

CHAPTER ONE: INTRODUCTION

1.0 INTRODUCTION

At schools, educational institutes and universities, a project is a research assignment given to a student which requires greater effort and independent work than is normally involved in a normal assignment, (Purwanto Sugeng, 2016). It requires students to undertake their own fact-finding and analysis, either from library/internet research or from gathering data empirically, (Gustafsson T, 2009). Projects inspire students to challenge themselves and learn problem solving and other skills indispensable to their future, for example time management and critical thinking.

The proposed application is based on making project research more efficient, specifically at the Copperbelt university for both the student undertaking the project, and the lecturer(s) supervising the project. It will challenge and encourage students to think outside the box and be creative as they select and implement a final year project – after which they will be graded accordingly.

1.1 PROBLEM STATEMENT

Recording, grading and storing of final student projects has proven to be hectic, time and resource consuming and takes up much needed space as the thesis books are kept in lecturers office spaces. Sometimes it so happens that students feel they have not been fairly graded for their project work which questions both the student effort and the lecturers grading. Students make mistakes in writing proposals because of broad and unclear topics, failure in methodology, (Sheila Fram, 2014) terminologies of research, problems in reporting the literature review. Furthermore, challenges faced by students in writing quality research proposals include absence of standard format, lack of knowledge in identifying clearly relevant literature review, lack of good, adequate, and regular feedback from supervisors, lack of materials related to selected topics, and finally the time arranged for writing proposals is not adequate (Qaseem & Zayid, 2019). In addition, given that majority students pick their own projects, it is no coincidence that sometimes multiple students pick the same project or a pick a project that has been done before which encourages plagiarism. This document proposes a web application that aims to solve all of the problems stated above.

1.2 THE PROPOSED SYSTEM

This document proposes a web application that will record all final year projects with student information to enable both students and lecturers to search and view the projects using either the project title or key words. The application will enable an independent number of lecturers to view and grade it according to the university criteria and automatically sum/average the total grades to obtain the final grade.

1.3 SCOPE OF THE PROJECT

The scope of this project will be focused on developing a web application that will record projects for final year students with the following student information; student name, student number, project title, key words, abstract, supervisor and a pdf copy of the project. The

application will enable other students to view the recorded projects and lectures to view and grade students using the school criteria.

1.4 OBJECTIVES

The following are the objectives of the proposed web application:

- To do a baseline study to understand the problem in detail from the relevant stakeholders- that is, the students and lecturers by interviewing both parties to find out the challenges they are facing.
- The application will record all final year projects with student information
- The application will allow other students and lectures to search for and view the recorded projects
- Lectures will be able to asses and grade student projects using the application.
- The application will automatically calculate the final grade of each students' project.
- The application should enable students to search for and upload projects.
- The application will enable the generalization of reports in form of graphs and charts, showing progress and other important statistics.

1.5 BENEFITS

- Projects will be recorded and reserved online which is a much more efficient way of storing information than the physical storing and sharing of thesis books which take up much needed space and sometimes get damaged and lost.
- Students will be able to search for and view previously done projects as well as projects currently being worked on so as to avoid doing projects that have been done before. Or impressively improve a previously done project.
- The cost of recording, documenting and storing projects will be greatly reduced.
- As they work, students will have many references as guidelines on how to go about the documentation as well as the implementation to produce projects that not only meet but exceed the expectations.
- Lecturers will be able to grade final students' projects using the application.
- Plagiarism will be completely eliminated as students will not be able to copy projects that have been done before.
- To the gain of the university, students will refer to and compare the previously done projects to their projects with an aim to do better than was done before which will gradually increase the standard of projects being produced.

1.6 METHODOLOGY

The methodology will be divided into 2; research methodology and System methodology.

1.6.1 RESEARCH METHODOLOGY

1. Baseline Study

This will involve interviewing the relevant stakeholders, that is, a few random lecturers from various schools will be interviewed to find out the major and minor challenges they faced while supervising and grading student project. A randomly selected number of students will also be interviewed to find out the challenges they faced while working on their projects. They will all be asked about what possible solutions they think would make project work and grading more efficient.

2. Observation

A lot of information will be gathered from basic observation. Observing the challenges that lecturers are going through as they grade projects and observing the challenges students face as they work on their projects. As well as observation from personal experience.

3. Data Analysis

Here, the data taken from the g is analyzed to gain information with an aim of using logical reasoning on the derived knowledge so as to make better, more informed decisions.

1.6.2 SYSTEM METHODOLOGY

1. Strategy

The incremental model from the software development life cycle (SDLC) involves breaking the requirements into standalone modules, (S shylesh, 2017) which are sequentially achieved starting with analysis design, implementation, testing and maintenance. After the first increment, a central product is delivered. Depending on customer feedback, a strategy is developed for the next increments, and modifications are made accordingly (Pressman R, 2010)). This process continues, with increments being delivered until the complete product is delivered.

1. Requirement analysis: In the first phase of the incremental model, the product analysis expertise identifies the requirements. And the system functional requirements are understood by the requirement analysis team. To develop the software under the incremental model, this phase performs a crucial role.

2. Design & Development: In this phase of the Incremental model of SDLC, the design of the system functionality and the development method are finished with success. When software develops new practicality, the incremental model uses style and development phase.

3. Testing: In the incremental model, the testing phase checks the performance of each existing function as well as additional functionality. In the testing phase, the various methods are used to test the behavior of each task.

4. Implementation: Implementation phase enables the coding phase of the development system. It involves the final coding that assist in the designing and development phase and tests the functionality in the testing phase. After completion of this phase, the number of the product working is enhanced and upgraded up to the final system product.

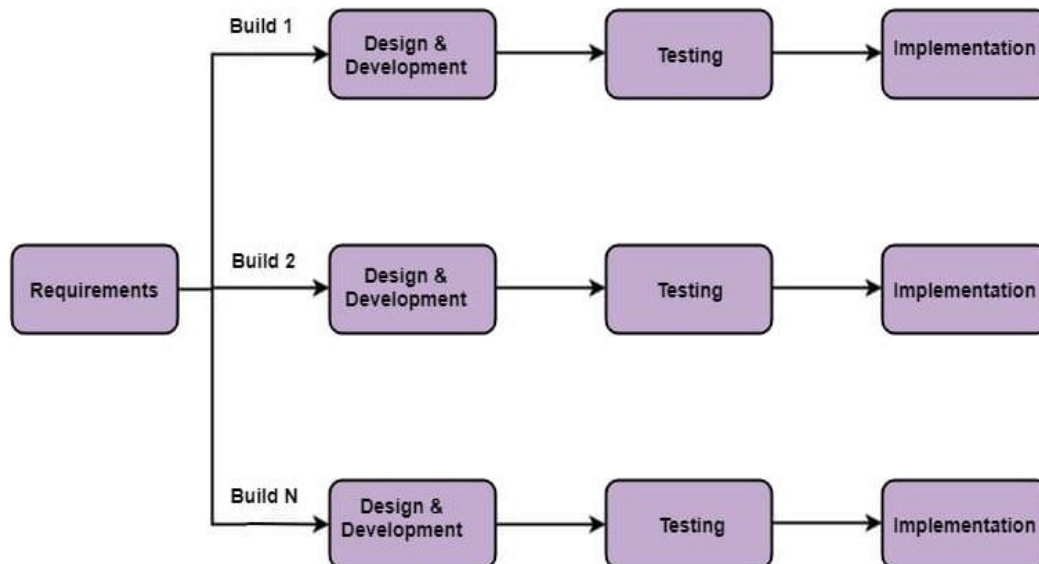


Figure 1: Incremental model, JavaTpoint(2011).

1.6.3 JUSTIFICATION OF THE SELECTED METHOD

The incremental model will be used in the creation of this model because it is a strategy that enables prioritized requirements to be executed first, (Kneuper, Ralf, 2017) and is generally easier to test and debug than other methods of software development because relatively smaller changes are made during each iteration. This allows for more selective and rigorous testing of each element within the overall product especially that this is a project with high-risk features and goals. In addition, returning to a previous stage is possible and flexibility to change is easy, making it low risk and cheaper. After each iteration, regression testing will be conducted. During this testing, faulty elements of the software can be quickly recognized because minimal changes are made within any single iteration.

CHAPTER 2: LITERATURE REVIEW

INTRODUCTION

A research project involves studying an area of interest in depth. The proposed application is based on making project research more efficient, specifically at the Copperbelt university for both the student undertaking the project, and the lecturer(s) supervising the project. This document assists in gaining an understanding of the existing systems and documented research related to the proposed web application. Previously done systems were studied and compared in order to take note of the concepts, research methods and implementation techniques used in an attempt to use the same or similar methods to realize the proposed web application. Different aspects such as grading, and supervision were studied and these were the findings;

2.0.1 GRADING

According to (Salvio Intravaia, 2009), grading in education is the attempt to apply standardized measurements of varying levels of achievement in a course. Grades can be assigned as letters (for example, A through F), as a range (for example, 1 to 6), as a percentage, or as a number out of a possible total (for example, out of 100) (Salvo Intravaia, 2009). Grades mean the actual score or mark that a student receives based on the quantity of measurement and decision-making of assessment defined by (Yesbeck, D, 2011). Grades are a means of communication concerning student performance. Grading can be further and more relatably defined as an attempt to determine how well a student has learnt the material taught in class and used it to do quality research. While absolute grading is when the quality of work is allocated a percentage. Grading is important in the proposed system as it will be given to the students as a measure of the quality of their research project.

2.0.2 SUPERVISOR

A supervisor, or also known as foreman, boss, overseer, facilitator, monitor, area coordinator, or sometimes gaffer, is the job title of a low-level management position that is primarily based on authority over a worker or charge of a workplace Roberson, (Quinetta M, 2019). A supervisor can also be one of the most senior in the staff at the place of work, such as a professor who oversees a PhD dissertation. Supervision is also defined as the act or function of overseeing something or somebody, (Writte J, 2015). It might include advising, educating, monitoring and supporting in relation to task output or task performance. Supervision is an important aspect of this application as supervisors are necessary, first as judges on a panel, and also as individual students' main supervisors.

2.0.3 PROJECT

A Project is a temporary, unique and progressive attempt or endeavour made to produce some kind of a tangible or intangible result (a unique product, service, benefit, competitive advantage, etc.). It usually includes a series of interrelated tasks that are planned for execution over a fixed period of time and within certain requirements and limitations such as cost, quality, performance, others. According to the PMBOK-Project Management Body of Knowledge (PMBOK, 2011), A project is defined as a "temporary endeavour with a beginning and an end and it must be used to create a unique product, service or result" (Prachi Juneja,

2020). The project is a major aspect of the proposed application as the application is based and dependent on the projects uploaded by the students.

2.0.4 STUDENT

Students are one of the major stakeholders necessary to the system. According to the definitions.net, a student is a person engaged in study Macan, Therese H, Shahani, (Comila, 1990); one who is devoted to learning; a learner; a pupil; a scholar; especially, one who attends a school, or who seeks knowledge from professional teachers or from books; as, the students of an academy, a college, or a university; a medical student; a hard student, (Chikopa J, 2021). Students enroll into a college or university with the aim of acquiring knowledge which they will later use to get a job. In the proposed application, students play the important role of being the ones to upload their approved project to the application.

2.1 OPUS

OPUS is the main system student system used by the Copperbelt university for all its online grading and university-student communication. In relation with the aspects stated above, OPUS can be compared as follows;

Grading in opus is done by lecturers, given they have a different permission level from students. Grading at the Copperbelt University is done with an aim to determine how well a student has learnt the material taught in class. Once the lecturers upload the results to the main student system, the students are able to view their individual results which are presented as letters A through F. An independent supervisor is assigned to each final year student. The supervisor can either pick the project idea for the student, or the student can pick a project they deem fit for themselves, which can then be approved by the supervisor. The role of the supervisor is to oversee the project as the student works on it and to educate, provide advice, monitor and support the student where needed. The student is required to use the available resources to research or experiment in order to produce good quality work which they will later upload to the proposed application to be graded.



Figure 2: OPUS

2.2 RELATED SYSTEMS

The systems vary according to the particular key points and aspects stated above and how they relate to each other. Other aspects compared include; the platform the application can run on, what operating system is needed to run it, and other small features. The following are the findings:

2.3 EASY GRADE PRO

Easy Grade Pro is spreadsheet-based gradebook software that keeps track of an unlimited number of classes and students, (Ispring, 2015). Like the proposed system, the easy Grade Pro application is used to store and preserve student work, (Meyer, Katrina A, 2014). Once the work is in the system, it can be manipulated by the teacher/lecturer, after which the students' final grade is calculated by the system. Unlike the proposed application, this application has no student access as it is not web based. Also, the application requires a user to have an understanding of the math behind the grade calculation to successfully use the system while the proposed application will not require a user to know the math, (Bastian K, Alexandre M. S. Costa, 2008).



Figure 3: Easy Go Pro

2.4 EDMODO

Edmodo has an intuitive student interface, reminiscent of Facebook and Twitter, which will help with student engagement, (Meyer Katrina A, 2015). Just like the proposed application, students are able to upload their work to the application as well as get feedback from the lecturer. The major difference with the proposed application and Edmodo is that with Edmodo, students have to enroll with a code before they can receive or upload any work, (St. Pierre, E. A., Jackson, A.Y, 2014) Also, this application enables supervisor to supervisor interaction while the proposed application will not offer that.

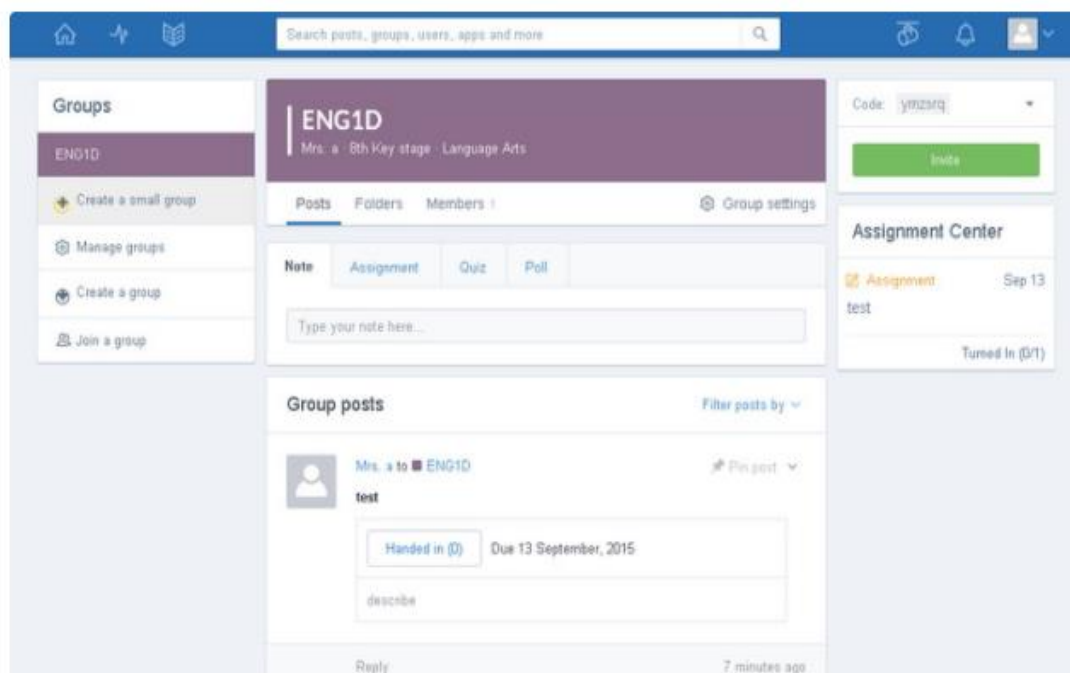


Figure 4: Edmodo

2.5 SCHOOLOGY

Schoology is an educational application that allows students to self-enrol in a class, as long as they have the access code. The interface combines features of Facebook and Twitter, making it easy for students to navigate, (Song Xiatao, Sun Hailong, 2019). Like the proposed application, students are able to upload files to the application to be graded by the respective lecturers. Also, both students and lecturers have access to previously uploaded information which is uploaded from various sources or linked to the application.

Like the proposed application, assignments, or in this case, projects, are uploaded to the web application. Lecturers can only see the progress and project of an already enrolled student, just as in the proposed application and can grade it according to the given criteria, Gorman, (Michael E, 2002). While the proposed system will only allow two-way communication between the lecturer and student, Schoology allows all communication between students, between lectures and between students and

lecturers. In addition, Schoology supports video and audio conferencing while the proposed application will not, (Emma F, Anne H, Michael G, 2001).

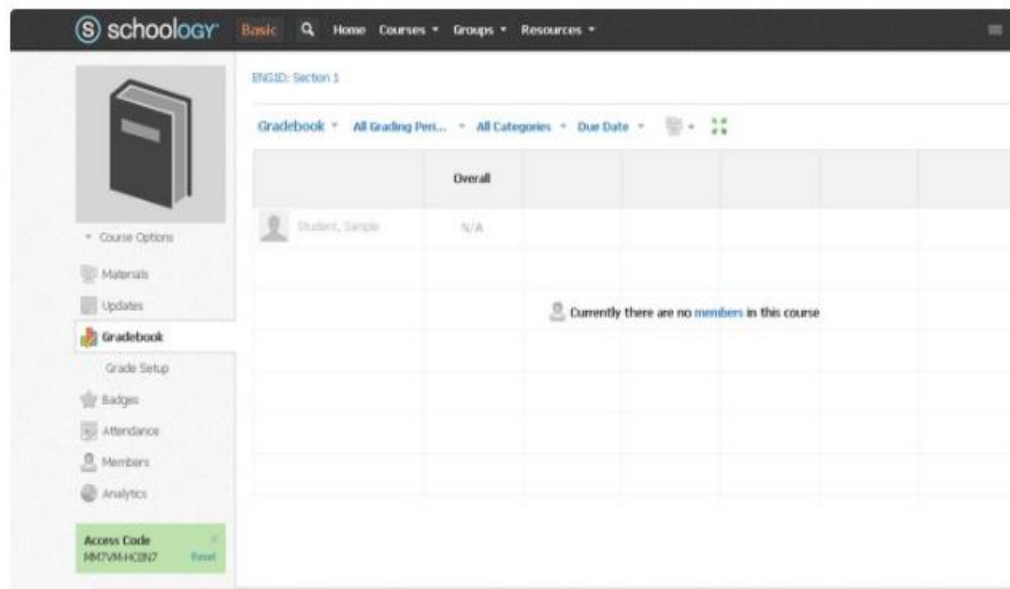


Figure 5: Schoology

2.6 THINKWAVE

Thinkwave is an application used to keep track of student grades and communicate student progress to parents, (Ispring, 2015). Just like the systems mentioned above, students are able to upload their work to the application, where it can be assessed, viewed, and graded by the respective lecturers. However, unlike the proposed application, it offers different types of report styles and files to be used while the proposed application will only deal with one type of file. The major difference between these two applications is that in Thinkwave, each student has to be manually added to the database while in the proposed application, as students register, they will be automatically added to the database.

The screenshot shows the 'Thinkwave' software interface. At the top is a navigation bar with links: Dashboard, Students, Classes, Gradebook, Attendance, Reports, Setup, Upgrade. Below this is a sidebar on the left with a yellow 'Add Class' button and a 'Courses' section containing a dropdown menu set to 'All Classes' and a list item 'English'. The main area is titled 'Class Info' and contains a 'Save' button at the top right. The form fields are as follows:

- Course:** English
- Credits:** 1.0
- Final grade:** Letter Grades (dropdown)
- Honors:** ☐
- Exclude:** ☐ (from report card, transcript, units & gpa)
- Optional Tags:**
- Section:**
- Terms:** ☒ Semester 1, ☒ Semester 2
- Location:**
- Schedule:**

Mon	First Period (dropdown)
Tue	First Period (dropdown)
Wed	First Period (dropdown)
Thu	First Period (dropdown)

On the right side, there is an 'Enrollment' section with a plus icon. It contains a list: b, a, d, c, f, e. Below this are two checkboxes: ☐ Enroll New Student and ☐ Enroll & Unenroll Existing Students.

Figure 6. Thinkwave

2.7 GRADEKEEPER

Gradekeeper is simple, spreadsheet-style software dedicated to one purpose: keeping grades (Ispring, 2015). The Gradekeeper, supports a number of online services to facilitate the efficient storage of student grades. Although it is the only aim it has, Gradekeeper and the proposed system have a main aim of storing and preserving student grades. The major difference is that Gradekeeper is not a web-based application while the proposed application is.

