# DEVELOPMENT OF A REPORTING SYSTEM FOR FACILITATING ONLINE ANALYTICS OF STUDENT ACADEMIC RECORDS

**By**

**ADEOTI, FAITH OLAMIDE**

**17010301040**

**A PROJECT SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE AND MATHEMATICS, COLLEGE OF BASIC AND APPLIED SCIENCES, IN PARTIAL FULLFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF DEGREE OF BACHELOR OF SCIENCE IN COMPUTER SCIENCE**

**2021**

# DECLARATION

I hereby declare that this project has been written by me and is a record of my own research work. It has not been presented in any previous application for a higher degree of this or any other University. All citations and sources of information are clearly acknowledged by means of reference.

**ADEOTI, FAITH OLAMIDE**

**Date**

# CERTIFICATION

This is to certify that the content of this project entitled '**Development of a Reporting System for Facilitating Online Analytics of Student Academic Records'** was prepared and submitted by **ADEOTI FAITH OLAMIDE** in partial fulfillment of the requirements for the degree of **BACHELOR OF SCIENCE IN COMPUTER SCIENCE.** The original research work was carried out by him under by supervision and is hereby accepted.

______________________________ (Signature and Date)

Mr. J.A Balogun
Supervisor

______________________________ (Signature and Date)

Dr. M.O. Adewole
Coordinator, Department of Computer Science and Mathematics

# DEDICATION

I dedicate this work firstly to God who has made it possible to have achieved such great feat amidst significant challenges. I also dedicate it to my parents who have put in their all to ensure that I get the best possible in my academic pursuit. Finally, I dedicate this to my biggest support system Olaiya Oreoluwa who was there to give me the encouragement I needed at times when strength failed me. Immense appreciation and recognition also go to Ajibade Benjamin who selflessly sacrificed his convenience to guarantee the completion of this work in record time.

# ACKNOWLEDGEMENTS

All thanks to God almighty, who is the source of my strength, wisdom and in whom I have my being. He alone has mercifully made it possible to take on such meaningful endeavor and deliver it appropriately. To him alone who doeth great wonders, for his mercies endures forever.

I sincerely appreciate the management and staff of Mountain Top University from the Chancellor in person of Dr. D.K Olukoya down to everyone who has contributed to this journey towards achieving success in my academic pursuit. Special recognitions go to Mr. J.A Balogun whom under his supervision and careful guidance I was able to surmount my fears and approach this project with the right mindset. I also recognize Professor, I.O Akinyemi, Mr. I.O Ebo, Dr. Funmilayo Kasali, Mr. O.J Falana, Mrs. Olutosin Taiwo and Late. Dr. M.O Oyetunji who through actions and good deeds have encouraged me and provided assistance to me in times of dire need.

Finally, my highest level of gratitude goes to my parents and siblings who have painstakingly invested so much in me in diverse ways than I can mention. For your financial support, spiritual counsels, moral and emotional support I cannot thank you enough. Only the good lord can reward you but I will definitely make you proud.

# ABSTRACT

This study identified an existing challenge in the area of analysis of student records and the use of reports for decision making process in higher institution. It thereafter developed a computerized web-based online reporting system that can be adopted by academic institutions to assess and review academic performance based on student academic records by identifying the requirements of the system, specifying the design and implementing the designed system.

In order to meet the outlined objectives, a review of literature was carried out to understand the existing systems and their limitations after which informal interviews were conducted with prospective users so as to identify the requirement of the system which was the designed using appropriate unified modelling language (UML) diagrams. Finally, the system was implemented with Microsoft's ASP.NET MVC framework for the frontend, Microsoft SQL Server for the database and C# as the scripting language.

This study at the end of its implementation delivered a web-based reporting system that allows users of three varying roles to access different levels reports of academic performance of student based on their roles dashboards as well as allow course lecturers to create and administer online quizzes to student which results will then be used to generate different new reports on the dashboard.

The study is then concluded with a summary of the result and recommendation of future actions in relation to this study

**Keywords:** Reporting system, Online analytics, Academic records

## TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# CHAPTER ONE

# INTRODUCTION

## 1.1 Background of Study

Student academic records refer to the information that relates to a student's admission and academic performance in a college or a university. These records include the information that is contained in an original transcript, in electronically stored records, and the official student academic record, as it is maintained by the office of the school registrar. An education record contains information directly related to a student; this means that the record is personally identifiable with and is maintained in various types of means including handwritten or printed documents, microfilm/fiche, a computer's main memory, magnetic tape, cassette, and disk or diskette. These records are used to assist offices in support of basic institutional objectives and to document every student's progress and achievement in the academic aspect within the institution.

In the academic field, there has always been the cogent need to harness data and infer meaningful insights through various analytic processes. Due to the diversity and peculiarity of the types of data gotten from the field, there is only so much progress made with the use and adoption of various techniques such as business analytics, educational data mining, academic analytics, predictive analytics, action analytics, learning analytics (LA) or online analytics. Online analytics is also known in some contexts as web analytics poses as the solution to the need for better analysis and consequent use of data by administrators, lecturers, and other stakeholders in academics in an easily accessible and comprehensible manner.

While learning analytics applications typically focus attention on individual courses and learners, there is a growing need and market for institution-wide analytic applications which offer online analytics solutions that allow institutional researchers and other administrators access to data and dashboards that compare student activity and learning metrics within and between courses, departments, and colleges across a university. Doing so helps an institution develop reports concerning student performance with respect to learning outcomes, departmental performance measures, and instructor performance over time.

With online analytics techniques, the instructor no longer needs to wait until the end of the course to download and analyze the data. The instructor also does not have to wait for another employee to run these analyses. The instructor can download the data anywhere he is, analyzes it, draw conclusions, and act on it immediately as learning is happening. This possibility to act on instantaneous data places the instructor in an ideal position in the quest to improve academic staff performance and student learning.

## 1.2 Statement of Problem

Despite the existence of several information systems for compilation and processing of computation of academic records, there are no systems capable of providing the much need reports that are relevant to both educators and the management body in academic institutions from this pool of available data. In the academic field, specifically higher institutions of learning there is a great divide between the sources of data generation and collation which majorly consists of the student's academic record, student's personal details, the lecturer's course content among many other things, and the stakeholders in this field. This divide is a result of the non-existence of a unified and holistic data collation, analysis, and reporting system which will collate the various data and analyze it using statistical or inferential methods and generate a timely and comprehensive report that can be adopted by lecturers, examination officers, management bodies and educational agencies at large to make new policies or adjust existing ones to achieve the primary goal of improving academic performance.

This study evolved from the initial investigation into the existing systems used to report student academic records and subsequently the decision-making infrastructure or process in universities. Generally, these universities just take a few samples of the academic records in this case test or examination results, and make assumptions upon which decisions to adopt new curriculum and knowledge delivery techniques or continue with existing ones are taken. This method is vague and does not represent an effective means of deriving knowledge from data. Identification of this need and the drawbacks of the existing system leads to the development of a web-based system that will easily collate all academic records and perform appropriate statistical analysis to generate reports displayed on user-friendly interfaces. The proposed system can easily be integrated with existing learning management systems (LMS) with permission to use the academic records provided by the LMS.

### 1.3 Aim and Objectives

This study aims to develop a computerized information system that can be used by academic institutions, management, and staff alike to assess and review the performance based on student's academic records. This study uses descriptive statistical methods to analyse student academic record with focus on performance records of individual students and groups of students. The objectives of the study are to;

i. identify the requirements of the system

ii. specify the design of the system

iii. implement the system

## 1.4 Proposed Methodology

In order to meet up with the aforementioned objectives of the study, the following methods will be adopted:

a. A review of pieces of literature will be done to identify and understand existing systems

b. The user and system requirements of the system will be identified from the system users via informal interview

c. The system design will be specified using the unified modelling language (UML) diagrams such as Use case, Sequence diagram, Class diagram, etc.

d. The frontend will be implemented using ASP.NET (MVC) framework with the razor syntax and C# as the scripting language for the application logic

e. The database will be implemented using relational database technology via Microsoft SQL Server.

## 1.5 Significance of Study

The review of existing works in the area of academic record processing shows that there is existing reporting system that make use of traditional reporting methods especially table but cannot be used to facilitate the decision-making process in academic institutions efficiently. Hence, this study is significant in several ways which include:

a. It will improve the decision-making process in academic institutions by providing useful information derived from the analysis of student academic records and presented using modern reporting techniques

b. It will spur responsiveness to pertinent issues that are recognized in academic record patterns from relevant stakeholders and concerned individuals.

c. It will increase the efficiency of the delivery of learning services within academic institutions by allowing the measurement of the lecturer's performance based on the performance of his student in each course.

## 1.6 Scope and Limitation

This study aims to develop a web-based application and is primarily focused on the generation of reports through statistical analysis of student's academic records of students in the Department of Computer Science and Mathematics, Mountain Top University. Specifically, this study focuses on the development of the proposed system that covers only the following modules: Module for Assessment, Module for Manual grade submission, Module for report generation and extraction.

Due to the time available for this study, it will only implement a test case for the academic record of students in the aforementioned department which validates the possibility of application of the system across other departments in the institution.

## 1.7 Definition of Terms

**Lecturer:** a lecturer is a member of the academic staff who is responsible for authoring unit content, including lectures and workshops, presenting the weekly lectures and some of the workshops (most units have multiple workshops due to class sizes).

**Unit:** a unit is a subject that is taught over the period of a semester, with a semester encompassing 12-13 teaching weeks normally followed by an examination.

**LMS:** LMS is an acronym for Learning Management System which is a software application for administration, documentation, tracking, reporting, automation, and delivery of educational courses, training programs, or learning and development programs

**ASP.NET:** .NET is a developer platform made up of tools, programming languages, and libraries for building many different types of applications. ASP.NET extends the .NET developer platform with tools and libraries specifically for building web apps.

**MVC:** The Model-View-Controller (MVC) is a software architectural design pattern that separates an application into three main interconnected logical components: the model, the view, and the controller.

**UML:** UML is an acronym for Unified Modelling Language which is a standardized modelling language consisting of integrated sets of diagrams for specifying, visualizing, constructing, and documenting the structure and design of a system.

**SQL:** Structured Query language is a standard domain-specific language used in programming and is mainly designed for managing data held in a relational database management system.

# CHAPTER TWO

# LITERATURE REVIEW

## 2.1 Information Systems

According to Rosca, Banica and Mirela (2010), Information systems are deployed within an organization with the goal of increasing the effectiveness and efficiency of that organization," The amount to which that objective is realized is determined by the capabilities of the information system, as well as the features of the organization, its work processes, its people, and its development and implementation methodologies.

An information system is everything within an organization that contribute to the flow and management of information primarily consisting of modes that hold information, channels that distribute information and actors that continuously act upon the information. The actors in which case are human are parts of the information rather than users of the information system. Hersh (2002) characterized an information system as a human activity system that is a summarized function of an organization, it's people and its' technological assets.

Information systems can be broadly defined in terms of their structure as a cohesive system consisting of a collection of people, processes, data, models and technology that is used to meet a specific organization function and in terms of their function as a technological means developed for the purpose of capturing, storing and distributing "linguistic expressions" and also provide the necessary support for inference derivation. All information system performs three major activities that provides organizations with the information necessary for informed decision making namely; Input, Processing and Output. The input activity takes data in various formats from within the organization and its external environment, which the process activity then turns into meaningful information based on the organization's rules as well as current conditions and limitations. In an information system, the output activity takes the meaningful information generated from processing and gives it to the users as feedback in the form of textual, graphical, or multimedia reports.

Information systems are defined by Buckingham, Hirschheim, Land and Tully (1987) as a system that collates, stores, processes, and retrieves information relevant to an organization (or society) in a manner that the information is accessible and relevant to those who wish to use it, including managers, staff, clients, and citizens. A human activity (social) system that may or may not include computer systems is called an information system.

Information system was also described as a collection of people, machines, and/or methods for gathering, processing, transmitting, and disseminating data, whether automated or manual. Data is acquired, stored, manipulated, managed, displayed, sent, or received using information systems. Both hardware and software are included. An information system is a work system that is dedicated to processing information, such as recording, sending, storing, retrieving, modifying, and presenting data.

Behavioral science and design science are two concepts that describe most of the study in the IS field. The behavioral science paradigm aims to create and test ideas that may be used to explain or predict human or organizational behavior. By creating new and innovative artifacts, the design science paradigm aims to push the boundaries of human and organizational capabilities. (Henver,March,Park,and Ram, 2004) Natural science research methods are at the heart of the behavioral science paradigm. Its goal is to create and validate ideas that explain or anticipate organizational and human events. The design science paradigm has its origins in engineering and artificial intelligence science. It is basically a problem-solving paradigm that aims to develop innovations that define the concepts, processes, technological capabilities, and products that may be used to effectively and efficiently analyze, design, execute, manage, and use information systems. (March and Smith 1995).

All these definitions present a conceptual point-of-view of an information system with reference to the human and behavioural components but with the popularity of computer systems came computer-based solutions to replace information systems that are completely human driven. Monika and Anju (2013) in their work titled "Information Systems and Software Development Life Cycle" categorized information systems into Computer information systems (CIS) and Business information systems (BIS) which collectively merge their peculiarities to develop the computer-based information system (CBIS). This computer-based information system (CBIS) attempt to eliminate or at least reduce the dependency of existing technologies on human input allowing them to perform most of the intended tasks without human intervention.

A computer-based information system is one that employs computer technology to fulfil part or all of its intended tasks. Information System as in narrow sense, is the specific application software used to record in computer systems and automates some information-processing operations of the organization. It refers to a data-processing system that is used for data records. CBIS (Computer Based Information System) encompasses the following types of information and support systems at various management levels. Data is processed using human efforts in an information system, which is a type of information and communication technology (ICT).

### 2.1.1 Concept of information management

Information management (IM) is the collection and management of information from one or more sources and the distribution of that information to those who have right to it (Robertson, 2005). Henczel, (2000) and Ravi, (2011) define information management as the systematic, imaginative, and responsible management of information in order to create and use information that contributes strategically to the achievement of an organization's goals and ensures that groups and individuals have efficient access to and make effective use of the information they need to do their work and to achieve their goals.

Technically, information management encompasses all systems and processes that support information management programs, such as web content management, document management, records management, digital asset management, learning management systems, and enterprise search (the technical infrastructure) (Reddy, Srinivasu, Rikkula, and Rao, 2009).

Robertson (2005) defines information management as the organizational, social, cultural, and strategic elements that must be considered in order to enhance information in organizations from a management standpoint. The importance of managerial and technical duties in any effective information management plan is highlighted by this. Earl (1989) divides the history of information management into two periods: the conventional era and the technology era. The traditional era encompasses the time when information was managed manually, with the use of human minds and hands, cabinets, papers, and pens and pencils. The technological era, on the other hand, is the time when information technology is incorporated into the administration of information programs in order to address some of the inherent issues with the manual method.

Kahraman and Cevilecan (2011) conducted a research in Turkey on intelligence decision systems in corporate information management. Intelligence approach was recognized as a novel tool for information management in the study. Intelligence techniques are characterized as systems that aid decision-making by gathering, analyzing, and diagnosing issues, as well as suggesting and assessing probable causes of actions. The research stressed the need of integrating cross-functional strategies for efficient information management, and that investment in information management should be led by both intelligence approaches and business strategy and needs.

The importance of the link between information as a resource and organizational success cannot be overstated. Esterhuizen, Schutte, and Du Toit (2012) investigated the influence of an information management framework on innovation capability. The research chose five industry and subject theory experts to assess the framework's applicability and appropriateness. It was discovered that organizations may utilize information/knowledge management technologies to foster innovation and growth, which can lead to increased organizational performance.

### 2.1.2 History of information systems

According to Gil, Dario and Raul (2010), Since the introduction of the first computer, information systems have been used in businesses as a vital and powerful instrument for optimizing and improving managerial functions. Herman Hollerith's census tabulator was the first large-scale mechanical information system. Hollerith's machine, which was developed in time to handle the 1890 U.S. census, was a major step forward in automation as well as an inspiration for the development of computerized information systems. (Zwass, 2020)

At the advent of information systems, they were merely reckoned for use in electronic data processing (EDP) powering simple data processing activities such as record-keeping, accounting and transaction processing. They were then referred to as: Automatic data processing system, transactions processing system or information processing system among many other things. In the 1960s, the addition of the role of data processing into useful report was introduced to computers, thereby raising the need to develop business application that will utilize this new role and provide managerial end user with a set of predefined reports from the available data resources and this marked the birth of Management Information System (MIS). This MIS soon became insufficient to support the decision-making need of organizations with it pre-specified management reports such as: sales analysis, cost and production trend reporting systems and gave birth to the concept of decision support systems.

Decision Support System (DSS) came into use in the 1970s providing interactive support of their decision-making processes through display of information such as profitability prediction charts, risk analysis, product pricing etc. The introduction of micro-computers and the increase in computing resources needed by organizations and their personnel called for the development of a system that could provide direct support through a decentralized information service. Executive Information Systems (EIS) came into to meet the aforementioned need by harnessing critical information from MIS reports and DSS analytical models specifically designed to meet the information need of organizational executives. The EIS was soon improved on following breakthroughs in artificial intelligence techniques in business allowing most information systems to be able to perform their primary tasks with little or no human intervention. This new breed of knowledge-based information systems was called Expert Systems and were adopted by many organizations as consultants in limited subject areas.

### 2.1.3 Types of information systems

a. **Transaction Processing Systems (TPS)**

TPS is a system that gathers and controls information about transactions. This is a computerized system that processes and records a company's everyday transactions. Billing systems, payroll systems, manufacturing and purchasing systems, stock management systems, and other regular operations are all processed swiftly and correctly. TPS are made up of four primary components: inputs in the form of transactions and events, processing activities such as filtering, indexing, merging, and updating, and outputs in the form of reports as well as users who are operational staff of the system. The primary responsibility of a transaction processing system is to process data created by transactions, maintain data correctness, and ensure timely delivery of documents and reports.

b. **Management Information Systems (MIS)**

MIS is concerned with the information required to manage various organizational operations. It is the most robust information system for managing organizational resources such as people, technology, and information. Management information systems, according to O'Brien (2003), involves three key resources: people, technology, and information or decision making. Management information systems differ from other types of information systems in that they are used to assess the organization's operational activities.

c. **Office Automation Systems (OAS)**

Office Automation systems are computer systems that are used to generate, collect, store, and alter office data that is required to complete tasks such as Archiving of raw data, electronic information exchange, and digital information organization, among other things. OAS streamlines office processes, improves communication at all levels, and boosts productivity. OAS assists anybody in efficiently preserving personal records utilizing basic computer-based tools such as spreadsheet applications, text and picture processing systems, database systems, and so on. Electronic document management systems, teleconferencing and videoconferencing systems, text processors, are common examples of OAS.

### d. **Expert Systems (ES)**

An expert system is a computer method or software that is used to simulate calculations and heuristics by gathering information from human experts for decision making. "An expert system has a distinctive structure, different from standard programs," (Stella and Chuks, 2011) It is separated into two parts: an inference engine, which is fixed and independent of the expert system, and the knowledge base, which is changeable. To run an expert system, the engine thinks like a person about the knowledge base. The knowledge base stores information on facts and rules used by the expert system and the inference engine being the process by which the system draws conclusions based on the rules and facts that apply to various problem domains.

### e. **Decision Support Systems (DSS)**

DSS is a computer-based information system that supports decision-making activities in an organization. It is an interactive and agile decision-making tool that is best utilized at the tactical and strategic levels of an organization and has a low frequency but great potential impact. DSS is made up of a data management database, data management models, and a user interface. "The idea of decision support has emerged from two primary areas of research: theoretical studies of organizational decision making and technological work on interactive computer systems," according to Keen and Morton (1978).

