**INTRODUCTION**

In the modern world that we live in, technology has undeniably become integrated into every aspect of our life. It is therefore highly important to sustain, spread and integrate this technological involvement in our lives due to its countless benefits. The Muranga cafeteria is a major location where campus students have their meals may it be lunch or breakfast. It is located in Muranga University campus area. Offers a variety of cuisines to students. The need for a computerized system was highly needed. The old system is a bit primitive and need for a database featured system is required.

CAFETERIA MANAGEMENT SYSTEM.

**Done by;**

*Name*: GIBSON NDUNGU KONGO.

*Registration number*: SC211/1318/2020.

*Course:* INFORMATION TECHNOLOGY.

*Unit*: PROGRAMMING AND DATABASE PRACTICUM.

**Abstract.**

The proposed cafeteria management system maintains information about the various meals offered in a campus institution. Since using the old system which is an electronic cash register is tedious as it does not store any data or track records of transactions. The new system has the ability to create food menus, place orders, generate bill slips and store purchased products for preparing the meals and show the profit/loss incurred.

The system will be able to access a local database and find what meals will be served every day and who is preparing.

**Acknowledgements.**

The project wouldn’t have been successful without the extended help from my colleagues and friends. I wish to extend my gratitude to my colleagues in participation of the research on the project. My gratitude goes to Muranga University for providing me with the resources needed to for the development of the project including internet resources. The ultimate responsibility of writing the project was entirely mine and will welcome any suggestions and criticism.

**Table of contents.**

ABSTRACT……………………………………………………………… (i)

ACKNOWLEDGEMENT………………………………………………… (ii)

**1.0 CHAPTER ONE**………………………………………………………

**INTRODUCTION**

1.1 Background of the study………………………………………6

1.2 Objectives……………………………………………………….

1.3 Purpose, Scope and Applicability…………………………….

1.3.1 Purpose……………………………………………………….

1.3.2 Scope…………………………………………………………

1.3.3 Applicability…………………………………………………..

**2.0 CHAPTER TWO**

**SURVEY OF TECHNOLOGIES**

2.1 Programming language…………………………………8

2.2 Database……………………………………………………

**3.0 CHAPTER THREE**

**REQUIREMENTS AND ANALYSIS**

3.1 Problem Definition……………………………………18

3.2 Requirements Specification……………………………

3.2.1 Hardware requirements……………………………..

3.2.2 Software requirements……………………………..

3.3.3 Preliminary product description……………………

**4.0** **CHAPTER FOUR**

**SYSTEM DESIGN**

4.1 The system design diagram……………………22

4.2 Data flow diagram……………………………..

4.3 Use case diagram…………………..

4.4 User interface design…………………..

**5.0 CHAPTER FIVE**

**IMPLEMENTATION AND TESTING**

5.1 Implementation approaches…………………30

5.2 Code efficiency and details…………………….

5.3 Testing the system……………………………..

5.4 Modifications……………………………………

**6.0** **CHAPTER SIX**

**RESULTS AND DISCUSSION.**

6.1 Test reports …………………………….34

6.2 User Documentation…………………….

**7.0** **CHAPTER SEVEN**

7.1 Conclusion………………………………………………………………………36

7.2 Limitations of the system………………………………………………………

7.3 REFERENCES…………………………………………………………………………

7.4 Glossary……………………………………………………………………………….

**TABLE OF FIGURES**

Figure 1. System design………………………………………………

Figure 2. Dataflow and activity diagram………………………………….

Figure 3. Use case diagram……………………………………………………..

Figure 4. Log in page……………………………………………………………….

Figure 5. Make sale page………………………………………………………….

Figure 6. Add meal page…………………………………………………………….

Figure 7. Update meal page………………………………………………………….

Figure 8. Remove meal page………………………………………………………….

**CHAPTER ONE**

**1.1 BACKGROUND OF STUDY**

As the most preferable place to have your meals within the school it has brought more students to the place meaning customers flow every day and the numbers keep increasing day by day. Work load has also increased and people need to be served with least amount of time possible. The old cash register system is not able to offer efficiency and the convenience.

Due to this it has come to realization that in order to maintain high standards of services there must be a system that will help the cafeteria maintain a healthy competition against outside hotels and restaurants.

**1.2 OBJECTIVES**

Aims and objectives to be achieved.

1. Create a menu of meals offered in the cafeteria.

2. Create a desktop application that will generate orders and make sales.

3. The system will print out bill slips of meals purchased.

4. Keep record of the amount of stock available to make meals in the database.

5. Create receipts and store them in a database.

6. Generate routine statements and report.

7. Calculate the profits/losses incurred.

1.3

**PURPOSE.**

The current system is not computerized thus the system will be developed in order to help the management team run the cafeteria effectively and efficiently. Comparing the manual system in place, the new system will break down the complexity of retrieving, updating and interpreting data and this will only be possible by coming up with a systematic database where information will be stored and thus generating reports will be easy. Therefore anybody in a position to access the system, of which not just anybody, but authorized personnel will have an easy time doing.

To make sure the right people get to access the system passwords will be implemented. Without the computerized system in place the management of the cafeteria will not be able to analyze sales and provide timely services all together since the system is also meant to break down the complexity of retrieving, updating and interpreting data.

