

**KWAME NKRUMAH UNIVERSITY OF SCIENCE AND  
TECHNOLOGY  
DEPARTMENT OF COMPUTER SCIENCE  
[BACHELOR OF SCIENCE IN INFORMATION TECHNOLOGY  
- IDL, KUMASI]**



**TOPIC:**  
**NUTRITION DATA COLLECTION AND CONVENTION SOFTWARE  
SYSTEM**

**PROJECT SUBMISSION BY:**  
**JAMES BAAH**  
**AND**  
**PIET ANKRAH ACHEAMPONG**

**INDEX NUMBERS:**

(20629959)

(29363534)

**PROJECT SUPERVISED BY:**

**DR. E. AHENE**

**JUNE, 2020**

**TOPIC SUMMARY:**

**DESIGN IMPLEMENTATION OF NUTRITION DATA COLLECTION AND  
COVENTION SYSTEM TO HELP PROPAGATE GOOD DIETRY EDUCATION.**

**DECLARATION**

**Candidate's Declaration**

**We, (JAMES BAAH and PIET ANKRAH), Do declare that this project work is our own work produced from research undertaken under supervision. Authors and Publishers whose works have been utilized in this study have been duly acknowledged in the text and list of references.**

**Name:**

**JAMES BAAH**

**Name:**

**PIET ANKRAH ACHEAMPONG**

**Candidate's Signature: .....**

**Date .....**

**Candidate's Signature: .....**

**Date .....**

**Supervisor's Declaration**

**I hereby declare that the preparation and presentation of this project report was supervised in accordance with the guidelines on supervision of project report laid down by the University.**

**Name:**

**DR. E. AHENE**

**Supervisor's Signature**

.....

**Date**

.....

## **TABLE OF CONTENTS**

**CONTENT      PAGE**

<b>DECLARATION.....</b>	
	<b>i</b>
<b>SUPERVISOR’S DECLARATION.....</b>	
	<b>ii</b>
<b>LIST OF CONTENTS.....</b>	
	<b>iii</b>

**CHAPTER ONE:**

<b>1.0 Background of the Study.....</b>	<b>1</b>
<b>1.1 Overview .....</b>	<b>1</b>
<b>1.2 Project Motivation .....</b>	<b>1</b>
<b>1.3 Project Definition .....</b>	<b>2</b>
<b>1.4 Aim of the Project .....</b>	<b>2</b>
<b>1.5 Specific Objectives .....</b>	<b>3</b>
<b>1.6 Definition of Terminologies.....</b>	<b>3</b>
<b>1.7 Project Beneficiaries.....</b>	<b>4</b>
<b>1.8 Project Scope .....</b>	<b>4</b>
<b>1.9 Project Deliverables.....</b>	<b>4</b>

## **CHAPTER TWO:**

<b>2.0 Review of similar system .....</b>	<b>5</b>
<b>2.1 Dfm online system.....</b>	<b>5</b>
<b>2.2 Proposed System.....</b>	<b>5</b>
<b>2.3 Development Tools.....</b>	<b>6</b>
<b>2.4 Software Features.....</b>	<b>6</b>
<b>2.5 Constraints.....</b>	<b>7</b>

## **CHAPTER THREE:**

### **3.0 REQUIREMENT SPECIFICATION**

<b>3.1 Requirement Specification Overview .....</b>	<b>7</b>
<b>3.2 Functional And Non Functional Requirement .....</b>	<b>8</b>
<b>3.3 User and System Requirement .....</b>	<b>8</b>
<b>3.4 Use Case Diagrams.....</b>	<b>8</b>
<b>3.5 Class Diagrams .....</b>	<b>9</b>
<b>3.6 Activity design.....</b>	<b>10</b>
<b>3.7 Sequence Diagram.....</b>	<b>10</b>

## **CHAPTER FOUR:**

### ***4.0 METHODOLOGY***

<b>4.1 Project Methodology .....</b>	<b>11</b>
<b>4.2 Boehm's Spiral Model .....</b>	<b>11</b>
<b>4.3 Waterfall Model Design.....</b>	<b>12</b>
<b>4.3 Incremental Model.....</b>	<b>12</b>
<b>4.4 User Interface Design.....</b>	<b>12</b>
<b>4.5 Database Designs.....</b>	<b>12</b>

## **CHAPTER FIVE:**

<b>5.0 System Implementation and Testing .....</b>	<b>13</b>
<b>5.1 Conclusion.....</b>	<b>13</b>
<b>5.2 Reference.....</b>	<b>13</b>

## **ABSTRACTS**

Medical Studies have revealed that consumption of healthy foods help the body to fight against Diseases. Food provides our body with essential nutrients needed by the body to sustain us for Our day-to-day activities. It is also important to note that different people have different tastes, Likes and dislikes on the choice of food to eat. It is therefore necessary to develop a Technological Nutritional Data collection and convention method software to provide knowledge to every individual with meals of his choice, while ensuring that the correct proportion of nutrients are

present in them. This project addresses this problem by developing a nutrition data collection and convention system. The system is made up of two parts: the first part provides content based diet data collection while the second part uses the data collected to be converted and compare food nutrients and recommend alternative food items. The goal of the Nutrition data collection and convention system is to educate and recommend a healthy and appropriate food quantity to users. The system is designed be used by nutritionist or dieticians to assist them in dieting education and recommendation for patients and also in different homes to suggest varieties of meals to users.



## CHAPTER 1

### **INTRODUCTION**

#### **1.0 BACKGROUND OF THE STUDY**

This section gives the framework of this study. It provides a piece of general knowledge about how nutritional growth and how affects daily meal and the growth and classification of data collection activities, considers the technical issues for each category, and examine the potential applications and challenges related to information and communication technology. The term nutrition is a process of providing or obtaining the food necessary for health and growth a guide to good nutrition. Most of the foods that you consume are made up of several classes of nutrients. These are some of the six food nutrients that are needed by the body all the time;

- Carbohydrates provide the body with glucose, which is converted to energy used to support bodily functions and physical activity.
- Protein to build and repair muscles and bones and to make hormones and enzymes
- Fats Protect yourself from the damage of chronic inflammation.
- Minerals build strong bones and teeth. Controlling body fluids inside and outside cells and turning the food you eat into energy.
- Vitamins help you resist infections and keep your nerves healthy, while others may help your body get energy from food or help your blood clot properly. By following the Dietary Guidelines, you will get enough of most of these vitamins from food.

Data collection is the process of gathering and measuring information on variables of interest, in an established systematic fashion that enables one to answer stated research questions, test hypotheses, and evaluate outcomes.

What is information and communication technology?

Diverse set of technological tools and resources used to transmit, store, create, and share or...

## **1.2 OVERVIEW**

Through the analysis, most people are not having a good balance diet because they do not have proper food by applying the six-food nutrition so most of the people are not growing well. So, we came up with this system by designing software that can help people who are lacking healthy growth and also serve proper checks on the food that they take daily and their weight. By the use of the software, you will know the amount of some food nutrition within the six food nutrition examples protein, iron, vitamins, minerals how it works in the body.

## **1.3 PROJECT MOTIVATION**

From the literature, research people take in food without considering six food groups. This results in unhealthy conditions just obesity .therefore is important to develop software that will make people become aware of the colonic. They ingest a particular food.

## **1.4 PROJECT DEFINITION**

From our research, we came across some important problems that affect most people who are lacking six food nutrients. These are the few problems most people are facing:

- How to balance daily meals with the six food nutrients in the right population.
- The lack of information about the number of nutrients that go into the human body after every meal

### **1.5 AIM OF THE PROJECT**

To help people know or understand how to mix the six food groups in the right population.

### **1.6 SPECIFIC OBJECTIVES**

- To develop software to know the amount of energy someone obtains from the food they take.
- To test the software to know if it is proving the desired result.

### **1.7 PROJECT TERMINOLOGY**

Calories are the amount of energy released when your body breaks down (digests and absorbs) food.

Energy is defined as the capacity to do work. Through the process of digestion, we convert the food we eat into energy.

A programmer is an individual that writes/creates computer software or applications by giving the computer specific programming instructions.

### **1.8 PROJECT BENEFICIARIES**

The project or nutrition data collection and convention system will help people especially children, old men, and women to know the correct food diet they should take in their daily meals for health and also for them strong as they grow.

- Improve health, fitness, and quality of life through daily physical activity
- Increase the quality, availability, and effectiveness of educational and community-based programs designed to prevent disease and injury, improve health, and enhance the quality of life.
- Improve the development, health, safety, and well-being of adolescents and young adults
- Reduce the consumption of calories from solid fats and added sugar in the population

### **1.9 SCOPE OF PROJECT**

This project will entail programming software development and it relates to a nutrient individual that will help people understand the need for a good balance diet.

- convention software system
- nutrition data collection tools examples

### **2.0 PROJECT DELIVERABLES**

This research will help to ensure that the health needs of individuals (in terms of the nation) will be met and it also enhance their health status.



## **CHAPTER 2**

### **REVIEW OF SIMILAR SYSTEM**

#### **2.1 THE DIETARY FOOD MANAGEMENT (DFM):**

Nutrition Management System allows for accurate and efficient nutritional analysis of ingredients, recipes, patient and cafeteria menus, as well as patient nutrient intake. This helps you to ensure that selected menu item combinations will offer the desired nutrients for a particular meal or diet order. No more manual calculations or updates ... DFM does it all for you!