Decision support systems are meant for use by a single manager or, more commonly, a group of managers at any organizational level in the process of making a quasi-structured decision (Asemi, Safari and Zavareh, 2011). "An interactive, flexible, and adaptive computer-based information system (CBIS) particularly created for assisting the solution of a non-structured management challenge for enhanced decision making," Turban and Aronson describe DSS. Decision-support systems, according to Khanore, Patil and Dand (2011), are especially intended to assist management in making decisions in situations when the probable results of such decisions are unknown. It aids an organization's mid- and high-level management by analyzing large amounts of unstructured data and compiling information into thorough reports that may aid problem solving and decision-making.

The components of a decision support system are: Hardware resources, Software modules and application packages, Data resources which is basically a database containing all data and information used by the system, Model resources including mathematical and analytical techniques in form of programs and subroutines and User interface which includes all the tools that help the user of the DSS to navigate through the system.

i. Alter (1980) identified three major characteristics of DSS as

ii. It is designed specifically to facilitate decision processes,

iii. It supports rather than automate decision making, and

iv. It should be able to respond quickly to the changing needs of decision makers.

## 2.2 Student Information Systems

According to Evangelista (2008), A Student Information System (SIS) is a secure, web-based interactive computer system that allows users to provide and view information such as grade reports, transcripts, class schedules, and remaining semester balances, as well as allow students to register for classes online by issuing students unique identifying number through the system which would be used for all data sent and received by the university.

According to Wang and Strong (1996), a School Information System (SIS) is a web-based application software developed to create a conducive and organized information dissemination environment for integrating students, parents, instructors, and school or college administration. Desousa (2008) emphasized the use of web-based application for student information systems, highlighting four major benefits that it has namely: Compatibility, Efficiency, Security of live data and Cost Effectiveness.

Student information systems, according to Benguet State University Online-Student Information System (2013), are a novel technique of record management and transaction processing that will improve the efficiency of processing student information. Administrative employees, academic professionals, stakeholders, and students would benefit greatly from it in terms of updating, retrieving, and creating student data.

Tertiary institutions use school information systems to collect and organize all student academic data in order to provide useful information that influences decision-making. Student information systems are frequently used to assist administrators in making decisions. To improve quality, administrators in tertiary institutions think that various data on student performance and enrolment should be included in student information systems.

### 2.2.1 Concept of student

A student as defined by Nandutu (2016). is “an individual who is registered for university credit course or program.” A student record/data contains information directly related to a student, which means that the record is personally identifiable. Student name, student ID, student address, parent/family member names, and a list of personal traits are all examples of personal identifiers that may be used to link a record to a student.

A student is any person who is enrolled in any educational institution with the primary goals of acquiring knowledge, developing skills or profession and achieving employment through continuous learning.

Handwriting, print, microfilm/fiche, computer main memory, magnetic tape, cassette, disk, or diskette might all be used to keep track of students. Student records and data may be provided by the student, submitted on their behalf, or generated by the university.

### 2.2.2 Academic records

Ap-azli, Safawi, Mohd Razilan, Mohd and Mohd (2016) defines academic records as data or information relating and concerning a student either in paper or electronic formats that presents evidence of events and actions that has occurred during the period of the students’ academic engagement

There are different types of academic records based on the interested party, office or the context in which it is discussed. Society of American Activists (SAA) identified some types of academic records which include: enrolment records, class schedules, graduation rosters which are usually kept the office of the registrar, minutes, reports and sample test questions, assessment scores and grades that are kept by the department.

Student information system deals with all these kinds of student details, academic related reports, college details, course details, curriculum, batch details, placement details and other resource related details too. It keeps track of all of a student's information, which may be utilized for reporting, attendance monitoring, course progress, semester completion, and year of completion. Academic institutions have access to a lot of student performance data, but they often lack the training and support they need to accurately interpret and apply those data to make the difference needed.

Killion and Bellamy (2000) stated: “Understanding and using data about school and student performance are fundamental to improving schools. Schools are unlikely to recognize and address problems that require attention, discover effective methods to address those problems, or know how they are going toward achieving their objectives and goals without analyzing data and transmitting intelligence. Reform is powered by data.”

Administrators must recognize that as the need to improve student achievement increases, the need for access to data in real-time grows. Academic institutions, particularly lecturers need immediate access to student performance data to determine the growth of individual students and subgroups of students to help create site-driven school improvement plans if they are going to better prepare students for the 21st century. In this context, academic institution management and lecturers need to be able to change their policies and instructional practices based on the reports of performance data from a variety of ongoing, summative assessments.

### 2.2.3 Reporting systems

Reporting systems, simply stated are software systems that are used to transform data into useful information. It uses an arrangement of data, processes, and an interface that interact to support and improve day-to-day operations in an organization as well as support the problem solving and decision-making needs of management and users. Laudon (2002) defines reporting systems technically as a set of interrelated components that collect (or retrieve), process, store, and distribute information to support decision making, coordination, and control in an organization. In the field of finance, Similarly, a financial reporting system is described as an information system that consists of a collection of interconnected parts that are responsible for external financial reporting. Organizations are being pushed to acquire, interpret, and utilize data in order to make practical decisions in order to enhance company operations. Reporting systems leverage on existing data repository or databases containing data that are unexplored and perform a primary function of aggregation after which analytical and/or statistical methods are applied to identify relevant information that needs to be reported.

Reporting systems have evolved from the conventional display of an unending list of records that are scarcely helpful for anything other than browsing into a technology whose relevance cannot be overstated. A modern technology popularly employed in commercial reporting systems is dashboards. A dashboard is used to manage information overload by summarizing key performance metrics to communicate performance against short and long-term interests. In research from Abduldaem and Gravell (2019) it stated that a dashboard implements the visual display of only the most important information from a vast collection of information corresponding to a set of objectives and arranges them in an organized manner so that it fits on a single screen and the information can be monitored and interpreted at a glance. Ioana et al. (2014) explained that dashboards present key performance indicator as well as key risk indicators using charts, graphs, and scorecards.

In academic institutions different reports can be generated based on vast options related to students, batch, course, faculty, exams, semesters, certification and even for the entire institution. The generated reports suggest several other areas for possible investigation and makes a number of other observations and recommendations. (Budhrani et al., 2018). Hamilton et al. (2009) highlighted the effects of using student's data systematically through analysis and reporting among which are:

a. prioritizing instructional time;
b. targeting additional individual instruction for students who are struggling with particular topics;
c. more easily identifying individual students' strengths and instructional interventions that can help students continue to progress;
d. gauging the instructional effectiveness of classroom lessons;
e. refining instructional methods; and
f. examining school-wide data to consider whether and how to adapt the curriculum based on information about students' strengths and weaknesses.

The adoption of reporting systems using the power of data to manage our decisions indicates that fact-based decision making is increasingly important within organizations (Mandinach, 2012).

### 2.2.4 Online analytics

Online analytics refers to the general activity of querying and presenting text and number data from data warehouses and/or data marts for analytical purposes. Online analytics tools are "read only" since they are only used to retrieve data from databases or repositories for use in decision-making. Online analytics is the real time extraction and analysis of data from one or more manual or automated systems, such as the CMS or a student information system with the primary aim of inducing goal-specific actions. The data, which may be stored in a data warehouse for ongoing use, is analyzed using statistical software, and a mathematical model is generated which could be presented either as a table of values or a visual representation of these values using chart or graphs. Based on the model a particular action may be triggered.

Online analytical tools allow users to swiftly analyse and comprehend data that has been collected and formatted particularly for analysis, and then make fact-based decisions. The word "online" as employed in the term “online analytics”, has nothing to do with the Internet or the World Wide Web. However, “online” merely refers to a form of computer processing in which the computer responds to user requests immediately (or extremely rapidly A typical business keeps and uses a variety of operational data sources. Databases and other data repositories that support the organization's day-to-day activities are examples of operational data sources. Online analytics can be used to analyze students’ different academic attributes to extract and display any hidden pattern in students’ academic performance. This knowledge can help educational institutions to improve their teaching or other approaches to the student which on the one side will improve students’ academic performance as well as career and on the other side will benefit all the other stakeholders of the institutions

### 2.2.5 Dashboard and visualizations

Dashboard used to be a fancy name for Executive Information Systems (EIS) when it was first developed in the 1980s with the main purpose of displaying a number of financial measures on a simple interface in a way that could be easily understood by the executives.

A dashboard is a visual representation of the most essential data necessary to achieve one or more goals, aggregated and displayed on a single screen so that the data may be watched at a glance using compact, precise, succinct, and intuitive display methods. The true value of dashboard solutions comes from their capacity to replace manual data collection with a constant, adaptive information flow mechanism. Data repositories are transformed into usable information through dashboards.

Dashboards and visualization are perceptual tools that help you get a better span of control over a large amount of data in your organization. These technologies assist individuals in seeing trends, correlations, and deviations visually, reasoning about what they see, and making successful judgments. As a result, these technologies must take advantage of people's visual ability. For users, dashboards often provide three fundamental functions:

a. They keep an eye on and track key metrics.

b. They provide analysis to determine trends and exception conditions.

c. They report information to aid study and diagnosis and indicate corrective activities as appropriate.

### 2.2.6 Common visualization techniques

a. **Bar graphs**

A bar graph is a graphical representation of qualitative data presented using a frequency distribution. On the graph's horizontal axis, labels for the qualitative variable's categories are displayed. A bar is placed above each label, with the height of each bar corresponding to the amount of data values in the category.

#### b. Pie charts

Pie chart is another graphical device for summarizing qualitative data. The size of each slice of the pie is proportional to the number of data values in the corresponding class.

#### c. Histogram

A histogram is a graphical representation of quantitative data presented in a frequency distribution. On the horizontal axis, the values of the quantitative variable are shown. Above each class, a rectangle with the base equal to the width of the class interval and the height proportionate to the number of data values in the class is displayed.

#### d. Line graph

A line graph, also known as a line plot, is a type of graph that shows data as a sequence of 'markers' linked by straight line segments. It's a simple chart that may be seen in a variety of areas. The measurement points are sorted (usually by their x-axis value) and connected by straight line segments, similar to a scatter plot. A line chart is frequently used to show a data trend over time periods.

### 2.2.7 Different visualization tools

#### a. Tableau

Tableau is a visual analytic platform that allows organizations and individual users to securely integrate visualization into their application using data from their local databases or one that is provisioned over the internet.

b. **Microsoft excel dashboard**

Excel dashboards contain different components that aid in the presentation of data, such as charts, tables, figures, and gauges. They may manage data from a variety of sources and for a variety of reasons, and the data can be utilized for marketing, financial, or other initiatives. The dashboard is particularly useful for big amounts of data because it would be difficult to sift through such vast amounts of data otherwise.

c. **Microsoft power BI**

Microsoft's Power BI is a business analytics service. Its goal is to deliver dynamic visualizations and business intelligence capabilities through an easy-to-use interface that allows end users to generate their own reports and dashboards.

d. **D3 JS**

D3.js is a JavaScript framework that allows web browsers to create dynamic, interactive data visualizations. Scalable Vector Graphics (SVG), HTML5, and Cascading Style Sheets (CSS) are all used. It creates visualizations by binding the data and graphical elements to the Document Object Model.

e. **High charts JS**

High charts JS is an SVG-based JavaScript charting toolkit featuring VML and canvas fall-backs for older browsers. Other than normal charts, it also offers a different package for stock charts called High Stock which is also feature rich. It allows exporting charts in PNG, JPG, SVG and PDF.

f. **Chart JS**

Chart.js is a free open-source JavaScript data visualization toolkit that supports eight different chart types: bar, line, area, pie, bubble, radar, polar, and scatter.

### 2.3. Software Development Life Cycle (SDLC)

Software development life cycle (SDLC) is a framework that provides a sequence of activities to be carried out by software designers, developers and other parties involved in the entire process of software development. SDLC is described as a highly structured step-by-step technique used for the development of any software. The software development life cycle is a set of processes for the entire design, development, and ultimate maintenance of software projects, which includes every action taken to gather customer requirements and guarantee that they are met. It is made up of several different phases that are well defined in terms of planning, designing, developing, testing, and deploying software systems.

According to Ruparelia (2010), Herbert Benington presented the first clear representation of the SDLC model in 1956, from which many other software development life cycles have been formed and redefined. The SDLC's main goal at the time was to create a structured process for producing high-quality software systems that meet or exceed customers' expectations based on the system's requirements, hence the approach of moving software projects through clearly defined phases, usually within specific time frames.

From conception, analysis, design, implementation, and maintenance through disposal, the system development life cycle is the whole process of creating, implementing, and decommissioning information systems. Although there are many distinct SDLC models and methodologies, they all have a set of clearly defined stages or phases in common. All software projects go through the phases of requirements gathering, business analysis, system design, implementation, and quality assurance testing. (Klopper, Gruner and Kourie, 2007)

### 2.3.1 Software Development Life Cycle (SDLC) phases

SDLC phases (also called SDLC processes) are logical collections of activities involved in software development life cycle in a manner that each collection interleaves the other. Nugoro, Waluyo and Hakin (2017) stated five general SDLC phases that cuts across various SDLC methods or models: They are:

a. **System planning**

This process involves activities such as problem identification, feasibility study, team consolidation and scope definition. This process identifies the problem that calls for a software solution and undergoes a preliminary investigation of the problem and proffers a suitable solution. It also entails weighing the solution's costs and benefits, as well as determining whether to enhance a current system or create a completely new one.

b. **System analysis**

An analysis of the existing system is also carried out through interviews and examination of existing documents as well as with items such as questionnaires and forms. The goal of this phase is to first identify the client's real requirements, capture the details of each requirement, and ensure that everyone knows how each requirement will be met. The deliverable artifact of this process is a Software Requirement Specification (SRS) document.

c. **System design**

In this process the core objective is to transform the requirement specification in an action plan and create a basis to build a system that satisfies these requirements.

This process delivers a logical design that can be sketch on a paper or on a computer and physical design of elements of the software system. Structure and formats to be used in the software are designed during this process with considerations on how data is to be manipulated in the system. The storage devices, input and output procedures, database structure and testing standards to be adopted by the software system are also stated in the design.

The technical specifics of the design are addressed with stakeholders, and many criteria such as risks, technologies to be employed, team capability, project limitations, time, and cost are examined before the optimal design strategy for the product is chosen. (Barjtya, Sharma and Rani. 2017). This process entails activities such as specific input and output descriptions, database schema, forms and reports design, system platforms (hardware and software), application architecture design, user interface design, system interface design, and system control design. The output of this process is a Software Design Document (SDD)

d. **System implementation**

This phase entails the development of the actual software system to suit the design specification defined in the system design phase. It also involves testing activities to validate that the system meets the user, business and system requirements. Software developer, engineers and testers work together in this phase creating the database to suit the database scheme, creating programs and application according to the system design as well as debugging the created programs and applications. Tests are used to identify system flaws and faults that need to be addressed. In case of software systems, after coding the whole programs, a test plan is developed and run on a given system.

For this aim, a variety of testing procedures are utilized. Two among many of such tests are: Black Box testing and White Box Testing. IEEE Standard Glossary of Software Engineering Terminology (1990) defines Black box testing (also called functional testing) as testing that ignores a system's or component's underlying mechanisms and focuses exclusively on the outputs produced in response to certain inputs and execution circumstances. and White box testing (also known as structural testing or glass box testing) is defined as testing that considers a system's or component's internal mechanism.”

e. **Software deployment and maintenance**

Users are taught on the system and begin to utilize it during this phase. The system is reviewed to determine its full capabilities. During this phase, needed adjustments for new needs are identified, and performance is assessed in terms of quality and efficiency. The system's developer supplies the organization with two types of documentation related to the system. These are:

i. Operator/user document: In this document, the user is provided a comprehensive description of the system, including how to run it, what error messages may arise, and how to resolve them.

ii. System document: This document provides the specifics of the system design, process flows, and other information that helps the organization understand the system and the modifications that need to be made, as well as the permissions given for those changes, in order to meet new user demands.

### 2.3.2 Software Development Life Cycle (SDLC) models

SDLC model is a theoretical outline of all the activities to be carried during software development usually stratified into phases. A software life cycle model is either a descriptive or prescriptive definition of how software is or should be created.

(Gajalakshmi P., 2006). It was pointed out that a descriptive model tells the story of how software is developed, providing a foundation for understanding the system's complexities and functions, as well as how the processes involved in software development can be improved, whereas prescriptive models suggest how software systems should be developed and serve as a guideline for managing and carrying out software development projects.

Different SDLC models have been developed to suit the various need of specific type of software taking into considerations the existing conditions and constraints. The most difficult task is to identify the model which suits the organization structure, for this every organization need to know the model and its real time application so the user can relate, understand and find out how its actually working. Every model in SDLC is the combination of various steps, which step leads to what and what is the basic requirement of that step that needs to be understood. Waterfall, RAD, spiral, incremental, V-shaped, and other SDLC models are employed in diverse companies based on the conditions that exist there. Each of these software development methods has its own set of benefits and drawbacks.

a. **Waterfall model**

Waterfall Model often termed as a linear sequential development model has been used widely in the last few decades. Waterfall model was proposed by Royce in 1970 which is a linear sequential software development life cycle (SDLC) model. The requirements analysis, design, coding, testing, and implementation phases are followed in such a way that the phases are not repeated and the development does not move on to the next phase until the previous phase is completely completed. (Roger, S.)

This model has a strict structure that necessitates the definition of all system requirements at the outset of a project. The design and development stages might then commence. In a waterfall model, each phase must be completed (completed) before the next one can begin, and the phases do not overlap. The development process is completed on a single project under this model, and the requirements must be known in advance, thus there is no room for change once the project development begins. As a result, the waterfall paradigm is not adaptable to changing requirements.

Waterfall technique spends the most of its time in the first phase, documenting the project, and the implementation phase also consumes a substantial amount of time, whereas extreme programming approach spends the most time in the testing phase. (Pooja and Nitasha, 2016)

Lewallen (2005) stated “This model is only used when the requirements are highly exceptional, clear, and unchanging, the product description is consistent, technology is well-understood, there are no ambiguous requirements, enough resources with needed expertise are readily available, and the work is brief. ”

b. **Spiral model**

The spiral model was defined by Barry Boehm in his 1988 article A Spiral Model of Software Development and Enhancement. The spiral model is like the incremental model, with more attention put on risk analysis. According to Boehm, 2000, the Spiral model has four phases, being: planning, risk analysis, engineering, and evaluation.

Iterations of these phases are repeated throughout a software project (called Spirals in this model). Starting with the planning phase, the baseline spiral collects requirements and evaluates risk. The baseline spiral is built upon in each spiral that follows.

i. **Planning phase**

During the planning phase, requirements are gathered. Business Requirement Specifications (BRS) and System Requirement Specifications (SRS) are examples of requirements gathered during this phase.

ii. **Risk analysis**

A procedure is conducted in the risk analysis phase to identify risk and alternative solutions. At the conclusion of the risk analysis phase, a prototype is created. Alternative solutions are offered and executed if any risks are discovered during the risk analysis.

iii. **Engineering phase**

Software is produced during this phase, which is followed by testing. As a result, development and testing are completed during this period.

iv. **Evaluation phase**

This phase allows the client to assess the project's output thus far before moving on to the following spiral.

c. **Iterative model**

The incremental model is a step forward from the waterfall paradigm. The waterfall model's phases are used in such a way that the output of each increment is used as the input for the following increment. As a result, with each iteration, some client feedback is included into the development of the following incremental product. After the first increment, the client receives a core product that is ready to use. A plan for the next increments is developed based on customer feedback, and changes are made as needed. The issues that arise in the waterfall approach, according to Munassar and Govardhan (2010), created a desire for software development models that may produce faster outcomes, require less information, and offer better flexibility, all of which are offered by the iterative model.