#	Student	Grade	9/3 Bulgarian Solitaire Problem Solving 10 points	9/4 Spiragraph Special Activities 10 points	9/5 Tootlick's Problem Solving 25 points	9/6 Cypsy Moths Activities 10 points	9/9 Chapter 7 Homework 45 points
1	Albright, Amy	94.6% A	10	9	23	10	45
2	Bravado, Barbara	69.0% D+			[2] 18		35
3	Cunningham, Cathy	92.3% A-	9	8	24	10	45
4	Deal, Danielle	79.8% C+	8	7	19	7	40
5	Earnest, Eileen	82.6% B-	8	8	22		45

Figure 7. Gradekeeper

2.8 ANALYSIS

All the systems were analyzed and compared for usage. All six systems, including the proposed application have similar usage. All of them allow students to upload files to the systems which could later be viewed, assessed, and graded. In the Proposed application however, students will not be able to view their grades as seen in the other applications as when the result has been automatically calculated it will be automatically forwarded to the student main system. In addition, most of the related applications are full classroom applications with full interaction and multiple courses, but the proposed application will strictly focus on the research project as the only course. Most of the compared applications run on all platforms and all operating systems except one, GradeKeeper which only runs on iOS.

2.9 CONCLUSION

This document proposes an application that will record final year student projects to assist in the recording of project information as well as solve the problem of repeated projects, plagiarism and assist students who do not know how to go about their projects. The proposed application will make project supervising easier and more efficient as lectures will be able to view and grade the projects.

From the data gathered above, a short analysis was done to compare the proposed system with previously done, relatable systems. A conclusion can be drawn to say the proposed application will have many features and functionalities like all the compared applications. It will also have notable differences depending on the requirements of the application. The

proposed application will be designed to work on a desktop platform and will be specifically designed for the Copperbelt University.

CHAPTER THREE: REQUIREMENTS SPECIFICATION

3.0 INTRODUCTION

This chapter outlines the functional and non-functional requirements of the proposed CBU Final Year Project Grading application. It will describe the various services the proposed application is expected to provide, and the constraints under which It must operate. It will show different characteristics of the system such as how it will work and what it will do.

3.1 THE PROPOSED SYSTEM

This document proposes a web application that will record all final year projects with student information to enable both students and lecturers to search and view the projects using either the project title or key words. The application will enable an independent number of lecturers to view and grade it according to the university criteria and automatically sum/average the total grades to obtain the final grade – after which students will be able to see their grade.

3.2 FUNCTIONAL REQUIREMENTS

The functional requirements of a system are what the system should do, i.e. statements briefly describing the services the system ought to provide, how the system should behave in particular situations, and how the system should react to particular inputs. The proposed system's functional requirements are as follows;

1.0 The application will record all final year projects with student information.

1.1 Students will be able to login to the application by logging into the main student system with correct credentials.

1.2 On the welcome page, the user will have the option of either uploading a project, or searching for projects.

1.3 Having chosen to upload the project or grade, the user will submit, and the information is sent to the database.

1.4 When the information is successfully uploaded, the user will receive a popup notification and the record will be added to the database.

2.0 The application will allow other students and lectures to search for and view the recorded projects.

2.1 After choosing to search at the home page, the user is taken to search page where they are prompted to input their search.

2.2 If a valid input is given, all possible results are brought forward. If an invalid input is searched or no results match the search, a negative result message is returned.

3.0 Lectures will be able to asses and grade student projects using the application.

3.1 On their dashboard, a lecturer will be able to view projects uploaded by students and grade it.

3.2 A lecturer will be able to grade a students' work either as a supervisor on the panel or as a supervisor giving a final grade.

4.0 The application will automatically calculate the final grade of each students' project.

4.1 The average of grades given by the lecturers on the panel will be added to the grade given by the supervisor to give the final grade.

5.0 The application will enable the generalization of reports.

5.1 The reports will be in form of graphs and charts, showing important statistics.

3.3 NON-FUNCTIONAL REQUIREMENTS

Non- functional requirements basically describe how a system works. They can be defined as the implementation constraints, quality attributes, design, and external interfaces which a product must have, which includes important behavioural properties such as performance and usability, (Cortés, U, 2003). They are constraints on the services or functions offered by the system that include constraints on the development process, and constraints imposed by standards. Non-functional requirements often apply to the system as a whole rather than individual system features or services. The following are the non-functional requirements of the system;

1.0 In terms of **availability**, the application will have a hosting database containing user and project information. Users will be able to access the information in the database through their various devices at all times of the day or night.

2.0 In terms of **maintenance** of the system database shall be updated and backed up periodically to ensure that there is no denial of service in the accessing of user information. The system will be made easy to operate, manage and update.

3.0 In terms of **recoverability**, the application will use the key elements of redundancy, i.e. shared nothing architecture. This will enable each node or module to function separately so that failure of one node does not mean failure of the whole system, and nodes will easily be added making scalability much easier.

4.0 In terms of **mobility**, the application will be operational on a wide spectrum of devices and operating conditions, such as performance, screen sizes and internet connection speed.

5.0 In terms of **browser compatibility**, the web application will be compatible with all major modern browsers i.e., Chrome, Safari, Firefox etc.

6.0 In terms of **security and privacy**, users will have to be authenticated before they can log in to the system and when they are logged in, will only have access to their share of information according to the permissions.

CHAPTER FOUR: SYSTEM MODELLING

4.0 PURPOSE OF SYSTEM MODELLING

The purpose of this system modelling is to conceptualize and develop the model that will represent various views of Project Management system. It will help in the visual representation of the overall system showing its interaction with the environment.

4.1 THE GENERAL SYSTEM MODELLING OVERVIEW

Systems modelling or system modelling is the interdisciplinary study of the use of models to conceptualize and construct systems in business and IT development which shows the communication (Alain Wegmann, 2006). System modelling is also described as the process of developing abstract models of a system, with each model presenting a different view or perspective of that system. System modelling in other words mean representing a system using some kind of graphical notation, which is now almost always based on notations in the Unified Modeling Language (UML). System modelling helps the analyst to understand the functionality of the system and models which are used to communicate between the system and its users (actors) (Silisque. I, 2003).

The models that will be covered in this system includes; use cases which shows the features of the system and how users are associated to the system, the context model which shows the working boundaries of the system, the sequence diagrams which illustrate how the different parts of a system interact with each other to carry out a function, and the order in which the interactions occur when a particular use case is executed. In simpler words, a sequence diagram shows different parts of a system work in a 'sequence' to get something done. It describes the interactions between actors, the system and the system components, the activity diagram which shows how processes are done in order to achieve a result.

4.2 CONTEXT MODEL

A context model describes the environment in which the system operates. It defines how context data are structured and maintained (It plays a key role in supporting efficient context management). It aims to produce a formal or semi-formal description of the context information that is present in a context-aware system. In other words, the context is the surrounding element for the system, and a model provides the mathematical interface and a behavioural description of the surrounding environment (Nicolas Guelfi, Anthony Savidis, 2006).

The logging in to the system will be through the main Copperbelt University System, so users will not be required to use other credentials. After the successful logging in, the link will appear on the main system which will navigate to the project management system, once clicked. The system will be responsible for enabling students to submit proposals, progress and completed projects. It will be able to achieve these objectives through the help of the database which will be able to store various records which will be submitted. Once the records have been submitted successfully, the email will be sent to the supervisors who then

evaluates and performs a corresponding action. The security will also be provided as only registered students will be able to interact with the system. The context diagram is shown below.

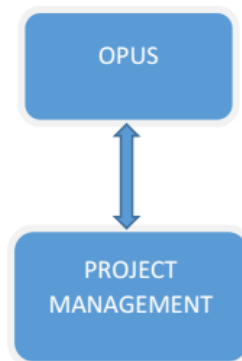


Figure 8: CONTEXT MODEL

4.3 USE CASE MODEL

The purpose of use case diagram is to capture the dynamic aspect of a system. Additional diagrams and documentation can be used to provide a complete functional and technical view of the system. They provide the simplified and graphical representation of what the system must actually do. A use-case diagram can help provide a higher-level view of the system (McLaughlin, 2006). It also describes which features the actors (users) will be associated to. It also shows the relationships between use

cases that always depend on other use cases to carry out a particular function and also use cases that only call other use cases when a certain condition has been met.

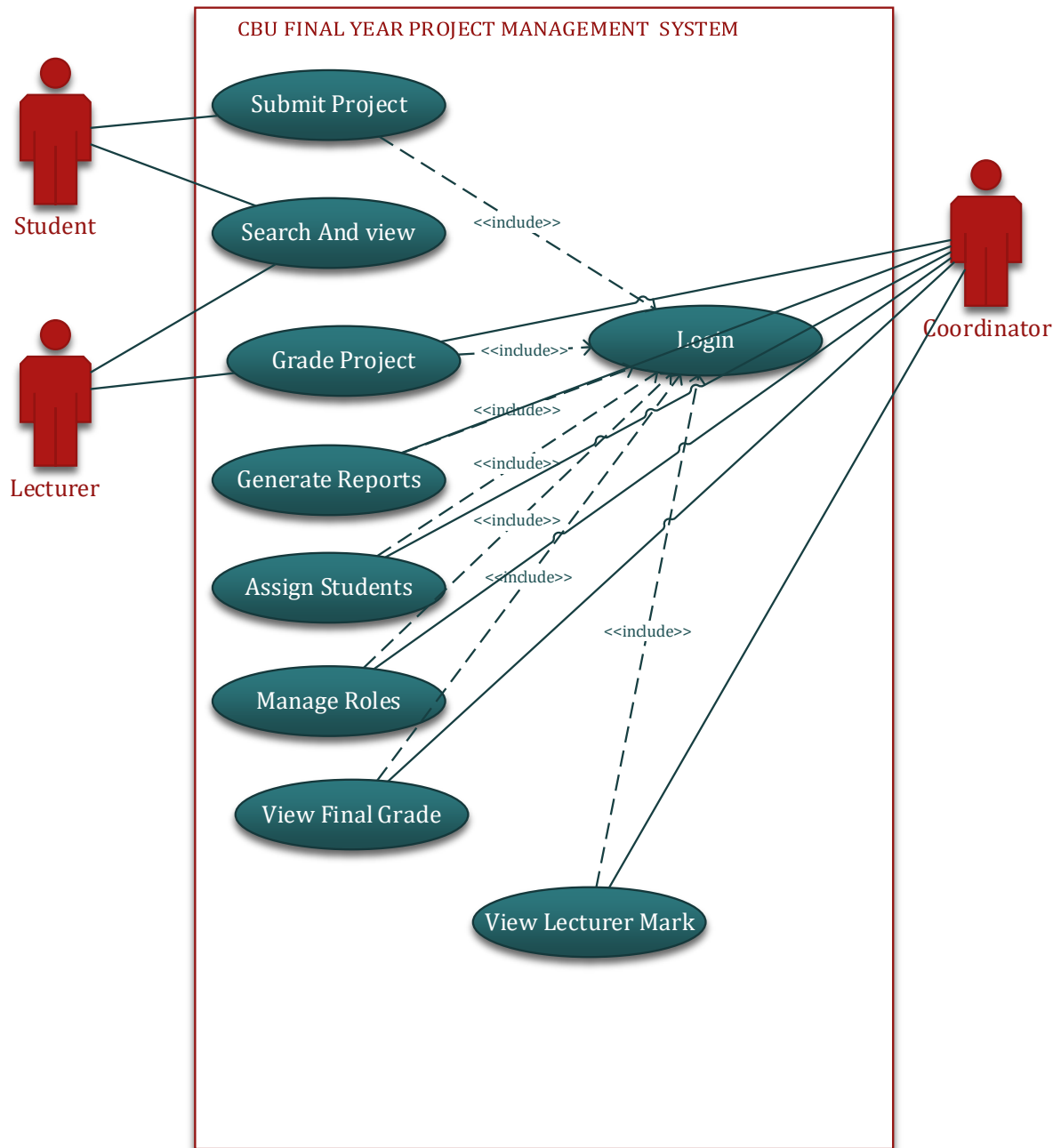


Figure 9: USE CASE MODEL

Table 1: SUBMIT PROJECT USE CASE DESCRIPTION

SYSTEM	Project Management
USE CASE	Submit Project
ACTORS	Students
DATA	The student enters correct data and the appropriate pdf document.
STIMULUS	The entered data and the document are then processed and recorded in the database.
RESPONSE	Once these are successfully stored, an email verification will be sent to both the student and the supervisor. The supervisor then carries an appropriate action.

Table 2: GRADING USE CASE DESCRIPTION

SYSTEM	Project Management
USE CASE	Grading
ACTORS	Lecturers and Coordinator
DATA	The supervisor and other lecturers will enter the marks the student scores from the presentation.
STIMULUS	The mathematical calculations are carried using the entered marks
RESPONSE	Then the system grades the project based on the calculations carried out and gives the grade between (D – A ⁺).

Table 3: GENERATE REPORT USE CASE DESCRIPTION

SYSTEM	Project Management
USE CASE	Generate Report
ACTORS	Lecturers and Coordinator
DATA	The users click on the generate report option and specify the academic year.
STIMULUS	The request is processed and sent to the database.
RESPONSE	The requested data is generated in form of a bar chart, pdf document etc.

Table 4: VIEW PROJECTS USE CASE DESCRIPTION

SYSTEM	Project Management
USE CASE	View Completed Projects
ACTORS	Students, Lecturers and Coordinator
DATA	The users' clicks on the view completed project option and specify the academic year.
STIMULUS	The request is processed and sent to the database.
RESPONSE	The corresponding data is generated and displayed as per request.

Table 5: SEARCH PROJECTS USE CASE DESCRIPTION

SYSTEM	Project Management
USE CASE	Search for projects
ACTORS	Students, Lecturers and Coordinator.
DATA	The users enters either the title of the project or key words.
STIMULUS	The request is processed and sent to the database.
RESPONSE	The corresponding data is generated and displayed as per request.

Table 6: DELETE PROJECT USE CASE DESCRIPTION

SYSTEM	Project Management
USE CASE	Delete Project
ACTORS	Lecturers and Coordinator
DATA	The actors will click on the delete feature
STIMULUS	The request is processed and sent to the database.
RESPONSE	The related data will be deleted once the it matches with what is in the database

4.4 SEQUENCE DIAGRAMS

As the name implies, a sequence diagram shows the sequence of interactions that take place during a particular use case or use case instance, Z. Pala, and N. Inanc, (2007). They are used to demonstrate the interactivity between the actors (stakeholders) and the objects in a system. In this case, the interaction between the user, the dashboard and the server.

4.4.1 SEARCH PROJECT SEQUENCE DIAGRAM

When a user chooses to search a project, a request is sent to the database to search for a project with the given title or key words. If the search yields positive results, the given project(s) are returned to the user.

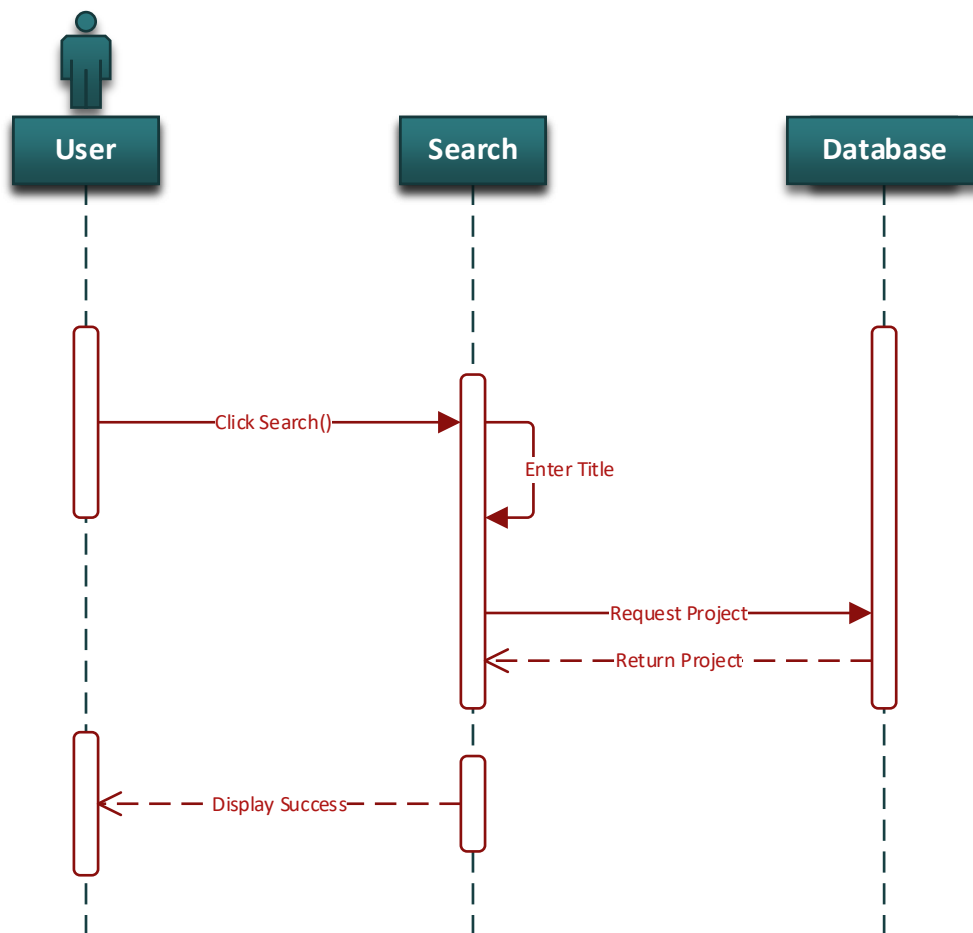


Figure 10: Search Project Sequence Diagram

4.4.2 VIEW PROJECT SEQUENCE DIAGRAM

After a user has searched a particular project, they are able to click the project they want to view from the results returned from the database.

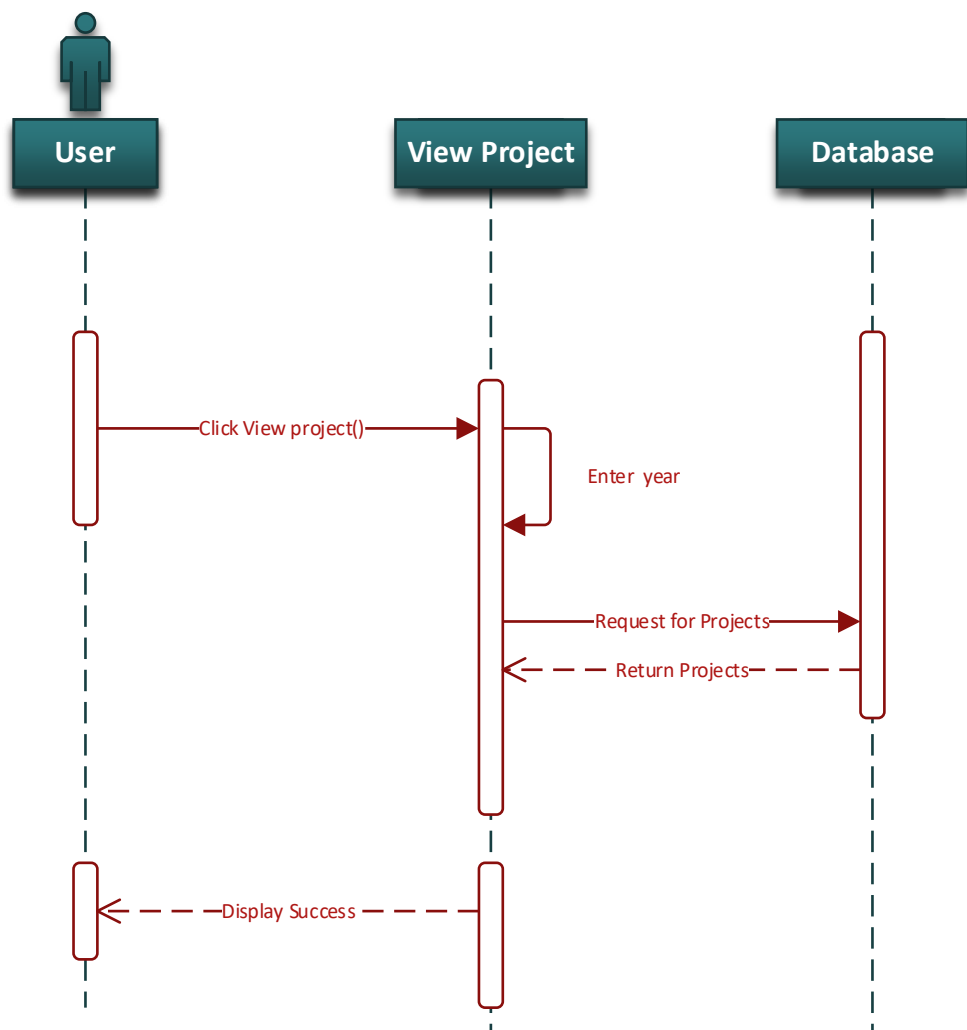


Figure 11: View Project Sequence Diagram

4.4.3 GRADING SEQUENCE DIAGRAM

When a supervisor or lecturer logs in to the application, they are taken to the main dashboard where they can select to grade a project. They will go through a series of drop-down menus until they find the particular student they want to grade. The project will be retrieved from the database and the grading sheet is brought forward. The lecturer then assesses it according to the various fields. After all the lecturers are done, their scores will be computed to make a final grade which is then stored in the database.