Standard Nutrient Analysis Features

☐ **Nutrient Database**

With DFM you may choose the USDA Standard Reference Nutrient Database or the Canadian Nutrient Data File, or you can manually enter in manufacturers labels as your source of nutrient information. Each database offers nutrient values for thousands of food items in raw or processed form. Nutrient information is provided in 100 gm, 1 LB EP/AP, and standard measures. Refuse factors are automatically applied where provided.

☐ **Ingredient and Recipe Analysis**

Nutrient composition of each recipe is calculated from the ingredients that make up the recipe. Changes to ingredient nutrient information are updated throughout the entire recipe file. When a recipe's ingredient, ingredient quantity, or serving yield is changed, DFM automatically recalculates the recipe's nutrient composition for you.

☐ **Menu Analysis**

Eliminate the time you spend manually calculating the nutritional composition of your patient menus for accreditation. Completely integrated with your Diet Office/Room Service, Management System, the Nutrition Management System will provide an analysis of your default ("house diet") menus, or any combination of menu items you choose, by cycle day and diet type. Menu Analysis options can also be applied to the cafeteria menu for healthy eating promotions.

#### ☐ **Nutrient Limit Screening**

The limit screening features are directly integrated with the Diet Office/Room Service Management System for automated menu checking. These options check patient menus to verify compliance with prescribed nutrient limits of the patient's diet order. Identifying those patients who fall outside prescribed limits promotes proactive dietary intervention -before the meal is served.

#### ☐ **Automatic Intake Analysis**

Minimize subjective assessments and professional staff time performing hand-calculations. With DFM, a patient intake analysis can be fully automated. An Intake Monitor Ticket is generated with the patient's meal ticket to quantify the patient's actual food intake for nutritional analysis of 21 nutrients - not just 3 or 4. The patient's intake history is stored until discharge, allowing intake analysis reporting for a single meal, day or period of time.

## **LITERATURE REVIEW**

Nutrition includes everything that happens to food from the time it is eaten until it is used for various functions in the body nutrition is a core pillar of human development and concrete large scale programming not only can reduce the burden under the food and deprivation but also advances the progress of nations, nutritional status is the state of our body as a result of the foods consumed and

their use by the body. Nutritional status can be good, fair or poor improved nutrition and health enhance the learning ability of children. In the long run it leads to an increase in the strength of the labor force and thereby it contributes positively to the economic growth and a good nutrition is essential for healthy, thriving individuals, families and a nation. Moreover, the nutritional status of children is a manifestation of a host factors, including household access to food and the distribution of this food within the household, availability and utilization of health services, and the care provided to the child.

The first step in the nutritional care process is the assessment of the patient's current status. Nutritional status expresses the degree to which physiologic needs for nutrients are being met (Mahen and Escott-Stump 1996). Studies indicate that upon admission to acute care facilities, 33 to 65% of all patients demonstrate some degree of malnutrition (Mahen and Escott-Stump 1996). Furthermore, in patients who are hospitalized for longer than 2 weeks, nutritional status deteriorates for reasons not identified (Mahen and Escott-Stump 1996). Current methods for nutritional status assessment include anthropometric, biochemical, dietary and clinical valuation. At this time there is no single test that provides both an accurate and reliable measure of nutritional status. For the purpose of this research, the focus of discussion is on anthropometric and biochemical evaluation.

Anthropometry is the science dealing with measurement of size, weight and proportion of the human body. A variety of anthropometric methods have been developed to measure body composition. Anthropometric measurements can be used to estimate subcutaneous fat, which is representative of 50% of body fat stores, and skeletal muscle bulk, which represents 60% of the total body protein pool and is the primary source of amino acids in times of starvation and stress (Manning and Shenkin 1995). The body is influenced by changes in nutritional status and



therefore, anthropometric measurements are important in identifying certain types of malnutrition that affect body size and composition. However, in critically ill patients who have rapid changes in body composition, skinfold anthropometry is of little value. For example, a patient who initially had a skinfold measurement above the normal range, but at time of assessment was within the normal range, may be mistakenly classified as healthy although the patient has lost body fat and protein stores (Manning and Shenkin 1995). In addition, fluid retention or dehydration can influence anthropometric measurements, which puts the patient at risk for a misdiagnosis of malnutrition. Anthropometric measurements performed every several months may help to evaluate a patient's progress when compared to previously recorded measurements. There still is a need for more suitable measurement of body composition in critically ill patients who can undergo rapid body composition changes.

Body weight is assessed upon admission to the hospital and weight loss reflects the immediate inability to meet nutritional requirements, and therefore may indicate nutritional risk. Weight change is assessed on a regular basis. Generally, a five-pound weight change over a six-month period is an indication of a need for further investigation (Simko et al. 1984). The degree of weight loss can be an extremely important index of change in nutritional status. Inadequate caloric intake induces loss of protein from the body cell mass. During starvation, there is mobilization of proteins, carbohydrates and fatty acids, which leads to a rapid decrease in lean body mass (LBM). However, decreases in LBM can be overlooked when edema exists. Acute weight loss in critically ill patients may only reflect changes in fluid balance. Weight change and other anthropometric measures need to be used in conjunction with other methods of assessment.

Biochemical parameters are often measured to assess both macro- and micronutrient status. In broad terms, indicators of nutritional status can be classified as static or functional indices. Static

indicators directly assess specific nutrient content of biological fluids or tissues. Functional indicators are indirect measures of a metabolic or physiological function that is dependent upon a specific nutrient.

Immune parameters are a subset of functional indicators that are used to measure nutritional status. Malnutrition is the most common cause of secondary immunodeficiency (Puri and Chandra 1985), which has led to the incorporation of many cell-mediated immunity (CMI) indexes, such as total lymphocyte count (TLC) and delayed cutaneous hypersensitivity (DCH), both of which are used in the assessment of nutritional status in the hospital (Harvey et al. 1981; Miller 1978; Pretsch et al. 1977). These tests are altered during the early stages of undernutrition and therefore provide a functional measure of mild deviations from normal nutritional health (Puri and Chandra 1985) and offer a broad-spectrum analysis of nutritional status that is appropriate since it is rare for a patient to be deficient in only one nutrient. However, the sensitivity and specificity of these assessments has not been well documented in acute nutrient deprivation.

In view of this a web based application will be need to help people access to know and be educated about balance diet.

## **2.2 PROPOSED SYSTEM**

As the number of users of computer increases every day, its use in different areas is also growing. One of the most powerful aspects of the Web is that anybody who has Internet access can browse on the net. This enables sharing the information worldwide.

One of the fast growing areas of the Web is distance education (or distance learning). The reason distance education on the Web is getting popular is because it has advantages over other types of distance education programs. It gives much more flexibility to the users. The users can take the courses they registered for at any computer connected to the Internet. They usually have a more flexible time frame to take their classes and their tests.

In terms of programming of these sites developers had until recently to use limited range of technology to choose from. These technologies usually involved Common Gateway Interface (CGI) programming, Java script, and Microsoft's Active Server Pages. This paper demonstrates that Java servlets and JDBC can be used programming for these sites.

This paper discusses this new technology and explains how this technology can be used in programming in dynamic software; presents an introduction to Java language, Java servlets and JDBC. It also compares the present technology used in creating dynamic Web sites to the new Java servlet technology. Explanation of the software used in this study and their configuration. It also describes in detail how this project is done by explaining each servlet and its relation to the database class. Lastly, it describes the database design and gives the results and conclusion of the study.

## **2.3 DEVELOPMENT TOOLS**

### **JAVA SERVLETS**

Java is platform independent (portable). Once the Java code is compiled into byte code, it runs on every machine which has Java Virtual Machine (JVM) on it. The Java compiler detects many

problems during compilation. Some of these problems are detected during run time with other language compilers. This makes Java a robust language. Java is also multithreaded. This feature gives Java a better real-time behavior. The Java language was originally intended for use in small, embedded devices. Its first use on the Web came in the form of applets. Until recently, Java has not been used in serious server-side Web development. Now, with the improvements in Java application programming interfaces (API's), especially the Servlet API, Java is becoming an important tool for server-side Web programming.

## **JAVA SERVLETS AND THEIR APPLICATIONS**

A servlet is a dynamically loaded module that services requests from a Web server and is a generic extension to Java-enabled servers. The most common use of servlets is to extend Web servers providing secure, portable, and an easy-to-use replacement for other server-side scripting methods, such as CGI. Servlets run entirely in the JVM and on the server-side; therefore they don't depend on browser compatibility (unlike Java applets). Servlets can be used for any number of Web-related applications where dynamic Web pages are needed. One of the biggest applications for servlets is developing E-commerce sites since they are one of biggest trend in Web development. Servlets can be connected to a database, either on the same server or on a different machine, and make accessible the contents of that database on the Web through JDBC which will be described later. Since the servlet API includes classes and interfaces for session tracking, servlets can be easily used on sites where session tracking is needed. Today, servlets are one of the most exciting and new technologies in server-side Web development. Servlets are efficient, persistent, portable, robust, extensible and secure. Servlets are efficient, because a servlet's initialization code is executed only once when the Web server loads it for the first time. Once the servlet is loaded,