An iterative life cycle model is a type of systems development life cycle model that does not start with a complete set of requirements. Rather, development begins with identifying and implementing a small portion of the product, which may then be tested to identify additional requirements. This method is then repeated, resulting in a new variation of the product for each model cycle. (Rastogi, 2015).

d. **V-shaped model**

The V-model is a software development method in which processes are executed sequentially in a V-form. The v-model (also known as verification and validation model) follows a fairly rigorous approach, with the next phase starting only when the previous one has finished. Gupta, Taya, Jai and Mukand (2012) described the V-model to be an extension of the Waterfall model that maps test phases to the general phases of development. The link between each development life cycle and the corresponding testing phase is shown by the V-shape. The V-shaped model is associated with three core phases that consists of set of activities in them. These phases are: Verification phase, Coding phase and Validation phase. After the coding phase, the V-model SDLC process stages are turned upwards. The verification phase consists of five activities namely: Business requirement analysis, System requirement analysis, Architecture engineering, Design and Detailed specification while the Validation phase is also made up of five testing activities which are: Unit testing, Component testing, System integration testing, System testing and Acceptance testing.

e. **Agile model**

The waterfall methodology is the polar opposite of agile. Instead of considering requirements, design, and testing as discrete phases, an agile approach treats them as continuous processes requiring participation from developers, management, and consumers. Agile software development promotes adaptable planning, evolutionary development, and continuous improvement, as well as quick and flexible change reaction. It emphasizes user satisfaction, simplicity, and the developers' and consumers' constant attention. The word "agile" has been used in software development to denote the capacity to adjust to changes — changes in requirements, technology, and people.

Agile development, according to Pilgrim (2012), is a social event involving programming-driven methods that focus on simplifying the SDLC. A significant portion of the displaying and documenting overhead is eliminated, in favor of extremely tight communication. Every accentuation is a finished process of programming endeavors, including orchestrating, requirements examination, layout, coding, testing, and documentation, in which an augment complements clear, iterative application progression.

A team of software developers published the Agile Manifesto in 2001, highlighting the importance of the development team, accommodating changing requirements and customer involvement.

### 2.3.3 Extreme programming

Extreme Programming (XP) made its debut in the software world in 1999, thanks to Kent Beck's book "Extreme Programming Explained." Extreme programming is a software development technique that is iterative. XP stands for "extreme programming," a flexible agile approach that emphasizes the connectivity of the proposal and implementation stages. Extreme programming is a software development approach that focuses on improving product quality and being responsive to changing client needs. It's a form of agile software development that promotes rapid development cycles and frequent releases. These releases are designed to increase software productivity and quality by using specific procedures that focus on establishing checkpoints at which client needs may be accepted and satisfied.

Tchidi and He (2010) define extreme programming as a rigorous approach to software development. The iterative life-cycle employed by XP includes the phases of analysis, planning, designing, implementation, and delivery. All of these steps are completed according to XP best practices. Extreme Programming (XP) was created to meet the unique challenges of small teams developing software in the face of ambiguous and shifting requirements.

Planning, Design, Coding, and Testing are the four key phases of the XP cycle. Oral communication, regular testing, code review, and design are all essential. Short development cycles, incremental planning, evolutionary design, and the flexibility to respond to changing business demands are all characteristics of the XP method. The approach is based on a series of practices that appear to be simple to grasp. Iterations are emphasized in XP, which allows for changes in needs even after the original planning is finished.

The XP approach, according to Beck (2000) and Lippert and Roock (2001), is designed to fulfill the demands of a competent small team of less than 10 developers working in a co-located office with the client building non-safety-critical software on an object-oriented technology. An XP project typically results in a software project in which software development begins immediately, virtually no documentation artifacts are created (except for “user stories” written on index cards), and the project proceeds in an iterative fashion in which prototypes are created daily with the direct input (and sometimes help) of stakeholders until the desired effect is achieved.

### 2.4 Unified Modelling Language (UML)

Object Management Group (OMG) specification of the UML states that The Unified Modelling Language (UML) is a graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system. The UML offers a standard way to write a system's blueprints, including conceptual things such as business processes and system functions as well as concrete things such as programming language statements, database schemas, and reusable software components.

The Unified Modelling Language (UML) is an object-oriented modelling language that was created in 1994 and 1995 by Rational Software's talented software engineers Grady Booch, Ivar Jacobson, and James Rumbaugh. It was in the works until 1996. The Object Management Group (OMG) designated UML as a standard in 1997. Since its adoption as a standard, the Object Management Group has been in charge of managing UML.UML was recognized as an ISO standard by the International Organization for Standardization in 2005. It's used to create object-oriented models in a variety of sectors.

The UML is a language for visualizing, specifying, building, and documenting software-intensive system artifacts. UML relies significantly on graphical constructs to form the foundation of the numerous UML diagrams as a visual modelling language. The development of UML version 1.0 began in 1994 and continued through 1996, ending in the release of UML version 1.0 in January 1997. (Booch, Rumbaugh and Jacobson, 1999). In November 1997, the Object Management Group (OMG) approved

UML 1.1 as a standard modellIng language. The most recent version is 2.5.1, which was released in 2017.

UML 1.x defines nine diagramming techniques: Class diagram, Object diagram, Component diagram, Deployment diagram, Use Case diagram, State-chart diagram, Activity diagram, Sequence diagram, and Collaboration diagram (Booch et al., 1999). Class Diagram, Object Diagram, Component Diagram, Composite Structure Diagram, Package Diagram, Deployment Diagram, Use Case Diagram, Sequence Diagram, Communication Diagram, State Diagram, Activity Diagram, Timing Diagram, and Interaction Overview Diagram are the thirteen diagramming techniques defined by UML 2.x. (Booch et al., 2005).

### 2.4.1 UML diagrams

The Unified Modelling Language produces UML Diagrams. It's a visual depiction of classes, objects, and their relationships. A UML diagram is a model that depicts a system component. It is used to define a system's functioning or design. A diagram must be clear and succinct in order for the viewer to understand it quickly. In UML 2.5, UML diagrams are divided into three categories based on the view of the system they represent: Behaviour diagrams, Structure diagrams, and Interaction diagrams. In addition, there are 14 other types of UML diagrams that span these three categories.

### 2.4.2 Types of UML diagrams

#### a. Class diagram

Class diagrams specify the system from both an analysis and design perspective. They depict what the system can do analysis, and provide a blueprint showing how the system will be built (Ambler, 2000). Class diagrams are self-describing, and include a listing of the attributes, behaviors, and responsibilities of the system classes. Class diagrams that are properly described can be directly converted into actual program code. Furthermore, well-designed class diagrams may aid in the software engineering process by providing thorough system documentation. (Lago, 2000).

b. **Object Diagrams**

Object diagrams represent specific occurrences or instances of class diagrams, and as such are generally seen as more concrete than the more abstract class diagrams.

c. **Component Diagrams**

Component diagrams depict the different parts of the software that constitute a system. This would include the interfaces of and between the components as well as their interrelationships. At a higher level, Ambler (2000) and Booch, Rumbaugh, and Jacobson (1999) described component diagrams as class diagrams.

d. **Composite Structure Diagrams**

Composite structure diagrams depicts the relationships and communications between the functional parts of a system. These diagrams depict high level and abstract views of the system being modeled.

e. **Package Diagrams**

Package diagrams are a subset of class or object diagrams. They are used to represent or demonstrate a collection of linked UML elements. Package diagrams make it simpler to see relationships between various components of the represented system (Pilone and Pitman, 2005). As a result, Package diagrams provide a high-level perspective of the systems being modelled, making them an excellent tool for communicating with users and other interested parties.

f. **Deployment Diagrams**

Deployment diagrams are a subset of class diagrams. The diagram depicts how the run-time processing units are coupled and collaborate in this case. The main distinction between component and deployment diagrams is that component diagrams focus on software components, whereas deployment diagrams represent the proposed system's hardware layout.

g. **Use-case Diagrams**

While class models and diagrams are the foundation of object -oriented programming, use case models and diagrams depict the system from the perspective of an end-user, and represent tasks that the system and users must do in order to fulfil their duties (Pooley and Stevens, 1999). Actors (those people or systems outside the system of interest who need to interact with the system under development), use cases, and interactions between the actors and use cases are all included in use case models and the accompanying use case diagrams.

Developers should initiate the analysis process with use cases, according to Booch, Rumbaugh, and Jacobson (1999), by interviewing end users, going over basic historical system documentation, and so on, and then developing use cases from those interviews and documents to drive the development of the class model and other system models. Use Case Narratives, which are textual explanations of Use Cases, are also widely used by developers to help in the assembly and comprehension of Use Case Diagrams, according to Dobing and Parsons (2006).

h. **Activity Diagrams**

Activity diagrams model the flow of control through activities in a system and, as such, are really just flow charts. In addition, activity diagrams are special instances of state chart diagrams.

i. **State Machine Diagrams**

State machines are used to simulate the transition between states inside an item, as well as the signals or events that cause or induce the change in state from one value to another (Booch, Rumbaugh and Jacobson, 1999). A thermostat, for example, is triggered by a change in air temperature to activate a heating or cooling system that adjusts the temperature in a room or building. The thermostat would detect an increase in air temperature in this scenario, causing the cooling system to switch from inactive (or idle) to active and begin the cooling process. When the ideal temperature is attained, the thermostat detects it and switches the cooling system back to inactive mode.

j. **Sequence Diagrams**

Sequence diagrams represent and verify the logical stages of use cases in depth (Ambler, 2000). The temporal ordering of communications between objects in the system is depicted in sequence diagrams, which feature lifelines for the objects participating in the sequence as well as the concentration of control at various points in time (Booch, Rumbaugh and Jacobson, 1999).

k. **Communication Diagrams**

Communication diagrams rather than depict the sequence and control flows, represents the objects engaged in the interactions. Because one may readily convert to the other, there is some amount of interdependence between Communication and Sequence Diagrams. The mapping, however, is not one-to-one. In other words, while converting from Sequence Diagrams to Communication Diagrams, certain information may be lost.

l. **Timing Diagram**

Timing diagrams are most commonly employed with real-time systems and aim to express the timing aspect linked to the messages being passed across the modelled system. Timing Diagrams depict a lifeline and the events that take place over time when the system runs. The elements indicated in these diagrams represent the specifics of time restrictions provided in messages.

m. **Interaction Overview Diagram**

This is a simplified version of Activity Diagrams and a sub-type of them. These diagrams can help a user comprehend how a system works, but they hide the specifics of the messages and information that the messages carry between objects. These are high-level diagrams that aren't meant to explain the nuances or subtleties of how one system interacts with another.

## 2.5. Model-View-Controller Architecture

The model view controller (MVC) pattern is a web-based application architecture paradigm. Trygve Reenskaug, who worked at Xerox Parc in the 1970s, was the first to think of the MVC design pattern. "The primary aim of MVC is to bridge the gap between the mental model of the human user and the digital model that resides in the computer," he says. Krasner and Pope later defined the MVC paradigm in full in their paper "A cookbook for utilizing the model-view controller user interface paradigm in Smalltalk-80" published in the Journal of Object-Oriented Programming in 1988. The model of the major application area, the display of data in that model, and user interaction are the three key categories of an application. The MVC pattern divides tasks into three primary roles, enabling for more effective cooperation. This pattern is used to create applications in most languages, including Java, PHP, Python, C#, and others. Spring-MVC framework is known in Java, Cake PHP is known in PHP, and Microsoft has a framework called ASP.Net MVC, and so on. MVC provide three types of classes that represents the three primary layers.:

**Model**: The model layer of an MVC application is crucial. It keeps track of information in the form of data, which is then utilized to display the result using views.

It is a representation of the database records. It is in charge of the application's data. It primarily consists of application data, logic definition, function specification, and business rule participation. A model can either be a single item or a collection of things. This layer is responsible for data management as well as database communication, which includes inserting, retrieving, and updating data in the database. The Model is a class that contains variables of various data kinds, as well as getter and setter methods. It doesn't connect to a database and just manages data. The major task of the model class is to respond appropriately to the request of the view's class and the controller's instruction.

**View**: Our application's user interface is built using views. Users engage with our web sites while utilizing the user interface. The outcomes of the data in the model are displayed in the view. A view is responsible for displaying all of the model's data. It only displays needed characteristics and conceals those that aren't. As a result, we get the benefit of presentation encapsulation. It uses script-based tags such as JSP, ASP, and PHP, and it's extremely easy to combine with AJAX. The view class's primary role is to provide graphical user interface output for model components that are especially intended for client needs. The data is contained in the View (the database records). A model's answer must be translated into an output form. It displays data in a certain format, such as JSP, ASP, or PHP.

**Controller**: The user's requests are handled by the controller layer. The controller will carry out the actions that the user has requested. The user and the system are connected by a controller. The Controller is in charge of both the Model and the View. It regulates the flow of data in the model and updates the display as the data changes. The distinction between the Model and the View is what it is. The controller gets the data, processes it, and then executes the code that changes the state of the data model. These components work with the model layer to choose the appropriate view to display to the user based on the user's requests.

The controller reacts to the user when a request is released. Controllers can read data from a view, control user input, transfer data to the model, and make model and view modifications. All of the controls are matched up with view and model classes in the controller class. The Controller binds all application logic and combines the display in the View with the functionality in the Model. It is in charge of retrieving data from the View and determining the application's execution route. The Controller will use the Model functionality to retrieve the data and parse it so that the View can show it. It is also in charge of error handling. 2011 (Freeman and Sanderson)

The MVC pattern architecture allows us to follow the separation of concerns principle by allowing us to implement the code in the model, views, and controller layers of the applications separately. Testing these components is much easier because we divide the logic of our program into three tasks (input logic, business logic, and interface logic). Since we can use any unit testing framework that is compatible with the MVC framework, testability is very quick and flexible.

The MVC design pattern is ideal for web application development since it combines a number of technologies that are often separated into layers. MVC-specific behavior could also include sending various views to different sorts of user-agents.

### 2.6 System Development Tools

This section discusses various system development tools that are to be adopted during the course of this study. It gives a clear understanding of the .NET framework, Microsoft SQL Server, C# and other related technologies.

### 2.6.1 ASP.NET

The.NET Framework (pronounced "dot net") is a Microsoft software framework that runs largely on Microsoft Windows. It contains the Framework Class Library (FCL), which is a big class library that allows language interoperability (each language may utilize code written in other languages) across a number of computer languages. The Common Language Runtime (CLR), an application virtual machine that offers services like as security, memory management, and exception handling, executes programs developed for the.NET Framework in a software environment (as opposed to a hardware environment). The.NET Framework is made up of two components: FCL and CLR. ASP.NET is a free to use server-side Web application framework that is used to create dynamic Web pages. Microsoft created it to help programmers create dynamic web pages, online applications, and web services.

The initial version of ASP.Net MVC was launched in May 2009, and it has since evolved to version 6.0, with the open-source version of ASP.Net Core MVC 1.0 being released in May 2016. In the ASP.NET web development framework, C# and VB.Net are utilized as code behind languages. ASP.NET is built on top of the HTTP protocol, and it makes use of HTTP commands and rules to establish bidirectional communication and collaboration between the browser and the server.

### 2.6.2 Microsoft SQL Server

Microsoft SQL Server is a relational database management system that Microsoft presently develops. It is a database server, which is a software product whose principal job is to store and retrieve data as required by other software programs, which may operate on the same computer or on a networked computer (including the Internet). It is used in corporate IT settings to handle a wide range of transaction processing, business intelligence, and analytics applications. The SQL Server Database Engine is the heart of Microsoft SQL Server, controlling data storage, processing, and security. It consists of a relational engine for processing instructions and queries, as well as a storage engine for managing database files, tables, pages, indexes, data buffers, and transactions. The Database Engine also creates and executes stored procedures, triggers, views, and other database objects.

### 2.6.3 C#

C# (C Sharp) is a Microsoft-developed general-purpose programming language that runs on the.NET Framework that is popularly used for creating mobile apps, games, and Windows programs. It is a type-safe, contemporary, object-oriented programming language that enables developers to create a wide range of safe and reliable applications for the.NET environment. C# comes from the C family of languages. It was first developed by a team led by Anders Hejlsberg at Microsoft in 2002 and was later approved by ECMA as standard. Its' first version C# 1.0 was launched with .NET 1.0 in the same year. Some of its major features include: Classes, Interfaces, Structs, Events, Properties, Operators and Expressions etc.

## 2.7 Related Works

There have been several studies on information systems used to manage student academic records as well as reporting systems some of which are reviewed in this study.

Maria and Dave (2016) developed an automated academic record management system using the business intelligence approach. The system used enterprise reporting, particularly the tabular type of business intelligence reporting. The system integrated business intelligence specifically in the query and reporting component. All the reports are dynamic and update in real-time if there are any transactions to be done by the users. Each report is generated by joining more than one table in the connectivity of the database. This ensures the accuracy and consistency of the data in the report. It is a web application built with C# through the Microsoft visual studio integrated development environment and hosted on windows Server 2008 R2. Microsoft SQL Server 2008 - It was used in the connectivity of databases in the programming application of the proposed system.

Dada et al. (2017) designed and implemented an integrated software system for result processing and transcript generation specifically for a networked environment. The work presented a system that can be used to check the performance of each student in various courses. It reduced the waiting time involved in the manual method of processing and generating results. The software delivered from the work was built on the web platform using HTML, CSS, and JavaScript for the frontend, PHP as the scripting language for the application logic, and MySQL for the database system. This work limited its research and solution to automating result processing specifically student grading as it made no effort to analyse the results and generate reports that may be useful for the management-level decision-making process.

Orobor (2015), adopted a new approach of cloud computing model for student result computation to replace traditional software that is usually installed locally on computer systems within the institution. The research developed a student result computational system as a cloud computing service which is a variant of the Software As A Service (SAAS) model. Thereby reducing the cost of owning a result computation system and the complexity of accessing the service locally. The research leveraged the Microsoft Azure Platform for hosting the result computation system making it available remotely to multiple institutions.

Obiniyi and Ezugwu (2010), in their work, identified the causes associated with delays in student's results processing and in extension the release of the result. They designed and implemented a student information system for tertiary institutions that made use of the neural network for processing results for designated departments through an improved centralized database system. Their work made use of Apache web server alongside PHP for server-side scripting to implement the system as well as an open-SSL library to ensure proper data encryption and role-based authentication.

After reviewing the aforementioned works, it is evident that different research and studies have developed different information systems for student's academic records to meet different purposes as their areas of interest demand. The use of various technologies and approaches including, web applications, cloud computing models, and neural networks justifies the need for information systems that are available remotely as against local stand-alone applications. Hence this work will develop a web application using Microsoft's ASP.NET MVC framework that will be hosted through IIS server making it accessible from any location as well as integrate a highly-scalable relational database through which data can be retrieved, processed, and reported in real-time.

In research from Hashim and Mohamed (2013), it developed a student information system for storing and updating students' data as well as creating reports that are made available to lecturers concerning a students' status. The study employed the rapid application development SDLC methodology to create a local stand-alone system that was developed based on the available features of an existing staff information system and a Microsoft Access 2007 database to store all the needed information on the system. It further recommended that future work in the area of student information systems adopt relevant technology to make the system available remotely over a network to ease access to the system.

Ali (2018) faulted the manual methods of student information management in Department of Student Affairs in the College of Medicine, University of Diyala and subsequently design and implemented a student information management system using the .Net framework environment of Microsoft Visual Studio 2010 and Microsoft SQL Server 2008 R2. It then concluded that the adoption of the system enables administrative users to efficiently update student data thereby increasing the speed of task completion and aiding decision-making process across the college where it is adopted.

# CHAPTER THREE

# METHODOLOGY

## 3.1 Method of Identification of User and System Requirement

During this work, the user requirement as well as system requirements were identified through informal interviews and review of other existing systems in the related context. In gathering the necessary data for the analysis, design and consequent effective development of the proposed system, the aforementioned techniques were adopted. The functional and non-functional requirements were identified and defined as well as roles needed to manage the system and the associated functions for each role. The hardware and software requirements for the development and deployment of the system were defined after which the design of the system was created using applicable UML diagrams such as: use case, sequence, activity, etc. The MVC method of software development was adopted to implement the front-end and database of the system using relevant development technologies.