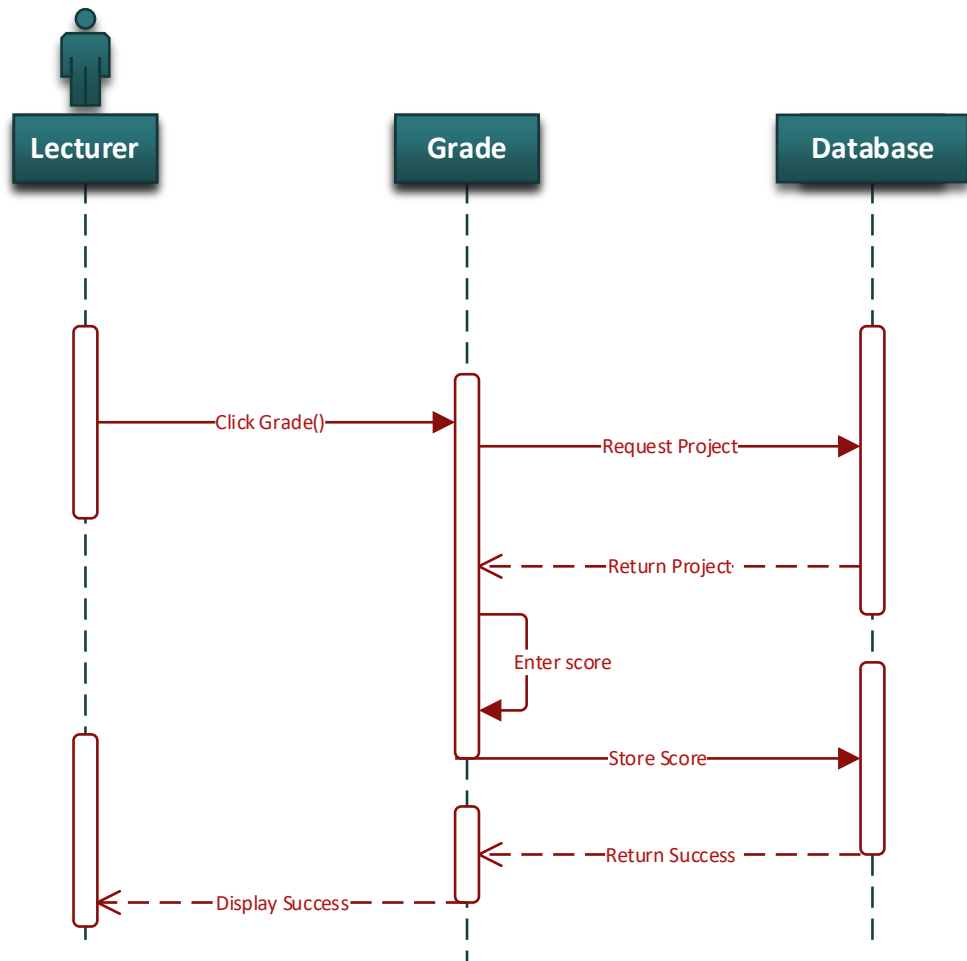


Figure 12: Grading sequence diagram

4.4.4 GENERATE REPORTS SEQUENCE DIAGRAM

When a coordinator or lecturer logs into the application, from their dashboard, they are able to request a data report. This request is sent to the database. Data in the database is calculated and used to generate reports in form of charts and graphs.

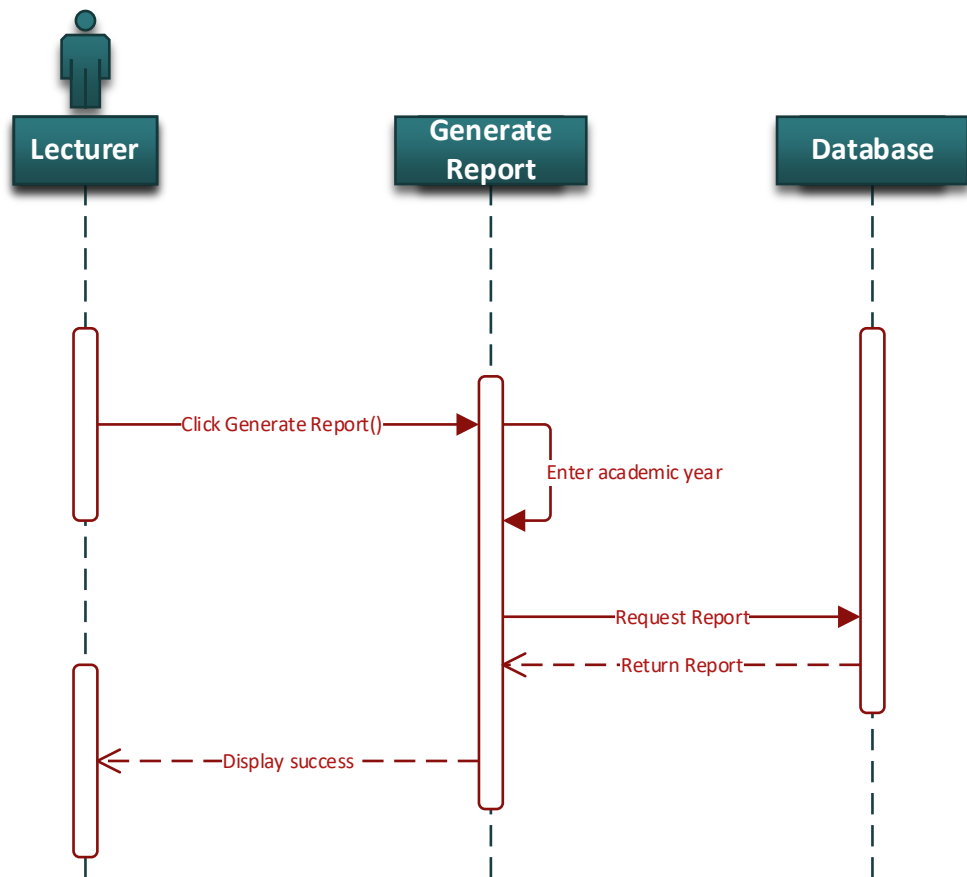


Figure 13: Generate Reports sequence diagram

4.4.5 SUBMIT PROJECT SEQUENCE DIAGRAM

The student will click on the submit feature from the menu and they will be prompted to enter the relevant details and upload the soft copy of their project. Upon a successful upload, both the student and lecturer receive a notification.

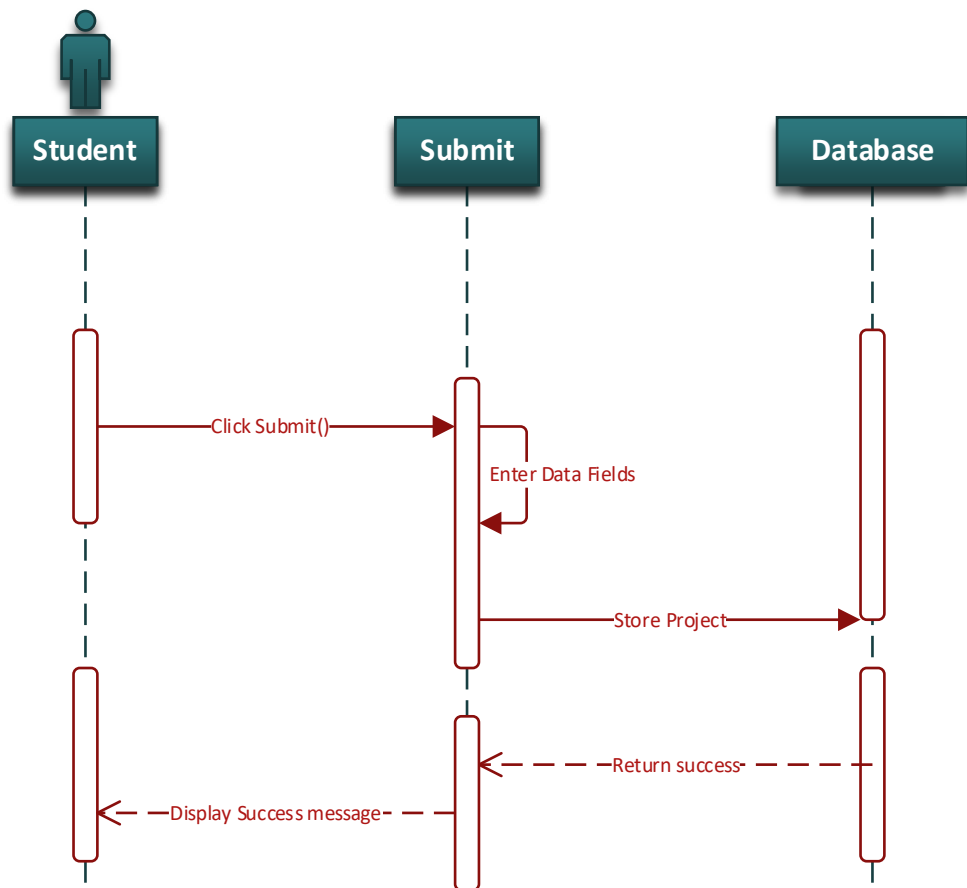


Figure 14: Submit Completed Project Sequence Diagram

4.4.6 DELETE PROJECT SEQUENCE DIAGRAM

In order to delete a project from the database, the user who has the permission will click the delete feature from the menu and another screen will be displayed asking the user to enter the title of the project to be deleted. If the project is found, then the project will be deleted otherwise the user will be asked to enter a valid project title.

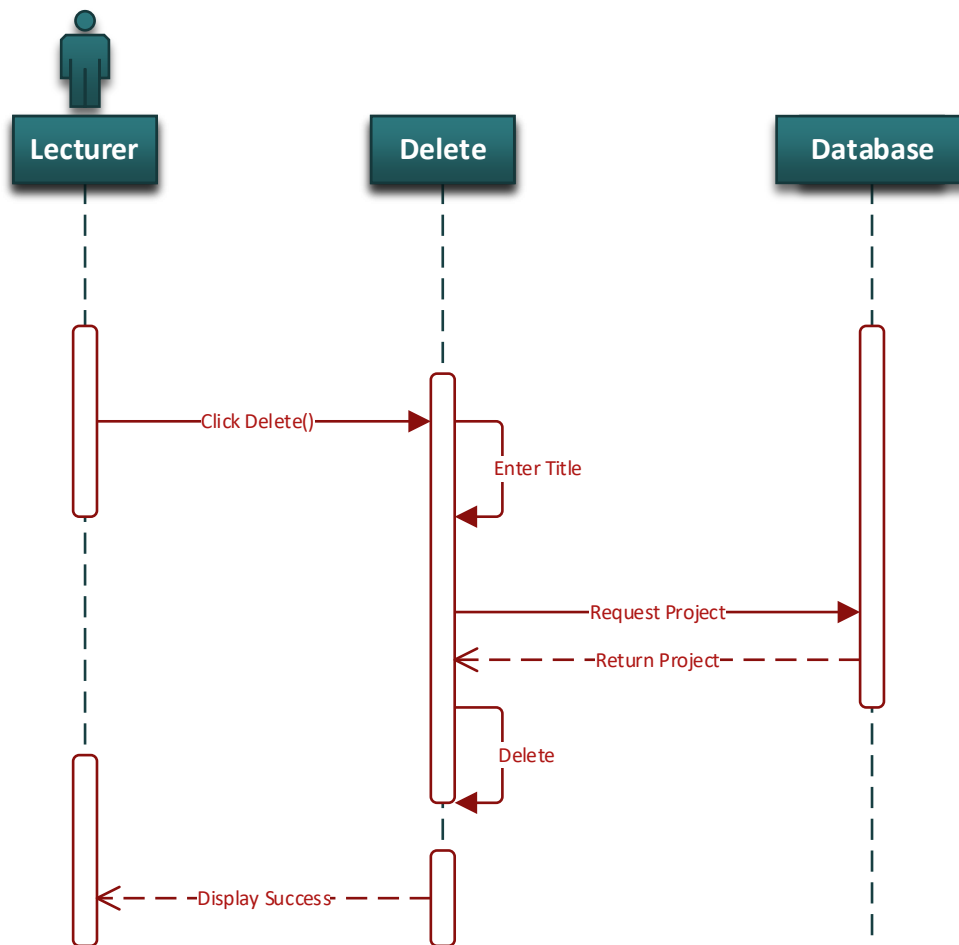


Figure 15: Delete Project Sequence Diagram

4.5 ACTIVITY DIAGRAMS

The Activity diagram describes processes including sequential tasks, conditional logic, and concurrency. It is used to define the logic or work flow or business process. It is useful when working with clients to determine how things should be done, also assessing the complexity of the application, and to verify the internal consistency of the use case definitions. When the details become complicated, being able to draw them out using the Activity diagram makes the logic much easier to understand. Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified modelling Language, activity diagrams are intended to model both computational and organizational processes (i.e., workflows), as well as the data flows intersecting with the related activities. Although activity diagrams primarily show the overall flow of control, they can also include elements showing the flow of data between activities through one or more data stores (J. Rumbaugh, I. Jacobson, 2009).

4.5.1 SUBMIT PROJECT ACTIVITY

When the project is completed and the supervisor is satisfied with the project, the student will then be given a go ahead to submit the project by clicking on the corresponding option on the menu bar and input correct data and then submit.

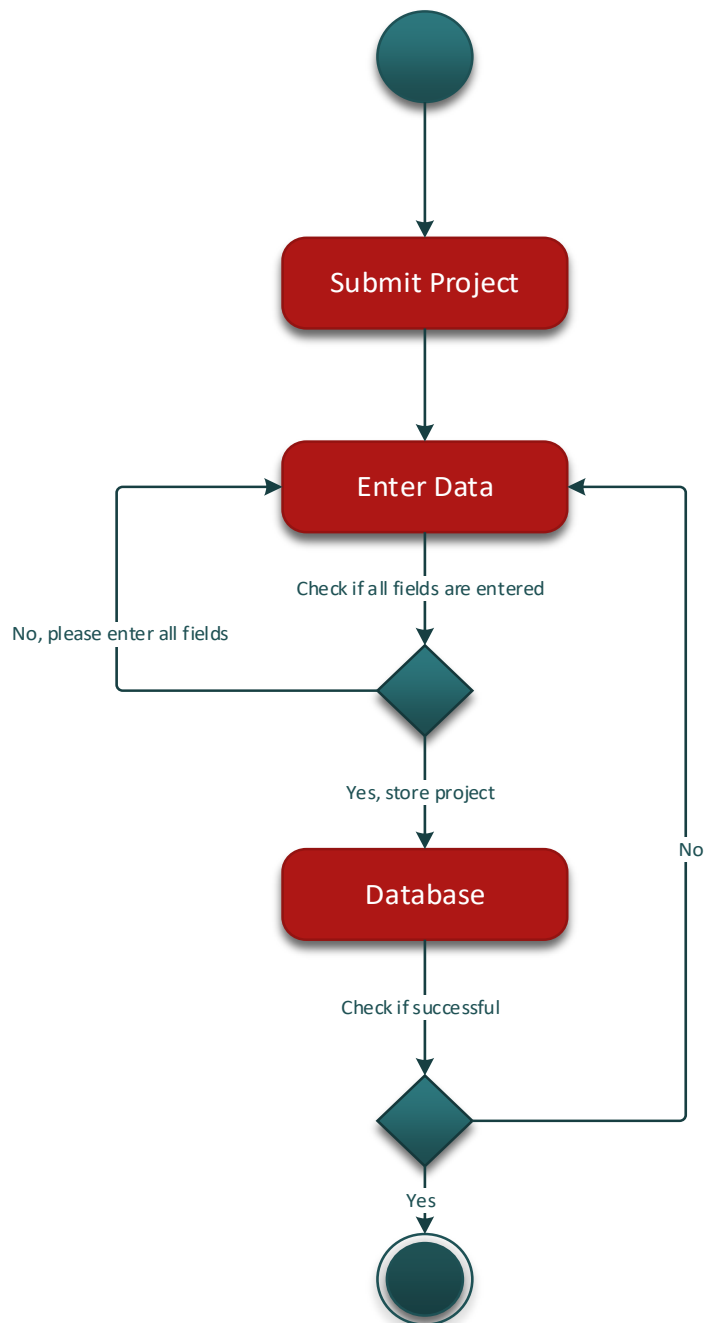


Figure 16: Submit Completed Project Activity Diagram

4.5.2 GRADING ACTIVITY DIAGRAM

Grading will be done by the lecturers on the panel and the supervisor. The lecturers will give scores during presentation according to their satisfaction and the supervisor give following the same criteria. The scores from the presentation will be submitted to the system which will be able to compute the overall mark and grade the project according to the level of satisfaction that will be found.

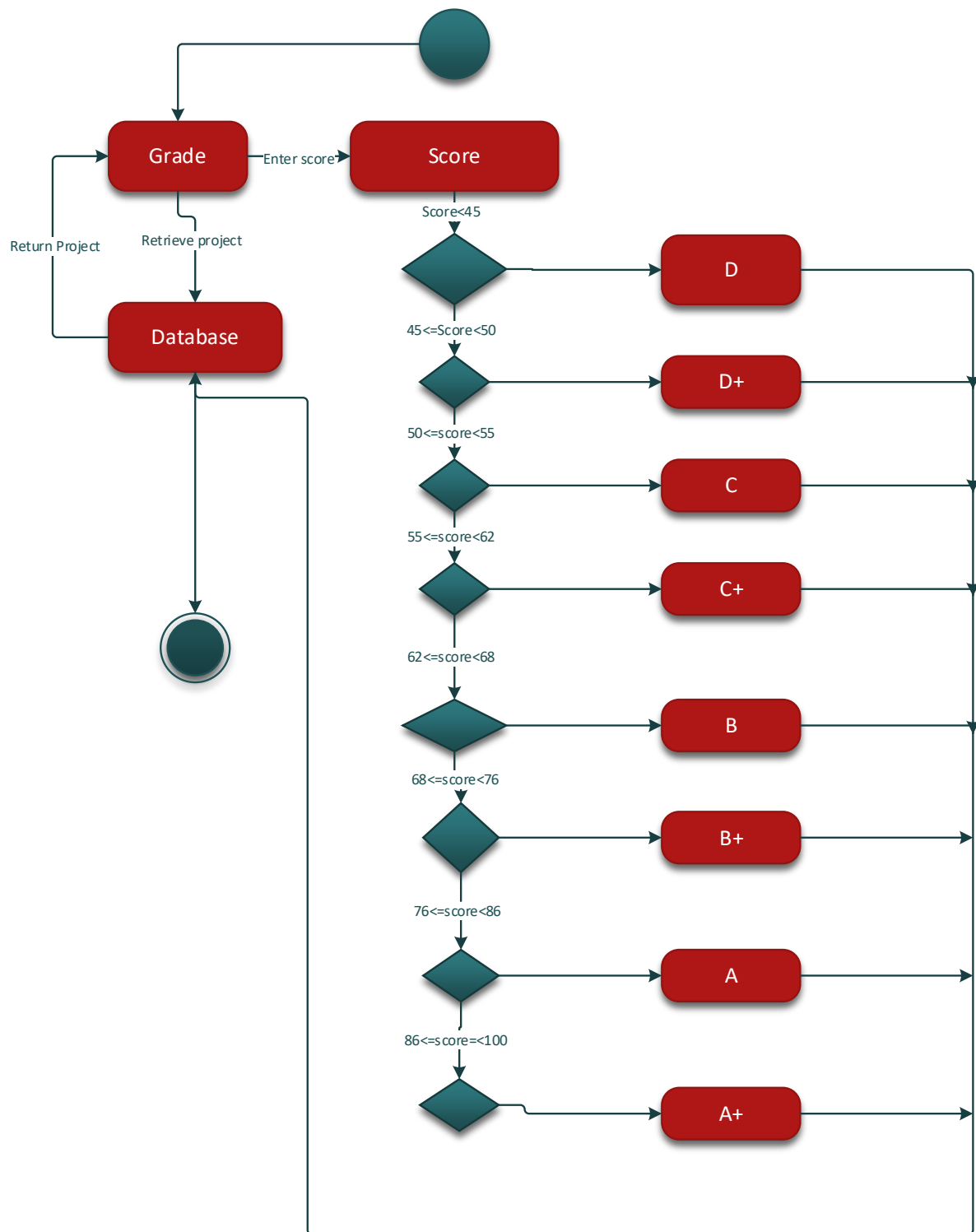


Figure 17: Grading Activity Diagram

4.5.3 VIEW PROJECT ACTIVITY DIAGRAM

All users of the system will be able to view projects from the system by supplying the academic year in which the projects were done.