handling new requests is only a matter of calling a service method. Since session tracking is built-in to the servlet API, servlets can maintain states between requests, which makes them persistent. Because servlets are written in Java, they are platform independent and portable. This feature enables servlets to be moved to a new operating system without changing the source code. Also because servlets are written in Java, they are developed with the access to the entire Java Development Kit (JDK) API. Therefore powerful Java packages, such as JDBC, Networking, Remote Method Invocation, etc., can be used in servlets. Servlets are secure, because they run on the server side, inheriting the security provided by the Web server. Servlets can also take advantage of the Java Security Manager API. There are two packages that constitute the Servlet API; these are `javax.servlet` and `javax.servlet.http`. The `javax.servlet` package contains classes to support generic, protocol-independent servlets. These classes are extended by the classes in the `javax.servlet.http` package to add HTTP-specific functionality. The top-level package name is `javax` instead of the familiar `java`, to indicate that the Servlet API is a standard extension. Every servlet must implement the `javax.servlet.Servlet` interface. Most servlets implement it by extending one of two special classes: `javax.servlet.GenericServlet` or `javax.servlet.http.HttpServlet`. A protocol independent servlet should subclass `GenericServlet`, while an HTTP servlet should subclass `HttpServlet`, which is itself a subclass of `GenericServlet` with added HTTP specific functionality. Unlike a regular Java program, and just like an applet, a servlet does not contain a `main()` method. Instead, certain methods of a servlet are invoked by the server in the process of handling requests. Each time the server dispatches a request to a servlet, it invokes the servlet's `service()` method. A generic servlet should override its `service()` method to handle requests as appropriate for the servlet. The `service()` methods accepts two parameters: a request object and a response object. The request object tells the servlet about the requests made by the clients (web

browsers), while the response object is used to return a response. In contrast, an HTTP servlet usually does not override the service() method. Instead, it overrides doGet() method to handle GET requests and doPost() method to handle POST requests. An HTTP servlet can override either or both of these methods. The service() method of HttpServlet handles the setup and dispatching to all the doXXX() methods. Figure 2[6] shows how an HTTP servlet handles GET and POST requests. The remainder in the javax.servlet and javax.servlet.http packages are largely support classes. For example, the ServletRequest and ServletResponse classes in javax.servlet provide access to generic server requests and responses, while HttpServletRequest and HttpServletResponse in javax.servlet.http provide access to HTTP requests and responses. The javax.servlet.http package also contains an HttpSession class that provides built-in session tracking functionality and a Cookie class that allows you to set up and process HTTP cookies.

## **JAVA SERVLET ENGINES**

As mentioned above, servlets are standard extensions to Java. However, the servlet API is not part of the core Java API. Therefore, the servlet API must be present on the server-side, in addition to JVM, in order for the servlets to work. The first servlet API was made available by the Java software division of Sun Microsystems. This product is called as Java Servlet Development Kit (JSDK). In addition to JSDK, there are many third party servlet engines (standalone and add-on) available from different manufacturers. A standalone engine is a Web server that includes built-in support for servlets. Some of the standalone servlet engines include Sun's Java Web Server, the World Wide Web Consortium's Jigsaw Server, and Netscape's Enterprise Server (version 3.51 and later). One advantage of these servers is that once the Web server is installed and configured on the server-side, the servlet engine is also installed. Therefore, an add on servlet engine does not

need to be separately installed and configured. An add-on servlet engine functions as a plug-in to an existing Web server. It adds servlet support to a server that was not originally designed with servlets in mind. Some of the add-on servlet engines include Java-Apache project's JServ module, Allaire Software's Jrun, and IBM's WebSphere Application Server.

## **JDBC DATA ACCESS**

JDBC is a Java API for executing Structured Query Language (SQL) statements. Sun Microsystems says that JDBC is a trademark and is not an acronym for Java Database Connectivity, however, it is often associated with Java Database Connectivity. It consists a set of classes and interfaces. JDBC makes it possible to write database applications using a pure Java API. In its simplest form, JDBC makes it possible to do three things: establish a connection with a database, send SQL statements and process the results. JDBC cannot be used to create databases. Therefore, in order to access a database using JDBC, a database has to be created with a Relational Database Management System (RDBMS). Open Database Connectivity (ODBC) is probably the most widely used programming interface for accessing the relational databases. ODBC, developed by Microsoft, was the first standard database driver. ODBC drivers provide a common API to database clients. However, using ODBC drivers in Java has its own drawbacks. It is written in C language, which is not an object-oriented language. It uses pointers and other programming structures that Java does not support. Also, ODBC drivers must be installed on the client side. This requires that an ODBC driver has to be present on the client side in order for Java applets to be run on the client's Web browser. As mentioned above, JDBC provides a common database programming API for Java programs. The JDBC API supports both two-tier and three-tier models for database access. In the two-tier model, a Java applet or application talks directly to the database.

This requires a JDBC driver that can communicate with the particular database management system being accessed. A user's SQL statements are delivered to the database, and the results of those statements are sent back to the user. Figure 1 shows a simplified version of a two-tier architecture.

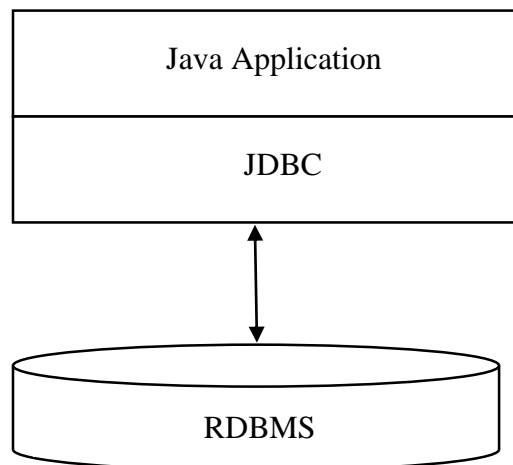
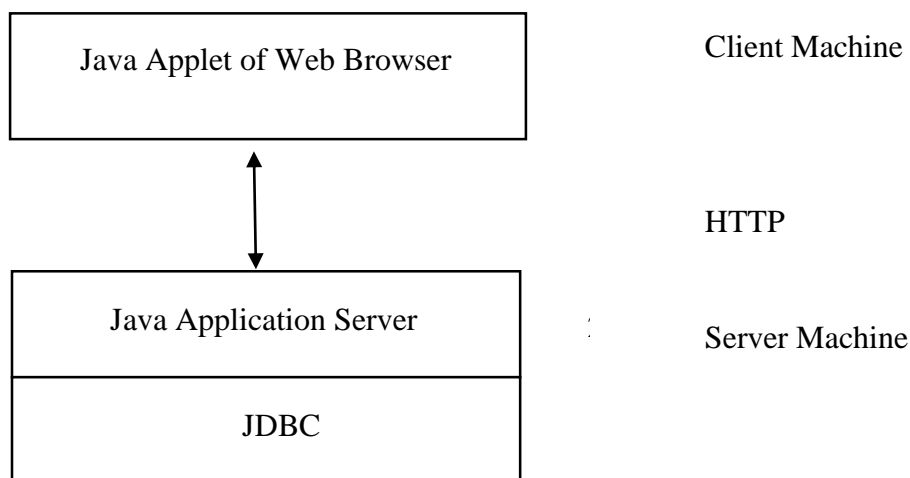


Fig 1 JDBC two-tier model

In the three-tier model, commands are sent to a “middle tier” of services, which then sends SQL statements to the database. The database processes the SQL statements and sends the results back to the middle tier, which sends them to the user. In many cases the three-tier architecture can provide performance advantages. Figure 4 show the three-tier model.





## COMMON GATEWAY INTERFACE

CGI is just a protocol, a formal process between a Web server and a separate server-side program. The server encodes the client's form input data, and the CGI program decodes the form and generates output. The protocol is independent from the programming language used to program CGI. CGI scripts called in two main ways (methods). The HTTP GET method is used in document retrievals where an identical request will produce an identical result, such as a dictionary lookup. The HTTP POST method sends form data separate from the request. The GET method is only safe for short read-only requests, whereas POST is safe for forms of any size, as well as for updates and feedback responses. Therefore, by default, the CGI module uses POST for all forms it generates.

## PROTOTYPING

Prototyping is defined as the process of developing a working replication of the system that has to be engineered. It offers a small scale facsimile of the end product and is used for obtaining customer feedback described below:

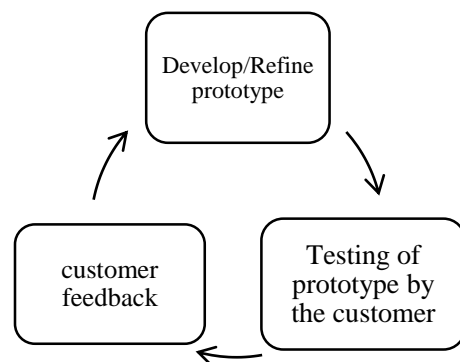


Figure 3

The prototyping model is one of the most popularly used Software Development Life Cycle Models (SDLC models). This model is used when the customers do not know the exact project requirements beforehand. In this model, a prototype of the end product is first developed, tested

and refined as per customer feedback repeatedly till a final acceptable prototype is achieved which forms the basis for developing the final product. In this model, the system is partially implemented before or during the analysis phase there by giving the customers an opportunity to see the product early in the life cycle. The process starts by interviewing the customers and developing the incomplete high-level paper model. This document is used to build the initial prototype supporting only the basic functionality as desired by the customer. Once the customer figures out the problems, the prototype is further refined to eliminate them. The process continues until the user approves the prototype and finds the working model to be satisfactory.