### 3.1.1 Identification of system requirement

This section discusses the functional and nonfunctional requirements of the system, identifying the essential features that the system should provide. It covers the data to be used by the system as well as the operations to be performed.

#### a. Functional requirement

These are the criteria that the end user expresses as essential features that the system should provide. These are expressed or described as input to be provided to the system, operation to be done, and expected output.

i. **Registration and authentication:** User should be able to register for the role of lecturer (Basic user) and login using their valid email address and password

ii. **Role management and authorization:** Admin user should be able to add roles as well as assign roles to registered users as necessary and thereafter authenticated users should be authorized to see dashboards for their specific roles.

iii. **Dashboard management:** Reports in form of charts (bar chart, pie chart, line chart etc.) should be generated and displayed on the dashboard based on the specified parameter (such as courses, programme, department, level etc.) and the current users' role.

iv. **Assessment management:** Users with the "lecturer" role should be able to create assessment for courses as well as administer the assessment to student and view reports of these assessments

b. **Non-functional requirement:**

Non-functional requirements are a solution's qualitative attributes. They are restrictions that apply to a set of functional requirements, allowing you to assess a solution's characteristics rather than its functional behaviors. It supports the functional requirements and determine how the system must fulfil them.

i. **Email verification:** The email provided by users during registration should be verified to validate the authenticity of the entry.

ii. **Assessment management for lecturer for only courses they manage:** Users with role "lecturer" should only be able to create and administer assessment for the courses that they are assigned to

iii. **Dashboard display based on roles:** The dashboard parameters should be available based on roles. For example: users with the "lecturer" role should only get dashboard reports of the courses they are assigned to and their respective assessments while users with the "Examination officer" role should get dashboard reports for all the courses in their department across all levels.

iv. **Application response time:** Each request processing and response must not exceed a 10 seconds limit

v. **Usability:** Users should be able to display different dashboard view based on filter made available through a list allowing seamless navigation between different views.

vi. **Data integrity:** The data used in generating reports for all dashboard is periodically refreshed to ensure that the system uses the update copy of the data present in the database for its' reports.

c. **Hardware requirement:**

Processor: x86 or x64, RAM: 512 MB (minimum), 1 GB (recommended), Hard disc: up to 3 GB of free space may be required

d. **Software requirement:**

.NET framework: Minimum .NET 4.5.1 up to .NET 4.7, SQL Server 2008 Express or higher, Windows server 2008 (Full server) or higher, Windows 8.1 operating system or higher.

### 3.1.2 Identification of user requirement

a. **System Admin requirement**

In this work, an admin user account will be created with the highest privilege to write and read data from any table in the system. The admin user should be able to assign roles to other registered users and delete the records of users if necessary. The admin should also have access to perform all other basic user functions such as view dashboards, creating and administering assessments

b. **Basic User requirement**

In this system the basic users are the users with either "examination officer" role or "lecturer" role. The have different levels of privilege on the system. Basic users with "examination officer" role have access to view dashboard of programme, courses and levels in the department he/she oversees while basic users with "lecturer" role have access to view dashboard of courses he/she is assigned to as well as assessments and the performance of students registered for the courses.

## 3.2 System Design Method

The system was designed using UML diagrams such as use case diagram, sequence diagram, class diagram and activity diagram. The system architecture was also designed as shown in figure 3.1. The various UML diagrams used in designing the system shows the system functions, interaction with users as well as processes involved in performing various action in the system.

### 3.2.1 System architecture

The system shall be made available through hosting on IIS server which supports the .Net framework, so that browser request to the URL of the system is directed to the server which interfaces with controller classes and mediates between the user requests from the browser and the application logic that is structured in classes using the MVC architecture. As shown in figure 3.1 the database of the system shall be managed using MS SQL server which holds all the tables and stored procedures used by the system.

### 3.2.2 Use case diagram

The use case diagram as shown in figure 3.2 presents users of this system and the various actions that they can perform. The users that make use of the system are:

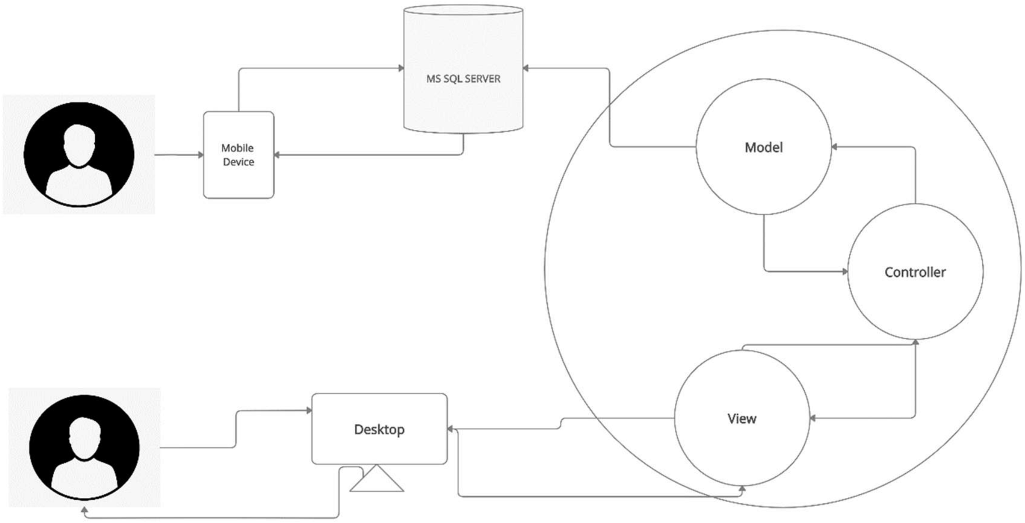


Figure 3.1: System architecture

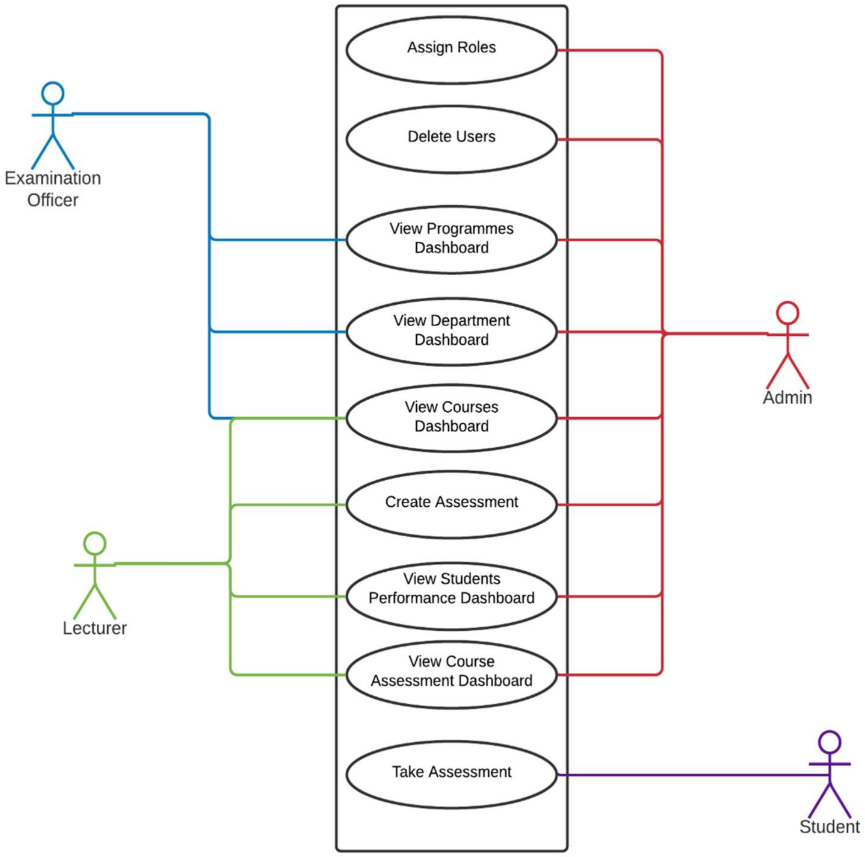

Assign Roles
Delete Users
View Programmes Dashboard
View Department Dashboard
View Courses Dashboard
Create Assessment
View Students Performance Dashboard
View Course Assessment Dashboard
Take Assessment
Examination Officer
Admin
Lecturer
Student

**Figure 3.2: Use case diagram**

administrator, examination officer, lecturer and student. The actions that can be performed on the system include: role assignment, user management, viewing dashboards by programme, courses, department and student performance. Also, students can take quizzes on the E-test module for courses that they are registered for.

a. **Assign roles:** The admin user is able to create accounts for users on the reporting system as well as assign roles for each of the users when their accounts are created and whenever there is a necessity to update the roles. This is further described in table 3.1.

b. **Delete users:** The admin user is also able to delete each user's account in event of the need to remove the account and its data from the system. This use case is fully specified in table 3.2.

c. **View programme dashboard:** As described in table 3.3, the admin user and the examination officers can view the dashboard that contains reports for programme. The examination officer can only view reports on this dashboard that pertains to the department he or she represents while the admin user can view for all programme.

d. **View department dashboard:** The admin user and examination officers can access the dashboard that shows reports for departments. In which case the examination officer can view reports for only his or her department while the admin user can view reports for all departments. This use case is fully described in table 3.4

e. **View courses dashboard:** Lecturers, examination officers and the admin user have access to the courses dashboard where they can view reports for each course. Each lecturer can view reports for the courses which he or she is assigned to while the examination officer can view reports for all the courses in his or her department and the admin user can view for all courses. A full specification off this use case is given in table 3.5

**Table 3.1: Description of assign roles use case**

| | |
|---|---|
| Use Case Name | Assign roles |
| Participating Actors | Admin |
| Flow of Events | ● The admin navigates to the role assignment page<br>● The admin selects a registered user<br>● The admin selects an available role to assign to the user |
| Entry Condition | The user must be logged in to his/her account |
| Exit Condition | The admin successfully assigns the role to the user.<br>If the process does not complete successfully the admin gets a response that the process was unsuccessful, why it was unsuccessful and a message to try again |
| Quality Requirement | The admin must successfully assign an available role to an existing user without errors |

**Table 3.2: Description of delete users use case**

| | |
|---|---|
| Use Case Name | Delete users |
| Participating Actors | Admin |
| Flow of Events | ● The admin navigates to the all-users page<br>● The admin selects a registered user<br>● The admin deletes users record from the system |
| Entry Condition | The admin must be logged in to his/her account |
| Exit Condition | The admin successfully deletes the user's record.<br>If the process does not complete successfully the admin gets a response that the process was unsuccessful, why it was unsuccessful and a message to try again |
| Quality Requirement | The admin must successfully delete an existing user 's record without errors |

**Table 3.3: Description of view programme dashboard use case**

| | |
|---|---|
| Use Case Name | View programme dashboard |
| Participating Actors | Admin, Examination officer |
| Flow of Events | ● The user navigates to dashboard page<br>● The user selects the “by programme” report parameter<br>● The user views reports by programme on the dashboard |
| Entry Condition | The user must be logged in to his/her account and be of role type admin or examination officer |
| Exit Condition | The user must successfully view the programme dashboard.<br>If the request is not completed successfully the user gets a response that the process was unsuccessful, why it was unsuccessful. |
| Quality Requirement | The user must successfully view the dashboard for the programme he presides over (in the case of an examination officer) |

**Table 3.4: Description of view department dashboard use case**

| | |
|---|---|
| Use Case Name | View department dashboard |
| Participating Actors | Admin, Examination officer |
| Flow of Events | ● The user navigates to dashboard page<br>● The user selects the “by department” report parameter<br>● The user views reports by department on the dashboard |
| Entry Condition | The user must be logged in to his/her account and be of role type admin or examination officer |
| Exit Condition | The user must successfully view the department dashboard.<br>If the request is not completed successfully the user gets a response that the process was unsuccessful, why it was unsuccessful. |
| Quality Requirement | The user must successfully view the dashboard for the department he/she presides over (in the case of an examination officer) |

**Table 3.5: Description of view courses dashboard use case**

| | |
|---|---|
| Use Case Name | View courses dashboard |
| Participating Actors | Admin, Examination officer, Lecturer |
| Flow of Events | ● The user navigates to dashboard page<br>● The user selects the “by courses” report parameter<br>● The user views reports by courses on the dashboard |
| Entry Condition | The user must be logged in to his/her account |
| Exit Condition | The user must successfully view the courses dashboard.<br>If the request is not completed successfully the user gets a response that the process was unsuccessful, why it was unsuccessful. |
| Quality Requirement | The user must successfully view the dashboard for the courses assigned to him/her. |

f. **View student performance dashboard:** This uses case as described in table 3.6 applies to lecturers, examination officers and admin users where the lecturer can only view the performance of student in the courses he or she is assigned to while the examination officers can view the performance of each student in the department in each course. The admin user can view the performance of every student in each of their courses.

g. **View Course assessment dashboard:** Lecturers, examination officers and admin users can view the reports or assessment for their assigned courses, all the courses in the department and all existing course respectively. A full description of this use case is given in table 3.7.

h. **Take assessment:** Students can take assessments (quiz) for each of the courses they register following a sequence of steps specified in table 3.8

### 3.2.3 Sequence diagram

Sequence diagram were used to illustrate the order of action for each process on the system. It shows the internal component and the user involved in the process. Some of the process designed with this diagram are: Login, Create assessment, View report by course, view report by programme etc.

#### a. Login

Figure 3.3 shows the sequence of actions performed by users for the login function on the system. The user is redirected to the URL of the login page once he/she is not logged in and a request is sent to the login controller for the login view, the login controller then fetches the login form entities and other attributes such as display name and error message and sends the login view to the browser. The user fills the login form with his/her login credentials and submits the credentials after which they are forwarded to the controller as a post request. The credentials are sent to the login model to validate

**Table 3.6: Description of view student performance use case**

| | |
|---|---|
| Use Case Name | View student performance dashboard |
| Participating Actors | Admin, Examination officer, Lecturer |
| Flow of Events | ● The user navigates to dashboard page<br>● The user selects the “student performance” report parameter<br>● The user views reports of student performance on the dashboard |
| Entry Condition | The user must be logged in to his/her account |
| Exit Condition | The user must successfully view the student performance dashboard.<br>If the request is not completed successfully the user gets a response that the process was unsuccessful, why it was unsuccessful. |
| Quality Requirement | The user must successfully view the dashboard for the student. In the case of examination officer, he views a dashboard of students in his department across all courses while for a lecturer he views a dashboard of students’ performance in his courses only. |

**Table 3.7: Description of view course assessment dashboard use case**

| | |
|---|---|
| Use Case Name | View course assessment dashboard |
| Participating Actors | Admin, Examination officer, Lecturer |
| Flow of Events | ● The user navigates to dashboard page<br>● The user selects the "course assessment" report parameter<br>● The user views report of course assessments on the dashboard |
| Entry Condition | The user must be logged in to his/her account |
| Exit Condition | The user must successfully view the course assessment dashboard.<br>If the request is not completed successfully the user gets a response that the process was unsuccessful, why it was unsuccessful. |
| Quality Requirement | The user must successfully view the dashboard for the course assessment. In the case of examination officer, he views a dashboard of course assessment of courses in his department while for a lecturer he views a dashboard of course assessments in the course assigned to him/her only. |

**Table 3.8: Description of take assessment use case**

| | |
|---|---|
| Use Case Name | Take assessment |
| Participating Actors | Student |
| Flow of Events | ● The user navigates to login page of the E-test Module<br>● The user logs in and enters the secret code for the quiz as made available by the lecturer<br>● The user attempts the questions in the quiz before the time elapses<br>● The user submits quiz |
| Entry Condition | The user must be logged in to his/her E-test account |
| Exit Condition | The user must successfully attempt the test before the time elapses else the system automatically submits on the expiration of the given time |
| Quality Requirement | The user must successfully view the questions in the test and be able to attempt them and submit before the set time expires. |

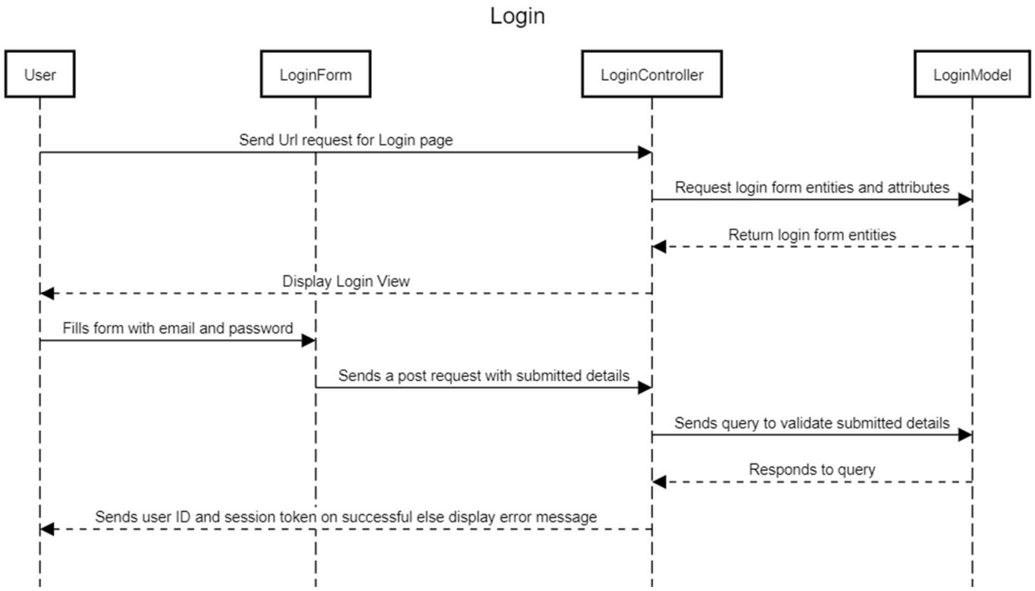

Login
User
LoginForm
LoginController
LoginModel
Send Url request for Login page
Request login form entities and attributes
Return login form entities
Display Login View
Fills form with email and password
Sends a post request with submitted details
Sends query to validate submitted details
Responds to query
Sends user ID and session token on successful else display error message

**Figure 3.3: Sequence diagram for login process**

them with records in the database and if the credentials are valid a session token as well as the user id is sent to the browser and the user is redirected to the dashboard

**b. Create Assessment**

Figure 3.4 shows the order of actions performed by a user logged in with the role of lecturer for the create assessment function. The lecturer clicks on the Create assessment button from the dashboard navigation menu, this sends a request to the assessment controller for the create assessment form which consequently requests for the data entities of assessments from the assessment model. Upon response from the assessment model the assessment controller displays the assessment form in a view on the browser and the user fills the form and submits, thereby sending a post request to the assessment controller with the submitted details. The assessment controller sends an insert query to the assessment model using the submitted details as the parameter and the model executes the query and responds with a message if operation is successful. Assessment successfully created message is then sent to the browser view by the assessment controller.

c. **View report by courses**

Figure 3.5 shows the series of actions performed by a logged in user to view reports by courses. The user selects the course whose report is to be shown from the course list. This action sends a request for the course report view to the report controller which maps the course data entities from the report model in JSON format to the report view for the course selected by the user.

d. **View report by programme**

Figure 3.6 shows the series of actions performed a logged in user to view reports by programme. In the case of the admin user, he/she selects the programme whose report is to be shown from the programme list. This action sends a request for the