1.3.2

**SCOPE**

The cafeteria management will help to investigate and analyze on students services, storage of data and retrieval of data. Among the many reasons that will hinder this project are;

**i)** Users newly presented to the system may not be conversant with computers.

For the system to be in place and work effectively and efficiently, the end users must be compacted with basic knowledge on computers and how the system works.

**ii)** Trying to propose the system to the management might be opposed.

**1.3.3**

**APPLIABILITY**

This system will not only improve the efficiency of the cafeteria but also the campus proficiency in purchasing the products needed by the cafeteria.

The system will gradually reduce the load to the cashier responsible of making sales and improve the digitization of the campus systems.

**CHAPTER 2**

**SURVEY OF TECHNOLOGIES**

There are various tools available to use in designing of the software. These tools include visual basic, *visual studio, RAD studio, android studio, Eclipse, Qt, Pycharm, Netbeans* among others are IDE’s with various tools to design the interfaces of a system. This tools also provide code editor where one can add functionality of an interface through coding. Languages such as JAVA, C#, C++, PYTHON, JAVA SCRIPT, SWIFT are languages used with the following IDE’s to create an intractable interface.

**PROGRAMMING LANGUAGE**

For this system I used *Microsoft visual studio 2017* due to its major use and a large community of developers ready to help, when you have any problem. Visual studio also has a lot of tools for developing any desktop application, it’s also known globally for its efficiency. I used C# in coding the system due to its huge libraries and the code is remarkably scalable and easy to maintain. C# was also the best compared to html and JavaScript since this project involves making of a desktop application and not a web application.

**DATBASE**

I used SQL Microsoft database since it syncs with the visual studio seamlessly.

The code has less execution time due to its efficiency, since it communicates with the computer processor directly and fast.

***Front end***

The front end is designed by Microsoft visual studio ide with the help of their tools thus creating a responsive interface that is appealing to the user.

Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop computer programs, as well as websites, web apps, web services and mobile apps. Visual Studio uses Microsoft software development platforms such as Windows API, Windows Forms, Windows Presentation Foundation, Windows Store and Microsoft Silverlight. It can produce both native code and managed code.

Visual Studio includes a code editor supporting IntelliSense (the code completion component) as well as code refactoring. The integrated debugger works both as a source-level debugger and a machine-level debugger. Other built-in tools include a code profiler, designer for building GUI applications, web designer, class designer, and database schema designer. It accepts plug-ins that expand the functionality at almost every level—including adding support for source control systems (like Subversion and Git) and adding new toolsets like editors and visual designers for domain-specific languages or toolsets for other aspects of the software development lifecycle (like the Azure DevOps client: Team Explorer).

Visual Studio supports 36 different programming languages and allows the code editor and debugger to support (to varying degrees) nearly any programming language, provided a language-specific service exists. Built-in languages include C, C++, C++/CLI, Visual Basic .NET, C#, F#, JavaScript, Type Script, XML, XSLT, HTML, and CSS. Support for other languages such as Python, Ruby, Node.js, and M among others is available via plug-ins.

**Code editor**

For the programming and coding I used the built in visual studio code editor to add functionality to the user interface.

Visual Studio (like any other IDE) includes a code editor that supports syntax highlighting and code completion using IntelliSense for variables, functions, methods, loops, and LINQ queries. IntelliSense is supported for the included languages, as well as for XML, Cascading Style Sheets, and JavaScript when developing web sites and web applications. Autocomplete suggestions appear in a modeless list box over the code editor window, in proximity of the editing cursor. In Visual Studio 2008 onwards, it can be made temporarily semi-transparent to see the code obstructed by it. The code editor is used for all supported languages.

The Visual Studio Code Editor also supports setting bookmarks in code for quick navigation. Other navigational aids include collapsing code blocks and incremental search, in addition to normal text search and regex search. The code editor also includes a multi-item clipboard and a task list. The code editor supports code snippets, which are saved templates for repetitive code and can be inserted into code and customized for the project being worked on. A management tool for code snippets is built in as well. These tools are surfaced as floating windows which can be set to automatically hide when unused or docked to the side of the screen. The Visual Studio code editor also supports code refactoring including parameter reordering, variable and method renaming, interface extraction, and encapsulation of class members inside properties, among others.

Visual Studio features background compilation (also called incremental compilation). As code is being written, Visual Studio compiles it in the background in order to provide feedback about syntax and compilation errors, which are flagged with a red wavy underline. Warnings are marked with a green underline. Background compilation does not generate executable code, since it requires a different compiler than the one used to generate executable code. Background compilation was initially introduced with Microsoft Visual Basic, but has now been expanded for all included languages.

Those are among the reasons I used the visual studio code editor.

*Back end.*

The back-end was coded using c# an object oriented language.

C# is designed to work with Microsoft's .NET platform. The aim is to facilitate the exchange of information and services over the Web, and to enable developers to build highly portable applications. C# simplifies programming through its use of Extensible Markup Language (XML) and Simple Object Access Protocol (SOAP) which allow access to a programming object or method without requiring the programmer to write additional code for each step. Because programmers can build on existing code, rather than repeatedly duplicating it, C# is expected to make it faster and less expensive to get new products and services to market.