## **USER INTERFACE DESIGN**

User interface design is the process designers use to build interface in software or computerized devices, focusing on looks of style. Designers aim to create interfaces which users find easy to use and pleasurable. UI design refers to graphical user interfaces and other forms:

**Graphical user interface;** users interact with visual representations on digital control panels.

**Gesture-based interface;** Users engage with 3D design spaces through bodily motions.

## **METHODOLOGY**

### **Phase One:**

The Discovery Phase is designed to capture the detail level view of requirements from the perspective of the system, creative and technical stakeholders. Fundamentally, the Discovery Phase is focused on answering the question: “What do you want your software to do?” In an ideal situation, a business case has

been developed and approved that can provide a guide to the entire Discovery Phase. Within the Discovery Phase are several steps:

- **Kicking Off the Project** - It is important, we conducted a formal kickoff meeting. The agenda described who will be involved, what topics will be discussed and what anticipated next steps look like. In this project kickoff meetings, the project manager was appointed as well as designated business, marketing and technology stakeholders who can speak to the concerns of the business.
- **Developing a Project Plan** - Once individual responsibilities for the team are defined, the project manager should develop a baseline project plan that documents the work activity tasks, dependencies, resources and timelines for the individual project elements.
- **Gathering Business Requirements** - For most projects, detailed business and user experience and technical requirements are not well defined. To ensure that you have a comprehensive view of the requirements, each component of the software should be further defined through a series of business stakeholder interviews.
- **Gathering Technical Requirements** - Once the core business requirements have been defined, it is important to identify third-party tools and applications that can be affected. This effort can include identifying whether the current infrastructure will support the project or whether new hardware is appropriate. As part of this effort, you should also focus on setting security requirements and reviewing licensing policies.
- **Gathering User Experience Requirements** - To begin, check to determine whether or not your organization has documented web or style standards. In the event that standards exist, it will be important to align creative deliverables within these requirements. While many people think the user experience consists of purely graphic design, the truth is that the field of information

architecture (IA) is equally important. IA affects the overall hierarchy and organization of information on the software, drives the navigational model and provides a consistent labeling scheme that aids in clarity for users.

➤ **Creating the Discovery Phase Deliverables** - Now that you've completed requirements gathering, the software development methodology calls for the creation of:

1. The functional requirements document
2. The information architecture document
3. The Content Management System (CMS) implementation guide

Once everything has been appropriately documented, it is time for the Implementation Phase.

## **Phase Two:**

The Implementation Phase; this is where we start building to specification. If we were to compare the building of the software to the building of a house, the Discovery Phase is where you meet with an architect, interior designer and landscaper to prototype your dream home. The implementation phase would be where we clear the land, lay the foundation and actually build the home. The steps involved in the Implementation Phase include:

- **Starting Development** - Focus first on the initial setup and configuration of the three environments: development, staging and production hosting environments. It is important to note there are a number of ways the software environment can be configured. While we mentioned a traditional three-tier hosting environment, each hosting topology is different and unique to each customer's specific environment.
- **Content Migration** - Once a framework of the software is complete, the next step is to begin loading content into the content management system (CMS). Depending on whether or not the

project relates to a new software or a redesign of an existing software, your approach to content migration may differ. For new software properties, content is typically developed in Microsoft Word documents. For redesign projects, there are multiple approaches to consider. For example, using SQL scripts or direct APIs, you can migrate content blocks directly from the previous installation.

### **Phase Three**

The Quality Assurance Phase; once implementation is complete, it is time to test. The testing phase of the methodology is intended to capture and resolve any issues, bugs or problems. If there is one overlooked aspect of software development, it is typically in the testing process. As you develop use cases, be expansive and remember to not only cover the tasks described in the business requirements, but also commonly used software functions, such as search and contact forms. There are two stages of testing:

- **System Acceptance Testing** - Using internal resources, begin the system testing phase by documenting specific test cases created using the business and functional requirements obtained during the Discovery Phase.
- **User Acceptance Testing** - The final phase of activity before deployment is to conduct user acceptance testing. With the previous testing phase, you used IT developers and QA staff to do the testing and issue resolution activities. In this phase, you will use actual end-users to validate the site experience.

### **Phase Four:**

Deployment; when the testing process is complete, training delivered and a final check of system functionality, you are ready to deploy.

Remember, once the software is launched, you still have ample opportunities to tweak, modify and enhance the content, layout and functionality of the software. Use this capability to measure your expected results and make changes as appropriate. No other medium has the ability to allow you to quickly change your mind and react to how your customers perceive and interact with your company.

It is also critically important to be detailed in the tracking and reporting of the KPI metrics as defined in the Discovery Phase of activities. If the initial investment in the software is based on demonstrable business impact, these KPIs help to prove that the expected result has been attained.

Always keep in mind, although many projects count success as the mere launching of a new software, real success is measured over time and in the newfound ability to react to the marketplace.

## **Conclusion**

It is critical to remember that the scope of the project must be carefully managed throughout the process or you are almost guaranteed to miss your budgetary and timeline constraints. This is a problem that every software development project faces. However, leveraging an effective change management process can mitigate scope creep by using the discovery deliverables as a baseline to measure against.

Also remember that a software is really never fully completed. Instead, software live in specific time frames, constantly evolving and changing to meet the expanding expectations and needs of your customer audiences. Remember to often revisit the functional requirements for their software, those items that were deemed appropriate for future expansion.

Making simple changes to the software that adds functionality will extend the life and business value of the software investment. These types of small enhancements increase the ROI of the

original business investment in the project and also serve to provide constant “little wins” related to the software that reminds people of the value of both the software as well as the platform it's built upon.

## **SOFTWARE USED IN THIS STUDY**

All the pieces of software used in this study are compatible with and installed on a Microsoft Windows operating system. We used the Apache Web Server. Although, Windows comes with a third party Java compiler, Virtual Machine, we downloaded and installed the JDK for Windows provided by Blackdown.org which follows the Sun Microsystems specifications for JDK. The version of JDK, we used, was 1.1.7v3. As a servlet engine, we downloaded and installed the Java Servlet Development Kit 2.1 available from Sun Microsystems. In order for servlets to function correctly, some features of this engine had to be configured before its use. Some of these features are the port number that the servlets run, servlet directory, and servlet properties file which holds the names of the compiled Java servlets. We used Mini SQL 2.0.11 as my database engine. This is a freely available RDBMS. This database server is very small and compared to its size very powerful. Mini SQL is designed and developed to run on Windows operating systems. Its installed version takes less than 3 megabytes space on hard disk. This RDBMS provides a SQL monitor and supports standard SQL commands. It possess the properties of many large RDBMS's without the overhead of those. In order to connect my Java code to this database server, we used MySQL-JDBC 1.0 driver. This driver is also freely available from the Center for Imaginary Environments to whoever wants to use it.

## **2.3 SOFTWARE FEATURES**

### **FEATURES OF NUTRITION DATA SYSTEMS FOR RESEARCH SOFTWARE:**

**Streamlined data entry and food coding:** Dietary intake data gathered by interview is entered directly into NDSR. The software searches for foods and brand products by name. Sophisticated search algorithms locate the food (e.g., fried egg), and interview prompts standardize requests for more detail (e.g., type of fat used in frying egg). The coding of foods and their variable ingredients and preparation methods occurs as data are entered, with calculation of nutrients occurring immediately.

**Comprehensive, complete, and current database:** The Nutrient Database serves as the source of food composition information in the program. This database includes over 18,000 foods, including 7,000 brand name products. Ingredient choices and preparation method options in NDSR provide more than 160,000 food variants. Values for 169 nutrient, nutrient ratios and other food components are generated from the database. Also, food group assignments (e.g., servings of fruit, vegetables, etc.) are provided. The database is updated annually to reflect marketplace changes and new analytic data.

**Dietary supplement assessment module:** Dietary supplement use may be assessed in conjunction with collection of in-person or telephone 24-hour dietary recalls using the Dietary Supplement Assessment Module included in NDSR. Use of all types of dietary supplements and non-prescription antacids are queried in the module. The database linked with the module includes over 2,000 dietary supplement products. A 'missing product' feature in the software allows the user to add products to the database.



**User support:** A variety of support services are available, including software and technical assistance (Monday through Sunday), a comprehensive user manual, and training and certification.

**Broad applications:** NDSR is licensed by hundreds of organizations including universities, research institutes, food companies, and medical centers (Client List). It is used as a research tool in an array of nutrition-related studies (Publication List).

## **2.4 CONSTRAINTS**

Web site can be created with H. T. M. L. authoring tools in particular with W. Y. S. I. W. I. G. (What You See Is What You Get) user interfaces. In addition web site design can appear to be an individual activity, it in fact requires the intervention of at least two other actors:

- The designer's clients, i.e. the persons who own the web site and fund its development.
- The site's future users, i.e. the future customers of the web site owners.

However, one or both of these actors may not be present throughout the design process, so designers have to anticipate these actors' expectations and needs in individual design activities. Designers' consideration of potential and/ or real expectations of these actors is reflected by their conformance to different kinds of constraints during the design process (Chevalier & Bonnardel, 2003).

Two main kinds of constraints can be distinguished (Chevalier & Ivory, 2003a): client-oriented constraints and user-oriented constraints.

1. Client-oriented constraints are explicitly prescribed or inferred from interactions between designers and their current or prior clients. These constraints can be grouped according to four categories:

- Site originality: site must be original to appease the clients.
- Branding usage: the site must respect characteristics of food and nutrition.
- Updates improvement: the site must present up to date information to favour and increase the number of clients.
- Site structure and content: these constraints refer directly to the web site's content and structure.