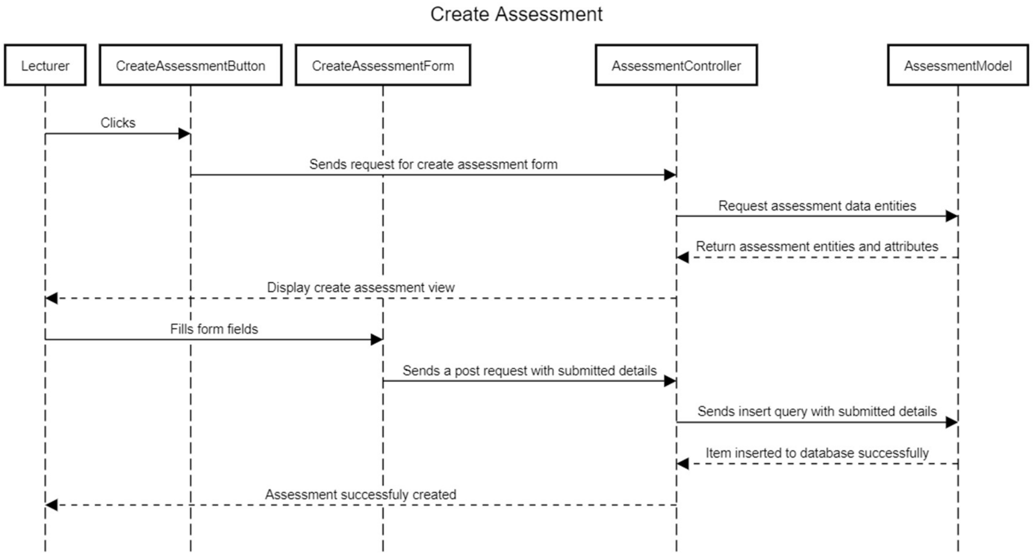

Create Assessment
Lecturer
CreateAssessmentButton
CreateAssessmentForm
AssessmentController
AssessmentModel
Clicks
Sends request for create assessment form
Request assessment data entities
Return assessment entities and attributes
Display create assessment view
Fills form fields
Sends a post request with submitted details
Sends insert query with submitted details
Item inserted to database successfully
Assessment successfuly created

**Figure 3.4: Sequence diagram for create assessment**

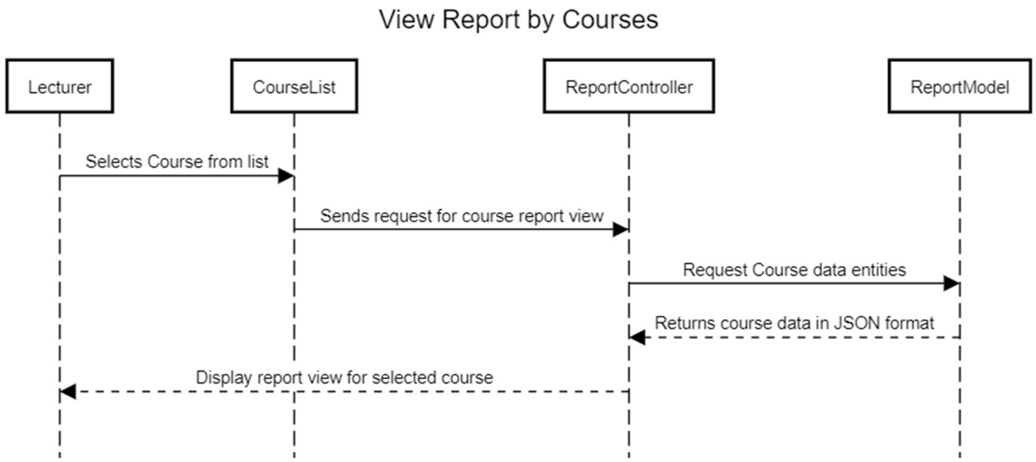

View Report by Courses
Lecturer
CourseList
ReportController
ReportModel
Selects Course from list
Sends request for course report view
Request Course data entities
Returns course data in JSON format
Display report view for selected course

**Figure 3.5: Sequence diagram for view reports by course process**

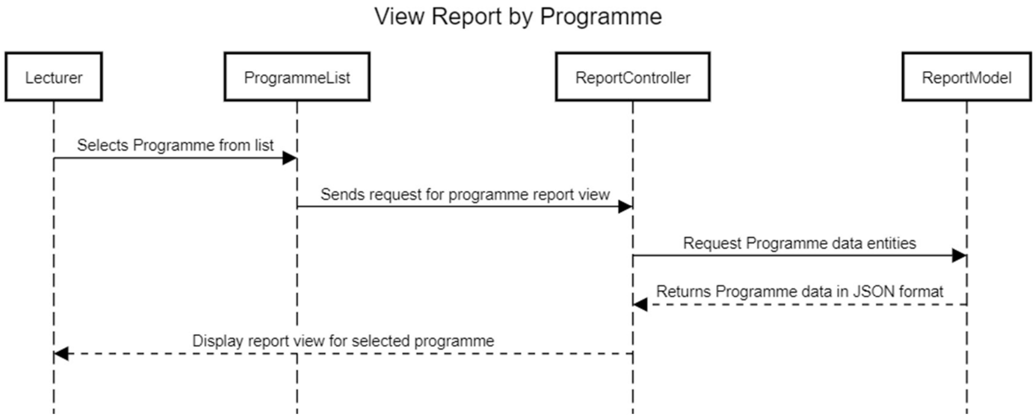

View Report by Programme
Lecturer
ProgrammeList
ReportController
ReportModel
Selects Programme from list
Sends request for programme report view
Request Programme data entities
Returns Programme data in JSON format
Display report view for selected programme

**Figure 3.6: Sequence diagram for view reports by programme process**

programme report view to the report controller which maps the programme report view data entities from the report model in JSON format to the report view for the programme selected by the user while for a user logged in as examination officer, the programme list only contains programme in his/her department and once he selects the programme from the programme list, the system then sends a request to the controller for the programme report view which then maps the data entities and displays the view of the selected programme.

e. **View report by department**

Figure 3.7 shows the series of actions performed a logged in user to view reports by department. In the case of the admin user, he/she selects the department whose report is to be shown from the department list. This action sends a request for the department report view to the report controller which maps the department data entities from the report model in JSON format to the report view for the department selected by the user while for a user logged in as examination officer the system by default sends a request to the controller for the department report view for the user's department and maps the data entities and displays the view for only that department.

f. **View report by level**

Figure 3.8 shows the series of actions performed a logged in user to view reports by level. In the case of the admin user, he/she selects the level whose report is to be shown from the level list. This action sends a request for the level report view to the report controller which maps the level report data entities from the report model in JSON format to the report view for the level selected by the user while for a user logged in as examination officer, once he selects the level from the level list, the system by default sends a request to the controller for the level report view for the user's

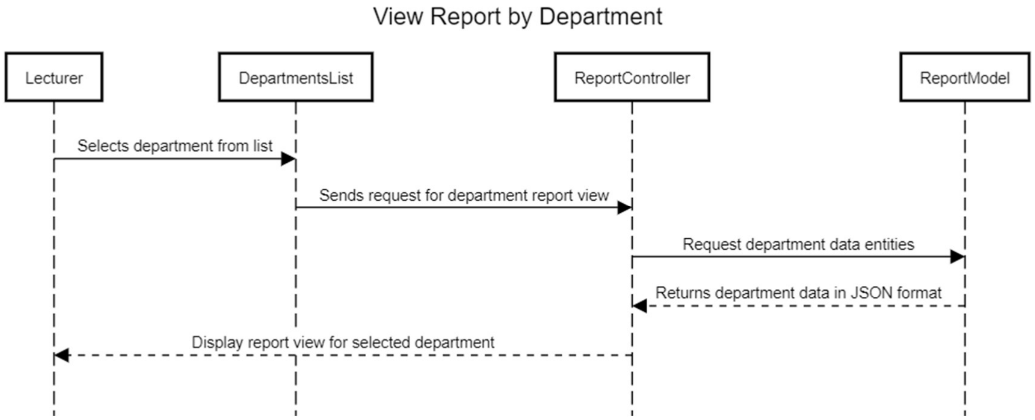

View Report by Department
Lecturer
DepartmentsList
ReportController
ReportModel
Selects department from list
Sends request for department report view
Request department data entities
Returns department data in JSON format
Display report view for selected department

**Figure 3.7: Sequence diagram for view reports by department process**

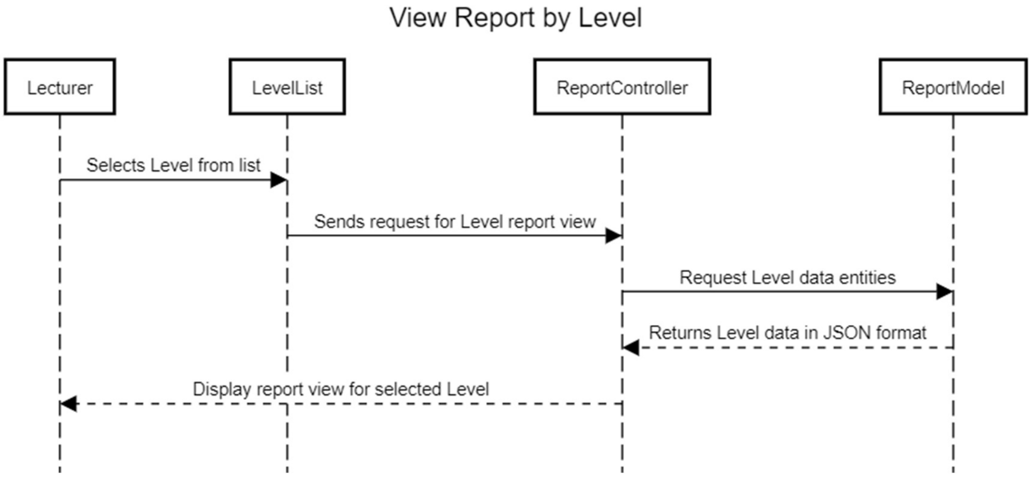

View Report by Level
Lecturer
LevelList
ReportController
ReportModel
Selects Level from list
Sends request for Level report view
Request Level data entities
Returns Level data in JSON format
Display report view for selected Level

**Figure 3.8 : Sequence diagram for view reports by level process**

department and maps the data entities and displays the view of the selected level for only that department.

g. **View report by assessment**

Figure 3.9 shows the series of actions performed a logged in user to view reports by assessment. The user selects the assessment whose report is to be shown from the assessment list. This action sends a request for the assessment report view to the report controller which maps the assessment report view data entities from the report model in JSON format to the assessment report view for the assessment selected by the user.

### 3.2.4 Activity diagram

The activity diagram shown in figure 3.10 illustrated the flow of activity within the system for all the roles existing in the system. A user must first log in with his/her credentials which is then check if it is valid or not it is valid the user proceed to the authorization process else the user has to retry the login. The authorization process checks the role of the logged in user and directs the user to the functions that are allowed for his/her role. A user with the admin role can then perform action such as role assignment, deleting existing users, view dashboard for courses, programme, departments and levels. A user logged in as an examination officer can view dashboards for courses, programme and level while a user with the lecturer role can create quiz, view the performance of student in quizzes and view a dashboard report of all assessment for his/her courses.

### 3.2.5 Class diagram

The class diagram of the reporting system as shown in figure 3.11 shows the classes for users (lecturer, examination officer and admin), students, quiz, course, questions, assessment, option and optionSelected. The user class represents the model

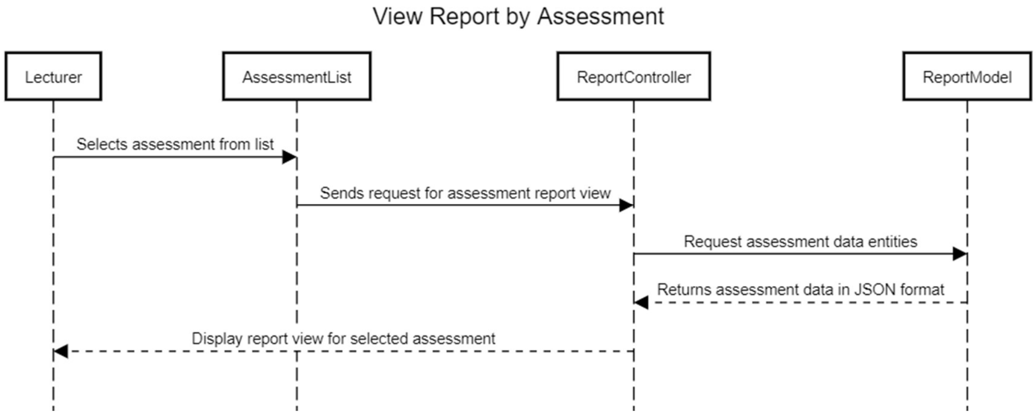

View Report by Assessment
Lecturer
AssessmentList
ReportController
ReportModel
Selects assessment from list
Sends request for assessment report view
Request assessment data entities
Returns assessment data in JSON format
Display report view for selected assessment

**Figure 3.9: Sequence diagram for view reports by assessment process**

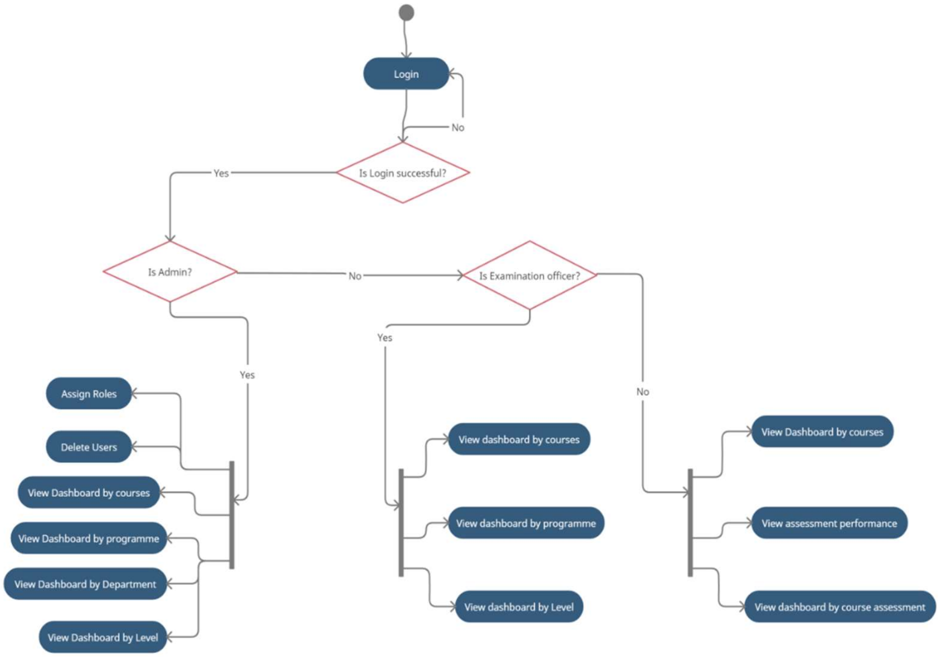

Login
No
Is Login successful?
Yes
Is Admin?
No
Is Examination officer?
Yes
Yes
No
Assign Roles
Delete Users
View Dashboard by courses
View Dashboard by programme
View Dashboard by Department
View Dashboard by Level
View dashboard by courses
View dashboard by programme
View dashboard by Level
View Dashboard by courses
View assessment performance
View dashboard by course assessment

**Figure 3.10: Activity diagram**

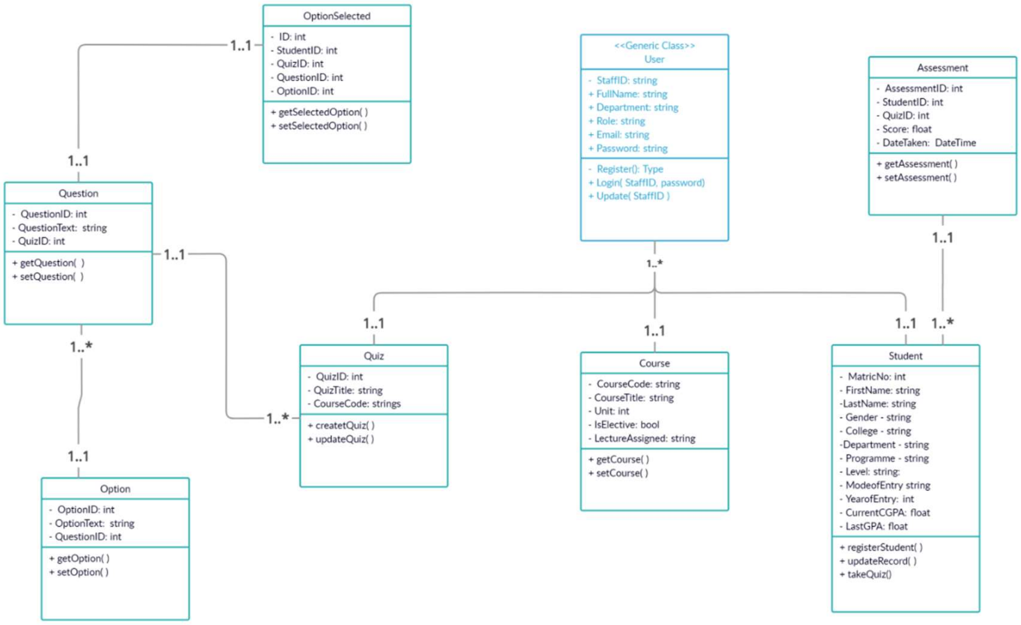


**Figure 3.11: Class diagram**

of admin and basic user in the system with properties StaffID, FullName, Department, Role, Email and password. The properties of this class form the columns of the user table in the database used by the system. Methods for user registration, login and record update are available for this class.

The student class instantiates the record of each student used in the reports generated by this system, it contains properties such as MatricNo, FirstName, LastName, Gender, College, Department, Programme, Level, ModeofEntry, YearofEntry, CurrentCGPA and LastGPA. Other classes such as quiz, course, option and optionSelected were used to build a simple E-test module in the system that allowed quiz creation by lecturers and attempt of the quiz by student registered for the course. The Quiz class has properties: QuizID, QuizTitle and CourseCode.

## 3.3 System Implementation

The system was implemented using the following tools and techniques. The database of the system was implemented using MS SQL server alongside entity framework. The fronted of the application which comprises of the user interface as well as the application logic were implemented using Microsoft's ASP.NET MVC 5, Microsoft ASP.NET Identity framework and visualization libraries such as Apex chart JS, Chart JS etc.

### 3.3.1 Database Implementation

#### a. MSSQL server

The database of the system was developed using MS SQL server's "." instance connected with Microsoft visual studio IDE via a configuration string that specifies the server engine instance to connect to and the credentials. MS SQL server was used to manage all database operation including, storage and retrieval of all records used by the system locally. It was also used to hold stored procedures for special queries that needed guarantee of execution by returning a promise and prevention of SQL injection. The database was also manually managed using MS SQL server management studio during implementation and testing.

b. **Entity Framework**

Entity framework is an object-relation mapping framework that allows development of system models and caters for the creating of tables that correlate to the model automatically. It was used in the implementation of this system to manage table creation in the database by running migration commands. It was also used to resolve type checking and validation of the data sent from the user interface and the corresponding column in the database

### 3.3.2 Front-end Implementation:

a. **ASP.NET MVC**

ASP.NET MVC was used as the framework for the development of the system. It was used to structure the system functions into views, models and controllers. The views were created as .NET files with ". cshtml" extension and were used to create the pages of the system. The models and controllers were created as .NET files with ".cs" extension containing C# classes for server-side data and data control and manipulation respectively. Each web page in the system is a view returned by a controller and has a corresponding model.

b. **Razor**

Razor is markup syntax than is available in ASP.NET MVC. It was used to embed server-side code into web pages (view files). It supports C# and uses @ symbol to transition between HTML markup and C# code so that it evaluates expressions written in C# and renders the output in HTML.

c. **Cascading Style Sheet (CSS)**

This was used to style the markup written in the view files. This was done by using the HTML link tag and ASP.NET script bundler to make style classes in the CSS files to be accessible within the view file.

d. **C#**

C# was used as the scripting language in this project. It is one of the languages supported in ASP.NET hence it was used to write the server-side code of the system including the models and controller classes.

e. **Apex charts**

Apex charts is a JavaScript charting library that was used to create some visualizations in the dashboards in the system. The library was used to create bar graphs and pie charts in different dashboards by sending the data for the report in JSON format to the appropriate class from the library.

f. **Chart JS**

Chart JS is another JavaScript library that was used in the system to create line graphs and grouped bar charts on various dashboards. It received data from the report controller in JSON and used them to create the chart.

g. **ASP.NET identity framework**

This is a claim-based authentication framework that was used to manage user registrations login, password reset and user details management. It was also user to manage user roles within the system.

