIMARY BUSINESS ACTOR:** | Programme Administrator | |
| **OTHER PARTICIPATING ACTORS:** | None | |
| **DESCRIPTION:** | This use case enables the programme administrator to update a programme’s details. | |
| **PRE-CONDITIONS:** | The programme administrator has logged onto the system. | |
| **TYPICAL COURSE** | 1. The programme administrator selects the “Update Programme” function.  2. The system displays the “Update Programme” form and displays a list of all the programmes (programme id and programme name).  3. The programme administrator selects the programme that has details that needs updating.  4. The system displays the programme’s details (programme id, programme name, and level).  5. The programme administrator updates the relevant details (programme name and level only).  6. The system validates the entries in the fields and prompts for confirmation to change the programmes details.  7. The programme administrator confirms the change of details.  8. The system saves the programme’s details.  9. The system displays the “Programme updated successfully” message.  10. The system displays the “Update another programme?” prompt.  11. The programme administrator elects to end the use case.  12. The system closes the form. | |
| **OF EVENTS:** |
|  |
|  |
|  |
|  |
| **ALTERNATE COURSES:** | 3a1. The programme administrator elects to cancel the operation.  3a2. The system closes the form. | |
|  | 7a1. The system identifies missing or incorrect fields.  7a2. The system prompts for completion of entry.  7a2. The system returns to step 4.  7b1. The programme administrator cancels the changes.  7b2. The system closes the form. | |
|  | 11a1. The programme administrator elects to update another programme.  11a2. The system returns to step 2. | |
| **POST CONDITIONS:** | None | |
| **ASSUMPTIONS:** | None | |

|  |  |  |
| --- | --- | --- |
| **USE CASE NAME:** | Delete Programme | **USE CASE TYPE** |
| **USE CASE ID:** | 24 | **Business Requirements: 🗹** |
| **PRIORITY:** | High |  |
| **PRIMARY BUSINESS ACTOR:** | Programme Administrator | |
| **OTHER PARTICIPATING ACTORS:** | None | |
| **DESCRIPTION:** | This use case enables the programme administrator to delete a programme’s details. | |
| **PRE-CONDITIONS:** | The programme administrator has logged onto the system. | |
| **TYPICAL COURSE** | 1. The programme administrator selects the “Delete Programme” function.  2. The system displays the “Delete Programme” form and displays a list of all the programmes (programme id and programme name) that have no courses assigned to them.  3. The programme administrator selects the programme that requires deleting.  4. The system displays the programme’s details (programme id, programme name, and level).  5. The programme administrator elects to delete the programme .  6. The system deletes the programme.  7. The system displays the “Programme deleted successfully” message.  8. The system displays the “Delete another programme?” prompt.  9. The programme administrator elects to end the use case.  10. The system closes the form. | |
| **OF EVENTS:** |
|  |
|  |
|  |
|  |
| **ALTERNATE COURSES:** | 3a1. The programme administrator elects to cancel the operation.  3a2. The system closes the form. | |
|  | 5a1. The programme administrator elects to cancel the operation.  5a2. The system closes the form. | |
|  | 9a1. The programme administrator elects to delete another programme.  9a2. The system returns to step 2. | |
| **POST CONDITIONS:** | None | |
| **ASSUMPTIONS:** | None | |