2. User-oriented constraints are inferred by designers from their experiences as web site designers and users. Such constraints may address aspects of general interest for users (i.e., aesthetics), or aspects related to web site usability (i.e., ease of navigation). We refer to the latter class of constraints as ergonomic constraints. Ergonomic constraints are categorized with regard to Ergonomic Criteria defined by Scapin and Bastien (1997), Scapin et al. (2000) and Nielsen (2000), such as guidance or consistency criteria. Constraints related to the site's general interest are grouped into two categories:

- Aesthetics, which refer to the look and feel of the site. These constraints concern mainly visual information; for example, the photographs or colours used (e.g. the colours of the web site must be pretty or to group harmoniously documents on the interface).

- Attractive content, which refers to the kind of verbal information to put on web pages (e.g. to not design a web site too technical and more to give technical advises).

## **CONSTRAINTS IN DESIGN ACTIVITIES**

Design is considered as a problem-solving activity in cognitive psychology. This is because designers have to produce an artifact that fits a specific function while satisfying various requirements (Malhotra, Thomas, Carroll, & Miller, 1980). These requirements define, to some extent, the goal to be reached, but problem-solving is required because designers cannot directly apply pre-defined procedures to reach this goal.

Design problems are also considered ill defined, because designers have, at least initially, only an incomplete and imprecise mental representation of the design goal and specifications (Eastman 1969, Simon 1973, 1995). It is only throughout the problem-solving process that designers can complete their mental representations by choosing design options. Therefore, design activities have been described as based on an iterative dialectic between problem-framing and problem-solving (Schön 1983, Simon 1995). To solve the problem, designers have to improve their mental representations so that they can satisfy a constraint condition, effectively transforming an ill defined problem into a better defined one. To solve any design problem, designers have to generate and introduce new constraints that contribute to satisfy the original constraint condition (Bonnardel, 1999; Darses, 2001; Guindon, 1990; Simon, 1973).

## **CHAPTER 3**

### **REQUIREMENT SPECIFICATION**

#### **3.1 REQUIREMENT SPECIFICATION OVERVIEW**

In this chapter, we define the requirements and attributes of the system. Also we shall present to you a number of use case diagram to demonstrate the how these requirements are key to the working of the system.

#### **3.2 FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENT**

##### **FUNCTIONAL REQUIREMENT**

This define what a product must do, what its features and functions are

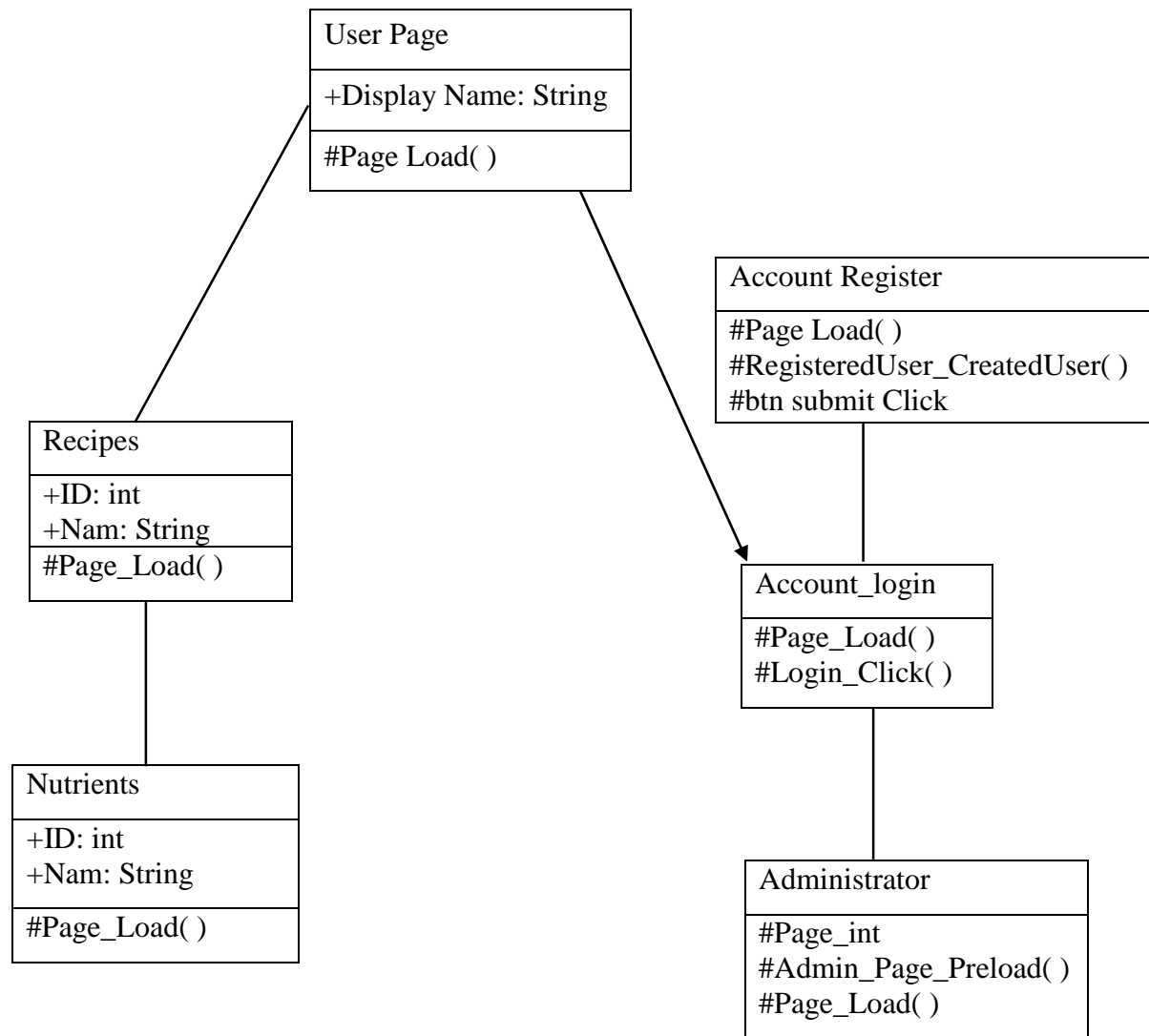
- Create forms
- Fill forms
- Download converted forms

##### **NON-FUNCTIONAL REQUIREMENT**

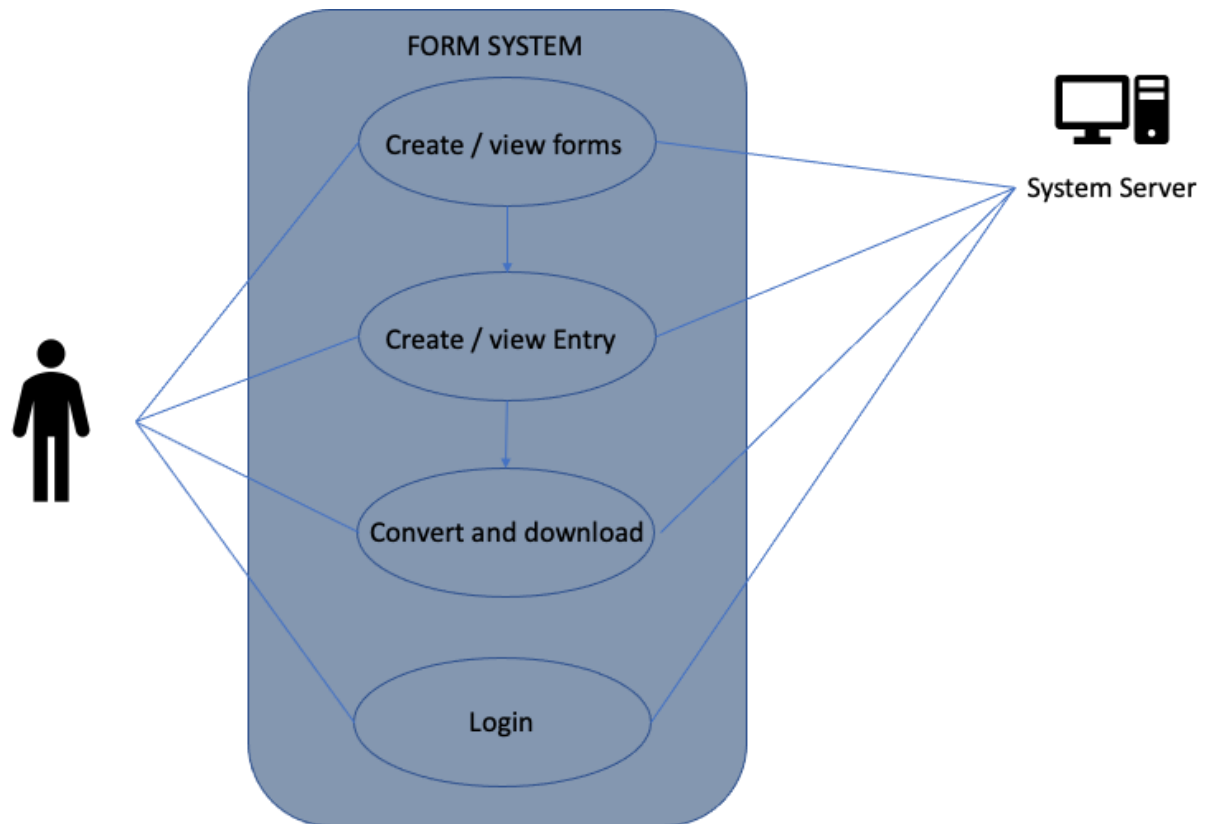
Describe the general properties of a system. They are also known as quality attributes