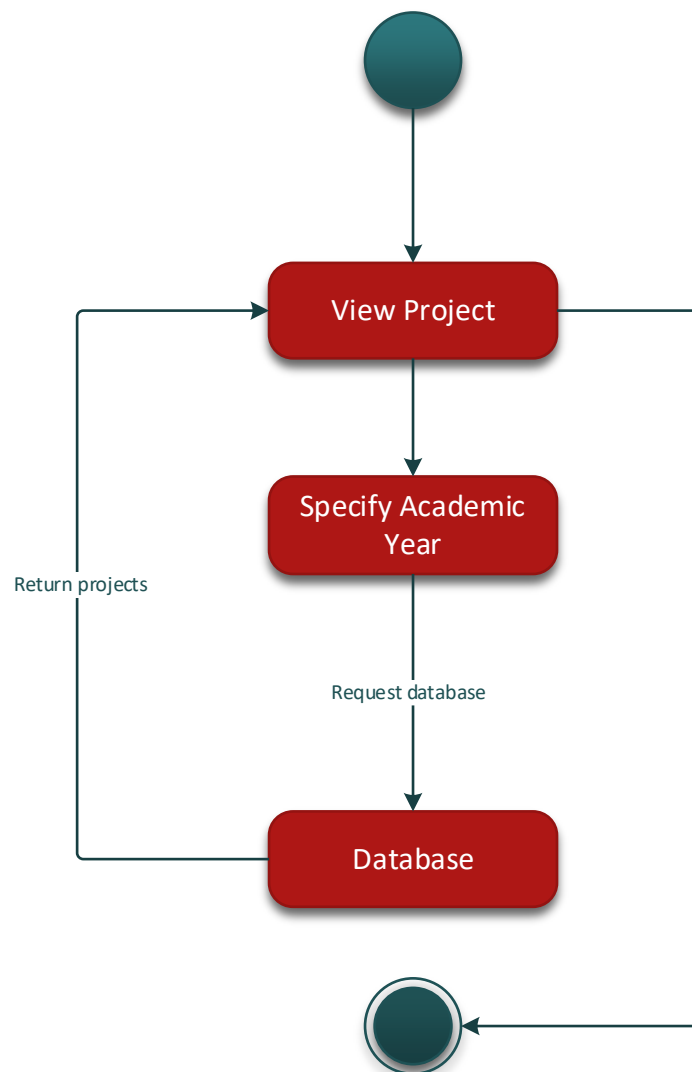


Figure 18: View Completed Projects Activity Diagram

4.4.4 GENERATE REPORT ACTIVITY DIAGRAM

Since all the projects will be stored in the database, either the supervisor, the coordinator or other lecturers will be able to generate the reports in form of either bar charts, pdf files etc. These reports will show the performance of the students on the completed project.

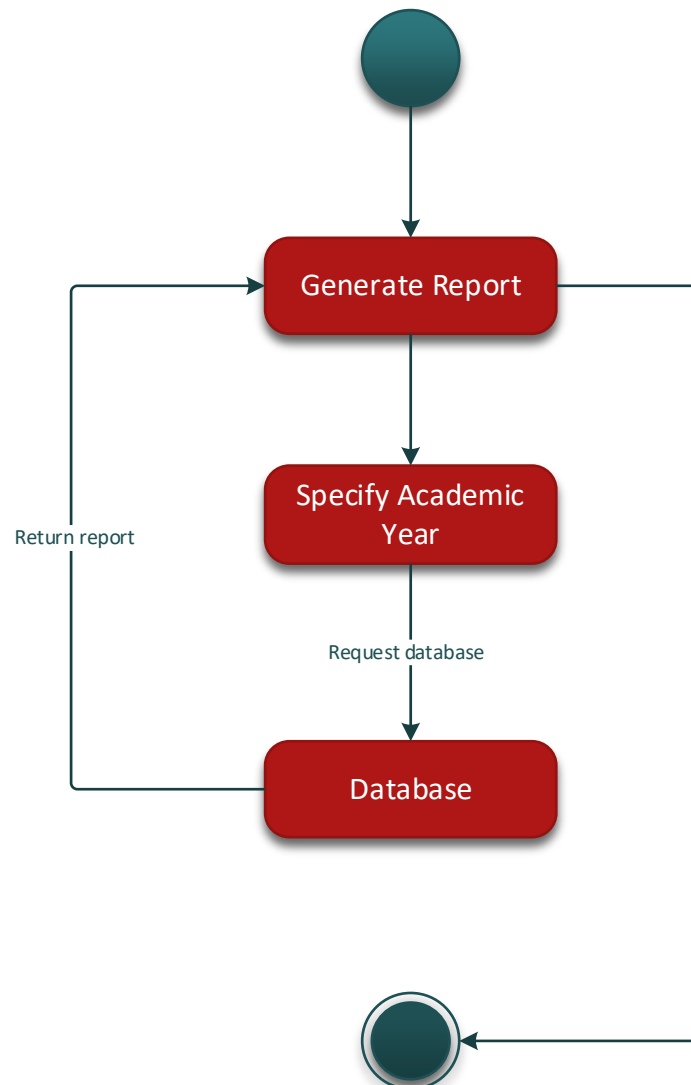


Figure 19: Generate Report Activity Diagram

4.4.5 SEARCH FOR THE PROJECTS ACTIVITY DIAGRAM

All the users will be able to search for the completed projects from the database by either supplying the project title or by using the key words contained in the project. The output of the project will be generated in form of the pdf files which users will read.

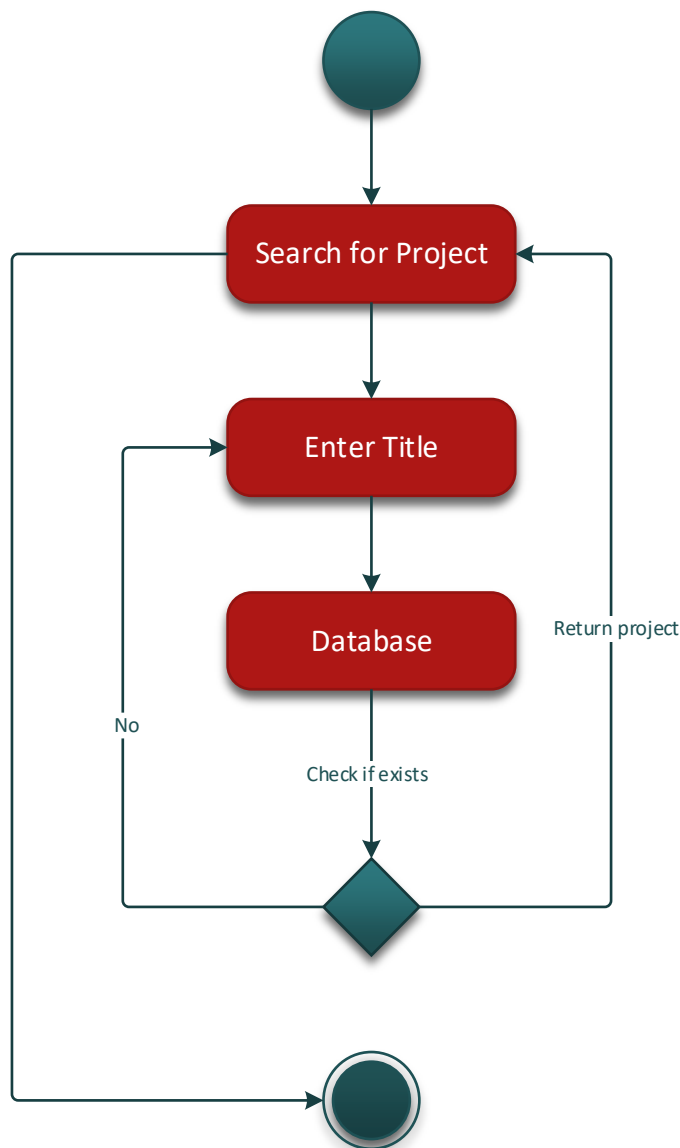


Figure 20: Search for Projects Activity Diagram

4.5.6 DELETE PROJECT ACTIVITY DIAGRAM

Lectures will have an option to delete projects from the system which will be of little satisfaction to them. They will be able to do this by supplying the title of the project from the delete feature in the main system of the project which they will be willing to delete. The activity diagram is shown below.

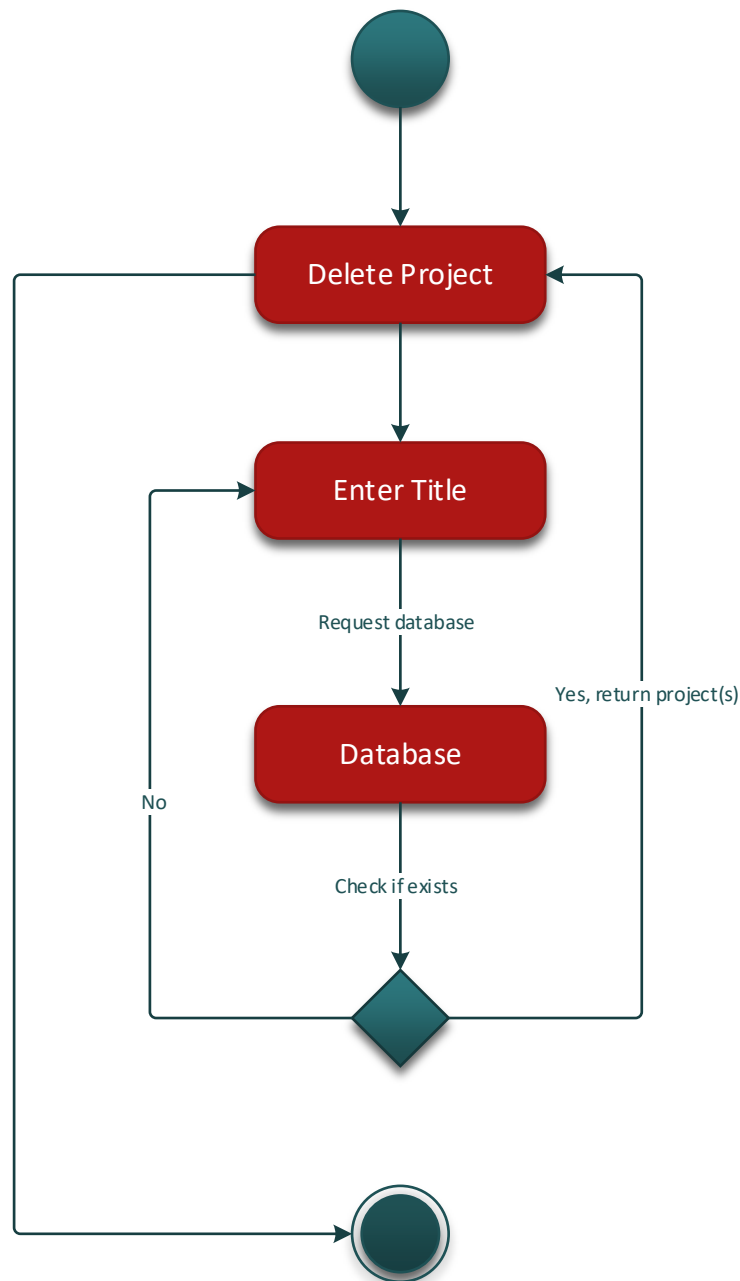


Figure 21: Delete Project Activity Diagram

4.6 CLASS DIAGRAMS

A class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application. Class diagrams describe the attributes and operations of a class and also the constraints imposed on the system.

Below are the class diagrams for the Copperbelt University Final Year Project Grading Application. The classes shown include the student class, the project class, the lecturer and

grade classes with all their given attributes and methods. The student class relates to the project class in that one student can upload and view one project. The lecturer relates to the grading class table and then to the project table in that a lecturer gives one grade to one project.

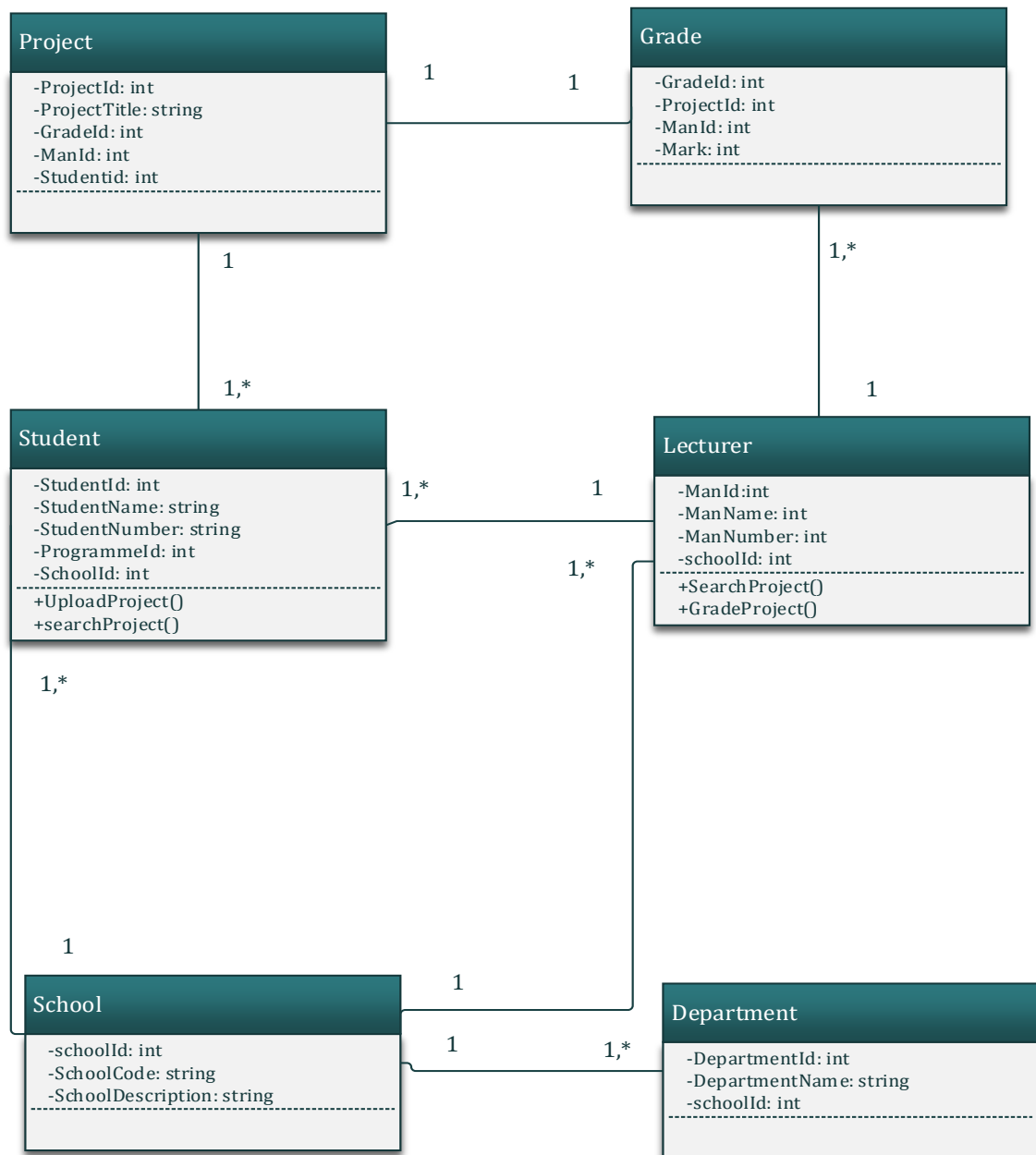


Figure 21: Class Diagrams

CHAPTER 5: ARCHITECTURAL DESIGN

5.0 INTRODUCTION

Architectural design involves the organization of a software system and how the overall design of the system will be. It involves identifying the main structural components of a system and the relationships between them.

5.1 PURPOSE OF CBU FINAL YEAR PROJECT GRADING APPLICATION ARCHITECTURE DESIGN

The purpose of the architecture is to identify the subsystems necessary for this system and the frameworks for the subsystems for the communication and control system to attain comprehensive software architecture for the CBU final year project grading system.

5.2 SYSTEM ARCHITECTURE

The system organization is according to the client-server architecture which will be realized as a 3-tier client server architecture consisting the user interface, the user interface management, and the application support system which is the system database.

5.3 CLIENT SERVER ARCHITECTURE

The client server architecture is a centralized resource system where server holds all the resources, (S.C. Chen, Sneh Gulati, 2003). The server receives numerous performances at its edge for sharing resources to its clients when requested. In this client server architecture, the functionality of the CBU Final Year Grading system is organized into a 3-tier architecture.

Tier 1: User Interface

The top-most layer of the application is the user interface which is a presentation layer. This is the layer that is visible to the end users. HTML and CSS are used to develop this layer. Multiple users will access the application using a web browser to query data and perform various tasks.

Tier 2: Business Logic

This is the application layer and is the middle layer of a web application. This layer supports the application running on a web server and will processes all commands from the end user at the presentation layer, perform calculations, and make all the necessary logical decisions. It will also move and process data between the two surrounding layers.

Tier 3: Data Management

The database is the third and final layer of the web application architecture. It is at this layer where the data will be stored and retrieved whenever necessary, D. Menasce, V. Almeida (2001). The information will then be passed back to the logic tier for processing and eventually, back to the end user at the presentation layer. In the CBU Final Year Project Grading Application, Microsoft SQL will be used for the storing and retrieving of information.

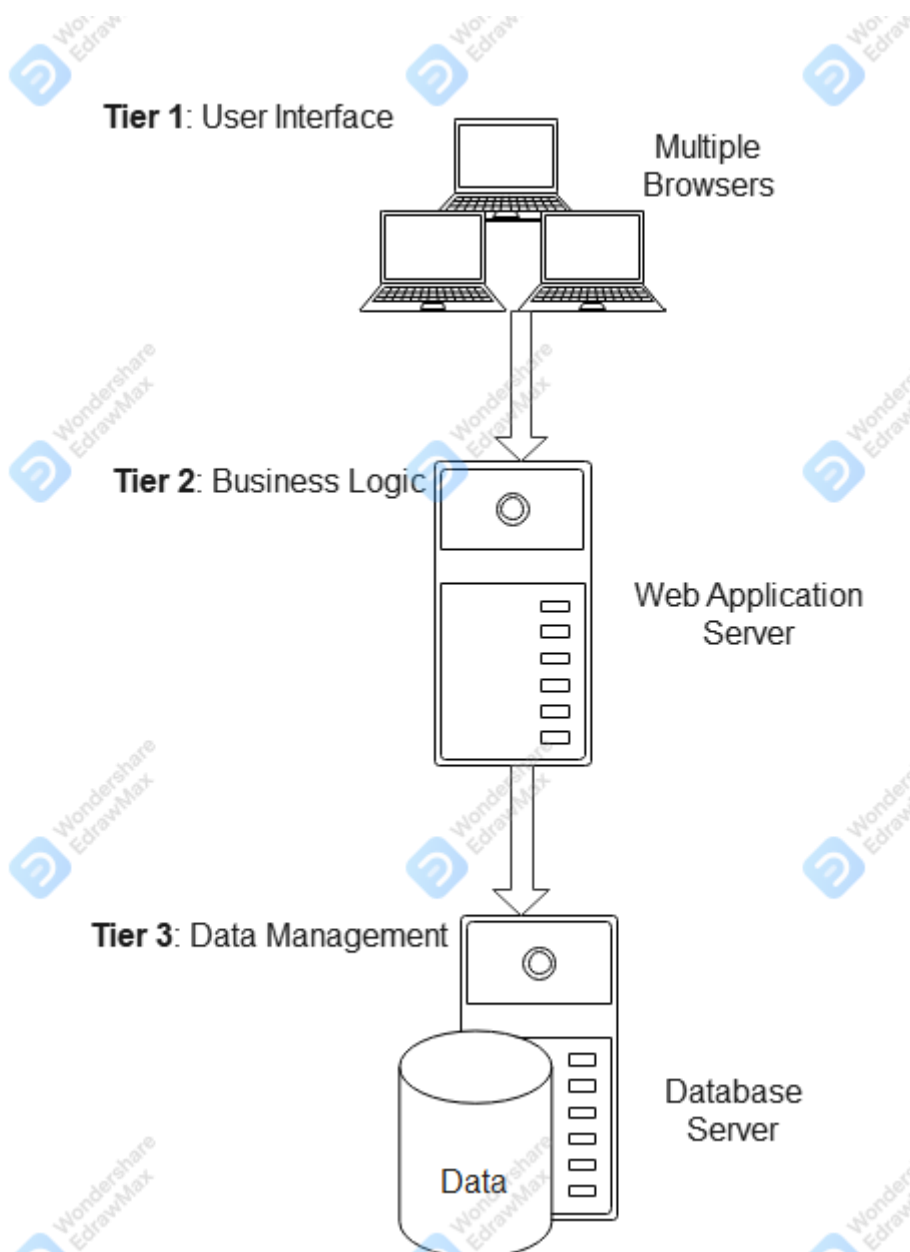


Figure 23: 3 Three Tier Client-Server Architecture

5.4 CBU FINAL YEAR PROJECT GRADING CONTROL MODEL

5.4.1 Lecturer Model

The system controller (application logic) will centrally co-ordinate and control the starting, stopping as well as co-ordination of other system processes which will include the upload process, project recording process, the grading process, and the user interface functionality encompassing the grading form and login functionality.