## 7.4 Activity Diagrams

## 7.5 Overall Class Diagram



# 8. Project Design

## 8.1 Introduction

## 8.2 Software List

## 8.3 Version Control Software

## 8.4 Design Use Case Narratives (Descriptions)

|  |  |  |
| --- | --- | --- |
| **USE CASE NAME:** | Add Course | **USE CASE TYPE** |
| **USE CASE ID:** | 11 | **Design Requirements: 🗹** |
| **PRIORITY:** | High |  |
| **PRIMARY BUSINESS ACTOR:** | Programme Administrator | |
| **OTHER PARTICIPATING ACTORS:** | None | |
| **DESCRIPTION:** | This use case enables the programme administrator to add a course’s details. | |
| **PRE-CONDITIONS:** | The programme administrator has logged onto the system | |
| **TYPICAL COURSE** | 1. The programme administrator selects the “Add Course” function.  2. The system displays the “Add Course” form with all fields blank.  3. The programme administrator enters the course’s details, course name, credits, fee, and status (current or suspended).  4. The system validates that all the fields are entered in correctly.  5. The system gets all the programmes.  6. The system displays a list of programmes (programme name and level) in a list box.  7. The programme administrator clicks on a programme.  8. The programme administrator clicks the “Add Course” button.  9. The system generates a course id.  10. The system saves the course’s details (course id, course name, credits, status, fee, and programme id).  11. The system displays the “Course added successfully” message.  12. The system displays the “Add another course?” prompt in a Yes No box.  13. The programme administrator clicks “No”.  14. The system closes the form. | |
| **OF EVENTS:** |
|  |
|  |
|  |
|  |
| **ALTERNATE COURSES:** | 5a1. The system identifies incorrect fields have been entered.  5a2. The system prompts the programme administrator with an error.  5a3. The system returns to step 2. | |
| 7a1. The programme clicks “Cancel”.  7a2. The system closes the forum. | |
|  | 13a1. The programme administrator clicks “Yes”.  13a2. The system goes to step 2. | |
| **POST CONDITIONS:** | None | |
| **ASSUMPTIONS:** | None | |

|  |  |  |
| --- | --- | --- |
| **USE CASE NAME:** | Update Course | **USE CASE TYPE** |
| **USE CASE ID:** | 12 | **Design Requirements: 🗹** |
| **PRIORITY:** | High |  |
| **PRIMARY BUSINESS ACTOR:** | Programme Administrator | |
| **OTHER PARTICIPATING ACTORS:** | None | |
| **DESCRIPTION:** | This use case enables the programme administrator to update a course’s details. | |
| **PRE-CONDITIONS:** | The programme administrator has logged onto the system. | |
| **TYPICAL COURSE** | 1. The programme administrator selects the “Update Course” function.  2. The system displays the “Update Course” form.  3. The system gets all the courses.  4. The system displays a list of all the courses (course id and course name) in a list box.  5. The programme administrator clicks on the course that has details that needs updating.  6. The system displays the course’s details (course id, course name, credits, status, fee, and programme name).  7. The programme administrator updates the relevant details (name, credits, fee, and status only).  8. The system validates the entries in the fields.  9. The system prompts for confirmation to change the courses details in a Yes No box.  10. The programme administrator clicks “Yes”.  11. The system saves the course’s details.  12. The system displays the “Course updated successfully” message.  13. The system displays the “Update another course?” prompt in a Yes No box.  14. The programme administrator clicks “No”.  15. The system closes the form. | |
| **OF EVENTS:** |
|  |
|  |
|  |
|  |
| **ALTERNATE COURSES:** | 4a1. The programme administrator clicks “Cancel”.  4a2. The system closes the form. | |
| 9a1. The system identifies missing or incorrect fields.  9a2. The system prompts for completion of the entry in a message box.  9a3. The system returns to step 4.  9b1. The programme administrator clicks “No”.  9b2. The system closes the form. | |
|  | 13a1. The programme administrator clicks “Yes”.  13a2. The system returns to step 2. | |
| **POST CONDITIONS:** | None | |
| **ASSUMPTIONS:** | None | |

|  |  |  |
| --- | --- | --- |
| **USE CASE NAME:** | Delete Course | **USE CASE TYPE** |
| **USE CASE ID:** | 13 | **Design Requirements: 🗹** |
| **PRIORITY:** | High |  |
| **PRIMARY BUSINESS ACTOR:** | Programme Administrator | |
| **OTHER PARTICIPATING ACTORS:** | None | |
| **DESCRIPTION:** | This use case enables the programme administrator to delete a course’s details. | |
| **PRE-CONDITIONS:** | The programme administrator has logged onto the system | |
| **TYPICAL COURSE** | 1. The programme administrator selects the “Delete Course” function.  2. The system displays the “Delete Course” form.  3. The system gets all the courses.  4. The system displays a list of all the courses (course id and course name) that have no lecturers, assessments, or enrolments assigned to them in a list box.  5. The programme administrator clicks the course that requires deleting.  6. The system displays the course’s details (course id, course name, credits, and status).  7. The programme administrator clicks “Delete Course”.  8. The system deletes the course.  9. The system displays the “Course deleted successfully” message.  10. The system displays the “Delete another course?” prompt in a Yes No box.  11. The programme administrator clicks “No”.  12. The system closes the form. | |
| **OF EVENTS:** |
|  |
|  |
|  |
|  |
| **ALTERNATE COURSES:** | 5a1. The programme administrator clicks “Cancel”.  5a2. The system closes the form. | |
| 7a1. The programme administrator clicks “Cancel”.  7a2. The system closes the form. | |
|  | 11a1. The programme administrator clicks “Yes”.  11a2. The system returns to step 2. | |
| **POST CONDITIONS:** | None | |
| **ASSUMPTIONS:** | None | |