- Notification
- Fad
- Settings

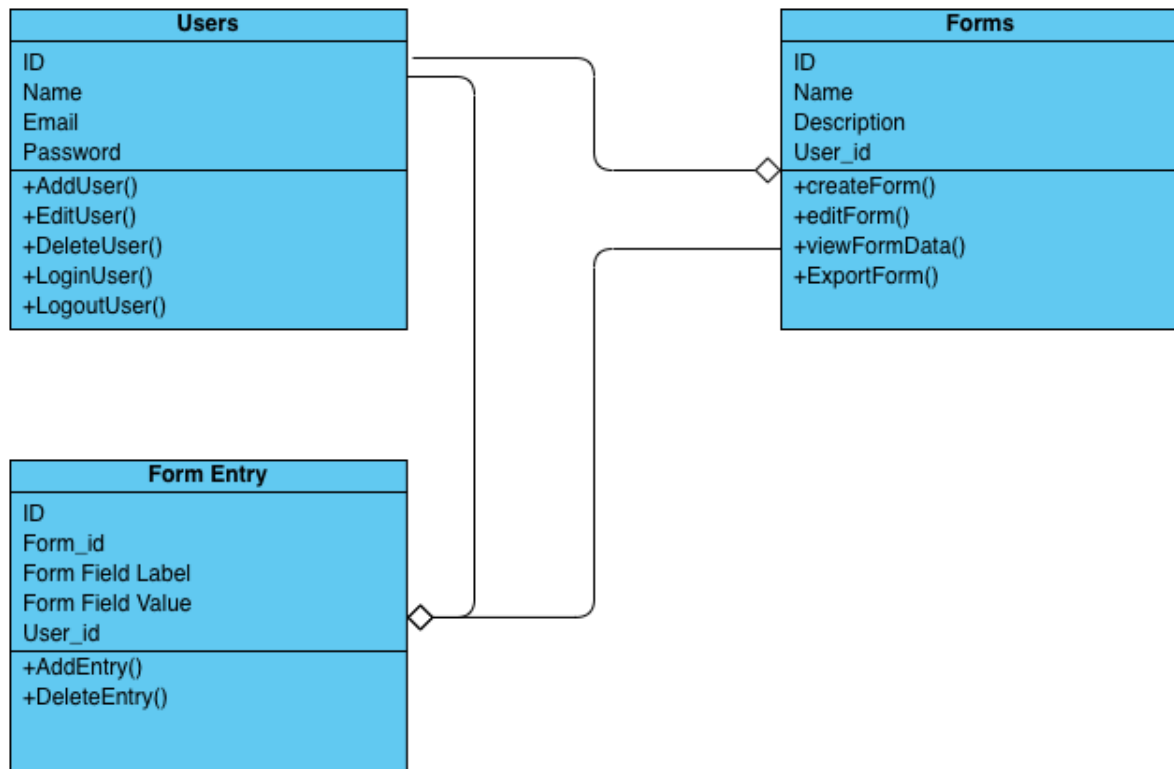
### 3.3 UML DIAGRAM



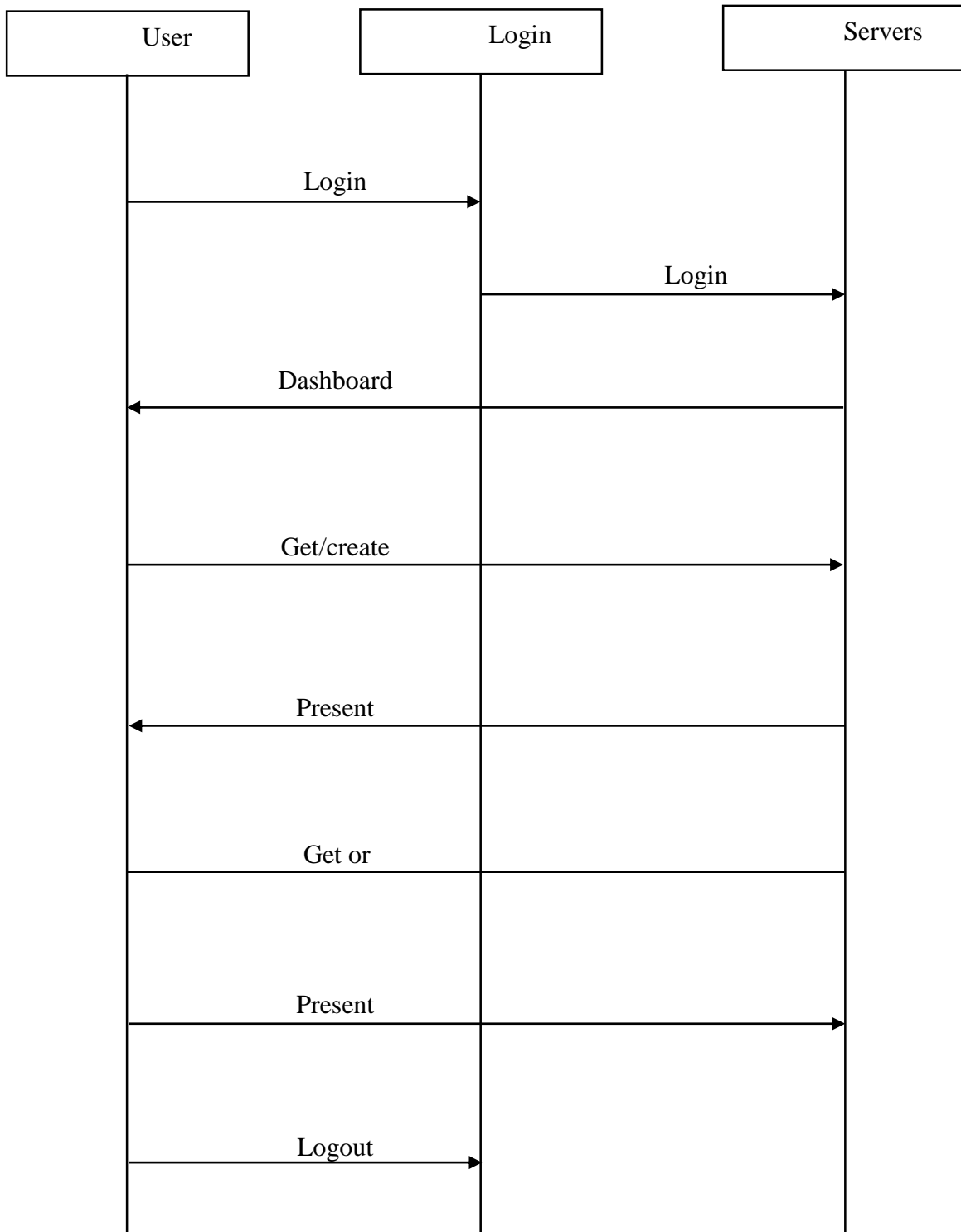
### 3.4 Use Case Diagram



### 3.5 Class Diagram

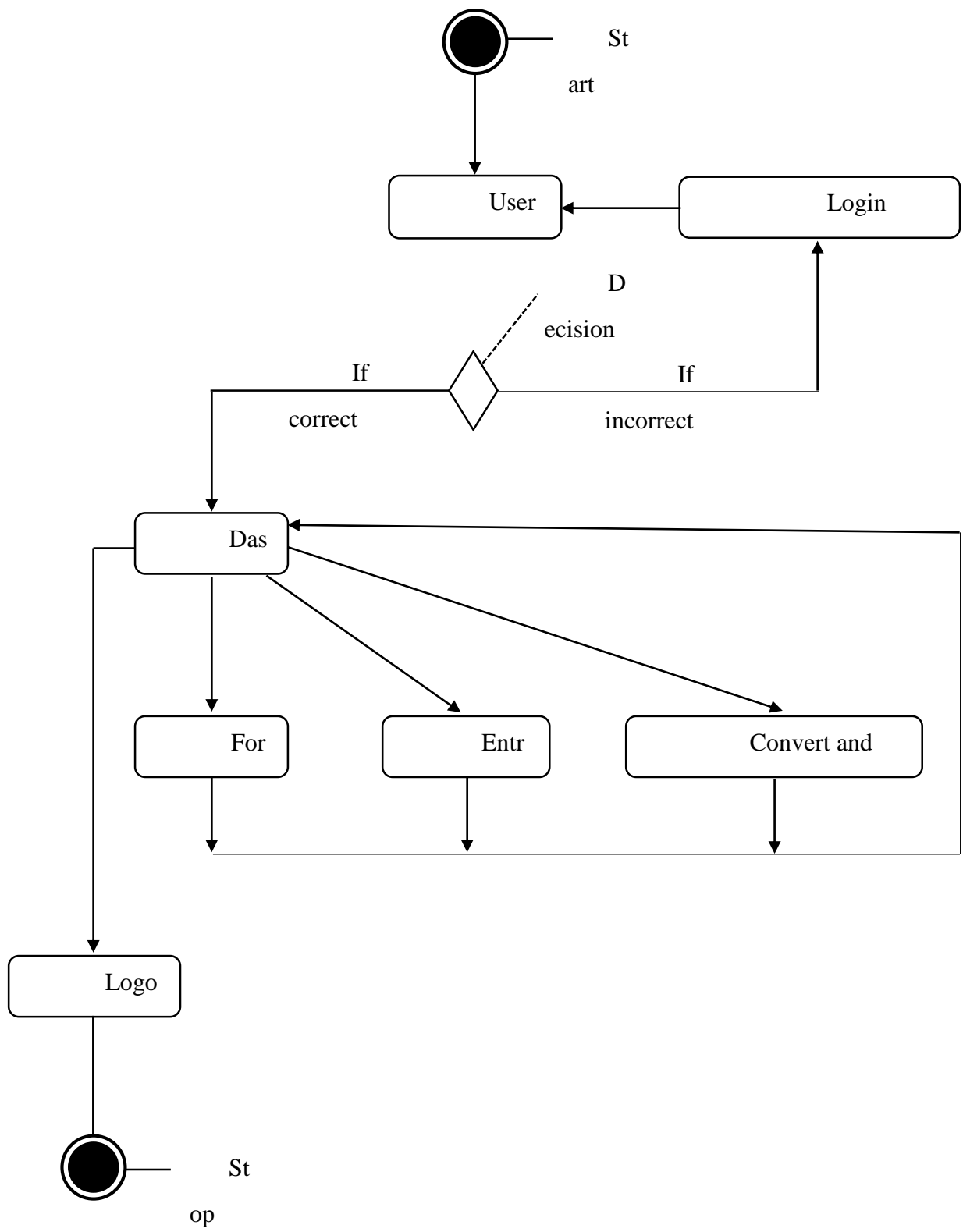


### 3.6 ACTIVITY DIAGRAM





### 3.7 SEQUENCE DIAGRAM



## CHAPTER 4

### METHODOLOGY

#### **4.1 PROJECT METHODOLOGY:**

##### **Phase One:**

The Discovery Phase is designed to capture the detail level view of requirements from the perspective of the system, creative and technical stakeholders. Fundamentally, the Discovery Phase is focused on answering the question: “What do you want your software to do?” In an ideal situation, a business case has been developed and approved that can provide a guide to the entire Discovery Phase. Within the Discovery Phase are several steps:

- ☐ **Kicking Off the Project** - It is important, we conducted a formal kickoff meeting. The agenda described who will be involved, what topics will be discussed and what anticipated next steps look like. In this project kickoff meetings, the project manager was appointed as well as designated business, marketing and technology stakeholders who can speak to the concerns of the business.
- ☐ **developing a Project Plan** - Once individual responsibilities for the team are defined, the project manager should develop a baseline project plan that documents the work activity tasks, dependencies, resources and timelines for the individual project elements.
- ☐ **Gathering Business Requirements** - For most projects, detailed business and user experience and technical requirements are not well defined. To ensure that you have a comprehensive view of the requirements, each component of the software should be further defined through a series of business stakeholder interviews.
- ☐ **Gathering Technical Requirements** - Once the core business requirements have been defined, it is important to identify third-party tools and applications that can be affected. This effort can include identifying whether the current infrastructure will support the project or whether new hardware is appropriate. As part of this effort, you should also focus on setting security requirements and reviewing licensing policies.

□ **Gathering User Experience Requirements** - To begin, check to determine whether or not your organization has documented web or style standards. In the event that standards exist, it will be important to align creative deliverables within these requirements. While many people think the user experience consists of purely graphic design, the truth is that the field of information architecture (IA) is equally important. IA affects the overall hierarchy and organization of information on the software, drives the navigational model and provides a consistent labeling scheme that aids in clarity for users.

□ **Creating the Discovery Phase Deliverables** - Now that you've completed requirements gathering, the software development methodology calls for the creation of:

1. The functional requirements document
2. The information architecture document
3. The Content Management System (CMS) implementation guide

Once everything has been appropriately documented, it is time for the Implementation Phase.

## **Phase Two:**

The Implementation Phase; this is where we start building to specification. If we were to compare the building of the software to the building of a house, the Discovery Phase is where you meet with an architect, interior designer and landscaper to prototype your dream home. The implementation phase would be where we clear the land, lay the foundation and actually build the home. The steps involved in the Implementation Phase include:

□ **Starting Development** - Focus first on the initial setup and configuration of the three environments: development, staging and production hosting environments. It is important to note there are a number of ways the software environment can be configured. While we mentioned a traditional three-tier hosting environment, each hosting topology is different and unique to each customer's specific environment.