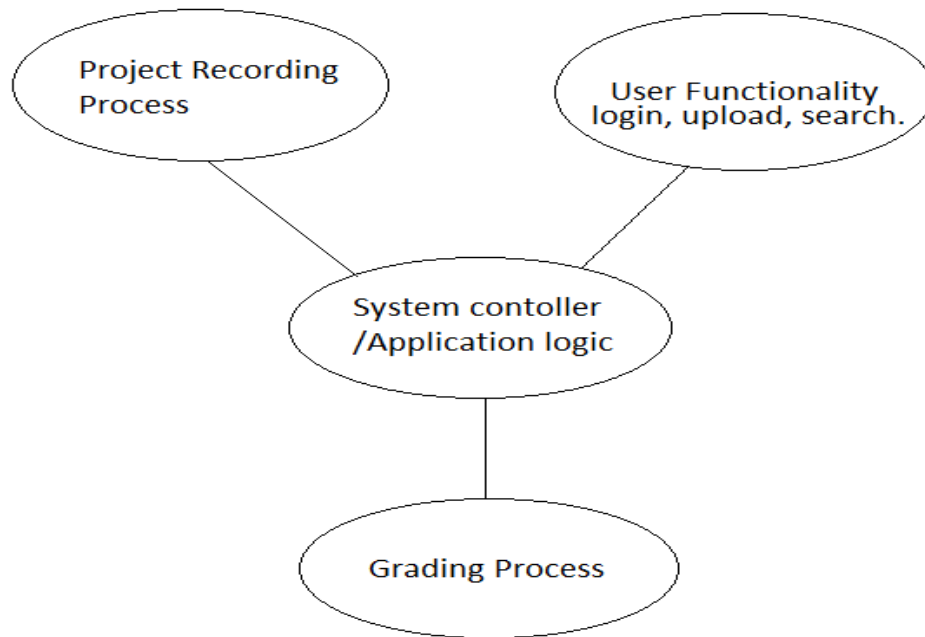


Figure 24: Lecturer control-model

5.4.2 Student Model

The student control-model is similar to the lecturers in that the system controller (application logic) will centrally co-ordinate and control the starting, stopping as well as co-ordination of other system processes, the difference is that students do not have a grading functionality as only lecturers will be able to grade.

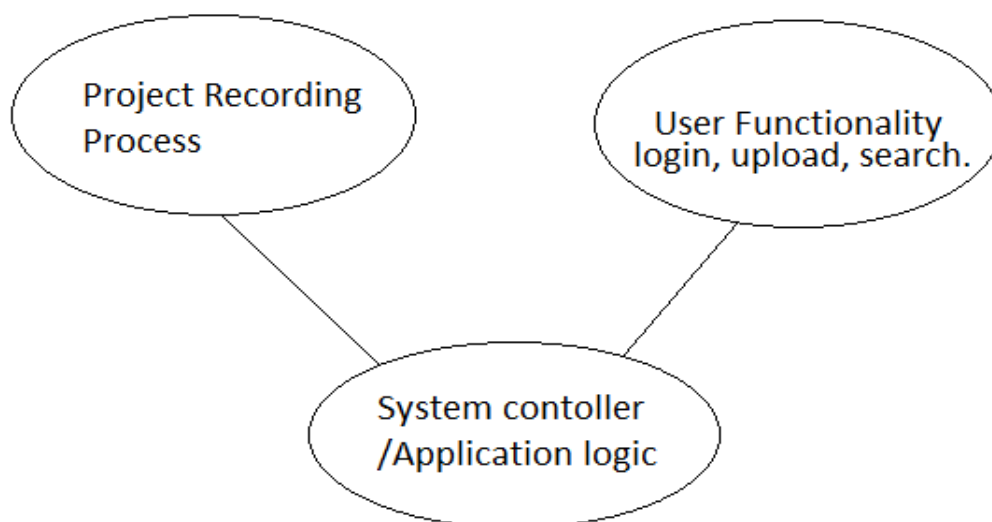


Figure 25: Student Control-Model

CHAPTER 6: IMPLEMENTATION

6.0 INTRODUCTION

This chapter contains the description of the implementation. The implementation stage involves the transformation of the software technical data package (TDP) into one or more fabricated, integrated, and tested software configuration items that are ready for software acceptance testing. Implementation describes the functions and operations that take place to bring the entire system project into reality.

6.1 DEVELOPMENT TOOLS

The Copperbelt University Final Year Project Grading Application was implemented using the following languages, tools, services and technologies:

6.1.1 MICROSOFT VISUAL STUDIO

Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop computer programs, as well as websites, web apps, web services and mobile apps. In Microsoft visual Studio, ASP.NET MVC framework was used.

ASP.NET MVC is a web framework based on Model-View-Controller (MVC) architecture. It was used in our project to build a dynamic web application which enabled a clean separation of concerns as well as fast development. MVC is a design pattern used to decouple user-interface (view), data (model), and application logic (controller). Using the MVC pattern for websites, requests are routed to a Controller that is responsible for working with the Model to perform actions and/or retrieve data. The Controller chooses the View to display and provides it with the Model. The View renders the final page, based on the data in the Model.

The following languages where used;

6.1.2 C#

C# is a general-purpose, multi-paradigm programming language. C# encompasses static typing, strong typing, lexically scoped, imperative, declarative, functional, generic, object-oriented (class-based), and component-oriented programming disciplines (Beasley Robert, 2020). This enabled us to build a dynamic web application using the .NET platform.

6.1.3 HTML

The Hypertext Markup Language, or HTML is the standard markup language for documents designed to be displayed in a web browser. It is the code that enabled us to structure a web page and its content.

6.1.4 JAVASCRIPT

JavaScript is a text-based programming language used both on the client-side and server-side that allows you to make web pages interactive. JavaScript gives web pages interactive elements that engage a user. Incorporating JavaScript enabled us to improve the web app user experience by converting it from a static web app to an interactive one.

6.1.5 CSS

CSS is the language for describing the presentation of Web pages, including colors, layout, and fonts. It allows one to adapt the presentation to different types of devices, such as large screens, small screens, or printers.

6.1.6 jQuery

jQuery is a JavaScript library designed to simplify HTML DOM tree traversal and manipulation, as well as event handling, CSS animation, and Ajax. It is a fast, small, cross-platform and feature-rich JavaScript library. It is designed to simplify the client-side scripting of HTML. While JavaScript allows websites to be interactive and dynamic, jQuery is a tool that helps streamline that process.

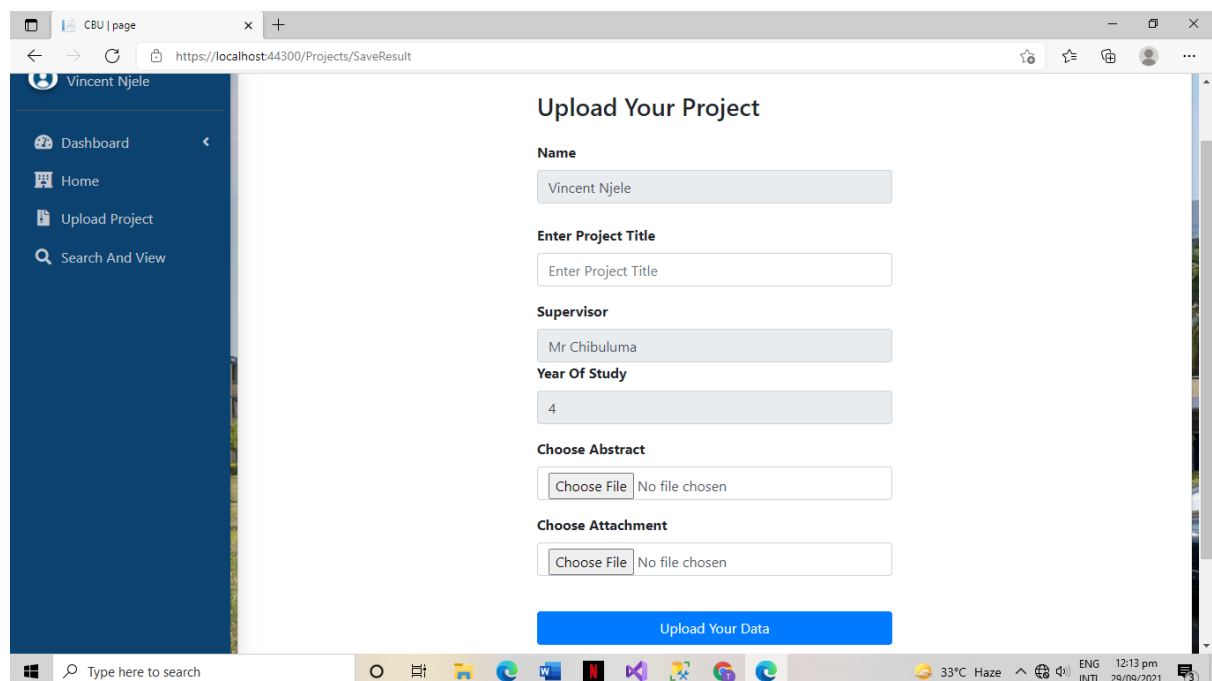
JavaScript is a text-based programming language used both on the client-side and server-side that allows you to make web pages interactive. JavaScript was used to update and change both HTML and CSS as well as calculate, manipulate and validate data.

6.2 PROCESSING OF DATA

This is how the data is processed to make the application work efficiently and effectively.

6.2.1 Submit Project

A student fills in the input fields with the required information, including a pdf of their project. All the information is stored in the database except the pdf. Instead, a project id is created and stored in its place. The code below shows how this is implemented.



The screenshot shows a web browser window with the URL `https://localhost:44300/Projects/SaveResult`. The application has a dark blue sidebar on the left with the user name "Vincent Njele" at the top. The sidebar contains links for "Dashboard", "Home", "Upload Project", and "Search And View". The main content area is titled "Upload Your Project" and contains several form fields: "Name" (filled with "Vincent Njele"), "Enter Project Title" (empty), "Supervisor" (filled with "Mr Chibuluma"), "Year Of Study" (filled with "4"), "Choose Abstract" (with a "Choose File" button and "No file chosen" text), and "Choose Attachment" (with a "Choose File" button and "No file chosen" text). At the bottom of the form is a large blue button labeled "Upload Your Data". The Windows taskbar is visible at the bottom of the screen.

Figure 26: Submit Project.

6.2.2 Search and View Project

Through a filtered search, a user is able to enter a project title into a search field. The value entered is compared to the already submitted projects. If the project title entered matches a project with the same title, the user will be able to click it and view the project pdf. If it doesn't, an error message is shown to the user indicating that no project with that title exists.

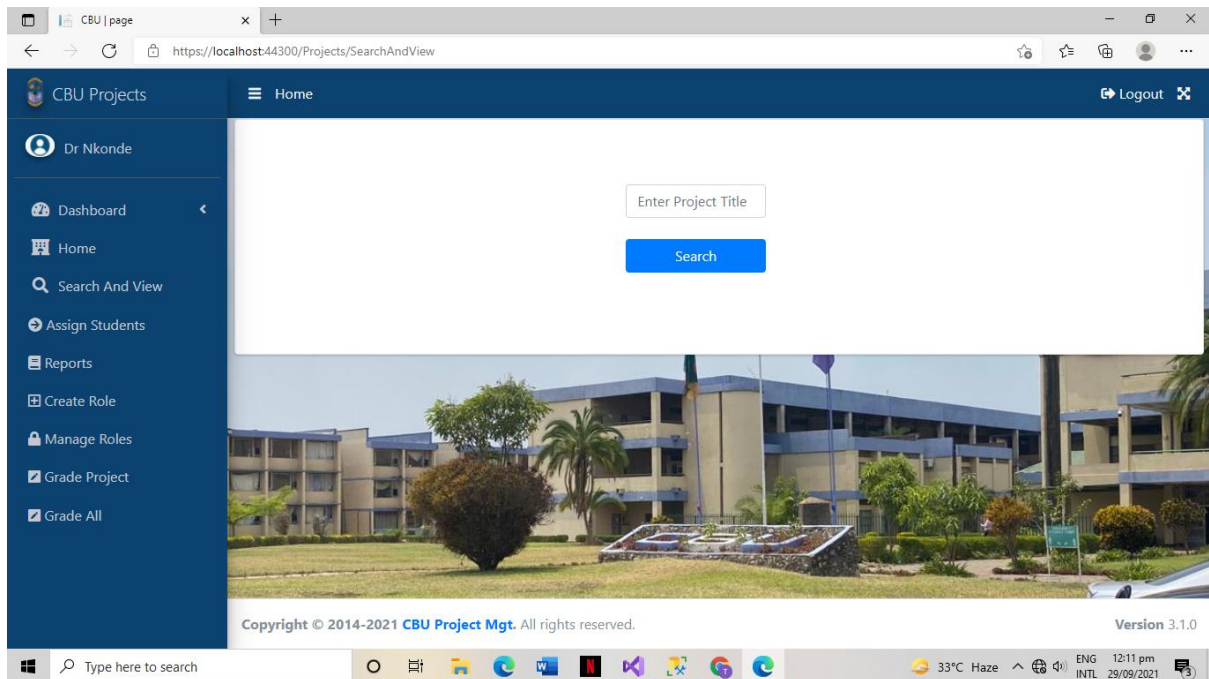


Figure 27: Search and view project

6.2.3 Grade Project

Through a filtered search, a lecturer enters the name of the student they want to grade. Then using the grading sheet provided, they enter the students deserving score according to each category.

The scores entered are summed up as they are being entered to give a final score out of 50 for an individual panelist. The different scores of each panelist are then averaged to give a 'panel mark' which is 50% of the students' final grade. Using the same grading sheet, the supervisor also enters the students mark as the other 50%. The panel mark is then added to the supervisor mark to give a final grade.

Computer Science And Computer Engineering Grading Sheet

Search For Student

	NAME	INTELLECTUAL QUALITY OF PROJECT	KNOWLEDGE OF RESEARCH AREA	QUALITY OF PRESENTATION	RESPONSE TO QUESTIONS	STRUCTURE OF REPORT	REFERENCE AND APPENDICES	TOTAL
	The following scale will help you allocate marks	Excellent 10 Very good 8 good 6 Fair 4 Bad 2	Excellent 10 Very good 8 good 6 Fair 4 Bad 2	Excellent 10 Very good 8 good 6 Fair 4 Bad 2	Excellent 10 Very good 8 good 6 Fair 4 Bad 2	Excellent 5 Very good 4 good 3 Fair 2 Bad 1	Excellent 5 Very good 4 good 3 Fair 2 Bad 1	50
1	Vincent Njele 17110127	<input type="text" value="Pick scc"/>	<input type="text" value="Pick sc"/>	<input type="text" value="Pick scc"/>	<input type="text" value="Pick s"/>	<input type="text" value="Pick s"/>	<input type="text" value="Pick sc"/>	
2	Thandiwe Ngoma 16102461	<input type="text" value="Pick scc"/>	<input type="text" value="Pick sc"/>	<input type="text" value="Pick scc"/>	<input type="text" value="Pick s"/>	<input type="text" value="Pick s"/>	<input type="text" value="Pick sc"/>	

Figure 28: Grading Project.

6.2.4 Generate Report

Based on the individual student grades stored in the database, a pie chart is generated showing the percentage of students in a particular grade range for a specified academic year.

Assign Students

Mr Chibuluma

Mrs Banda

Dr Ntalasha

Dr Nkonde

Figure 29: Assign Students

6.2.6 Managing Roles

Aside from being able to assign students, the coordinator needs to assign and manage the user roles. They should be able to add and delete users to different roles.

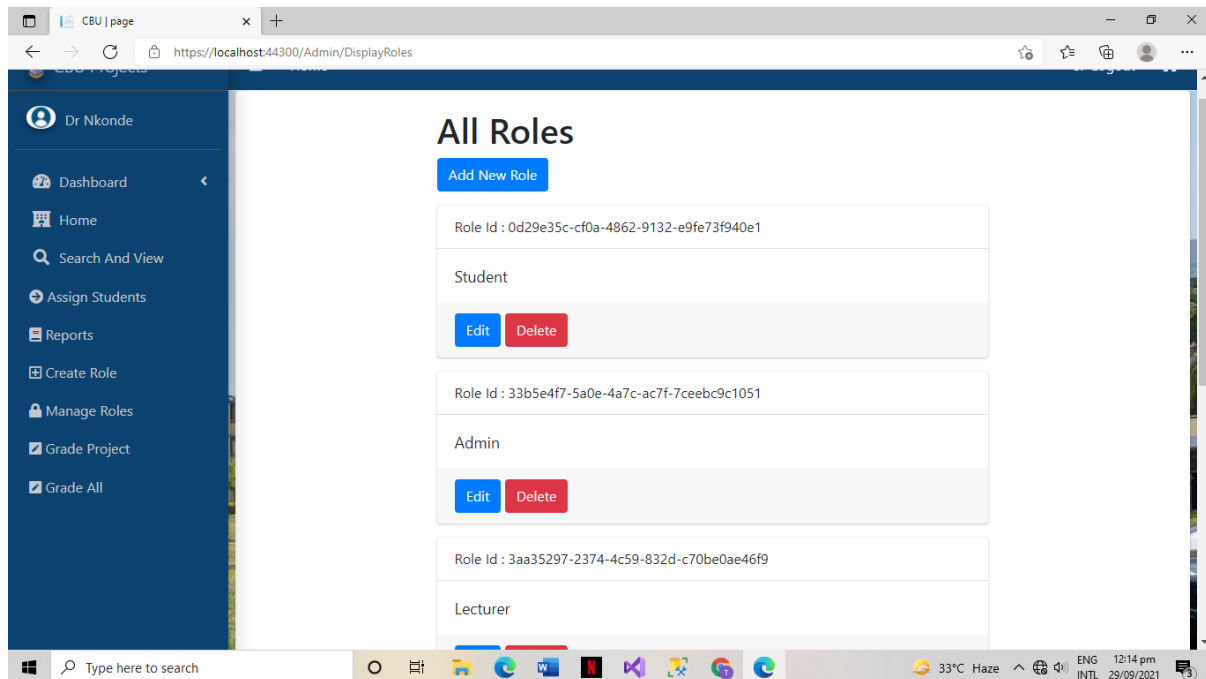


Figure 30: Managing roles.

6.2.6 Grade All

Another role the coordinator has is to grade all student projects. The click of this button generates the student grades by adding the supervisor mark to the panel mark.

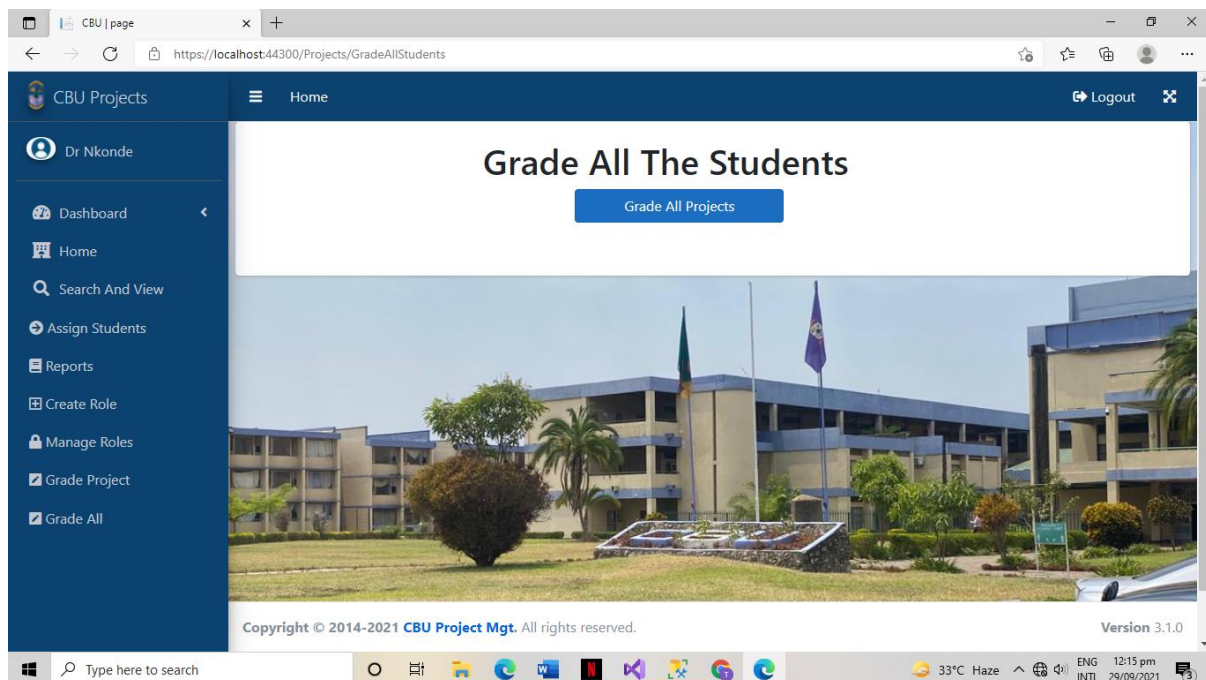


Figure 31: Grade all projects.