|  |  |  |
| --- | --- | --- |
| **USE CASE NAME:** | Produce Courses Report | **USE CASE TYPE** |
| **USE CASE ID:** | 14 | **Design Requirements: 🗹** |
| **PRIORITY:** | High |  |
| **PRIMARY BUSINESS ACTOR:** | Programme Administrator | |
| **OTHER PARTICIPATING ACTORS:** | None | |
| **DESCRIPTION:** | This use case enables the programme administrator to produce the courses report. | |
| **PRE-CONDITIONS:** | The programme administrator has logged onto the system | |
| **TYPICAL COURSE** | 1. The programme administrator selects the “Courses Report” function.  2. The system displays the “Courses Report” form.  3. The programme administrator clicks “Generate Report”.  4. The system gets the details (course ID, course name, credits, status, fee, and programme name) of each course.  5. The system counts the number of enrolments for each course.  6. The system counts the number of assessments for each course.  7. The system then generates the courses report (course ID, course name, credits, status, and programme name) with the course sorted by name.  8. The system displays the courses report.  9. The programme administrator clicks “Close”.  10. The system closes the forum and ends the use case. | |
| **OF EVENTS:** |
|  |
|  |
|  |
|  |
| **ALTERNATE COURSES:** | 3a1. The programme administrator clicks “Cancel”.  3a2. The system closes the form. | |
| **POST CONDITIONS:** | None | |
| **ASSUMPTIONS:** | None | |

|  |  |  |
| --- | --- | --- |
| **USE CASE NAME:** | Enter Result | **USE CASE TYPE** |
| **USE CASE ID:** | 21 | **Design Requirements: 🗹** |
| **PRIORITY:** | High |  |
| **PRIMARY BUSINESS ACTOR:** | Course Administrator | |
| **OTHER PARTICIPATING ACTORS:** | None | |
| **DESCRIPTION:** | This use case enables the course administrator to enter a result for a student against an assessment. | |
| **PRE-CONDITIONS:** | The course administrator has logged onto the system. | |
| **TYPICAL COURSE** | 1. The course administrator selects the “Enter Result” function.  2. The system displays the “Enter Result” form  3. The system gets all the assessments.  4. The system displays a list of all assessments (assessment id, assessment number, assessment name and course name).  5. The course administrator selects the assessment that is to be assigned.  6. The system displays the assessment’s details (assessment number, assessment name, maximum mark, and course name).  7. The system gets the assessments course.  8. The system gets all enrolments for the course.  9. The system displays a list of all the enrolments (student name, year, semester, and status).  10. The course administrator clicks an enrolment.  11. The course administrator enters a mark and a result date.  12. The system checks if a result already belongs to the assessment.  13. The system saves the result (assessment, enrolment, result date, and mark).  14. The system displays the “Result entered successfully” message box.  15. The system displays the “Enter another result?” prompt in a Yes No box.  16. The course administrator clicks “No”.  17. The system closes the form. | |
| **OF EVENTS:** |
|  |
|  |
|  |
|  |
| **ALTERNATE COURSES:** | 5a1. The course administrator clicks “Cancel”.  5a2. The system closes the form. | |
|  | 9a1. The course administrator clicks “Cancel”.  9a2. The system closes the form. | |
|  | 12a1. Upon a result already belonging to the assessment and enrolment, an error message is displayed.  12a2. The system returns to step 4. | |
|  | 16a1. The course administrator clicks “Yes”.  16a2. The system returns to step 2. | |
| **POST CONDITIONS:** | None | |
| **ASSUMPTIONS:** | None | |