***Database.***

The system uses a local access database for easier transfer of data.

I used an access database and SQL database.

The following sections are short descriptions of the database.

i)Tables

ii)Forms

iii)Reports

iv)Queries

V)Macros

vi)Modules

*Tables*

The database

Database table is similar in appearance to a spreadsheet, in that data is stored in rows and columns. As a result, it is usually quite easy to import a spreadsheet into a database table. The main difference between storing your data in a spreadsheet and storing it in a database is in how the data is organized.

To get the most flexibility out of a database, the data needs to be organized into tables so that redundancies don't occur. For example, if you're storing information about employees, each employee should only need to be entered once in a table that is set up just to hold employee data. Data about Meals will be stored in its own table, and data about price will be stored in another table, using the process of normalization.

*Forms*

Forms allow you to create a user interface in which you can enter and edit your data. Forms often contain command buttons and other controls that perform various tasks. One can create a database without using forms by simply editing your data in the table datasheets. However, most database users prefer to use forms for viewing, entering, and editing data in the tables.

One can program command buttons to determine which data appears on the form, open other forms or reports, or perform a variety of other tasks. For example, One might have a form named "Customer Form" in which you work with customer data. The customer form might have a button which opens an order form where you can enter a new order for that customer.

Forms also allow you to control how other users interact with the data in the database. For example, you can create a form that shows only certain fields and allows only certain operations to be performed. This helps protect data and to ensure that the data is entered properly.

*Reports*

Reports are what One use to format, summarize and present data. A report usually answers a specific question, such as "How much money did we receive from each student this year?" or "What are the most consumed meals?" Each report can be formatted to present the information in the most readable way possible.

A report can be run at any time, and will always reflect the current data in the database. Reports are generally formatted to be printed out, but they can also be viewed on the screen, exported to another program, or sent as an attachment to an e-mail message.

*Queries*

Queries can perform many different functions in a database. Their most common function is to retrieve specific data from the tables. The data you want to see is usually spread across several tables, and queries allow One to view it in a single datasheet. Also, since you usually don't want to see all the records at once, queries let you add criteria to "filter" the data down to just the records you want.

Certain queries are "updateable," meaning you can edit the data in the underlying tables via the query datasheet. If you are working in an updateable query, remember that your changes are actually being made in the tables, not just in the query datasheet.

Queries come in two basic varieties: select queries and action queries. A select query simply retrieves the data and makes it available for use. You can view the results of the query on the screen, print it out, or copy it to the clipboard. Or, you can use the output of the query as the record source for a form or report.

An action query, as the name implies, performs a task with the data. Action queries can be used to create new tables, add data to existing tables, update data, or delete data.

*Macros*

Macros in Access can be thought of as a simplified programming language which you can use to add functionality to your database. For example, you can attach a macro to a command button on a form so that the macro runs whenever the button is clicked. Macros contain actions that perform tasks, such as opening a report, running a query, or closing the database. Most database operations that you do manually can be automated by using macros, so they can be great time-saving devices.

*Modules*

Modules, like macros, are objects you can use to add functionality to your database. Whereas you create macros in Access by choosing from a list of macro actions, you write modules in the Visual Basic for Applications (VBA) programming language. A module is a collection of declarations, statements, and procedures that are stored together as a unit. A module can be either a class module or a standard module. Class modules are attached to forms or reports, and usually contain procedures that are specific to the form or report they're attached to. Standard modules contain general procedures that aren't associated with any other object. Standard modules are listed under Modules in the Navigation Pane, whereas class modules are not.

**CHAPTER 3**

**REQUIREMENTS AND ANALYSIS**

**3.1 PROBLEM DEFINITION**

The lack of a computerized system really slows functionality down and thus the modernized computer system will be applied to fix the problem.

The problem occurred before having computerized system includes:

*• File loss*- When computerized system is not implemented files are always lost because of human environment. Sometimes due to some human error there may be a loss of records.

• *File damaged*- When a computerized system is not there files are likely to be lost due to some accident. Besides some natural disaster like floods or fires may also damage the files.

• *Difficult to search record* -When there is no computerized system there is always a difficulty in searching of records if the records are large in number.

• *Space consuming*- After the number of records become large the space for physical storage of file and records also increases if no computerized system is implemented.

• *Cost consuming*- As there is no computerized system the need to add each record paper will be needed which will increase the cost for the management of cafeteria.

•*Time consuming-* the old cash register is time consuming since there is delay in input of what a student/customer has ordered.

•*Duplication of data entry-*this happens when a sale record is made twice thus leading to a loss of the cafeteria resources.

•*Lack of security-* there is no security in using a cash register system since unauthorized user can make a sale and that’s a threat.

**3.2 REQUIREMENTS SPECIFICATION**.

*Functional requirements*

1- System must be able to search the database based on select search type

2- System must be able to filter meals based on keyword entered.

3- System must be able to show the filtered meals in table view.

4-System will enable the authorized user to select categories.

*Existing system*

The old system is a bit primitive since it does not offer the database functionality thus hindering the retrieval of information for future reference.

Keeping of data and information manually is a tedious way of handling the logistics thus an automated system is required.

*Proposed system.*

To solve the inconveniences as mentioned in the existing system an automated cafeteria system is required.