CHAPTER 7: SYSTEM TESTING

7.0 Introduction

Testing is done to ensure the development of high-quality software that can easily be adopted by end users. This document describes the various methods used were used to test the developed system. Two main methods of testing were used, namely unit testing and integration testing.

7.1 Unit Testing

Unit testing is a software testing method in which individual units of source code are tested in isolation to determine if they are fit for use. As shown below, tests were carried out on the login module for the users and different tests were carried out to check that the data entered was validated before being stored in the database.

7.2 Unit Testing User Authentication:

TEST CASE	EXPECTED RESULTS	ACTUAL RESULTS
No Password, no username Entered.	Access should be denied. Error message displayed.	Access denied. Error message indicating use of wrong credentials is displayed.
Entering wrong credentials	Access should be denied. Error message displayed.	Access denied. Error message indicating use of wrong credentials is displayed.
Entering Correct credentials	Access should be granted.	Access was granted. User was directed to home page.

Table 7: Testing User Authentication

7.3 Data Validation Testing

Data validation testing is done to ensure that the input fields only accept the specified data type for that field and that all required fields are entered and there are no null values. The following table shows how validation testing was carried out.

TEST CASE	EXPECTED RESULTS	OBSERVED RESULTS
Null value submitted	Error message alerting user to enter the required fields.	Error message alerted user to enter the required fields.
Incorrect value submitted	Error message alerting user to enter the required fields.	Error message alerting user to enter the required fields.
Correct values submitted	System accepts user input and displays successful submission message.	System accepted user input and displayed successful submission message.

Table 8: Data validation Testing.

7.4 Integration Testing

Integration testing is done to ascertain if the combined parts of the application are functioning correctly. The combined modules are tested to check if they are correctly functioning together. The emphasis in integration testing is on checking that modules that worked in isolation can function together. The following table shows how integration testing was conducted.

TEST CASE	EXPECTED RESULTS	OBSERVED RESULTS
Submit project	Pop-up message to show that document has been successfully submitted.	The pop-up message was displayed indicating successful submission.
Generate Grades	All grades should be generated and displayed.	All grades were generated and displayed.
Search Project	Display project.	Project was displayed and ready for viewing.

Table 9: Integration Testing

CHAPTER 8: CONCLUSION AND RECOMMENDATION

8.0 Introduction

This Chapter gives a general overview of the CBU Final Year Project Grading Application and what features may be added in future to enhance the versions.

8.1 Conclusion

The Copperbelt University Final Year Project Grading Application is a web application that records all final year projects and enables both students and lecturers to search and view the projects using the project title. The application enables an independent number of lecturers to view and grade a project according to the university criteria.

The system is designed to use the client-server paradigm. It was implemented using the ASP.NET MVC framework with an SQL database. Guidelines on how to properly use and navigate the system area shown in the user manual.

8.2 Recommendation

Future enhancements to the Copperbelt University Final Year Project Grading Application include;

- Adding a plagiarism checker to ensure that students are submitting original work.
- Adding an algorithm that creates a balance between the project grade and a students' other grades.
- Expanding the application so that it includes all students doing projects instead of the final years alone.
- Expanding the application so that this form of grading is possible for other courses as well.

REFERENCES

1. Purwanto, Sugeng. 2016, *Improving Academic Writing Skills through Online Mode of Task-Based Assignments*, Canadian Center of science and Education.
2. Gustafsson T, 2009. *How to do your research project*, Sage Publications Inc.
3. (Qaseem, F & Zayid, E., 2019, p.33, *The Challenges and Problems faced by students in the early stage of writing research projects in L₂*, University of Bisha, Saudi Arabia (volume 4), Open Access Publishing group.
4. Pressman, Roger, 2010. *Software Engineering: A Practitioner's Approach*. Boston: McGraw Hill, Sage Publications.
5. S, Shylesh, 2017. *SSRN Electronic Journal: A Study of software Development Life Cycle Process Models*, Srinivas Institute of Management Studies.
6. Yesbeck, D. 2011, *Grading Practices: Teachers' Considerations of Academic and Non-Academic Factors* (p.22), Virginia Common wealth University)
7. Abdul Majid Wazwaz, 2007. *Computers and Mathematics with applications*, Elsevier
8. Jin-Tae Park, 2018. *International Journal of Advanced Science and Technology*, IEEE Computer Society
9. Roberson, Quinetta M, (2019). Annual Review of Organizational Psychology and Organizational Behaviour, Villanova University.
10. Kneuper, Ralf, 2017. *Sixty Years of Software Development Life Cycle Models*, IEEE Computer Society.
11. Macan, Therese H., Shahani, Comila, 1990. College students' time management: Correlations with academic performance and stress, Journal of Educational Psychology
12. Meyer, Katrina A, 2014. Student Engagement in online Learning: What works and why, Modern language Journal.
13. Bastian K, Alexandre M. S. Costa, 2007. Computer Applications in Engineering Education, Wiley Periodicals Inc.
14. Elizabeth A, Alecia Y. J, 2014. Qualitative Data Analysis after Coding. Sage Publications.
15. Song X, Sun H, 2019. A Survey of Automatic Generation of source Code comments: Algorithms and Techniques, IEEE Access.
16. Gorman, Michael E, 2002. Turning Students into Professionals: Types of knowledge and ABET Engineering Criteria, Journal of Engineering Education.
17. Emma F. F, Anne H, Michael, G, 2001. The Impact of Status and audio conferencing on business meetings, Academic Press.

18. Beasley Robert E, 2020. Essential ASP.NET Web Forms Development (Full Stack Programming with C#, SQL, Ajax, and JavaScript), Apress.
19. Craig Grannell, 2007. The Essential Guide to CSS and HTML Web Design, Apress Company.
20. Almirall, E., Brito, I., Silisque, A., Cortés, U., 2003: *From Supply Chains to Demand Networks*, pg. 13
21. Z. Pala, and N. Inanc, "Smart Parking Application using RFID Technology", in *RFID Eurasia, 1st Annual*, pp. 1-3, 2007.
22. McLaughlin, B., Pollice, G., West, D. (2006). *Head First Object Oriented Analysis and Design*.
23. Alain Wegmann, (2006). *Systems Sciences*.
24. Silisque. I, A., Cortes. 2003. *Supply Chains to Demand Networks*, pg. 13
25. Nicolas Guelfi; Anthony Savidis (2006). *Rapid integration of software engineering techniques*
26. J. Rumbaugh, I. Jacobson, and G. Booch, (1999). *The Unified Modeling Language Reference Manual*. Addison-Wesley.
27. S.C. Chen, ,Sneh Gulati (2003), *A three-tier system architecture design and development for hurricane occurrence simulation*, IEEE Xplore
28. D. Menasce and V. Almeida (2001.), *Capacity Planning for Web Services: Metrics, Models, and Methods*: Prentice Hall PTR.
29. Jerry Gao; H.-S. J. Tsao; Ye Wu (2003). *Testing and Quality Assurance for Component-based Software*. Artech House. pp. 170-. ISBN 978-1-58053-735-3.
30. Patton, Ron 2005). *Software Testing (2nd ed.)*. Indianapolis: Sams Publishing. ISBN 978-0672327988

APPENDIX A: SAMPLE CODE


```

using CBUProjects.DAL.Models;
using CBUProjects.WEB.Models;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Data.SqlClient;
using System;
using System.Collections.Generic;
using System.Dynamic;
using System.IO;
using System.Linq;
using System.Threading.Tasks;

namespace CBUProjects.WEB.Controllers
{
    [Authorize]
    public class ProjectsController : Controller
    {
        private readonly CbufinalprojectsContext _context;
        private readonly IWebHostEnvironment _environmet;
        public UserManager<IdentityUser> UserManager { get; }

        string connect = $"Data Source=THANDI\\SQLEXPRESS;Initial
Catalog=CBUFinalProjects;Integrated Security=True; MultipleActiveResultSets=True";

        public ProjectsController(CbufinalprojectsContext context, IWebHostEnvironment environment,

            UserManager<IdentityUser> userManager)
        {
            _context = context;

```

```

        _environmet = environment;
        UserManager = userManager;
    }
    public IActionResult Index()
    {
        return View();
    }

```

```

[HttpGet]
public IActionResult SaveResult()

{
    ViewBag.Username = Utilities.GetCurrentLoggedUserDetails(_context, User.Identity.Name);

    ViewBag.Name = _context.Student.Where(v => v.StudentNumber == User.Identity.Name)
        .Select(y => new Student { StudentName = y.StudentName }).FirstOrDefault().StudentName;

    int ManId = Convert.ToInt32(_context.Student.Where(v => v.StudentNumber ==
User.Identity.Name)
        .Select(x => new Student { ManId = x.ManId }).FirstOrDefault().ManId);

    ViewBag.ManName = _context.Lecturer.Where(v => v.ManId ==
ManId).FirstOrDefault().ManName;

    return View();
}

```

```

[HttpPost]

public async Task<IActionResult> SaveResult(ProjectModels model)
{
    ViewBag.Username = Utilities.GetCurrentLoggedUserDetails(_context, User.Identity.Name);

    Project project = new Project();

    ViewBag.Name = _context.Student.Where(v => v.StudentNumber == User.Identity.Name)
        .Select(x => new Student
        {
            StudentName = x.StudentName,
        }).FirstOrDefault().StudentName;

    int ManId = Convert.ToInt32(_context.Student.Where(v => v.StudentNumber ==
User.Identity.Name)

        .Select(x => new Student
        {
            ManId = x.ManId,
        }).FirstOrDefault().ManId);

    ViewBag.ManName = _context.Lecturer.Where(v => v.ManId ==
ManId).FirstOrDefault().ManName;

    int StudentId = _context.Student.Where(v => v.StudentNumber == User.Identity.Name)
        .Select(x => new Student
        {
            StudentId = x.StudentId,
        }).FirstOrDefault().StudentId;

    int ManId2 = Convert.ToInt32(_context.Student.Where(v => v.StudentNumber ==
User.Identity.Name)

```

```

        .Select(x => new Student
        {
            ManId = x.ManId,
        }).FirstOrDefault().ManId);

    ViewBag.ManName = _context.Lecturer.Where(v => v.ManId ==
    ManId).FirstOrDefault().ManName;

    if (_context.Project.Any(x => x.StudentId.Equals(StudentId)))
    {

        ViewData["Message"] = "YOU HAVE ALREADY SUBMITTED..";

    }

    else

    {
        if (model.Attachment != null)
        {
            string attachmentpath = "Projects/Attachments/";
            attachmentpath += Guid.NewGuid().ToString() + "_" + model.Attachment.FileName;
            project.Attachment = "/" + attachmentpath;
            string serverPath = Path.Combine(_environmet.WebRootPath, attachmentpath);
            await model.Attachment.CopyToAsync(new FileStream(serverPath, FileMode.Create));
        }

        if (model.Abstract != null)

```

```

{

    string abstractpath = "Projects/Abstracts/";
    abstractpath += Guid.NewGuid().ToString() + "_" + model.Abstract.FileName;
    project.Abstract = "/" + abstractpath;
    string serverpath = Path.Combine(_environmet.WebRootPath, abstractpath);
    await model.Abstract.CopyToAsync(new FileStream(serverpath, FileMode.Create));
}

project.ManId = ManId;
project.StudentId = StudentId;
project.ProjectTitle = model.ProjectTitle;
project.YearOfStudy = 4;
project.AcademicYear = 2020;
_context.Project.Add(project);
await _context.SaveChangesAsync();

if (_context.SaveChangesAsync().IsCompleted)
{
    ViewData["Message"] = "Successfully Uploaded data to the database";
}

else
{
    ViewData["Message"] = "Data Could Not Be Uploaded";

}

}

```

```

        return View();
    }

    [HttpGet]
    public IActionResult GradingSheet()
    {

        ViewBag.Username = Utilities.GetCurrentLoggedUserDetails(_context, User.Identity.Name);

        try
        {
            var manid = _context.Lecturer.Where(x => x.ManNumber.ToString() ==
User.Identity.Name).Select(v => new Lecturer
            {
                ManId = v.ManId,
            }).FirstOrDefault().ManId;

            ViewBag.Manid = manid;

            var result = _context.Student.Select(x => new Student
            { StudentName = x.StudentName, StudentNumber = x.StudentNumber, StudentId =
x.StudentId }).ToList();

            return View(result);
        }
        catch(Exception)
        {
            throw;
        }
    }

```

```

    }

    [HttpPost]
    public IActionResult GradingSheet(string Name)
    {
        ViewBag.Username = Utilities.GetCurrentLoggedUserDetails(_context, User.Identity.Name);

        var Man = UserManager.GetUserName(HttpContext.User);

        if (Man.Length < 5)
        {
            int ManNumber = Convert.ToInt32(Man);

            int ManId = _context.Lecturer.Where(x => x.ManNumber.ToString() ==
User.Identity.Name).FirstOrDefault().ManId;

            ViewBag.ManId = ManId;
        }
        else
        {
            int ManNumber = Convert.ToInt32(Man);

            int? ManId = Int32.Parse(_context.Student.Where(x => x.StudentNumber.ToString() ==
User.Identity.Name)
                .FirstOrDefault().StudentNumber);

        }
    }

```

```
var list = _context.Student.Where(x => x.StudentName.Contains(Name)).Select(v => new
Student { StudentName = v.StudentName, StudentNumber = v.StudentNumber, StudentId =
v.StudentId }).FirstOrDefault();
```

```
List<Student> mylist = new List<Student>();
```

```
if (list != null)
```

```
{
```

```
    mylist.Add(new Student
```

```
    {
```

```
        StudentName = list.StudentName,
```

```
        StudentNumber = list.StudentNumber,
```

```
        StudentId = list.StudentId,
```

```
    });
```

```
}
```

```
int StudentId = list.StudentId;
```

```
int ProjectId = _context.Project.Where(v => v.StudentId.Equals(StudentId)).Select(v => new
Project { ProjectId = v.ProjectId }).FirstOrDefault().ProjectId;
```

```
ViewBag.ProjectId = ProjectId;
```

```
return View(mylist);
```

```
}
```



```

[HttpPost]
public JsonResult Grade([FromBody] List<GradingModel> ListOfData)
{

    ViewBag.Username = Utilities.GetCurrentLoggedUserDetails(_context, User.Identity.Name);
    List<EvaluationTransaction> transaction = new List<EvaluationTransaction>();
    EvaluationTransaction evaluation = new EvaluationTransaction();

    int CategoryId = 0;
    var studentName = "";

    foreach (var i in ListOfData)
    {
        var student = _context.Project.Where(v => v.ProjectId == i.ProjectId).
            Select(x => new Project
            {
                StudentId = x.StudentId
            }).FirstOrDefault().StudentId;
        studentName = _context.Student.Where(v => v.StudentId == student).
            Select(x => new Student
            {
                StudentName = x.StudentName
            }).FirstOrDefault().StudentName;

        if(_context.EvaluationTransaction.Any(v => v.ProjectId == i.ProjectId && v.ManId ==
i.ManId))
        {

```

```
        return Json("You have already graded " +studentName);  
    }  
}
```

```
        CategoryId = _context.Category.Where(x =>  
x.CategoryName.Equals(i.CategoryName)).Select(v => new Category { CategoryId = v.CategoryId  
}).FirstOrDefault().CategoryId;
```

```
int? studentId = _context.Student.Where(x => x.ManId == i.ManId)  
    .Select(v => new Student { ManId = v.ManId }).FirstOrDefault().ManId;
```

```
int? student2 = _context.Project.Where(x => x.ProjectId == i.ProjectId).  
    Select(v => new Project { ManId = v.ManId }).FirstOrDefault().ManId;
```

```
if (student2 == studentId)  
{  
    evaluation.IsSuperVisor = 1;  
}  
else  
{  
    evaluation.IsSuperVisor = 0;  
}
```

```
evaluation.CategoryId = CategoryId;  
evaluation.ManId = i.ManId;  
evaluation.ProjectId = i.ProjectId;  
evaluation.Status = "Complete";  
evaluation.Score = i.Score;
```

```

transaction.Add(new EvaluationTransaction
{
    IsSuperVisor = evaluation.IsSuperVisor,
    ManId = evaluation.ManId,
    CategoryId = evaluation.CategoryId,
    Score = evaluation.Score,
    Status = evaluation.Status,
    ProjectId = evaluation.ProjectId,

});

foreach (var value in transaction)
{

    _context.EvaluationTransaction.Add(value);

}
}
_context.SaveChanges();

return Json("Successfully Graded");
}

```

[HttpGet]

```

public IActionResult SearchAndView()
{
    ViewBag.Username = Utilities.GetCurrentLoggedUserDetails(_context, User.Identity.Name);
    //ViewBag.Username = Utilities.GetCurrentLoggedUserDetails(_context, User.Identity.Name);
    return View();
}

[HttpPost]
public IActionResult SearchAndView(string ProjectTitle)
{
    ViewBag.Username = Utilities.GetCurrentLoggedUserDetails(_context, User.Identity.Name);

    var data = _context.Project.Where(x => x.ProjectTitle.Contains(ProjectTitle)).Select(v => new
Project { Attachment = v.Attachment, ProjectTitle = v.ProjectTitle }).ToList();

    return PartialView("SearchAndViewPartial", data);
}

public IActionResult AssignStudents()
{
    ViewBag.Username = Utilities.GetCurrentLoggedUserDetails(_context, User.Identity.Name);

    return View(_context.Lecturer.ToList());
}

[HttpGet]
public IActionResult Assign(int ManId)
{
    ViewBag.Username = Utilities.GetCurrentLoggedUserDetails(_context, User.Identity.Name);

```

```

List<AuthenticationModel> authenticationModels = new List<AuthenticationModel>();

try
{
    var authenticate = _context.Student.Where(x => x.ManId == null).
        Select(v => new Student
        {
            StudentId = v.StudentId,
            StudentName = v.StudentName,
        }).ToList();

    foreach (var i in authenticate)
    {
        authenticationModels.Add(new AuthenticationModel
        {
            Id = i.StudentId,
            studentName = i.StudentName,
            IsSelected = false
        });
    }
}

catch (Exception)
{
    return View("NotFound");
}

return View(authenticationModels);
}

```

```

[HttpPost]
public IActionResult Assign(List<AuthenticationModel> check)
{
    ViewBag.Username = Utilities.GetCurrentLoggedUserDetails(_context, User.Identity.Name);

    try
    {
        for (int i = 0; i < check.Count; i++)
        {

            int manId = Convert.ToInt32(check[i].ManId);
            var result = check[i].Id;
            if (check[i].IsSelected == true)
            {

                using (SqlConnection con = new SqlConnection(connect))
                {
                    using (SqlCommand cmd = new SqlCommand($"update Student set ManId = @manId
where StudentId = @result ", con))
                    {
                        cmd.Parameters.Add("@manId", System.Data.SqlDbType.Int).Value = manId;
                        cmd.Parameters.Add("@result", System.Data.SqlDbType.Int).Value = result;
                        con.Open();
                        cmd.ExecuteNonQuery();

                        con.Close();
                    }
                }
            }
        }
    }
}

```

```

        }
    }

}

}

catch (Exception)
{
    return View("NotFound");
}

ViewBag.Message = "Successfully Assgined";

return RedirectToAction("AssignStudents", "Projects");
}

public IActionResult GradeSingleStudent()
{

    return View();
}

[HttpPost]

public IActionResult GradeSingleStudent(int academicYear)
{
    List<EvaluationLookUp> listOfMarks = new List<EvaluationLookUp>();

    EvaluationLookUp lookUp = new EvaluationLookUp();