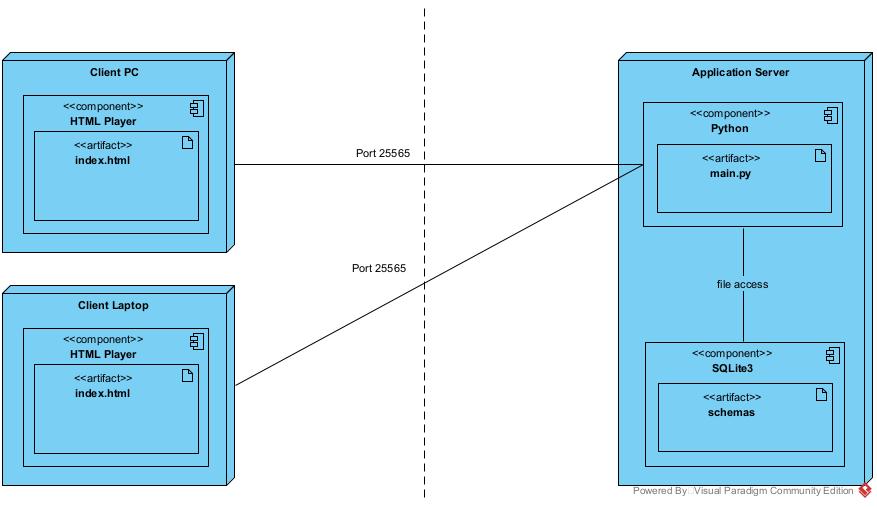
|  |  |  |
| --- | --- | --- |
| **USE CASE NAME:** | Add Programme | **USE CASE TYPE** |
| **USE CASE ID:** | 22 | **Design Requirements: 🗹** |
| **PRIORITY:** | High |  |
| **PRIMARY BUSINESS ACTOR:** | Programme Administrator | |
| **OTHER PARTICIPATING ACTORS:** | None | |
| **DESCRIPTION:** | This use case enables the programme administrator to add a programme’s details. | |
| **PRE-CONDITIONS:** | The programme administrator has logged onto the system. | |
| **TYPICAL COURSE** | 1 .The programme administrator selects the “Add Programme” function.  2. The system displays the “Add Programme” form with all fields blank.  3. The programme administrator enters the programme’s details (programme id, programme name and level.  4. The system confirms that the details are filled in correctly.  5. The programme administrator clicks “Add Programme”.  6. The system saves the programme’s details (programme id, programme name, and level).  7. The system displays the “Programme added successfully” message box.  8. The system displays the “Add another programme?” prompt in a Yes No box.  9. The programme administrator clicks “No”.  10. The system closes the form. | |
| **OF EVENTS:** |
|  |
|  |
|  |
|  |
| **ALTERNATE COURSES:** | 5a1. The system detects incorrect fields.  5a2. The system displays an error message box.  5a3. The system returns to step 2.  5b1. The course administrator clicks “Cancel”.  5b2. The system closes the form. | |
|  | 9a1. The programme administrator clicks “Yes”.  9a2. The system returns to step 2. | |
| **POST CONDITIONS:** | None | |
| **ASSUMPTIONS:** | None | |

|  |  |  |
| --- | --- | --- |
| **USE CASE NAME:** | Update Programme | **USE CASE TYPE** |
| **USE CASE ID:** | 23 | **Design Requirements: 🗹** |
| **PRIORITY:** | High |  |
| **PRIMARY BUSINESS ACTOR:** | Programme Administrator | |
| **OTHER PARTICIPATING ACTORS:** | None | |
| **DESCRIPTION:** | This use case enables the programme administrator to update a programme’s details. | |
| **PRE-CONDITIONS:** | The programme administrator has logged onto the system. | |
| **TYPICAL COURSE** | 1. The programme administrator selects the “Update Programme” function.  2. The system displays the “Update Programme” form.  3. The system gets all programmes.  4. The system displays a list of all the programmes (programme id and programme name).  5. The programme administrator clicks the programme that has details that needs updating.  6. The system gets the selected programme’s details (programme id, programme name and level).  7. The system displays the programme’s details.  8. The programme administrator updates the relevant details (programme name and level only).  9. The system validates the entries in the fields.  10. The system prompts for confirmation to change the programmes details in a Yes No box.  11. The programme administrator confirms the change of details.  12. The system saves the programme’s details.  13. The system displays the “Programme updated successfully” message box.  14. The system displays the “Update another programme?” prompt in a Yes No box.  15. The programme administrator clicks “No”.  16. The system closes the form. | |
| **OF EVENTS:** |
|  |
|  |
|  |
|  |
| **ALTERNATE COURSES:** | 5a1. The programme administrator clicks “Cancel”.  5a2. The system closes the form. | |
|  | 10a1. The system identifies missing or incorrect fields.  10a2. The system prompts for completion of entry in a message box.  10a3. The system returns to step 6. | |
|  | 11a1. The programme administrator clicks “No”.  11a2. The system closes the form. | |
|  | 13a1. The programme administrator clicks “Yes”.  13a2. The system returns to step 2. | |
| **POST CONDITIONS:** | None | |
| **ASSUMPTIONS:** | None | |