**3.2.1 HARDWARE REQUIREMENTS.**

An intel core i5 5th generation or the AMD Ryzen 3 are used as the processors because they are faster than others and provide reliable and stable system performance .

By using this either of the processors we can keep on developing the project with no worries.

A 4gb ram DDR4 is used as it will provide fast reading and writing and will in turn support in processing.

The best storage will be an SSD or a faster hard-drive.

**3.2.2 SOFTWARE REQUIREMENTS.**

The system requires a windows 10 operating system to run efficiently, since it is stable and used globally.

The system is compiled, built and debugged by the visual studio 2017 or higher.

The system requires .net framework and SQL database extension to be installed.

**3.3.3 PRELIMINARY PRODUCT DESCRIPTION.**

The proposed system contains the following features:

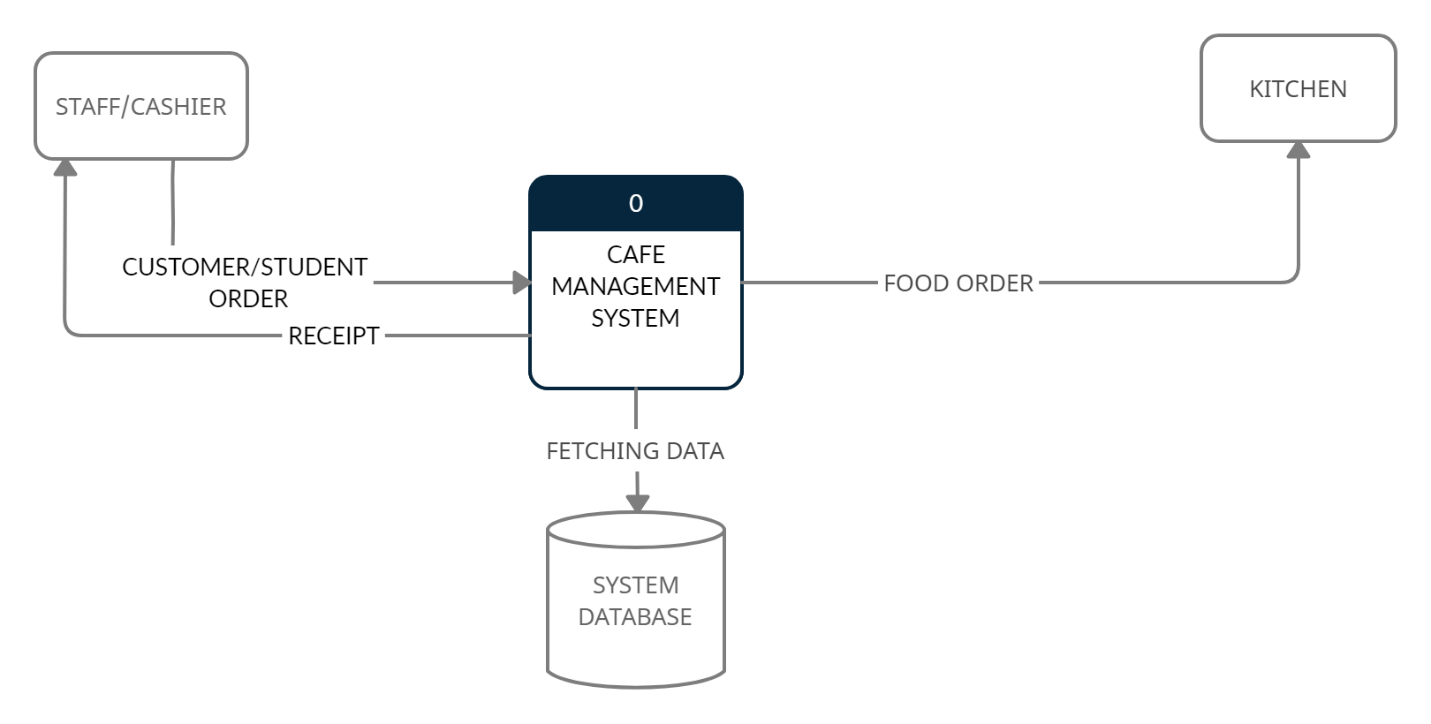
1. Keep a list of a recipes to be used in different days and occasions.
2. Have a login protocol where only the authorized staff can have access to the system since they are responsible for printing the receipts of the food sold.
3. The admin/manager and authorized staff can edit the meals available;
4. Add meals
5. Remove meals
6. Updating the price.
7. Time consumption is minimized, gives accurate results, reliability can be improved with the help of security.