```

```

int value = 0;

int? manId = 0;

int? studentId = 0;

int? projectId = 0;


var _results = Utilities.ProjectCloseOut(2020, _context);
var res = _results.GroupBy(v => v.ProjectId).ToList();


foreach (var i in res)
{
    foreach (var data in i)

    {
        projectId = data.ProjectId;

        manId = _context.Project.Where(x => x.ProjectId == data.ProjectId).
            Select(y => new Project { ManId = y.ManId }).FirstOrDefault().ManId;


        studentId = _context.Project.Where(x => x.ProjectId == data.ProjectId).
            Select(v => new Project { StudentId = v.StudentId }).FirstOrDefault().StudentId;


        int score = Convert.ToInt32(data.ProjectScore);

        value = value + score;

    }


var programmeAndCourseIddata = _context.Student.Where(x => x.StudentId == studentId).
    Select(v => new Student
    {
        ProgrammId = v.ProgrammId,
        CourseId = v.CourseId,
        YearOfStudy = v.YearOfStudy,

```



```
}).ToList();
```

```
if (value > 85 && value <= 100)
```

```
{  
    lookUp.Grade = "A+";  
}
```

```
else if (value > 75 && value <= 85)
```

```
{  
    lookUp.Grade = "A";  
}
```

```
else if (value > 67 && value <= 75)
```

```
{  
    lookUp.Grade = "B+";  
}
```

```
else if (value > 61 && value <= 67)
```

```
{  
    lookUp.Grade = "B";  
}
```

```
else if (value > 55 && value <= 61)
```

```
{  
    lookUp.Grade = "C+";  
}
```

```
else if (value > 49 && value <= 55)
```

```

{
    lookUp.Grade = "C";
}

else if (value > 39 && value <= 49)
{
    lookUp.Grade = "D+";
}

else if (value >= 0 && value <= 39)
{
    lookUp.Grade = "D";
}
else
{
    lookUp.Grade = "Fail";
}

foreach (var loop in programmeAndCourseIddata)
{
    lookUp.ProgrammId = loop.ProgrammId;
    lookUp.CourseId = loop.CourseId;
    lookUp.AcademicYear = 2020;
    lookUp.ProjectId = projectId;
    lookUp.ManId = manId;
    lookUp.YearOfStudy = 4;
    lookUp.Mark = value;
}

value = 0;

```

```

listOfMarks.Add(new EvaluationLookUp
{
    ProjectId = lookUp.ProjectId,
    CourseId = lookUp.CourseId,
    AcademicYear = lookUp.AcademicYear,
    ProgrammId = lookUp.ProgrammId,
    ManId = lookUp.ManId,
    YearOfStudy = lookUp.YearOfStudy,
    Mark = lookUp.Mark,

});

foreach (var list in listOfMarks)
{
    _context.EvaluationLookUp.Add(list);
}

}

_context.SaveChanges();

var gradelId = _context.EvaluationLookUp.Where(x => x.Mark != 0).
    Select(v => new EvaluationLookUp
    {
        GradelId = v.GradelId
    }).ToList();

foreach (var i in gradelId)
{
    using (SqlConnection con = new SqlConnection(connect))
    {

```

```

        using (SqlCommand cmd = new SqlCommand($"update Project set GradeId= {i.GradeId}
where GradeId=null", con))

        {
            cmd.ExecuteNonQuery();
        }
    }

}

return View();
}

[HttpGet]
public IActionResult GradeAllStudents()
{
    ViewBag.Username = Utilities.GetCurrentLoggedUserDetails(_context, User.Identity.Name);
    return View();
}

[HttpPost]
public JsonResult GradeAllStudents(int academicYear)
{
    ViewBag.Username = Utilities.GetCurrentLoggedUserDetails(_context, User.Identity.Name);
    List<EvaluationLookUp> listOfMarks = new List<EvaluationLookUp>();

    EvaluationLookUp lookUp = new EvaluationLookUp();

    int value = 0;
    int? manId = 0;
    int? studentId = 0;
    int? projectId = 0;

```

```

var _results = Utilities.ProjectCloseOut(2020, _context);
var res = _results.GroupBy(v => v.ProjectId).ToList();

foreach (var i in res)
{
    foreach (var data in i)

    {
        projectId = data.ProjectId;
        manId = _context.Project.Where(x => x.ProjectId == data.ProjectId).
            Select(y => new Project { ManId = y.ManId }).FirstOrDefault().ManId;

        studentId = _context.Project.Where(x => x.ProjectId == data.ProjectId).
            Select(v => new Project { StudentId = v.StudentId }).FirstOrDefault().StudentId;

        int score = Convert.ToInt32(data.ProjectScore);
        value = value + score;

    }

    var programmeAndCourseIddata = _context.Student.Where(x => x.StudentId ==
studentId).
    Select(v => new Student
    {
        ProgrammId = v.ProgrammId,
        CourseId = v.CourseId,
        YearOfStudy = v.YearOfStudy,

```

```
}).ToList();
```

```
if (value > 85 && value <= 100)
```

```
{  
    lookUp.Grade = "A+";  
}
```

```
else if (value > 75 && value <= 85)
```

```
{  
    lookUp.Grade = "A";  
}
```

```
else if (value > 67 && value <= 75)
```

```
{  
    lookUp.Grade = "B+";  
}
```

```
else if (value > 61 && value <= 67)
```

```
{  
    lookUp.Grade = "B";  
}
```

```
else if (value > 55 && value <= 61)
```

```
{  
    lookUp.Grade = "C+";  
}
```

```
else if (value > 49 && value <= 55)
```

```
{
```

```

        lookUp.Grade = "C";
    }

    else if (value > 39 && value <= 49)
    {
        lookUp.Grade = "D+";
    }

    else if (value >= 0 && value <= 39)
    {
        lookUp.Grade = "D";
    }
    else
    {
        lookUp.Grade = "Fail";
    }

    foreach (var loop in programmeAndCourseIddata)
    {
        lookUp.ProgrammId = loop.ProgrammId;
        lookUp.CourseId = loop.CourseId;
        lookUp.AcademicYear = 2020;
        lookUp.ProjectId = projectId;
        lookUp.ManId = manId;
        lookUp.YearOfStudy = 4;
        lookUp.Mark = value;

    }

    value = 0;

```

```

listOfMarks.Add(new EvaluationLookUp
{
    ProjectId = lookUp.ProjectId,
    CourseId = lookUp.CourseId,
    AcademicYear = lookUp.AcademicYear,
    ProgramId = lookUp.ProgramId,
    ManId = lookUp.ManId,
    YearOfStudy = lookUp.YearOfStudy,
    Mark = lookUp.Mark,
    Grade = lookUp.Grade

});

foreach (var list in listOfMarks)
{
    if(!_context.EvaluationLookUp.Any(v=> v.ProjectId==projectId))
    {
        break;
    }
    else
    {
        _context.EvaluationLookUp.Add(list);
    }
}

}

_context.SaveChanges();

var gradeId = _context.EvaluationLookUp.Where(x => x.Mark != 0).
    Select(v => new EvaluationLookUp

```



```

        {
            GradeId = v.GradeId,
            ProjectId = v.ProjectId
        }).ToList();

int gradeIds = 0;
int projectIds = 0;
using (SqlConnection con = new SqlConnection(connect))
{
    foreach (var i in gradeId)
    {
        using (SqlCommand cmd = new SqlCommand("USE [CBUFinalProjects]
UPDATE[dbo].[Project] SET [GradeId] =@gradeIds WHERE ProjectId = @projectIds ", con))
        {
            cmd.Parameters.Add("@gradeIds", System.Data.SqlDbType.Int).Value = i.GradeId;
            cmd.Parameters.Add("@projectIds", System.Data.SqlDbType.Int).Value = i.ProjectId;
            con.Open();
            cmd.ExecuteNonQuery();

        }
    }

    con.Close();
}

return Json("You have successfully graded all the students!!");
}

```

[HttpGet]

```

public IActionResult Reports()
{
    ViewBag.Username = Utilities.GetCurrentLoggedUserDetails(_context, User.Identity.Name);
    return View();
}

[HttpPost]
public IActionResult Result(int academicYear)
{
    ViewBag.Username = Utilities.GetCurrentLoggedUserDetails(_context, User.Identity.Name);

    dynamic model = new ExpandoObject();

    var studentReports = Utilities.ProjectReports(2020, _context);
    var lecturerStudentReports = Utilities.LecturerStudentReports(_context);
    var arrangedLecturerStudentReports = lecturerStudentReports.GroupBy(v => v.ManId);

    List<Reports> reports = new List<Reports>();

    List<LecturerStudents> lecturerStudents = new List<LecturerStudents>();

    foreach (var i in studentReports)
    {
        reports.Add(new Reports
        {
            FailedProjects = i.FailedProjects,
            TotalNumberOfPassed = i.TotalNumberOfPassed,
            TotalProjects = i.TotalProjects

        });
    }
}

```

```

    }

    foreach (var i in arrangedLecturerStudentReports)
    {
        foreach (var result in i)
        {
            lecturerStudents.Add(new LecturerStudents
            {
                LecturerName= result.LecturerName,
                StudentsSuperVised= result.StudentsSuperVised

                });
        }
    }

    model.Reports = reports;
    model.LecturerStudents = lecturerStudents;
    return PartialView("ReportsPartialView", model);
}
}
}

```

APPENDIX B: PROJECT PROPOSAL

2.0 INTRODUCTION

At schools, educational institutes and universities, a project is a research assignment given to a student which requires greater effort and independent work than is normally involved in a normal assignment, Purwanto Sugeng, (2016). It requires students to undertake their own fact-finding and analysis, either from library/internet research or from gathering data empirically, Gustafsson T, (2009). Projects inspire students to challenge themselves and learn problem solving and other skills indispensable to their future, for example time management and critical thinking.

The proposed application is based on making project research more efficient, specifically at the Copperbelt university for both the student undertaking the project, and the lecturer(s) supervising the project. It will challenge and encourage students to think outside the box and be creative as they select and implement a final year project – after which they will be graded accordingly.

1.1 PROBLEM STATEMENT

Recording, grading and storing of final student projects has proven to be hectic, time and resource consuming and takes up much needed space as the thesis books are kept in lecturers office spaces. Sometimes it so happens that students feel they have not been fairly graded for their project work which questions both the student effort and the lecturers grading. students make mistakes in writing proposals because of broad and unclear topics, failure in methodology, Sheila Fram, (2014) terminologies of research, problems in reporting the literature review. Furthermore, challenges faced by students in writing quality research proposals include absence of standard format, lack of knowledge in identifying clearly relevant literature review, lack of good, adequate, and regular feedback from supervisors, lack of materials related to selected topics, and finally the time arranged for writing proposals is not adequate (Qaseem, F & Zayid, E. (2019). In addition, given that majority students pick their own projects, it is no coincidence that sometimes multiple students pick the same project or a pick a project that has been done before which encourages plagiarism. This document proposes a web application that aims to solve all of the problems stated above.

1.2 THE PROPOSED SYSTEM

This document proposes a web application that will record all final year projects with student information to enable both students and lecturers to search and view the projects using either the project title or key words. The application will enable an independent number of lecturers to view and grade it according to the university criteria and automatically sum/average the total grades to obtain the final grade.

1.3 SCOPE OF THE PROJECT

The scope of this project will be focused on developing a web application that will record projects for final year students with the following student information; student name, student number, project title, key words, abstract, supervisor and a pdf copy of the project. The application will enable other students to view the recorded projects and lectures to view and grade students using the school criteria.

1.4 OBJECTIVES

The following are the objectives of the proposed web application:

- To do a baseline study to understand the problem in detail from the relevant stakeholders- that is, the students and lecturers by interviewing both parties to find out the challenges they are facing.
- The application will record all final year projects with student information
- The application will allow other students and lectures to search for and view the recorded projects
- Lectures will be able to asses and grade student projects using the application.
- The application will automatically calculate the final grade of each students' project.
- The application should enable students to search for and upload projects.
- The application will enable the generalization of reports in form of graphs and charts, showing the important statistics.

1.5 BENEFITS

- Projects will be recorded and reserved online which is a much more efficient way of storing information than the physical storing and sharing of thesis books which take up much needed space and sometimes get damaged and lost.
- Students will be able to search for and view previously done projects as well as projects currently being worked on so as to avoid doing projects that have been done before. Or impressively improve a previously done project.
- The cost of recording, documenting and storing projects will be greatly reduced.
- As they work, students will have many references as guidelines on how to go about the documentation as well as the implementation to produce projects that not only meet but exceed the expectations.
- Lecturers will be able to grade final students' projects using the application.
- Plagiarism will be completely eliminated as students will not be able to copy projects that have been done before.
- To the gain of the university, students will refer to and compare the previously done projects to their projects with an aim to do better than was done before which will gradually increase the standard of projects being produced.

1.6 METHODOLOGY

The methodology will be divided into 2; research methodology and System methodology.

1.6.1 RESEARCH METHODOLOGY

4. Baseline Study

This will involve interviewing the relevant stakeholders, that is, a few random lecturers from various schools will be interviewed to find out the major and minor challenges they faced while supervising and grading student project. A randomly selected number of students will also be interviewed to find

out the challenges they faced while working on their projects. They will all be asked about what possible solutions they think would make project work and grading more efficient.

5. Observation

A lot of information will be gathered from basic observation. Observing the challenges that lecturers are going through as they grade projects and observing the challenges students face as they work on their projects. As well as observation from personal experience.

6. Data Analysis

Here, the data taken from the g is analyzed to gain information with an aim of using logical reasoning on the derived knowledge so as to make better, more informed decisions.

1.6.2 SYSTEM METHODOLOGY

2. Strategy

The incremental model from the software development life cycle (SDLC) involves breaking the requirements into standalone modules, S shylesh, (2017) which are sequentially achieved starting with analysis design, implementation, testing and maintenance. After the first increment, a central product is delivered. Depending on customer feedback, a strategy is developed for the next increments, and modifications are made accordingly (Pressman R, (2010)). This process continues, with increments being delivered until the complete product is delivered.

1. Requirement analysis: In the first phase of the incremental model, the product analysis expertise identifies the requirements. And the system functional requirements are understood by the requirement analysis team. To develop the software under the incremental model, this phase performs a crucial role.

2. Design & Development: In this phase of the Incremental model of SDLC, the design of the system functionality and the development method are finished with success. When software develops new practicality, the incremental model uses style and development phase.

3. Testing: In the incremental model, the testing phase checks the performance of each existing function as well as additional functionality. In the testing phase, the various methods are used to test the behavior of each task.

4. Implementation: Implementation phase enables the coding phase of the development system. It involves the final coding that assist in the designing and development phase and tests the functionality in the testing phase. After completion of this phase, the number of the product working is enhanced and upgraded up to the final system product.

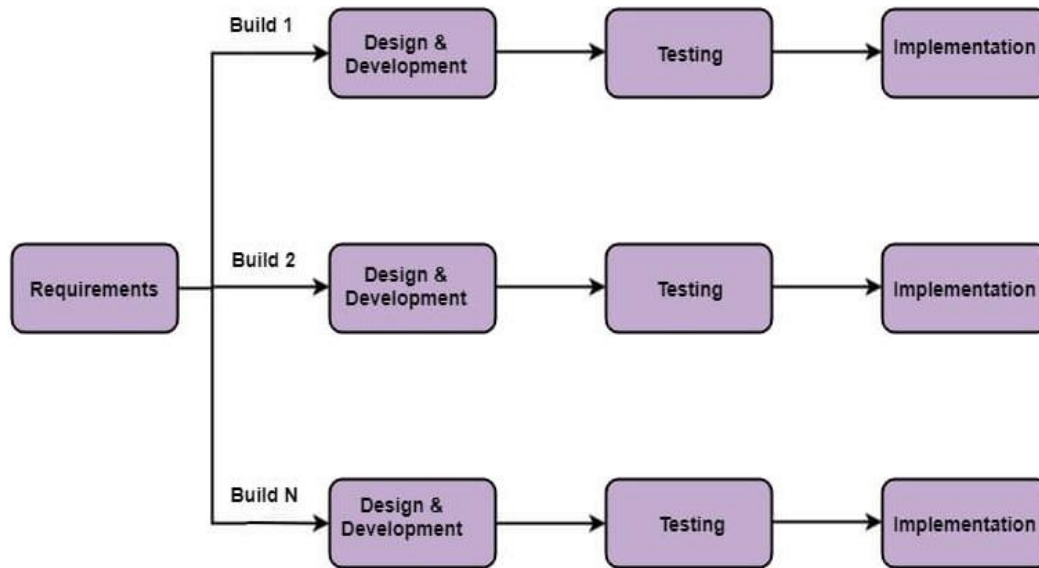


Figure 32: Incremental model, JavaTpoint(2011).

1.6.3 JUSTIFICATION OF THE SELECTED METHOD

The incremental model will be used in the creation of this model because it is a strategy that enables prioritized requirements to be executed first, Kneuper, Ralf, (2017) and is generally easier to test and debug than other methods of software development because relatively smaller changes are made during each iteration. This allows for more selective and rigorous testing of each element within the overall product especially that this is a project with high-risk features and goals. In addition, returning to a previous stage is possible and flexibility to change is easy, making it low risk and cheaper. After each iteration, regression testing will be conducted. During this testing, faulty elements of the software can be quickly recognized because minimal changes are made within any single iteration.

1.7 PROJECT PLAN

MONTHS	FEBRUARY				MARCH				APRIL				MAY				JUNE				JULY			
WEEKS	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Project Proposal																								
Literature Review																								
Requirements Specification																								
Design Specification																								
Coding & Implementation																								
Testing																								
Installation																								
Project Report																								
Handover of product																								

Table 10: Project Plan Gantt Chart

1.8 BUDGET

Below are the proposed costs for the completion of my project:

ITEM	COST(K)
Stationary	100
Data collection and internet facility	500
Printing and Binding	700
Incidental cost	200
TOTAL:	1500

Table 11: Budget

APPENDIX C: USER MANUAL

1.0 System Overview

The Copperbelt University Final Year Project Grading application is a web application that records all final year projects with student information to enable both students and lecturers to search and view the projects using either the project title or key words. The application enables an independent number of lecturers to view and grade it according to the university criteria and automatically sum/average the total grades to obtain the final grade.

2.0 Register

A user who does not have an already existing account is inclined to register so as to gain access to the system.

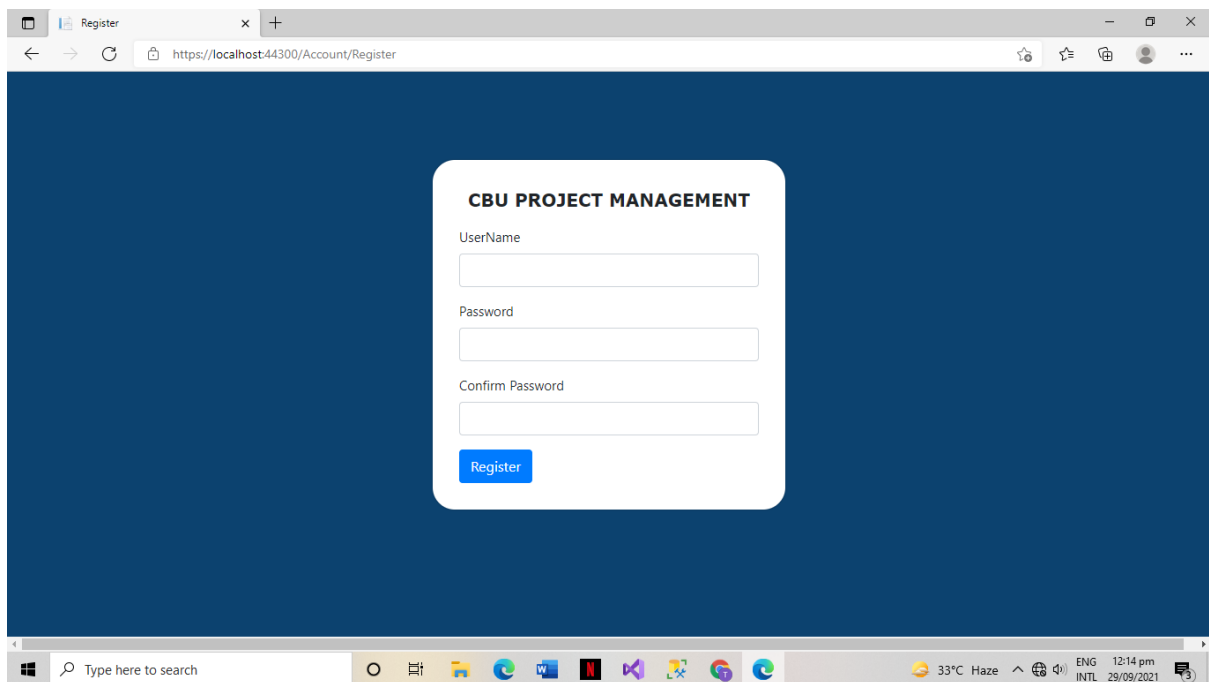
A screenshot of a web browser displaying the registration page for 'CBU PROJECT MANAGEMENT'. The browser's address bar shows 'https://localhost:44300/Account/Register'. The page has a dark blue background. In the center, there is a white rounded rectangle containing the title 'CBU PROJECT MANAGEMENT' in bold. Below the title are three input fields labeled 'UserName', 'Password', and 'Confirm Password'. At the bottom of this rectangle is a blue button labeled 'Register'. The browser's taskbar at the bottom shows various application icons and system information: 33°C Haze, ENG INTL, and 12:14 pm 29/09/2021.