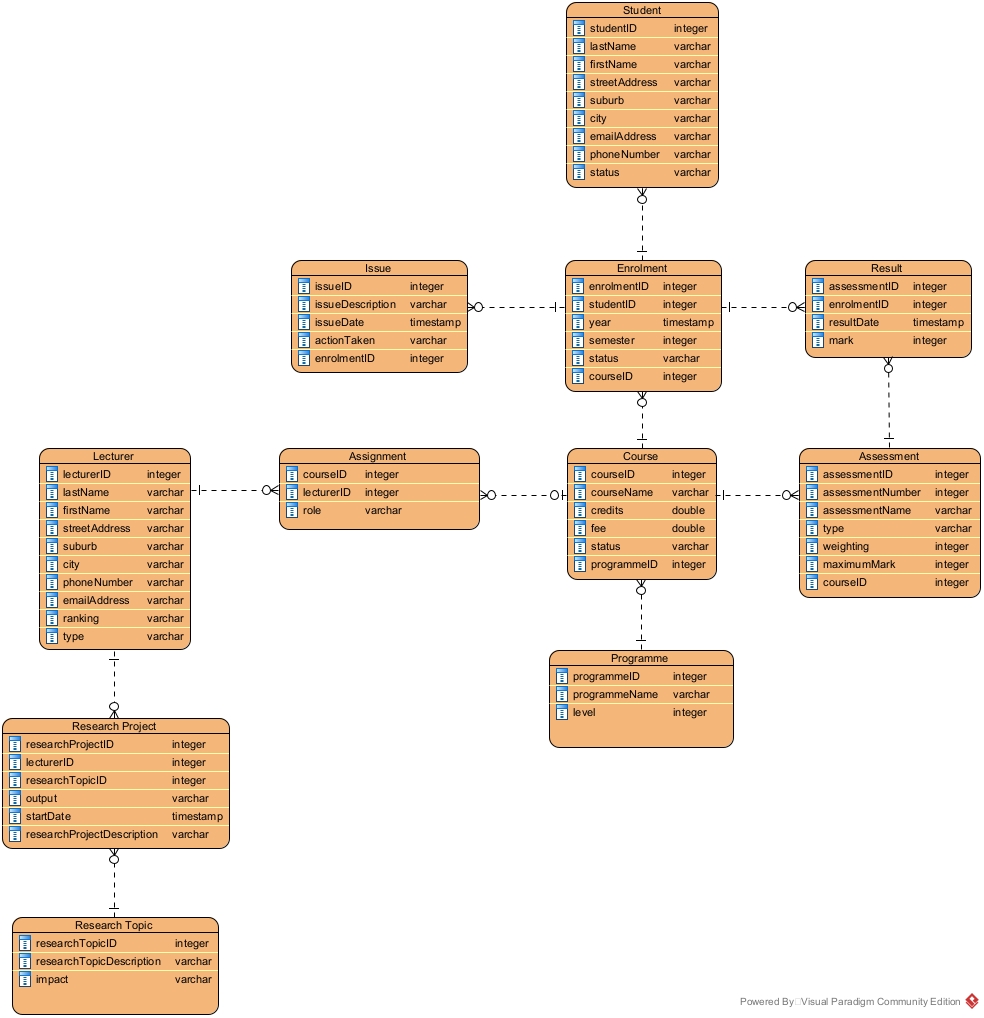
|  |  |  |
| --- | --- | --- |
| **USE CASE NAME:** | Delete Programme | **USE CASE TYPE** |
| **USE CASE ID:** | 24 | **Design Requirements: 🗹** |
| **PRIORITY:** | High |  |
| **PRIMARY BUSINESS ACTOR:** | Programme Administrator | |
| **OTHER PARTICIPATING ACTORS:** | None | |
| **DESCRIPTION:** | This use case enables the programme administrator to delete a programme’s details. | |
| **PRE-CONDITIONS:** | The programme administrator has logged onto the system. | |
| **TYPICAL COURSE** | 1. The programme administrator selects the “Delete Programme” function.  2. The system displays the “Delete Programme” form.  3. The system gets all the programmes that have no courses assigned to them.  4. The system displays a list of all the programmes (programme id and programme name).  5. The programme administrator clicks on the programme that requires deleting.  6. The system gets the programme’s details (programme id, programme name and level).  7. The system displays the programme’s details.  8. The programme administrator clicks “Delete Programme”.  9. The system deletes the programme.  10. The system displays the “Programme deleted successfully” message box.  11. The system displays the “Delete another programme?” prompt in a Yes No box.  12. The programme administrator clicks “No”.  13. The system closes the form. | |
| **OF EVENTS:** |
|  |
|  |
|  |
|  |
| **ALTERNATE COURSES:** | 5a1. The programme administrator clicks “Cancel”.  5a2. The system closes the form. | |
|  | 8a1. The programme administrator clicks “Cancel”.  8a2. The system closes the form. | |
|  | 12a1. The programme administrator clicks “Yes”.  12a2. The system returns to step 2. | |
| **POST CONDITIONS:** | None | |
| **ASSUMPTIONS:** | None | |