**4.0 CHAPTER FOUR.**

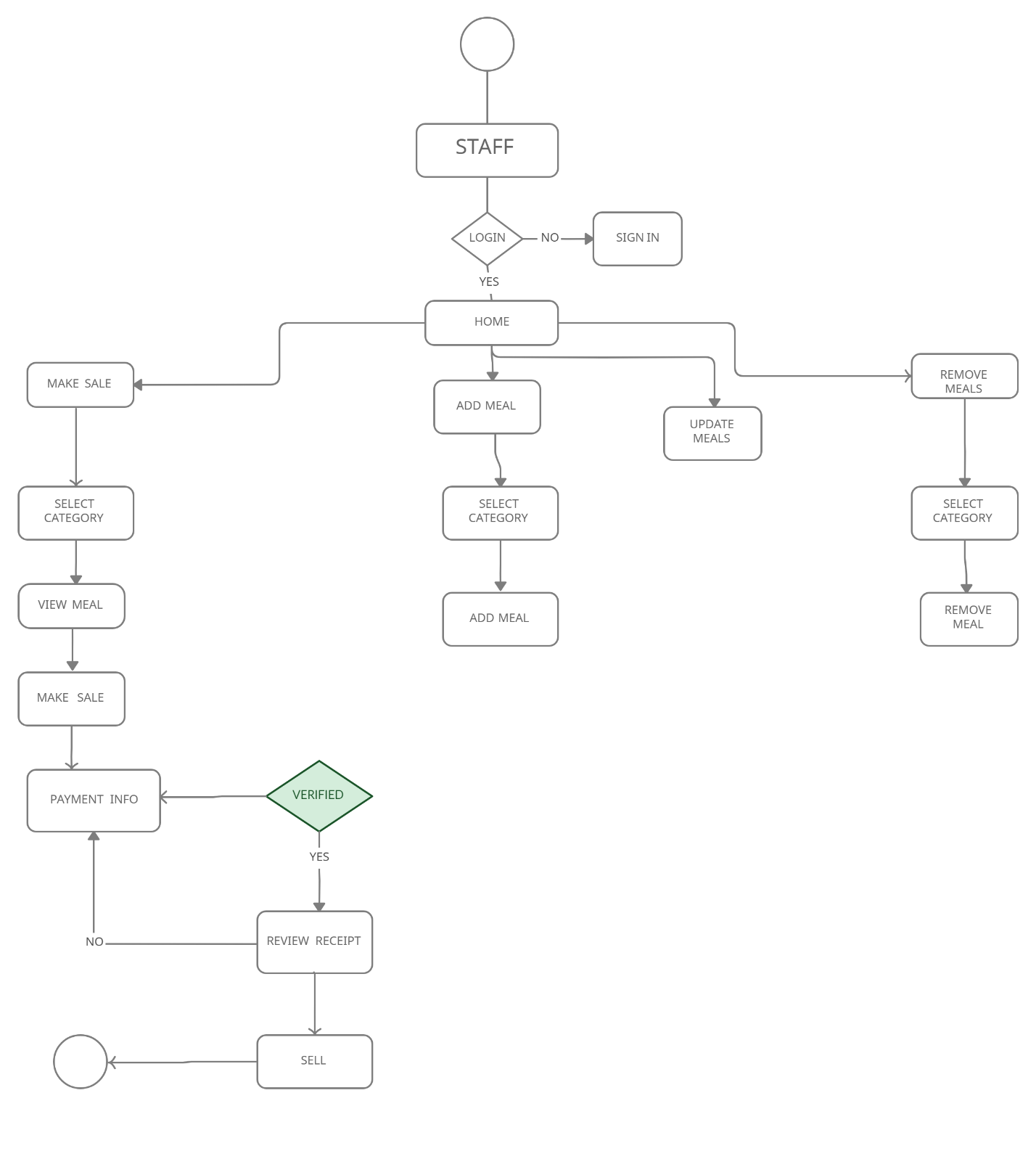
**4.1 SYSTEM DESIGN**

Designing a system requires that someone think about the right way to decompose the functionality, and how to create a small set of abstractions that can be re-used and re-combined to provide the needed functionality. It involves definition of the elements of a system such as modules, architecture and components.