Figure C.1: Register

3.0 login

Using any browser, the student can log into the web application using their correct login credentials.

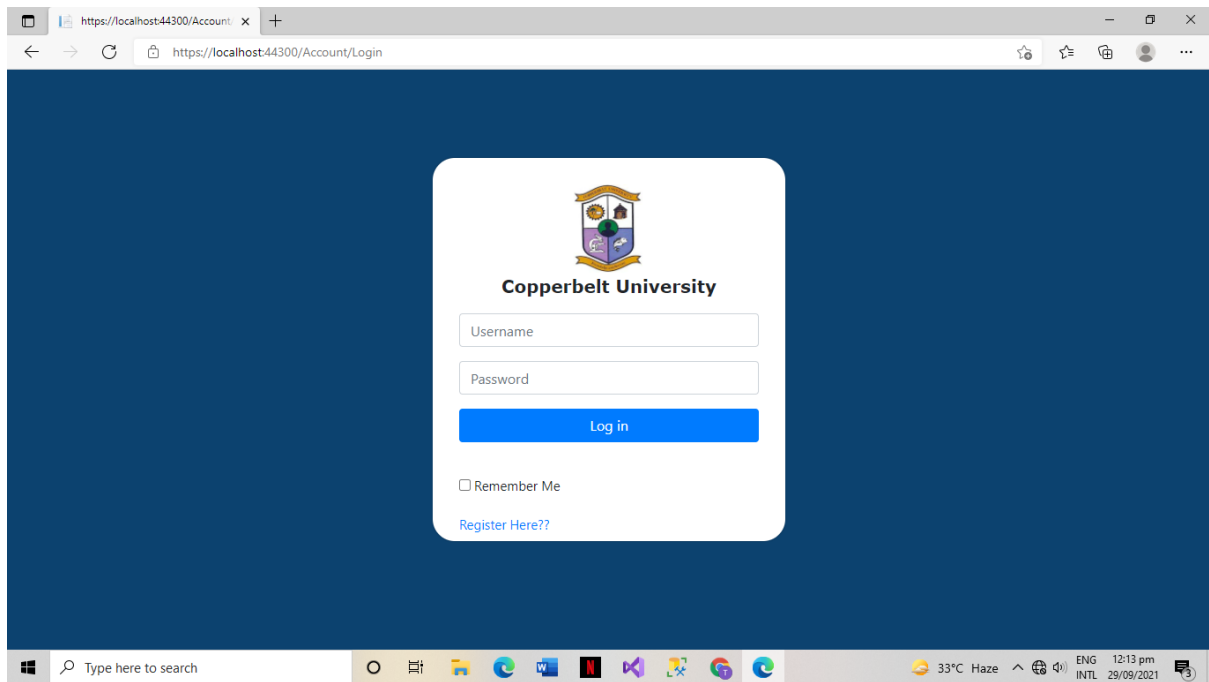


Figure C.2: Login

3.0 Submit Project

To submit a project, a user clicks the submit project option on the side navigation bar. A form pops up where the user is able to fill in the required fields and attach a pdf copy of the project.

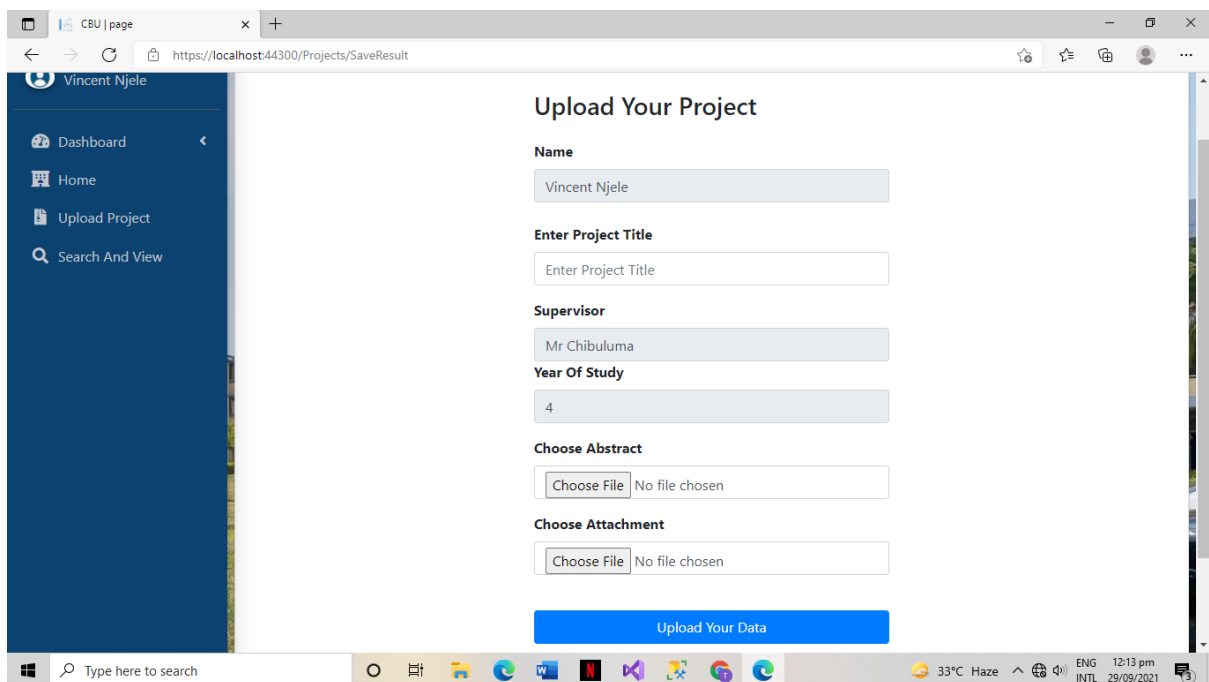


Figure C.3: Submit Project

4.0 Search Project

To search for a project, a user must click the Search option in the side navigation bar. A page with a search bar will pop up where a user can enter a project title. If the project exists, the user can click it to view.

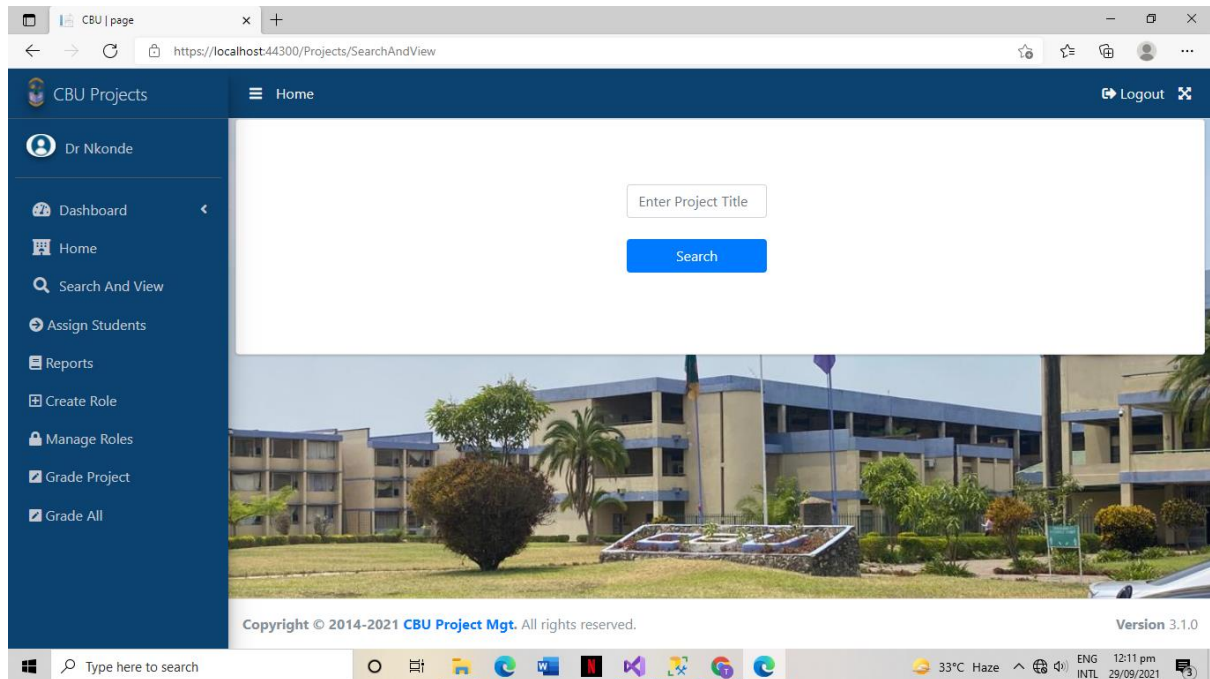


Figure C.4: Search project

5.0 Grade Project

To grade a project, a lecturer clicks the grade project option in the side navigation bar. On the next page, the lecturer must enter the name of the student they want to grade. The grading sheet will be updated to grade only the selected student. The lecturer must then give the student the deserving grade according to each category after which they must click the grade button to send their score.

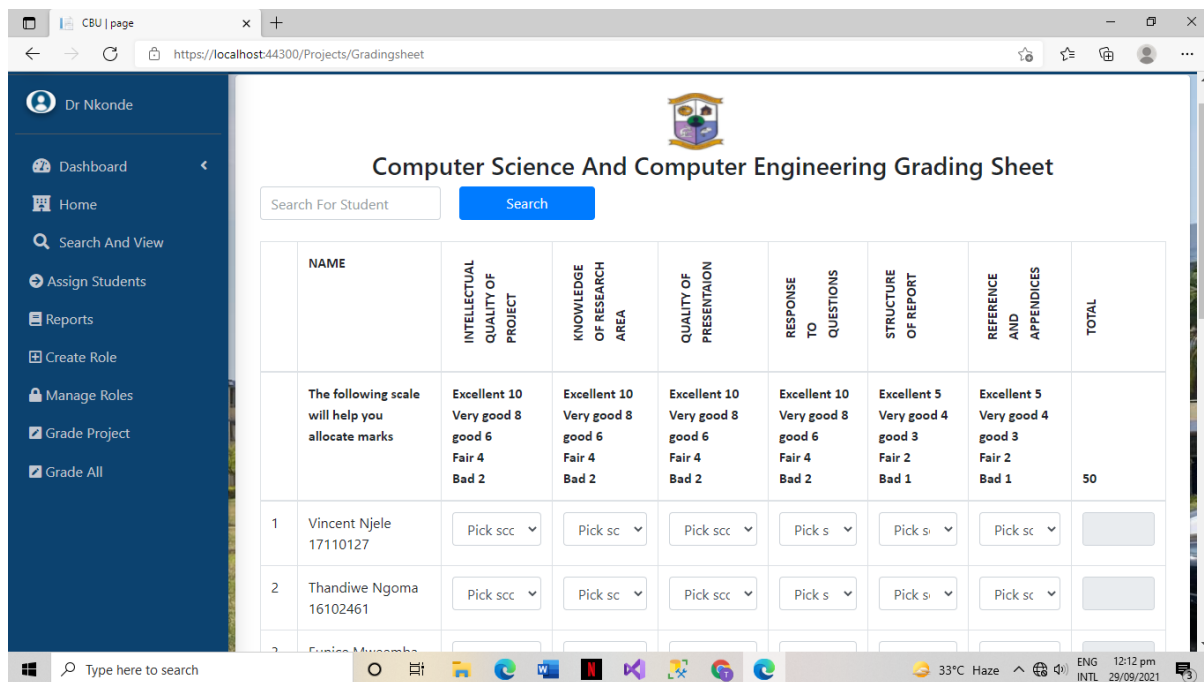


Figure C.5: Grade project

6.0 Generate Report

To Generate a report, the user clicks the Reports option in the side navigation bar. On the next page, the user enters the academic year they wish to view a report for in the provided input field. The report will be displayed.

7.0 Manage Roles

A coordinator can manage roles by creating, deleting and adding users to various roles. To achieve this, click the manage roles option in the side navigation bar. On the following page, the options to create, edit or delete roles is made available.

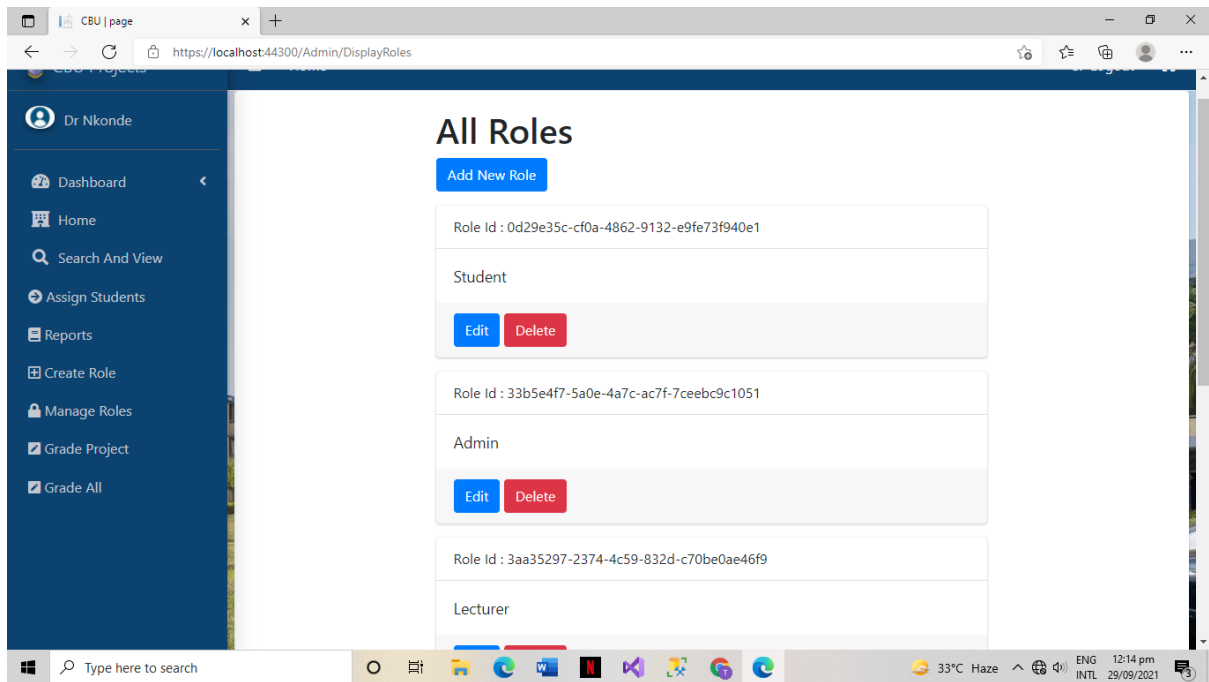


Figure C.7: Manage Roles

8.0 Assign Students

In order to assign students, the coordinator clicks the assign students option in the side navigation bar. A list of available lecturers will appear of which the user must click the one to which he intends to assign a student to.

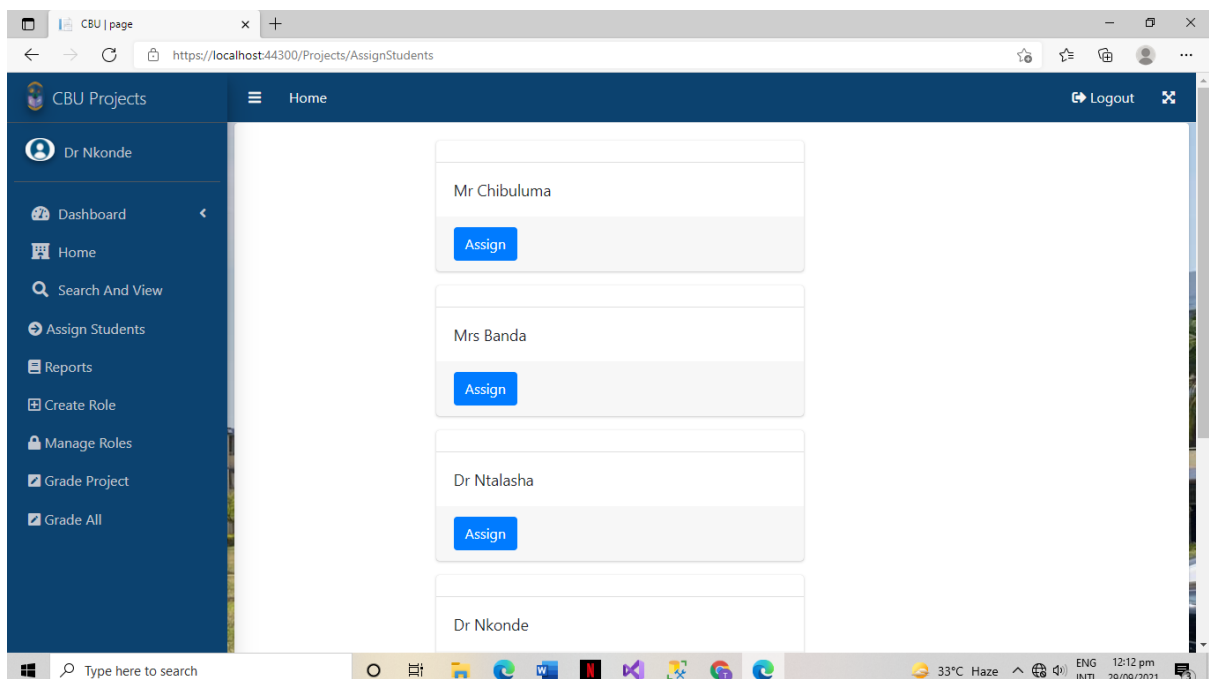


Figure C.8 Assign Students

9.0 Log Out

to log out of the system, the user must click the log out option in the top right corner. The user will be logged out. The log out option is displayed by the red circle in the image below.

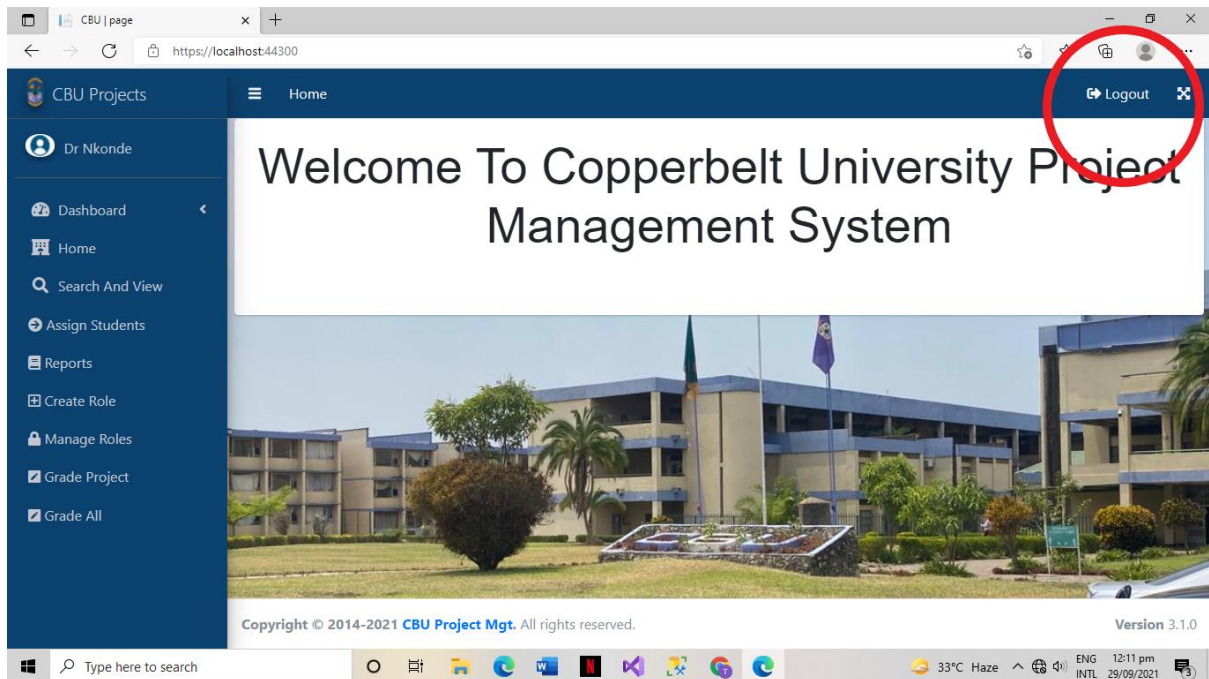


Figure C.9 Log out.