## 8.5 Sequence Diagrams

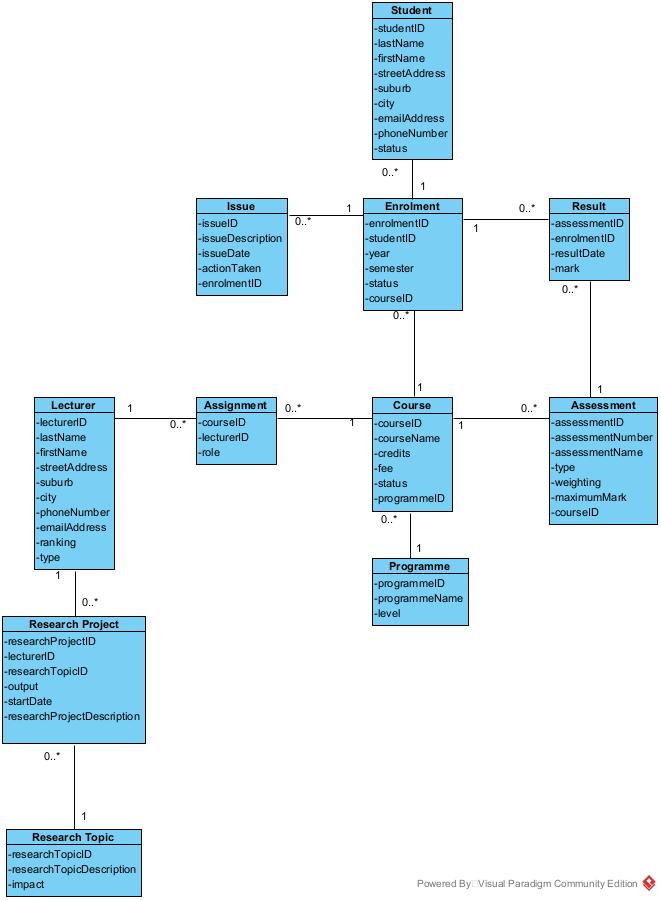
## 8.6 Deployment Diagram



## 8.7 Database Design

8.7.1 Logical Entity Relationship Diagram

### 8.7.2 Class Diagram

****

### 8.7.3 Data Dictionary

**COURSE**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Required** | **Data Type** | **Maximum Length** | **Range/List** |
| CourseID | Yes | Auto-number | 4 | Primary Key |
| CourseName | Yes | Text | 32 | - |
| Credits | Yes | Integer | 3 | 5 to 120 Inclusive |
| Fee | Yes | Currency | 4(2) | 200.00 to 8000.00 inclusive |
| Status | Yes | Text | 9 | current, suspended |
| ProgrammeID | Yes | Foreign Key (PROGRAMME) | 2 | - |