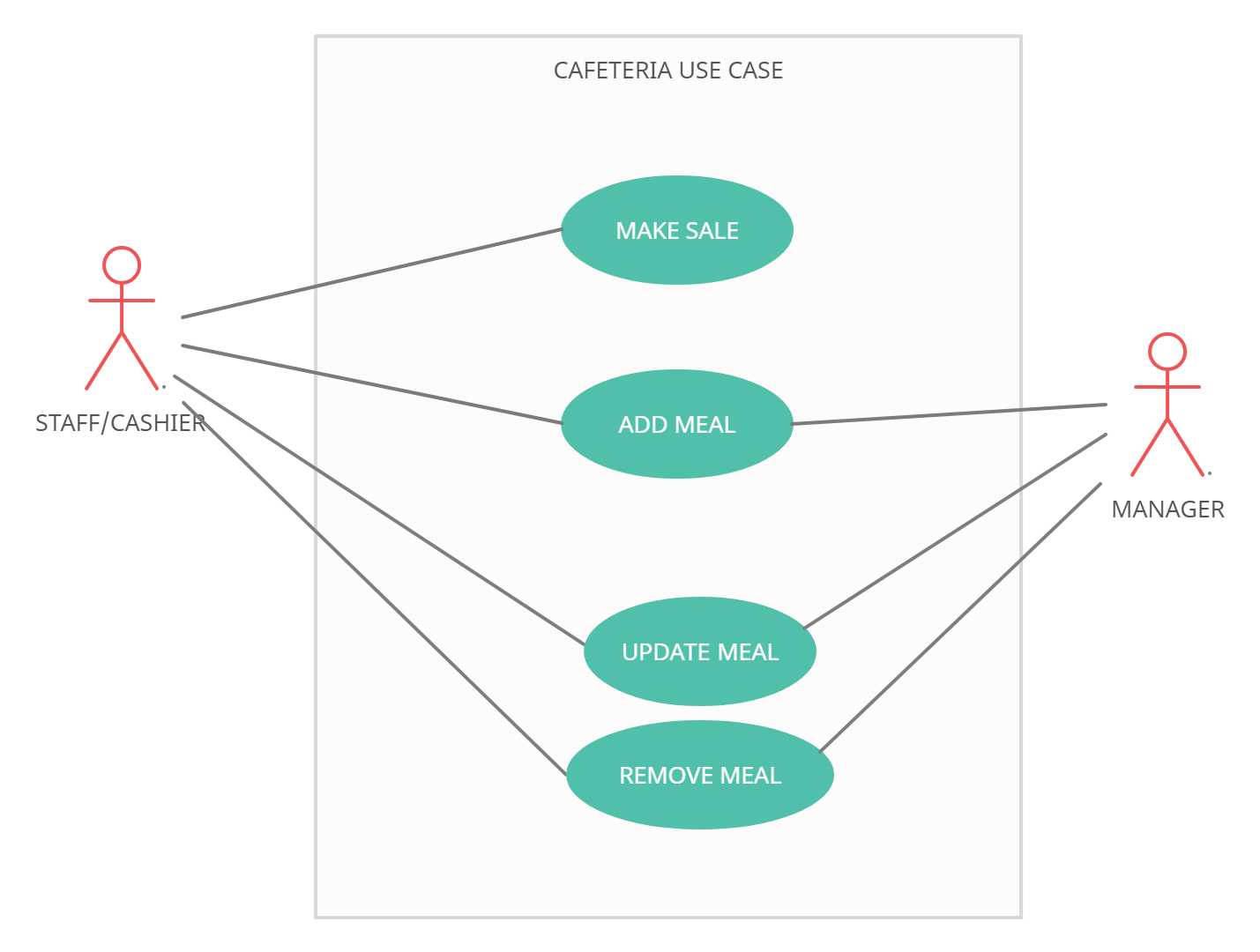
***System design diagram***

****

**4.2 A dataflow diagram of the system.**

****

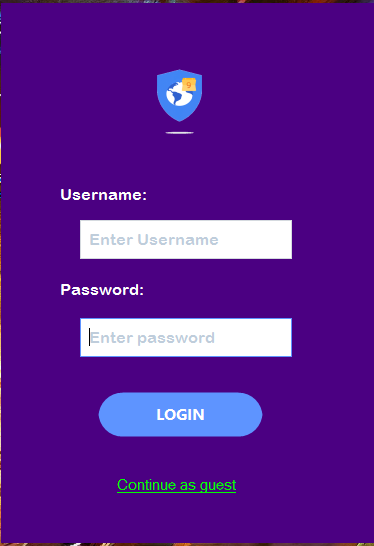
**4.3 *A use case diagram***

****

**4.4 Graphical User Interface**

The graphical interfaces from the system:

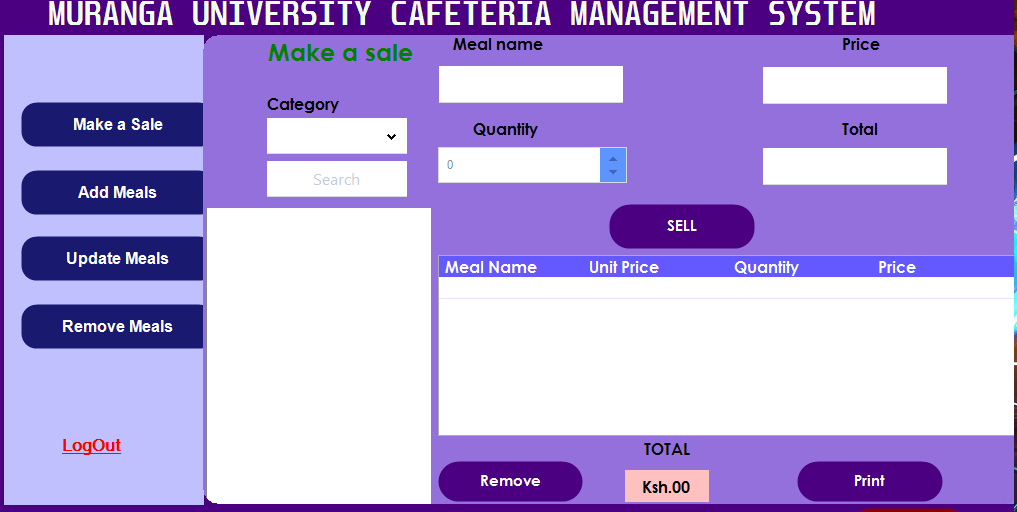
The following shows the login page of an administrator;