# CHAPTER FOUR

# IMPLEMENTATION AND RESULT

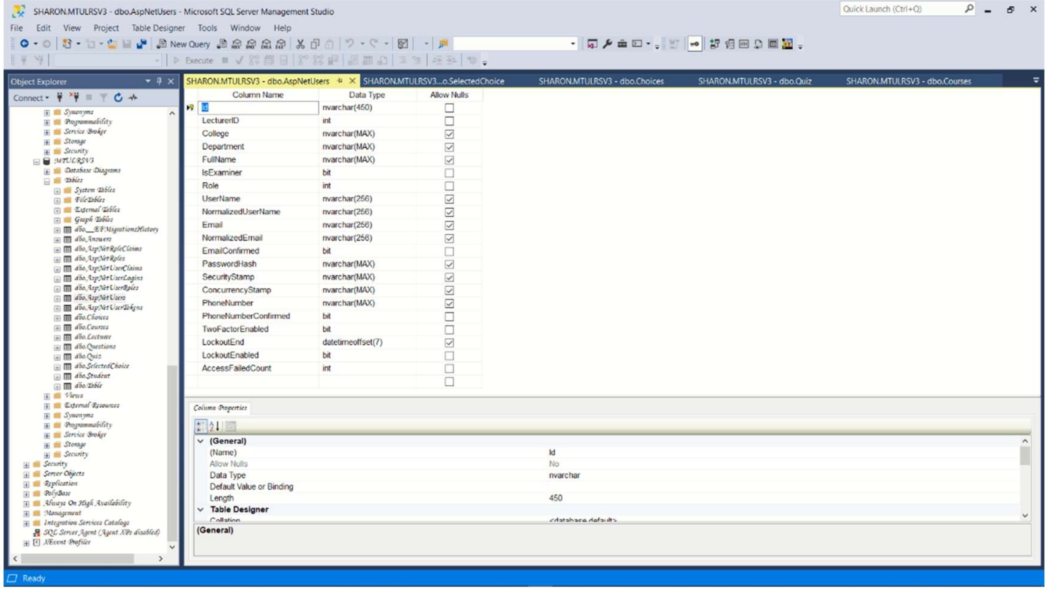
This section presents the results of the system and a discussion of the results. It covers the results of the database of the system that was implemented using MS SQL server as well as the user interface of the web application for the learning reporting system which was implemented using web technologies such as: ASP.NET MVC, CSS and some JavaScript charting libraries. It also presents the user interface of the E-test module of the system that was implemented using HTML, CSS and JS.

## 4.1 Database Implementation

Figure 4.1 shows the result of the user table as created in the database by ASP.NET identity framework through entity framework migration command which was used for user table and user data management. The table contains columns: ID which is the primary key and LecturerId which is a candidate key in the table. Other columns in the table include: college, department, fullName, role, email. This table holds the data of each user of the learning reporting system namely: lecturers, examination officer and admin user.

Also related to the user table is the roles table shown in figure 4.2. It was also created by ASP.NET identity framework with entity framework migration command and is used for holding roles data. It contains columns for roleID, roleName and concurrency stamp that is auto-generated by the migration command.

Figure 4.3 shows the result of the student table in the SQL server management studio. It contains columns: MatricNo, FirstName, LastName, College, Department, Programme, Email, YearofEntry, ModeofEntry, CurrentCGPA, LastGPA and AcademicStanding. The primary key for this table is MatricNo based on the

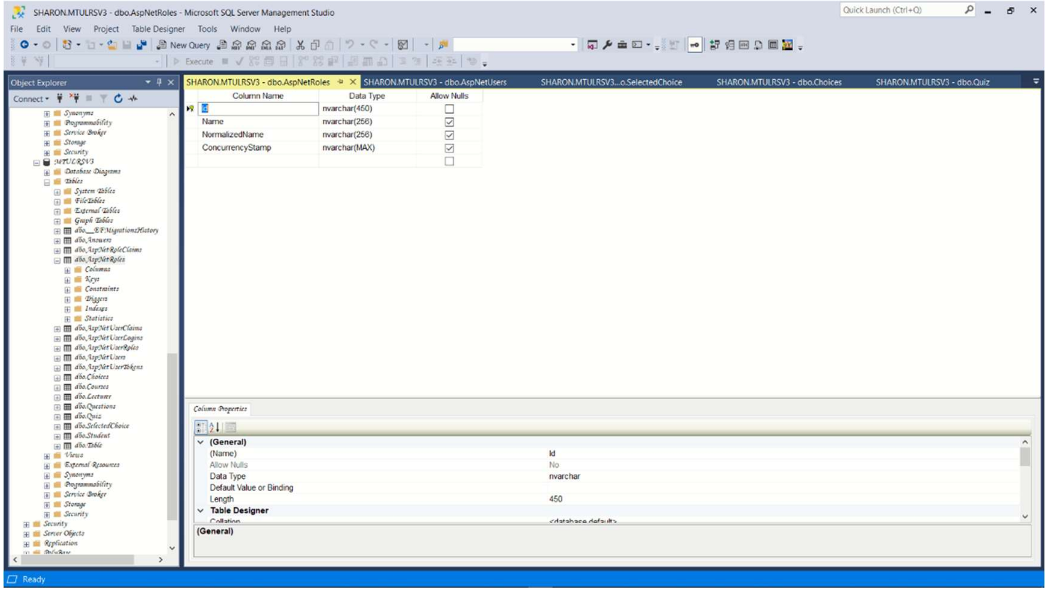


| Column Name | Data Type | Allow Nulls |
|---|---|---|
| Id | nvarchar(450) | ☐ |
| LecturerID | int | ☐ |
| College | nvarchar(MAX) | ☑ |
| Department | nvarchar(MAX) | ☑ |
| FullName | nvarchar(MAX) | ☑ |
| IsExaminer | bit | ☐ |
| Role | int | ☐ |
| UserName | nvarchar(256) | ☑ |
| NormalizedUserName | nvarchar(256) | ☑ |
| Email | nvarchar(256) | ☑ |
| NormalizedEmail | nvarchar(256) | ☑ |
| EmailConfirmed | bit | ☐ |
| PasswordHash | nvarchar(MAX) | ☑ |
| SecurityStamp | nvarchar(MAX) | ☑ |
| ConcurrencyStamp | nvarchar(MAX) | ☑ |
| PhoneNumber | nvarchar(MAX) | ☑ |
| PhoneNumberConfirmed | bit | ☐ |
| TwoFactorEnabled | bit | ☐ |
| LockoutEnd | datetimeoffset(7) | ☑ |
| LockoutEnabled | bit | ☐ |
| AccessFailedCount | int | ☐ |
| | | ☐ |

| | |
|---|---|
| **(General)** | |
| (Name) | Id |
| Allow Nulls | No |
| Data Type | nvarchar |
| Default Value or Binding | |
| Length | 450 |
| **Table Designer** | |
| Collation | <database default> |

**(General)**

Figure 4.1: Users Table


SHARON.MTULRSV3 - dbo.AspNetRoles - Microsoft SQL Server Management Studio
File Edit View Project Table Designer Tools Window Help
New Query
Execute
Object Explorer
Connect
SHARON.MTULRSV3 - dbo.AspNetRoles
SHARON.MTULRSV3 - dbo.AspNetUsers
SHARON.MTULRSV3...o.SelectedChoice
SHARON.MTULRSV3 - dbo.Choices
SHARON.MTULRSV3 - dbo.Quiz
Quick Launch (Ctrl+Q)
Ready

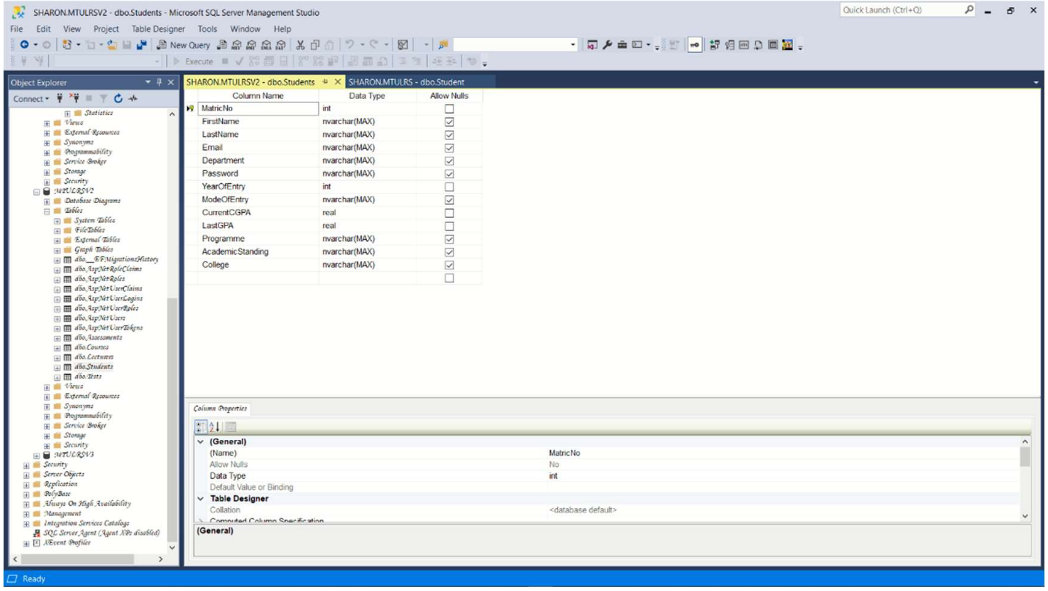
| Column Name | Data Type | Allow Nulls |
|---|---|---|
| Id | nvarchar(450) | □ |
| Name | nvarchar(256) | ☑ |
| NormalizedName | nvarchar(256) | ☑ |
| ConcurrencyStamp | nvarchar(MAX) | ☑ |
|  |  | □ |

Column Properties

| | |
|---|---|
| **(General)** | |
| (Name) | Id |
| Allow Nulls | No |
| Data Type | nvarchar |
| Default Value or Binding | |
| Length | 450 |
| **Table Designer** | |
| Collation | <database default> |

**(General)**

Figure 4.2: Roles Table



| Column Name | Data Type | Allow Nulls |
|---|---|---|
| MatricNo | int | ☐ |
| FirstName | nvarchar(MAX) | ☑ |
| LastName | nvarchar(MAX) | ☑ |
| Email | nvarchar(MAX) | ☑ |
| Department | nvarchar(MAX) | ☑ |
| Password | nvarchar(MAX) | ☑ |
| YearOfEntry | int | ☐ |
| ModeOfEntry | nvarchar(MAX) | ☑ |
| CurrentCGPA | real | ☐ |
| LastGPA | real | ☐ |
| Programme | nvarchar(MAX) | ☑ |
| AcademicStanding | nvarchar(MAX) | ☑ |
| College | nvarchar(MAX) | ☑ |
|  |  | ☐ |

| (General) |  |
|---|---|
| (Name) | MatricNo |
| Allow Nulls | No |
| Data Type | int |
| Default Value or Binding |  |
| **Table Designer** |  |
| Collation | <database default> |
| Computed Column Specification |  |

(General)

**Figure 4.3: Students Table**

observation of the uniqueness of the matriculation number for each students' academic record.

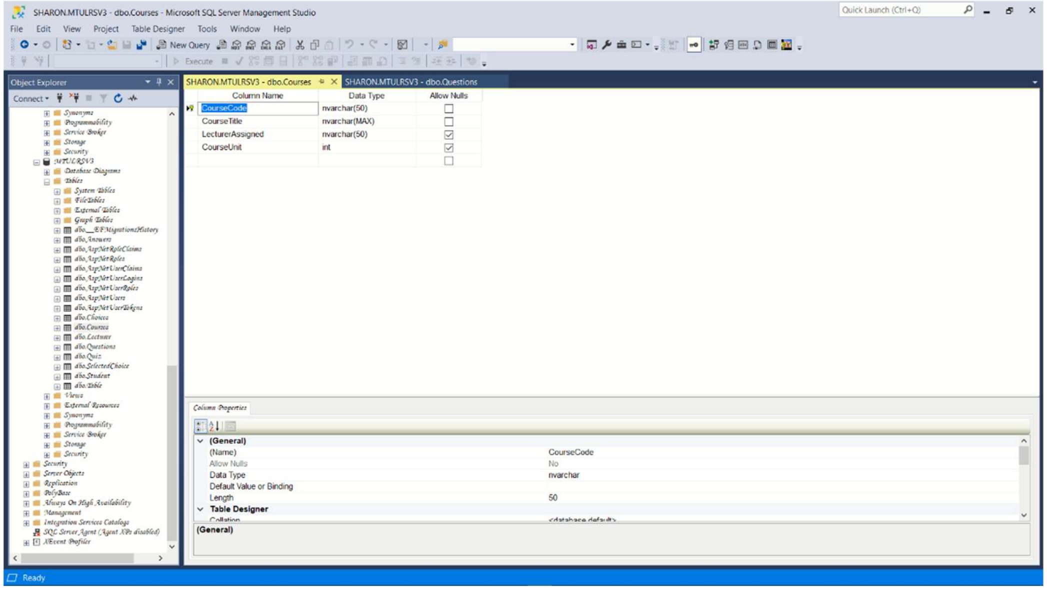
Figure 4.4 show the result of the courses table that was used in the system to hold data for each course. The table contain columns: CourseCode which is the primary key, CourseTitle, CourseUnit and LecturerAssigned.

Figure 4.5 presents the result of the quiz table that holds the record of quizzes for courses. It contains columns for: QuizID, QuizName and CourseCode. The quiz table works along with other tables such as Questions and Choices which holds records of question, options and options selected by users respectively for each quiz. The Questions table shown in figure 4.6 contains columns: QuestionID, QuestionText and QuizID which references the primary key of the Quiz table. The choices table shown in figure 4.7 also has columns: ChoiceID, ChoiceText and QuestionID with QuestionID been a foreign key in this table.

## 4.2 Front-end Implementation

Figure 4.8 shows the result of the login interface of the learning reporting system. It displays a form that has fields for email address and password as well as perform validation of form entry function to ensure user enters details and the validity of the details entered by the user in the fields. Once the user clicks the login button the user is directed to the dashboard that is authorized for his/her role.

Figure 4.9 shows the result of the admin dashboard after a successful login. The interface displays reports in form of charts of academic status for all levels, student performance per department, grade point distribution across level for each department, courses pass and failure rates, lecturer performance based on performance in the course taught by the lecturer and tables showing list of students by programme,

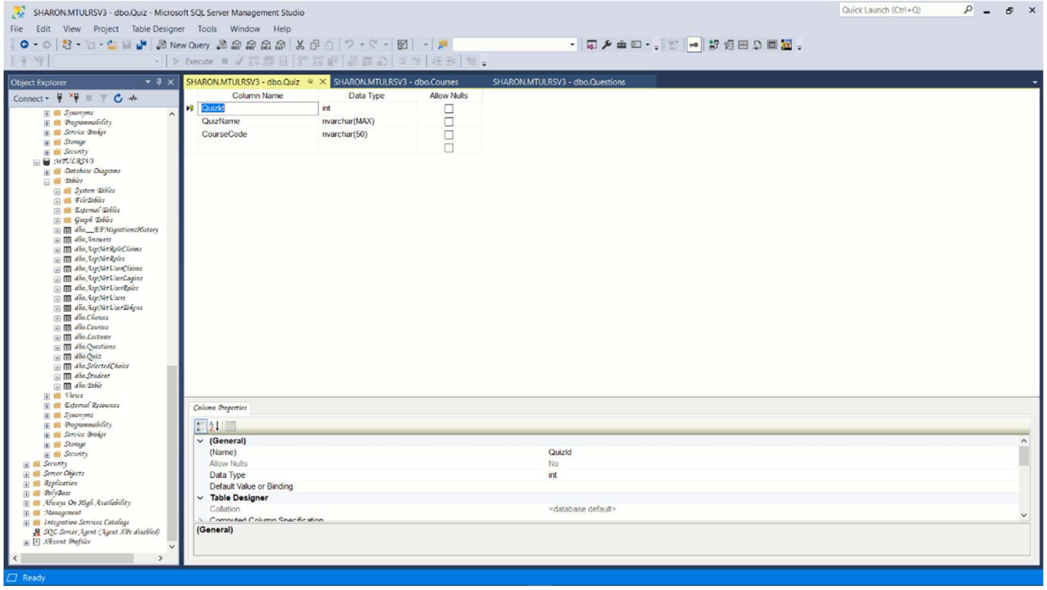


| Column Name | Data Type | Allow Nulls |
|---|---|---|
| CourseCode | nvarchar(50) | ☐ |
| CourseTitle | nvarchar(MAX) | ☐ |
| LecturerAssigned | nvarchar(50) | ☑ |
| CourseUnit | int | ☑ |
|  |  | ☐ |

| | |
|---|---|
| **(General)** | |
| (Name) | CourseCode |
| Allow Nulls | No |
| Data Type | nvarchar |
| Default Value or Binding | |
| Length | 50 |
| **Table Designer** | |
| Collation | <database default> |

**(General)**

Figure 4.4: Courses Table

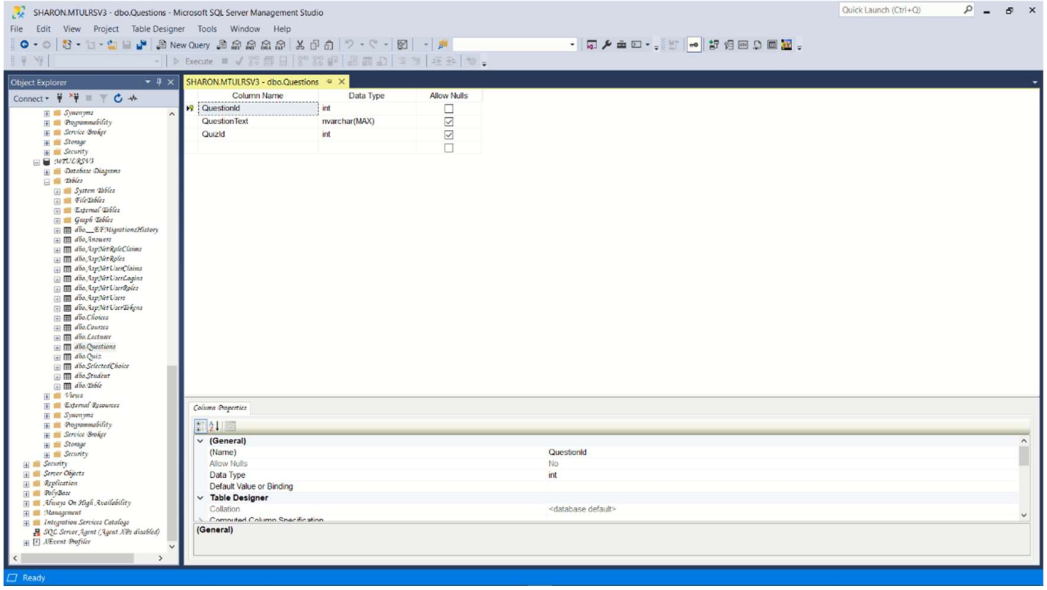

SHARON.MTULRSV3 - dbo.Quiz - Microsoft SQL Server Management Studio
File Edit View Project Table Designer Tools Window Help
New Query
Execute
Object Explorer
Connect
SHARON.MTULRSV3 - dbo.Quiz
SHARON.MTULRSV3 - dbo.Courses
SHARON.MTULRSV3 - dbo.Questions
Column Properties
Ready

| Column Name | Data Type | Allow Nulls |
|---|---|---|
| QuizId | int | □ |
| QuizName | nvarchar(MAX) | □ |
| CourseCode | nvarchar(50) | □ |
| | | □ |

| | |
|---|---|
| (General) | |
| (Name) | QuizId |
| Allow Nulls | No |
| Data Type | int |
| Default Value or Binding | |
| Table Designer | |
| Collation | <database default> |
| Computed Column Specification | |