□ **Content Migration** - Once a framework of the software is complete, the next step is to begin loading content into the content management system (CMS). Depending on whether or not the project

relates to a new software or a redesign of an existing software, your approach to content migration may differ. For new software properties, content is typically developed in Microsoft Word documents. For redesign projects, there are multiple approaches to consider. For example, using SQL scripts or direct APIs, you can migrate content blocks directly from the previous installation.

### Phase Three

The Quality Assurance Phase; once implementation is complete, it is time to test. The testing phase of the methodology is intended to capture and resolve any issues, bugs or problems. If there is one overlooked aspect of software development, it is typically in the testing process. As you develop use cases, be expansive and remember to not only cover the tasks described in the business requirements, but also commonly used software functions, such as search and contact forms. There are two stages of testing:

- **System Acceptance Testing** - Using internal resources, begin the system testing phase by documenting specific test cases created using the business and functional requirements obtained during the Discovery Phase.

- **User Acceptance Testing** - The final phase of activity before deployment is to conduct user acceptance testing. With the previous testing phase, you used IT developers and QA staff to do the testing and issue resolution activities. In this phase, you will use actual end-users to validate the site experience.

### Phase Four:

Deployment; when the testing process is complete, training delivered and a final check of system functionality, you are ready to deploy.

Remember, once the software is launched, you still have ample opportunities to tweak, modify and enhance the content, layout and functionality of the software. Use this capability to measure your expected results and make changes as appropriate. No other medium has the ability to allow you to quickly change your mind and react to how your customers perceive and interact with your company.

It is also critically important to be detailed in the tracking and reporting of the KPI metrics as defined in the Discovery Phase of activities. If the initial investment in the software is based on demonstrable business impact, these KPIs help to prove that the expected result has been attained.

Always keep in mind, although many projects count success as the mere launching of a new software, real success is measured over time and in the newfound ability to react to the marketplace.

## **Conclusion**

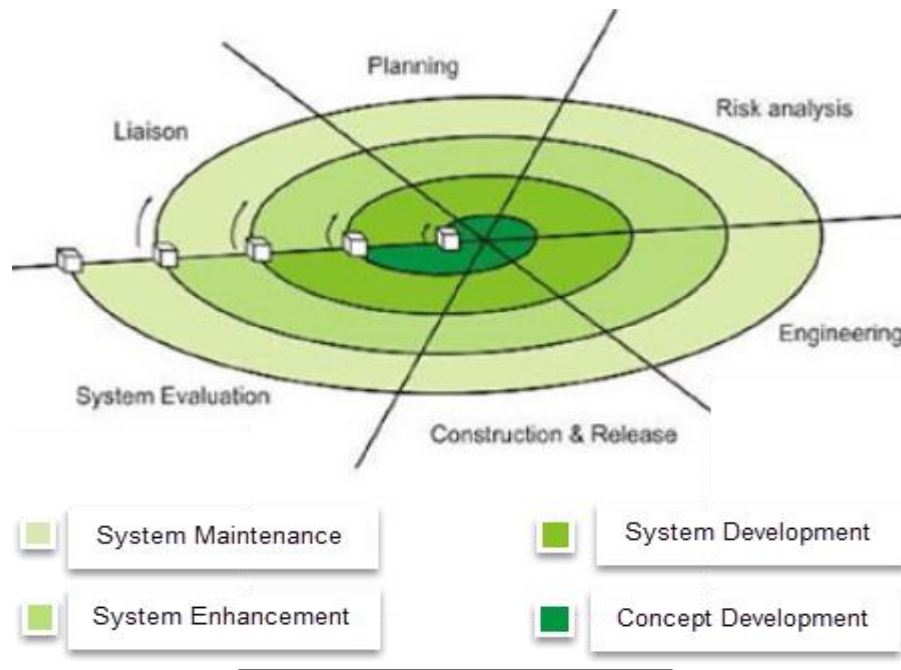
It is critical to remember that the scope of the project must be carefully managed throughout the process or you are almost guaranteed to miss your budgetary and timeline constraints. This is a problem that every software development project faces. However, leveraging an effective change management process can mitigate scope creep by using the discovery deliverables as a baseline to measure against.

Also remember that a software is really never fully completed. Instead, software live in specific time frames, constantly evolving and changing to meet the expanding expectations and needs of your customer audiences. Remember to often revisit the functional requirements for their software, those items that were deemed appropriate for future expansion.

Making simple changes to the software that adds functionality will extend the life and business value of the software investment. These types of small enhancements increase the ROI of the original business investment in the project and also serve to provide constant “little wins” related to the software that reminds people of the value of both the software as well as the platform it's built upon.

## **4.2 BOEHM'S SPIRAL MODEL**

is a risk-driven software development process model which is a combination of waterfall model and iterative model. Spiral Model helps to adopt software development elements of multiple process models for the software project based on unique risk patterns ensuring efficient development process.