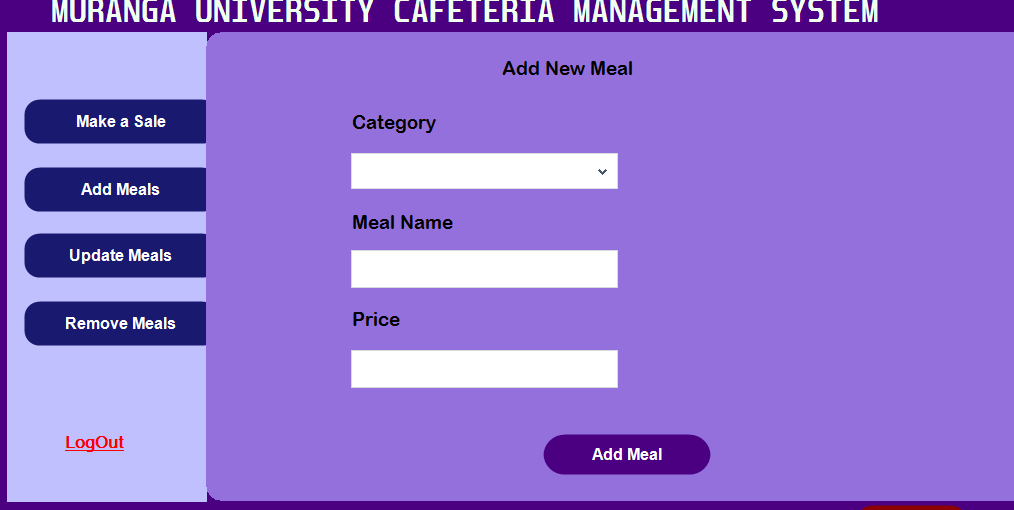
The welcome screen.



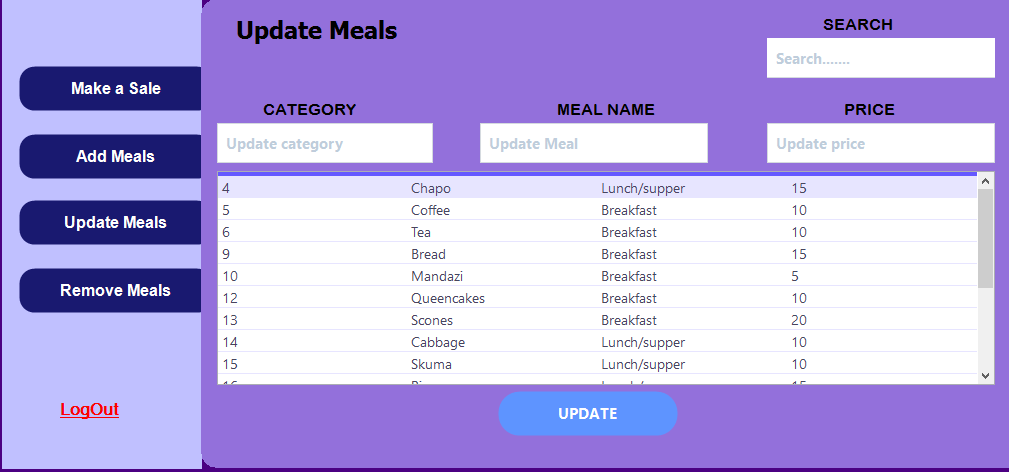
Shows the make sale option;



The Add meal option;



Update and remove meals option;





**CHAPTER 5:**

**IMPLEMENTATION AND TESTING.**

**5.1 Implementation approach**

The implementation of the system was implemented using Windows 10 pro, where selected user verified everything is working as planned.

**5.2 CODING DETAILS AND CODE EFFICIENCY.**

//This code hides the contents of the buttons when on the welcome screen as shown below//

private void Dashboard\_Load(object sender, EventArgs e)

{

uC\_Addmeals1.Visible = false;

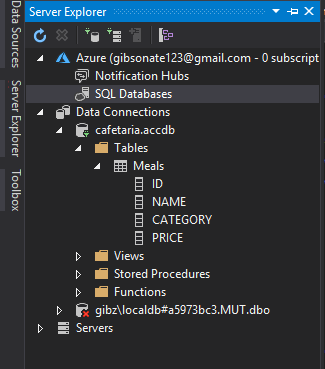
uC\_Makesale1.Visible = false;

uC\_UpdateMeals1.Visible = false;

uC\_RemoveMeals1.Visible = false;

}

I have simplified the code for better understanding and efficiency, by avoiding string conversions.

****

//this code connects the system with the SQL DATABASE//

class function

{

protected SqlConnection getConnetion()

{

SqlConnection con = new SqlConnection(@"Data Source=(LocalDb)\LocalDBCafe;Initial Catalog=MUT;Integrated Security=True");

return con;

}

public DataSet getData(string query)

{

SqlConnection con = getConnetion();

SqlCommand cmd = new SqlCommand();

cmd.Connection = con;

cmd.CommandText = query;

SqlDataAdapter da = new SqlDataAdapter(cmd);

DataSet ds = new DataSet();

da.Fill(ds);

return ds;

}

public void setData (string query)

{

SqlConnection con = getConnetion();

SqlCommand cmd = new SqlCommand();

cmd.Connection = con;

con.Open();

cmd.CommandText = query;

cmd.ExecuteNonQuery();

con.Close();

MessageBox.Show("Data processed succesfully.", "SUCCESS", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

}

**5.3**

**TESTING APPROACH**

System testing is to ensure that a system meets its specification and any other requirements that have been agreed upon by its users.