(General)

Figure 4.5: Quiz Table

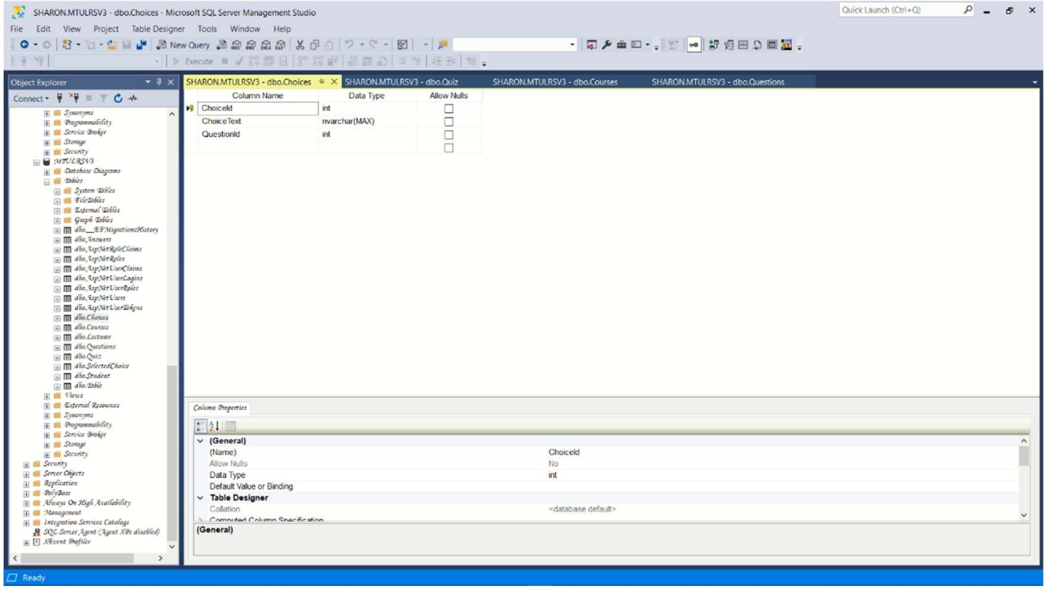


| Column Name | Data Type | Allow Nulls |
|---|---|---|
| QuestionId | int | □ |
| QuestionText | nvarchar(MAX) | ☑ |
| QuizId | int | ☑ |
| | | □ |

(General)
(Name) QuestionId
Allow Nulls No
Data Type int
Default Value or Binding
Table Designer
Collation <database default>
Computed Column Specification

(General)

Figure 4.6: Questions Table



| Column Name | Data Type | Allow Nulls |
|---|---|---|
| ChoiceId | int | □ |
| ChoiceText | nvarchar(MAX) | □ |
| QuestionId | int | □ |
| | | □ |

(General)

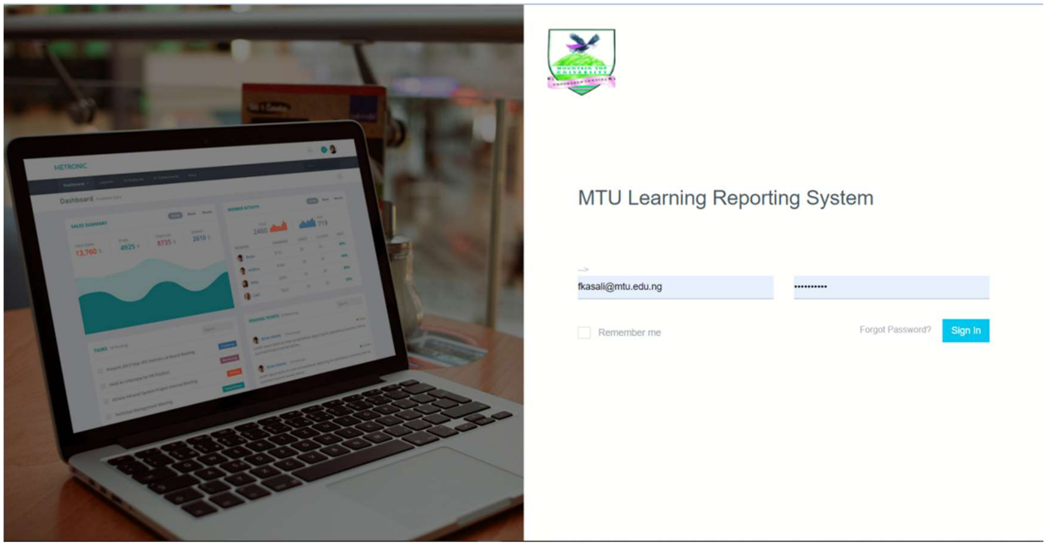
| | |
|---|---|
| (Name) | ChoiceId |
| Allow Nulls | No |
| Data Type | int |
| Default Value or Binding | |

Table Designer

| | |
|---|---|
| Collation | <database default> |
| Computed Column Specification | |

(General)

Figure 4.7: Choices Table

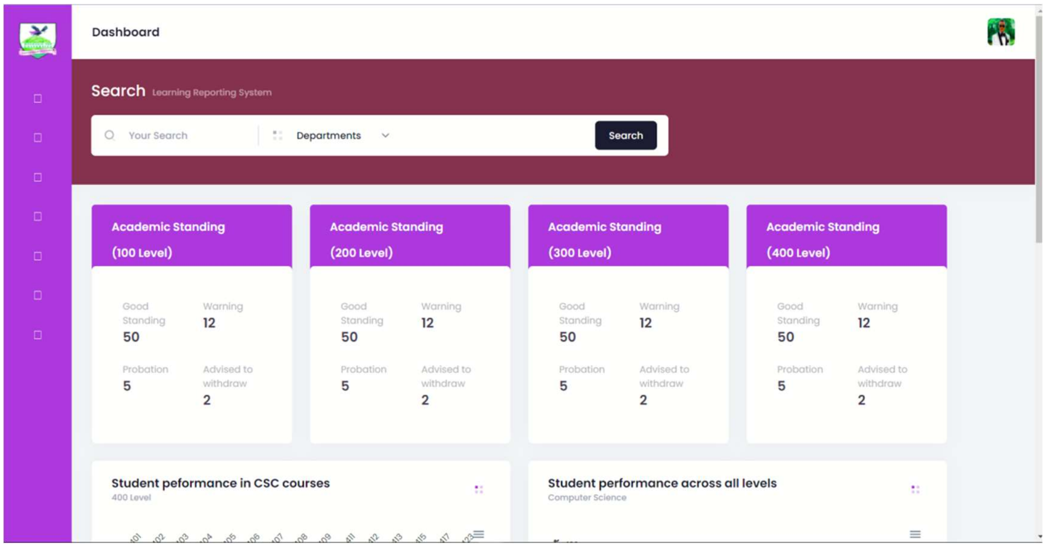


MTU Learning Reporting System

fkasali@mtu.edu.ng

••••••••••

**Figure 4.8: Login interface**



Academic Standing (100 Level)

Good Standing 50 | Warning 12 | Probation 5 | Advised to withdraw 2

Academic Standing (200 Level)

Good Standing 50 | Warning 12 | Probation 5 | Advised to withdraw 2

Academic Standing (300 Level)

Good Standing 50 | Warning 12 | Probation 5 | Advised to withdraw 2

Academic Standing (400 Level)

Good Standing 50 | Warning 12 | Probation 5 | Advised to withdraw 2

Student peformance in CSC courses
400 Level

Student performance across all levels
Computer Science

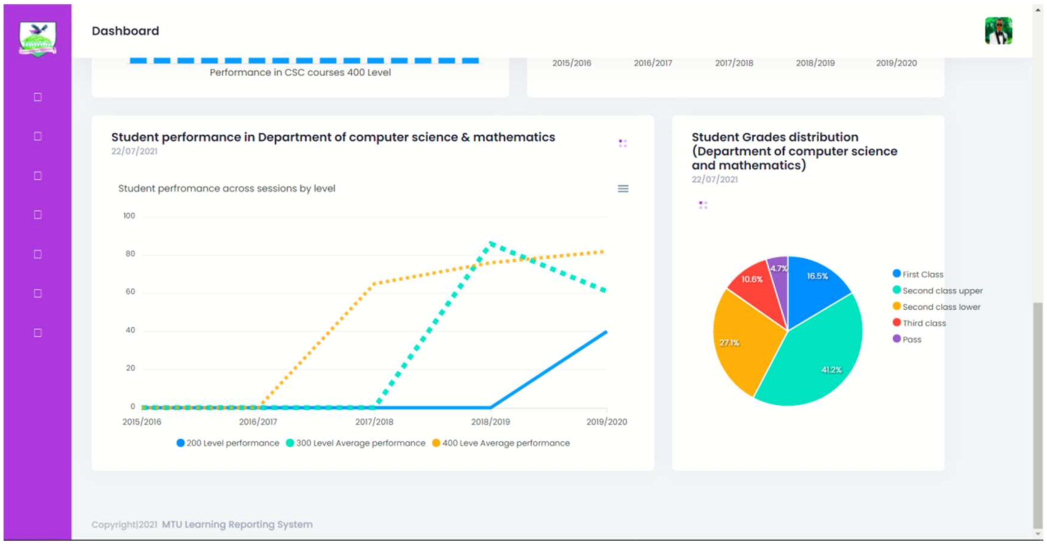
**Figure 4.9: Admin dashboard**

department, and college in order of their cumulative grade point average (CGPA) among other things.

Figure 4.10 shows the interface for the examination officer after successfully logging into the system. The interface contains reports for academic status of students, average performance per course, average performance per level, average performance per programme, grade point distribution per level, courses failure and pass rate, performance of lecturers based on student performance in the courses they teach, performance of each student in all the courses/he or she registered and a table showing the list of students per programme in the examination officer's department in order of their CGPA. This interface displays reports for only data relating to the examination officer's department.

Figure 4.11 shows the result of the interface for lecturers after a successful login into the system. It displays reports of average performance for the courses taken by the lecturer, list of scores per assessment for each course taken by the lecturer and reports of right and wrong response per question in each assessment. It allows the lecturer to get detailed information on student's response and performance in his/her courses.

Figure 4.12 shows the result of the interface for creating assessment by lecturers for their courses. This interface contains a form that contains fields for setting a course title for a selected course, as well as fields for lists of question and their corresponding options. Once the user submits this assessment creation form the fields of this form are validated if they are correctly filled and are then sent to the database.



Dashboard

Performance in CSC courses 400 Level

Student performance in Department of computer science & mathematics
22/07/2021

Student perfromance across sessions by level

Student Grades distribution (Department of computer science and mathematics)
22/07/2021

Copyright|2021 MTU Learning Reporting System

Figure 4.10: Examination officer dashboard

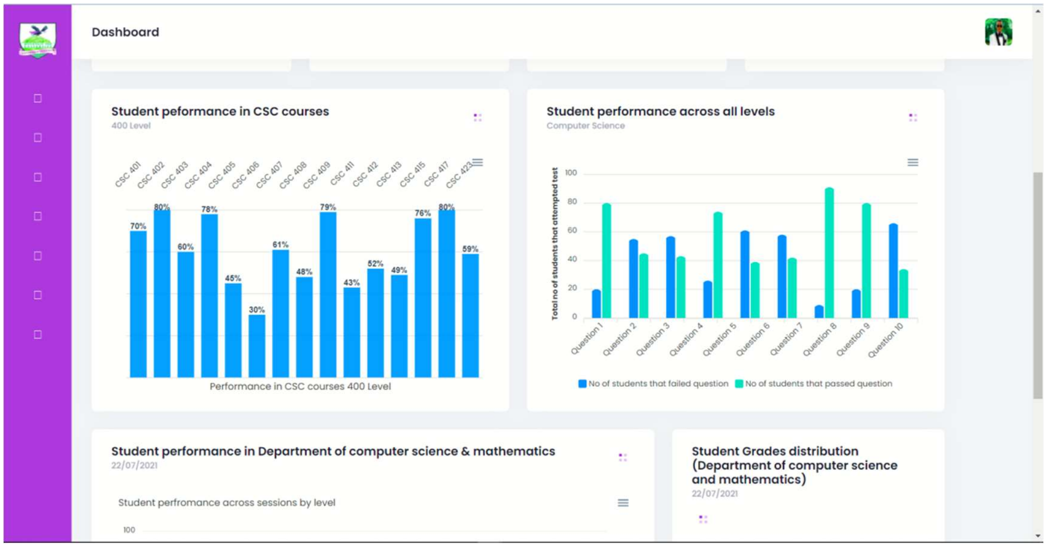
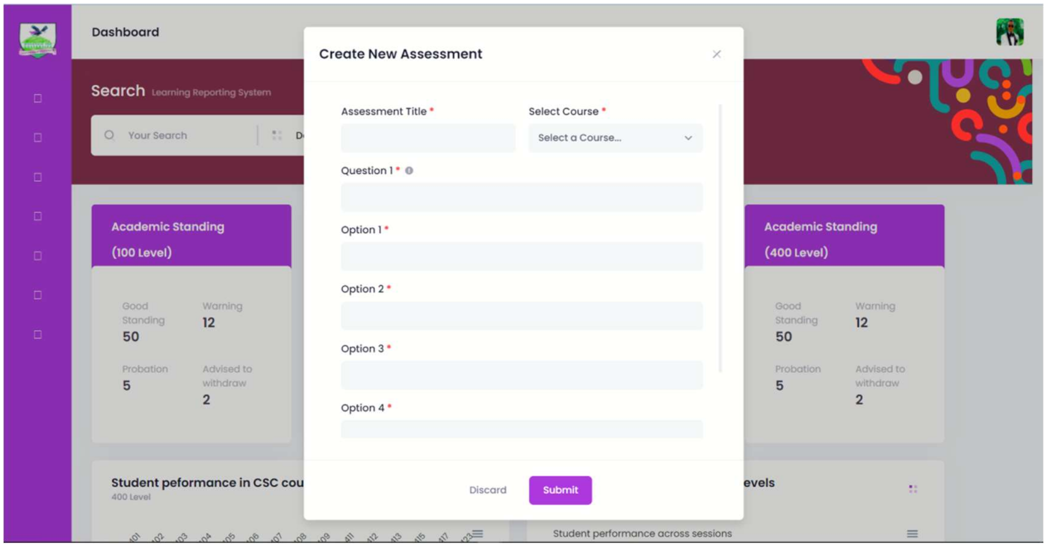

Dashboard
Student peformance in CSC courses
400 Level
CSC 401
CSC 402
CSC 403
CSC 404
CSC 405
CSC 406
CSC 407
CSC 408
CSC 409
CSC 411
CSC 412
CSC 413
CSC 415
CSC 417
CSC 423
70%
80%
60%
78%
45%
30%
61%
48%
79%
43%
52%
49%
76%
80%
59%
Performance in CSC courses 400 Level
Student performance across all levels
Computer Science
Total no of students that attempted test
100
80
60
40
20
0
Question 1
Question 2
Question 3
Question 4
Question 5
Question 6
Question 7
Question 8
Question 9
Question 10
No of students that failed question
No of students that passed question
Student performance in Department of computer science & mathematics
22/07/2021
Student perfromance across sessions by level
100
Student Grades distribution (Department of computer science and mathematics)
22/07/2021

**Figure 4. 11: Lecturer dashboard**



Create New Assessment

Assessment Title *

Select Course *

Select a Course...

Question 1 *

Option 1 *

Option 2 *

Option 3 *

Option 4 *

Discard

Submit

**Figure 4. 12: Create assessment interface**

Figure 4.13 show the result of the login interface for the E-test module which contains a form with fields for students' matric number and the quiz password as set by the lecturer. Once the student clicks login and is successfully authenticated he/she is directed to the quiz page as shown in figure 4.14 where questions for the logged in quiz are displayed with their respective options. The interface is simple and user-friendly as it immediately notifies student of right and wrong choice by color variations as shown in figure 4.15(a) and 4.15(b) respectively.

## 4.3 Discussion of Result

The result of the study as presented, shows the different expectation of this study based on the objectives that were stated in the earlier chapters of this study. The results of the identification of the system and user requirements allowed for the identification of the different users of the proposed system such as primary and secondary users of the system duly defined as lecturers, examination officer and administrator for primary users and students as the secondary users of the system.

The results shows that the admin user was responsible for the oversight function on the system having access to every kind of report generated on the system. The results also shows that the users can only access the system using their school email address and password for example admin@mtu.edu.ng was used for the admin user while other users used their respective school email address and password provided by the institutions' system administrator. The result also shows that the user can view various reports on their dashboard based on the roles that they have, however the assessment creation function is only accessible to examination officers and lecturers by virtue of their academic function. The results shows that the approach adopted by this was able to cater for the system and user requirements as defined during the requirement identification phase of this study. The result of the system implementation showed

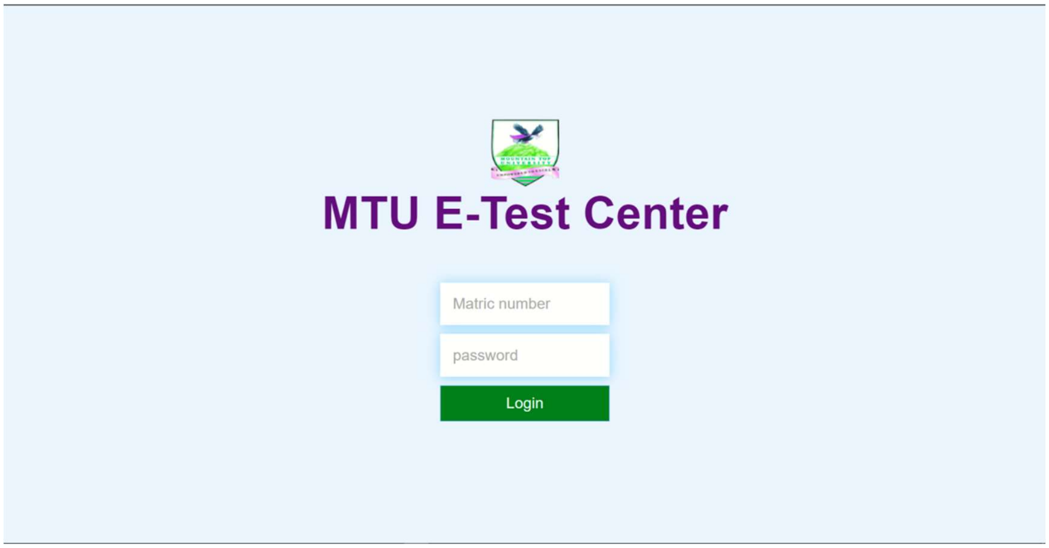
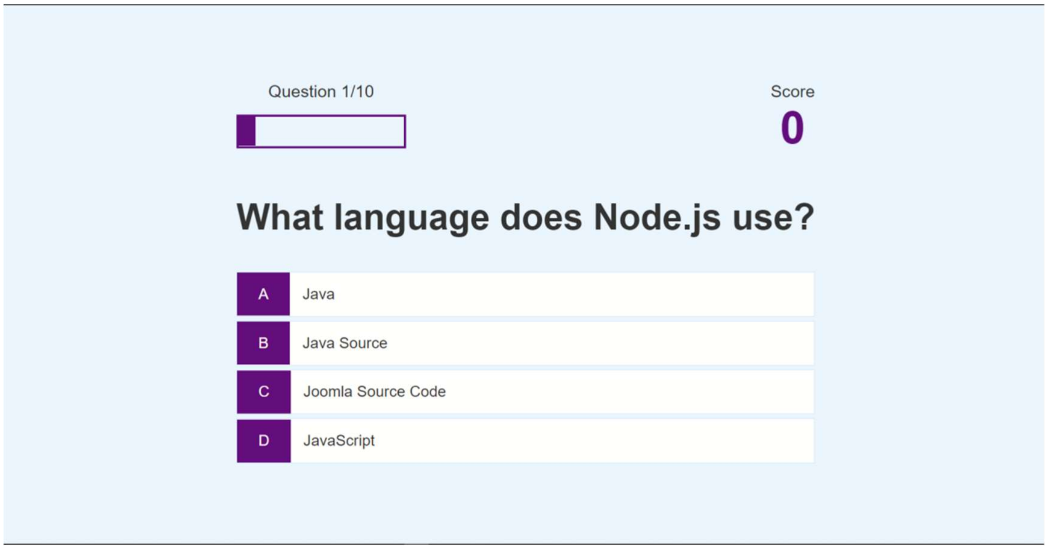


Figure 4. 13: E-Test login interface



Question 1/10

Score

0

What language does Node.js use?