**PROGRAMME**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Required** | **Data Type** | **Maximum Length** | **Range/List** |
| ProgrammeID | Yes | Auto-number | 2 | Primary Key |
| ProgrammeName | Yes | Text | 32 | - |
| Level | Yes | Integer | 2 | 1 to 10 Inclusive |

**ASSESSMENT**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Required** | **Data Type** | **Maximum Length** | **Range/List** |
| AssessmentID | Yes | Auto-number | 3 | Primary Key |
| AssessmentNumber | Yes | Integer | 1 | 1 to 5 Inclusive |
| AssessmentName | Yes | Text | 32 | - |
| Type | Yes | Text | 14 | Assignment, Written Test, Practical Test, or Final Exam |
| Weighting | Yes | Integer | 3 | 10 to 100 Inclusive |
| MaximumMark | Yes | Integer | 3 | 50 to 200 Inclusive |
| CourseID | Yes | Foreign Key (COURSE) | 4 | - |

**RESULT**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Required** | **Data Type** | **Maximum Length** | **Range/List** |
| AssessmentID | Yes | Foreign Key (ASSESSMENT) | 3 | Primary Key |
| EnrolmentID | Yes | Foreign Key (ENROLMENT) | 12 |
| ResultDate | Yes | Date | 10 | Format: DD/MM/YYYY |
| Mark | Yes | Integer | 3 | 0 to 200 Inclusive |

**ENROLMENT**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Required** | **Data Type** | **Maximum Length** | **Range/List** |
| EnrolmentID | Yes | Auto-number | 12 | Primary Key |
| StudentID | Yes | Foreign Key (STUDENT) | 8 | - |
| Year | Yes | Date | 10 | DD/MM/YYYY, 2020 to 2050 Inclusive |
| Semester | Yes | Integer | 1 | 1 to 3 Inclusive |
| Status | Yes | Text | 8 | Pending, Paid or complete |
| CourseID | Yes | Foreign Key (COURSE) | 4 | - |

**STUDENT**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Required** | **Data Type** | **Maximum Length** | **Range/List** |
| StudentID | Yes | Auto-number | 8 | Primary Key |
| LastName | Yes | Text | 16 | - |
| FirstName | Yes | Text | 16 | - |
| StreetAddress | Yes | Text | 32 | - |
| Suburb | No | Text | 16 | - |
| City | Yes | Text | 16 | - |
| EmailAddress | Yes | Text | 32 | - |
| PhoneNumber | Yes | Text | 10 | - |
| Status | Yes | Text | 9 | Part-time or full-time |

**ISSUE**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Required** | **Data Type** | **Maximum Length** | **Range/List** |
| IssueID | Yes | Auto-number | 5 | Primary Key |
| IssueDescription | Yes | Text | 64 | - |
| IssueDate | Yes | Date | 10 | DD/MM/YYYY |
| ActionTaken | Yes | Text | 64 | - |
| EnrolmentID | Yes | Foreign Key (ENROLMENT) | 12 | - |

**ASSIGNMENT**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Required** | **Data Type** | **Maximum Length** | **Range/List** |
| CourseID | Yes | Foreign Key (COURSE) | 4 | Primary Key |
| LecturerID | Yes | Foreign Key (LECTURER) | 4 |
| Role | Yes | Text | 9 | Primary or Secondary |

**LECTURER**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Required** | **Data Type** | **Maximum Length** | **Range/List** |
| LecturerID | Yes | Auto-number | 4 | Primary Key |
| LastName | Yes | Text | 16 | - |
| FirstName | Yes | Text | 16 | - |
| StreetAddress | Yes | Text | 32 | - |
| Suburb | No | Text | 16 | - |
| City | Yes | Text | 16 | - |
| PhoneNumber | Yes | Text | 10 | - |
| EmailAddress | Yes | Text | 32 | - |
| Ranking | Yes | Text | 1 | A, B or C |
| Type | Yes | Text | 8 | Academic or Contract |