The aim of the system testing process was to determine all the defects in our project. The program was subjected to a set of test inputs and various observations were made and based on these observations it will be decided whether the program behaves as expected or not.

*Unit testing*

I tested to see if the system is able to add the meals in the database and the input was reflected in the database.

*Integration testing*

I was able to test all components of the cafeteria system. The first test I conducted was registering a user as a normal user and an administrator user, thereafter I logged in into the system. Once a registration was made it reflected in the database and in the back-end where the administrator can add or remove a user or upgrade a user.

**MODIFICATION AND IMPROVEMENTS.**

The code had database accessing issues it had to be changed to avoid lagging issues in the future.

Since the system did not have a windows frame an exit button was added manually to help with the process.

**6.0 RESULTS AND DISCUSSION**

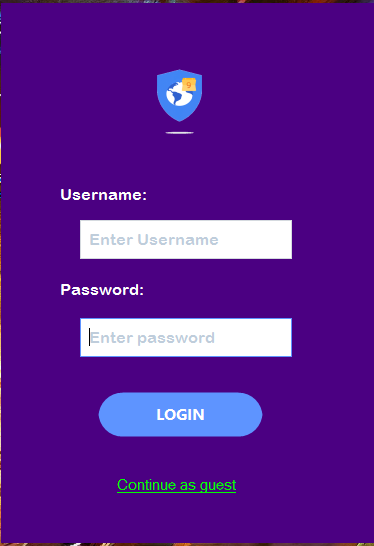
*6.1 Test Reports*

The system was subjected to various situations such as large data and was able to handle it conveniently.

The retrieving, updating and removing of data was tested on a use case scenario and the process was fast.

*6.2 User documentation.*

This is a login page of the admin.

**

Once the admin logs in to the system the welcome Home page is opened.

**//for more interfaces**



**check *graphical***

***use interfaces 4.4//***

**7.0 CONCLUSION AND FUTURE SCOPE.**

In conclusion the modern world that we live in, has undeniably become integrated with technology. You cannot escape it. It has been meshed into every aspect of our life. It is therefore highly important to sustain, spread and integrate this technological involvement in our lives due to its countless benefits. The food industry greatly relies on technology and I believe such systems are the future of the hospitality industry. The cafeteria would greatly benefit from this system even at its early stages.

**Recommendations**

The following will be implemented at a later stage of this project.

• Payment method- Instead of having the customer pay to the till number directly, I will integrate the system with M-PESA APIs in order to enable payments to reflect on the system.

• Allow processing of orders as a guest.

• Enhance User Interface by adding more features and better graphics.

• Add a blog section where users can read up on recipes and other related stories.

**7.1**

**Challenges experienced and limitations.**

• I altered a line of code and some of the graphics began to fail.

•Initiating a payment system was tedious and was not implemented.

•The system does not have a maximize option due to graphical issues so it cannot be viewed in full screen.

**7.2 REFERENCES**

**i) Microsoft resources -**[**https://www.google.com/search?q=microsoft+visual+studio+explained&sxsrf=APq-WBsEnqDV74cIuzTmYo8\_X8k4ueCqEw%3A1649622499283&ei=4z1TYtX6ELKP9u8Ptom1iAs&oq=microsoft+visual+studio+expla&gs\_lcp=Cgdnd3Mtd2l6EAEYADIFCAAQgAQyBggAEBYQHjoHCCMQsAMQJzoHCAAQRxCwAzoHCAAQsAMQQzoKCAAQ5AIQsAMYAToSCC4QxwEQ0QMQyAMQsAMQQxgCOhUILhDHARCjAhDUAhDIAxCwAxBDGAI6BAgjECc6BAgAEENKBAhBGABKBAhGGAFQ3RdYsShgszVoAXABeACAAZcCiAHKDJIBAzItNpgBAKABAcgBEcABAdoBBggBEAEYCdoBBggCEAEYCA&sclient=gws-wiz**](https://www.google.com/search?q=microsoft+visual+studio+explained&sxsrf=APq-WBsEnqDV74cIuzTmYo8_X8k4ueCqEw%3A1649622499283&ei=4z1TYtX6ELKP9u8Ptom1iAs&oq=microsoft+visual+studio+expla&gs_lcp=Cgdnd3Mtd2l6EAEYADIFCAAQgAQyBggAEBYQHjoHCCMQsAMQJzoHCAAQRxCwAzoHCAAQsAMQQzoKCAAQ5AIQsAMYAToSCC4QxwEQ0QMQyAMQsAMQQxgCOhUILhDHARCjAhDUAhDIAxCwAxBDGAI6BAgjECc6BAgAEENKBAhBGABKBAhGGAFQ3RdYsShgszVoAXABeACAAZcCiAHKDJIBAzItNpgBAKABAcgBEcABAdoBBggBEAEYCdoBBggCEAEYCA&sclient=gws-wiz)

**ii) Creately -**[**https://app.creately.com/d/2uQwNdbltlD/edit**](https://app.creately.com/d/2uQwNdbltlD/edit)

**iii) Visual studio tutorials -** [**https://www.youtube.com/watch?v=1CgsMtUmVgs**](https://www.youtube.com/watch?v=1CgsMtUmVgs)