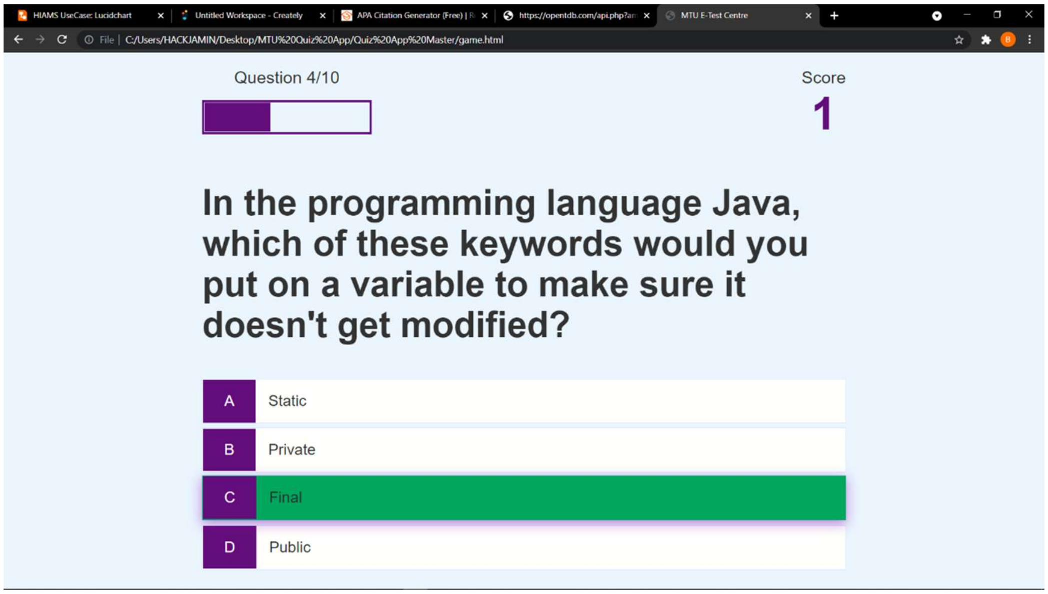
A Java

B Java Source

C Joomla Source Code

D JavaScript

**Figure 4. 14: Quiz interface**


HIAMS UseCase: Lucidchart
Untitled Workspace - Creately
APA Citation Generator (Free) | R
https://opentdb.com/api.php?am
MTU E-Test Centre
File | C:/Users/HACKJAMIN/Desktop/MTU%20Quiz%20App/Quiz%20App%20Master/game.html

Question 4/10

Score
1

In the programming language Java, which of these keywords would you put on a variable to make sure it doesn't get modified?

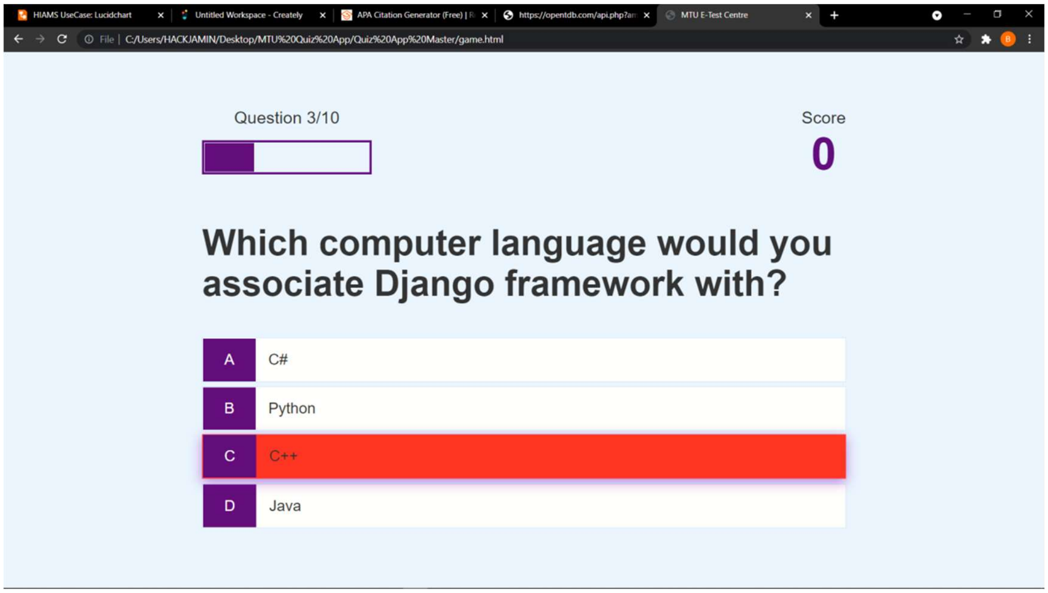
A Static

B Private

C Final

D Public

**Figure 4. 15 (a): Quiz interface when user selects right option**


HIAMS UseCase: Lucidchart
Untitled Workspace - Creately
APA Citation Generator (Free) | R
https://opentdb.com/api.php?am
MTU E-Test Centre
File | C:/Users/HACKJAMIN/Desktop/MTU%20Quiz%20App/Quiz%20App%20Master/game.html

Question 3/10

Score
0

Which computer language would you associate Django framework with?

A C#

B Python

C C++

D Java

**Figure 4. 16 (b): Quiz interface when user selects wrong option**

the database of the system was able to manage the data storage need. For each table in the database a significant number of records were inserted and retrieved during report generation.

The result of the web interface shows that the system was able to provide an interface that efficiently suits the system and user requirement that were identified in the course of the study. The system implementation allows users of various roles to perform their various academic and administrative functions using the systems' report removing the challenges of manual data analysis and vague decision-making process.

# CHAPTER FIVE

# SUMMARY, CONCLUSION AND RECOMMENDATION

## 5.1 Summary

This study developed a reporting system of student academic records that enables lecturers, examination officer, administrative staff and other stake holders of higher institutions to effectively monitor performance metrics of student and staff alike which are key to the process of decision making and subsequent policy implementation towards achieving their set goals. The study identified the user and system requirements that were necessary to be met by the system which were identified alongside the software and hardware requirements of the system.

The identified requirements were further specified using relevant unified modelling language diagrams such as use case diagram, activity diagram and sequence diagrams for the user requirements as well as class diagram for data modelling. The system was developed using Microsoft's ASP.NET MVC framework for the frontend as well as packages such as identity and identity for implementing various features. It also made use of MS SQL server and entity framework for the management database of the system.

## 5.1 Conclusion

In conclusion, this study has designed and implemented a reporting system that solves the challenges in analysis of student academic record and informed decision making in higher institutions. The study was able to identify the respective user and system requirements and the appropriate designs were used to specify these requirements provided by the users using UML diagrams. The database was implemented to suit the workings of the proposed system.

## 5.2 Recommendation

Following the conclusion of this study, I hereby recommend the adoption of this reporting system by higher institution with an existing student academic record system to drive meaningful discussions that are influenced by visual reports of from statistical analysis. I also recommend that future works in this area be done to integrate various additional performance metrics that are not covered within this study as well as the adoption of machine learning methods to reveal important patterns that are not discovered by the statistical methods adopted by this work.

## References

Abduldaem, A., & Gravell, A. (2019). PRINCIPLES FOR THE DESIGN AND DEVELOPMENT OF DASHBOARDS: LITERATURE REVIEW.

Ali, I. H. (2018). Design and implement a Novel Student Information Management System – Case Study. *International Journal of Computer Science and Mobile Computing*, *7*(7), 20–31.

Alter, S. Decision Support Systems: Current Practice and Continuing Challenges. Reading, Mass.: Addison-Wesley, Inc., 1980.

Ambler, S. (2000). *How the UML Models Fit Together*. DrDobbs. http://www.sdmagazine.com/articles/2000/003/003z/003z1.htmp?topic=uml

Ap-Azli, B., Safawi, A., Mohd Razilan, A., Mohd, E., & Mohd, N. (2016). *Is it a document? Or is it a record?* [E-book].

Asemi, A., Safari, A., & Zavareh, A. A. (2011). The Role of Management Information System (MIS) and Decision Support System (DSS) for Manager's Decision Making Process. *International Journal of Business and Management*, 6(7), 164–173.

Barjtya, S., Sharma, A., & Rani, U. (2017). A detailed study of Software Development Life Cycle (SDLC) Models. *International Journal of Engineering and Computing Science*, *6*(7). http://www.ijecs.in/index.php/ijecs/article/view/2830

Beck, K. (2000). *Extreme programming explained: Embrace change*. Addison Wesley Longman, Inc.

Boehm, B. (1968). A Spiral Model of Software Development and Enhancement. *ACM SIGSOFT Software Engineering Notes*, *11*(4), 14–24.

Boehm, B. (2000) 'Spiral Development: Experience, Principles and Refinements', Carnegie Mellon Software Engineering Institute, Special Report CMU/SEI-2000-SR-008.

Booch, G., Rumbaugh, J., & Jacobson, I. (1999). The Unified Modeling Language User Guide. Reading, MA: Addison Wesley.

Booch, G., Rumbaugh, J., & Jacobson, I. (2005). *The unified modeling language user guide* (2nd ed.). Reading MA: Addison Wesley.

Buckingham, R. A., Hirschheim, R. A., Land, F. F., & Tully, C. J. (1987). Information systems education: Recommendations and implementation. *Cambridge University Press*, 204–214.

Budhrani, K., & Espiritu, J. (2019). Cultivating an e-learning culture. *Scientia Pedagogica Experimentalis*, *56*(1), 3–32.

Choudhari, J., & Suman, U. (Eds.). (2010). *Iterative Maintenance Life cycle using Extreme Programming*. Communication and Computing.

Dada, O. M., Raji, A. K., Oyedepo, F. M., Yusuf, I. T., & Saka, T. O. (2017). Design and Implementation of an Integrated Result Processing System in a Networked Environment. *Biomedical Statistics and Informatics*, *2*(4), 131–137. https://doi.org/10.11648/j.bsi.20170204.11

Dobing, B., & Parsons, B. (2006). How UML is Used. Communications of the ACM. *ACM*, *49*(5), 109–113.

Earl, M. J. (1989). Management Strategies for Information Technology. Mayland Avenue : Prentice Hall International (UK) Ltd Campus 400.

Esterhuizen, D., Schutte, C. & Du Toit, A. (2012). A knowledge management framework to grow innovation capacity maturity. SA Journal of Information Management , 14 (1).

Freeman, A., & Sanderson, S. (2011). *Pro ASP.NET MVC 3 Framework*. APress.

Gil, H. G., Dario, M. A., & Raul, O. (2010). Evolution and trends of information systems for business management: The mbusiness. A review. *Dyna*, *77*, 181–193.

Goto, T., Tsuchida, K., & Nishino, T. (Eds.). (2014). *An iterative programming method for innovative software based on system designs*. 3rd International Conference on Advanced Applied Informatics.

Gupta, S., Taya, S., Jai, S., & Mukand, P. (2012). Descriptive approach to software development life cycle models.

Hamilton, L., Halverson, R., Jackson, S. S., Mandinach, E., Supovitz, J. A., Wayman, J. C., Pickens, C., Martin, E., & Steele, J. L. (2009). Using Student Achievement Data to Support Instructional Decision Making. *United States Department of Education,* Retrieved from https://repository.upenn.edu/gse_pubs/279

Hashim, N. M. Z., & Mohamed, S. N. K. S. (2013). Development of Student Information System. *International Journal of Science and Research (IJSR)*, *2*(8), 256–260.

Henczel, S. (2000). The Information Audit as a First Step towards Effective Knowledge Management: An Opportunity for the Special Librarian. Worldwide Conference on Special Librarianship (pp. 210-226). Brighton: Special Libraries Association (SLA).

Henver, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science research in information systems. *MIS Quarterly*, 75–100.

Hersh, M. A. (2002). THE APPLICATION OF SOFT SYSTEMS METHODOLOGIES TO UNDERSTANDING AND RESOLVING CONFLICTS. *IFAC Proceedings Volumes*, *35*(1), 201–206. https://doi.org/10.3182/20020721-6-ES-1901.01424

IEEE, IEEE Standard 610.12-1990, IEEE Standard Glossary of Software Engineering Terminology, 1990

Kahraman, C., Kaya, I. and Cevikcan, E. (2011). Intelligence decision systems in enterprise information management. Journal of Enterprese Information Management (Emerald Insight) , 24 (4), 350 379.

Kaya, I. (2017, November 13). *What is a Learning Management System?* CMS WIRE. https://www.cmswire.com/digital-workplace/what-is-a-learning-management-system/

Keen, P.G.W., & Morton, M.S.S (1978). Decision support systems: an organizational perspective. Reading, Mass., Addison-Wesley Pub. Co. ISBN 0-201-03667-3

Khanore, S., Patil, R., & Dand, H. (2011). Management information system. *Institute of Distance and Open Learning*. Published.

Killion, J., & Bellamy, G. T. (2000). On the job: Data analysts focus school improvement efforts. *Journal of Staff Development*, 21(1), 12–18.

Klopper, R., Gruner, S., & Kourie, D. (2007). Assessment of a framework to compare software development methodologies. *SAICSIT '07*.

Krasner, G. E., & Pope, S. T. (1988). A cookbook for using the model-view controller user interface paradigm in Smalltalk-80. *A Cookbook for Using the Model-View Controller User Interface Paradigm in Smalltalk-8*, *1*(3), 26–49.

Lago, P. (2000). Rendering distributed Systems in UML. *Unified Modeling Language: Systems Analysis, Design*. Published.

Wikipedia contributors. (2021, July 20). *Learning management system*. Wikipedia. https://en.wikipedia.org/wiki/Learning_management_system

Lehman, T.J., & Sharma, A. (2011), Software Development as a Service: Agile Experiences, 2011 Annual SRII Global Conference, pp. 749-758, doi: 10.1109/SRII.2011.82.

Lewallen, R. (2005). Software Development Life Cycle.

Lippert, M., & Roock, S. (Eds.). (2001). *Adapting XP to complex application domains*. 8th European software engineering conference.

Mandinach, E. (2012). A perfect time for data use: Using data-driven decision making to inform practice. *Educational Psychologist*, *47*, 71–85.

March, S. T., & Smith, G. (1995). Design and natural science research on information technology. *Decision Support System*, *15*, 251–261.

Maria, D. C. S., & Dave, E. M. (2016). Developing An Automated Student Academic Record Management With Business Intelligence Approach. *INFORMATIKA*, *12*(2), 111–123. https://doi.org/10.21460/inf.2016.122.453

Monika, S., & Anju, S., (2013) Information system and System development life cycle

Munassar, N. M. A., & Govardhan, A. (2010). A Comparison Between Five Models Of Software Engineering. *International Journal of Computer Science Issues*, *7*(5).

*MVC framework - introduction - tutorialspoint*. (n.d.). Tutorials Point. Retrieved July 14, 2021, from https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm

Nandutu, J. (2016). *STUDENT RECORD MANAGEMENT SYSTEM*. Livingstone international university.

Noraziah, A., Nawsher, K., Ahmed, N. A., & Abul, H. B. (2010). A Novel Database Design for Student Information System. *Journal of Computer Science*, *6*(1), 43–46.

Nwigbo, S., & Agbo, O.C., School of Science Education, Expert system: a catalyst in educational development in Nigeria

O’Brien, J.A. (1999). Management Information Systems – Managing Information Technology in the Internetworked Enterprise. Boston: Irwin McGraw-Hill. ISBN 0-07-112373-3.

Obiniyi, A. A., & Ezugwu, E. A. (2010). design and Implementation of Students' Information System for Tertiary Institutions using neural networks: An Open Source Approach. *International Journal of Green Computing*, *1*(1), 1–15. https://doi.org/10.4018/jgc.2010010101

O'Brien, J.A. (2003). Introduction to information systems: essentials for the e-business enterprise. McGraw-Hill, Boston, MA

Orobor, A. I. (2015). A Novel Framework for Student Result Computation as a Cloud Computing Service. *American Journal of Systems and Software*, *3*(1), 13–19. https://doi.org/10.12691/ajss-3-1-2

Pilone, D., & Pitman, N. (2005). *UML 2.0 in a Nutshell.* O’Reilly Media.

Pooja, S., & Nitasha, H. (Eds.). (2016). *Analysis of Linear Sequential and Extreme Programming Development Methodology for a Gaming Application*. International Conference on Communication and Signal Processing.

Pooley, R., & Stevens, P. (1999). *Using UML: Software engineering with objects and components*. Addison Wesley Longman Limited.

Ravi, T. M. (2011). The Path to Information Management Nirvana. Information Management Daily .

Reddy, G. S., Srinivasu, R., Rikkula, S. R., & Rao, V. S. (2009). Management Information Systems to help Managers for Providing decision making in an Organisation. International Journal of Reviews in Computing , 2076-3336.

Robertson, J. (2005). 10 Principles of Effective Information Management. Step Two Design pty Limited

Roger, S. (2005). *Software Engineering: A Practitioner's Approach*. Pressman.

Rosca, D., Banica, L., & Mirela, S. (2010). Building successful information systems – a key for successful organization. *Annals of Dunărea de Jos University. Fascicle I : Economics and Applied Informatics.* Published.

Ruparelia, N. (2010). Software development lifecycle models. *ACM SIGSOFT Software Engineering Notes*, *35*(3), 8–13.

Sharma, N. (2017, December 22). *The 14 best data visualization tools*. TNW | Dd. https://thenextweb.com/news/the-14-best-data-visualization-tools

Stella, N., & Chuks, A.O. (2011). Expert System : A Catalyst in Educational Development in Nigeria.

Tchid, M. F., & Zhen, H. (2010). The Requirements Engineering Process Model Based on Design for Six Sigma. *Institute of Electrical and Electronics Engineers (IEEE)*. Published.

Trauth, E. M. (1989). The Evolution of Information Resource Management . Information and Management 16 , 257 - 268.

Trygve Reenska, http:// heim.ifi.uio.no / ~trygver/themes/mvc/mvc-index.html.

Turban, E., & Aronson, J. E. (1995). *Decision Support and Intelligent Systems* (5th ed.). Prentice-Hall, Inc.

Wang, R. Y., & Strong, D. M. (1996). Beyond accuracy: What data quality means to data consumers. *Journal of Management Information System*, *12*(4), 5–33. https://doi.org/10.1080/07421222.1996.11518099

Zwass, V. (2020, November 2). Information system. Encyclopedia Britannica. *https://www.britannica.com/topic/information-system*