**RESEARCH PROJECT**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Required** | **Data Type** | **Maximum Length** | **Range/List** |
| ResearchProjectID | Yes | Auto-number | 4 | Primary Key |
| LecturerID | Yes | Foreign Key (LECTURER) | 4 | - |
| ResearchTopicID | Yes | Foreign Key (RESEARCH TOPIC) | 2 | - |
| Output | Yes | Text | 16 | - |
| StartDate | Yes | Date | 10 | DD/MM/YYYY |
| ResearchProjectDescription | Yes | Text | 64 | - |

**RESEARCH TOPIC**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Required** | **Data Type** | **Maximum Length** | **Range/List** |
| ResearchTopicID | Yes | Auto-number | 2 | Primary Key |
| ResearchTopicDescription | Yes | Text | 64 | - |
| Impact | Yes | Text | 6 | High, medium or low |

## 8.8 Annotated User Interface Designs

## 8.9 Test Plan

# 9. Project Training

## 9.1 End User Background and Training Objectives

## 9.2 Training Materials

## 9.3 Training Deliverables

## 9.4 Evaluation

# 10. Conclusion & Lessons Learned

# References

Kanchev, K. (2021). 10 Interesting Facts About JavaScript Each JS Programmer May Want to Know. *Techva Me,* [*https://techva.me/ten-interesting-facts-about-javascript/*](https://techva.me/ten-interesting-facts-about-javascript/)

Microsoft. (2021, August 23). *A Tour of the C# Language*. <https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/>

FutureLearn. (2021, April 9th). *What is Python Used For? 10 Practical Python Uses.* [*https://www.futurelearn.com/info/blog/what-is-python-used-for*](https://www.futurelearn.com/info/blog/what-is-python-used-for)

Boyd, M. (2021, May 25th). *What is Go? An Intro to Google’s Go Programming Language (aka Golang). A Cloud Guru,* [*https://acloudguru.com/blog/engineering/what-is-go-an-intro-to-googles-go-programming-language-aka-golang*](https://acloudguru.com/blog/engineering/what-is-go-an-intro-to-googles-go-programming-language-aka-golang)

Red Hat.(2019, January 8th). *What is an IDE?.* [*https://www.redhat.com/en/topics/middleware/what-is-ide*](https://www.redhat.com/en/topics/middleware/what-is-ide)

JetBrains.(2021). *PyCharm Features*.[*https://www.jetbrains.com/pycharm/features/*](https://www.jetbrains.com/pycharm/features/)

Microsoft. (2021, September 14th). *Welcome to the Visual Studio IDE.* <https://docs.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide>

Tutorials Point. (2021). *Sublime Text – Introduction.* [*https://www.tutorialspoint.com/sublime\_text/sublime\_text\_introduction.htm*](https://www.tutorialspoint.com/sublime_text/sublime_text_introduction.htm)

Python. (2021, September 28th). *IDLE*. <https://docs.python.org/3/library/idle.html>

Tsai, D. (2021, April 9th). *Microsoft Access is Good for Databases Management. Trust Radius*. <https://www.trustradius.com/reviews/microsoft-access-2021-04-03-16-54-24>

SQLite. (2021, September 24th). *About SQLite.* [*https://www.sqlite.org/about.html*](https://www.sqlite.org/about.html)

Talend. (2021). *What is MySQL : Everything You Need to Know.* [*https://www.talend.com/resources/what-is-mysql*](https://www.talend.com/resources/what-is-mysql)

Titus. (2020). *What is Moodle : A Comprehensive Guide to The Worlds Most Popular LMS.* [*https://www.tituslearning.com/what-is-moodle-lms-a-comprehensive-guide/*](https://www.tituslearning.com/what-is-moodle-lms-a-comprehensive-guide/)

Budram, M. (2021, August 24th), *Moodle Reviews & Product Details. G2.* [*https://www.g2.com/products/moodle/reviews/moodle-review-4961478*](https://www.g2.com/products/moodle/reviews/moodle-review-4961478)

# Appendices