### **Planning:**

It includes estimating the cost, schedule and resources for the iteration. It also involves understanding the system requirements for continuous communication between the system analyst and the customer

### **Risk Analysis**

Identification of potential risk is done while risk mitigation strategy is planned and finalized

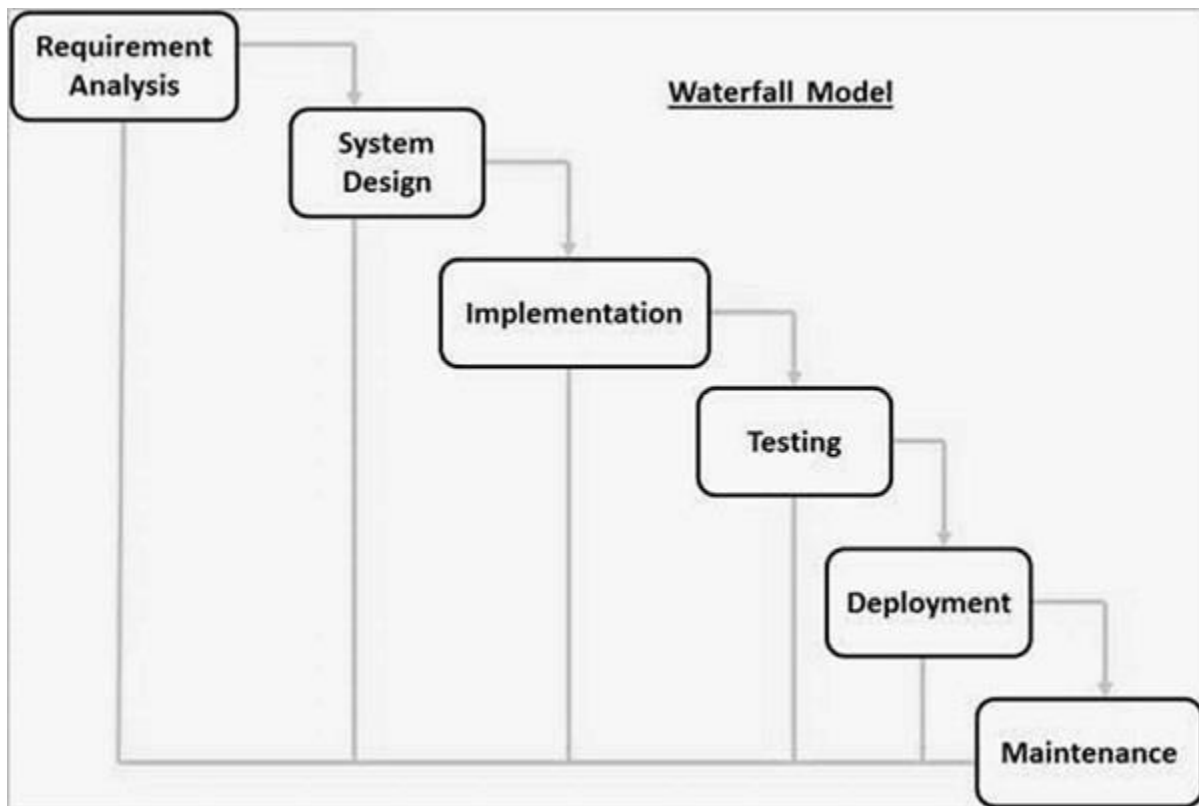
### **Engineering**

It includes testing, coding and deploying software at the customer site

### **Evaluation**

Evaluation of software by the customer. Also, includes identifying and monitoring risks such as schedule slippage and cost overrun

### **4.3 WATERFALL MODEL DESIGN**



#### **Requirement Gathering and analysis –**

All possible requirements of the system developed are captured in this phase and documented in a requirement specification document.

#### **System Design –**

The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.

#### **Implementation –**

With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.

Integration and Testing – All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.

#### **Deployment of system –**

Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.

#### **Maintenance –**

There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

All these phases are cascaded to each other in which progress is seen as flowing steadily downwards (like a waterfall) through the phases. The next phase is started only after the defined set of goals are achieved for previous phase and it is signed off, so the name "Waterfall Model". In this model, phases do not overlap.

#### **Waterfall Model - Application**

Every software developed is different and requires a suitable SDLC approach to be followed based on the internal and external factors. Some situations where the use of Waterfall model is most appropriate are –

Requirements are very well documented, clear and fixed.



Product definition is stable.

Technology is understood and is not dynamic.

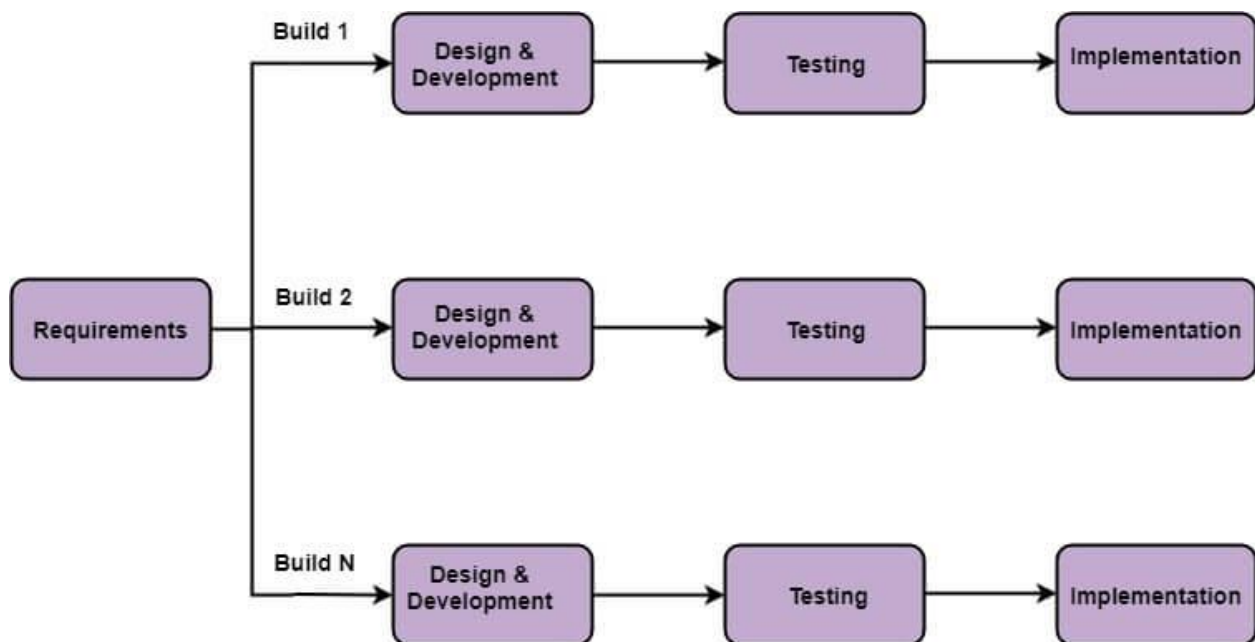
There are no ambiguous requirements.

Ample resources with required expertise are available to support the product.

The project is short.

#### **4.4 INCREMENTAL MODEL**

Incremental Model is a process of software development where requirements divided into multiple standalone modules of the software development cycle. In this model, each module goes through the requirements, design, implementation and testing phases. Every subsequent release of the module adds function to the previous release. The process continues until the complete system achieved.



**Requirement analysis:**

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

### **Design & Development:**

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

### **Testing:**

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

### **Implementation:**

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

## **4.5 USER INTERFACE DESIGNS:**

This nutrition system's website was created and designed with the front-end tools below and the backend was developed using Node.js. The backend involves connecting the front-end(website) to the Database(MySQL), allowing for account registration and adding data to it.

## **TECHNOLOGIES USED**

**PROGRAMING LANGUAGE:** The programming language used for the design and creation of this software is JAVASCRIPT (NODE.JS FRAMEWORK). The Node.js has an event-driven architecture capable of asynchronous I/O. These design choices aim to optimize throughput and scalability in web applications with many input/output operations, as well as for real-time web applications (e.g., real-time communications program and browser applications).

**DATABASE:** MySQL (RDBMS) was used for the development of this project

### **FRONT-END DESIGN:**

HTML5,

CSS3,

JAVASCRIPT

**CDN : TAILWINDCSS**

User interface design is the process designers use to build interface in software or computerized devices, focusing on looks of style. Designers aim to create interfaces which users find easy to use and pleasurable.

UI design refers to graphical user interfaces and other forms:

Voice-controlled interface; Users interact with these through their voices.

Graphical user interface; users interact with visual representations on digital control panels.

Gesture-based interface; Users engage with 3D design spaces through bodily motions.

## **4.6 DATABASE DESIGN:**

## **MySQL Database**

Relational Database Management system (RDBMS)

Was used because of its relational functionalities which is it been able to link different tables which are small spaces within a database for data storage.

It was appropriate to be used since relational database management system (RDBMS) is a collection of programs and capabilities that enable IT teams and others to create, update, administer and otherwise interact with a relational database. RDBMSes store data in the form of tables, with most commercial relational database management systems using Structured Query Language (SQL) to access the database. However, since SQL was invented after the initial development of the relational model, it is not necessary for RDBMS use.

The RDBMS is the most popular database system among organizations across the world. It provides a dependable method of storing and retrieving large amounts of data while offering a combination of system performance and ease of implementation.

### **USER TABLE:**

All user's accounts for logins and registration are been received and checked on this table.

### **FORMS TABLE:**

All forms created will be placed here with a unique form ID which helps to identify the forms

### **ENTITY TABLE:**

All entities created will be placed here with a unique entity ID and the associated with a form ID thus all entities are linked to forms.

### **RESPONSE TABLE:**

Dietary data is stored here with a unique ID and its linked to an entity and form

### **COMPOSITION TABLE**

It contains all the nutrient components for each food and has all the data about the food entered which is converted from the response table.

## **CHAPTER 5**

### **5.1 SYSTEM IMPLEMENTATION AND TESTING**

#### **PROJECT CODES**

##### **DATABASE CODE**

```
const { response } = require('express');
```

```
const mysql = require('mysql');
```

```
function connection(){
```

```
    const db = mysql.createPool({
```

```
        connectionLimit : 100,
```

```
        host: process.env.HOST,
```

```
        user: process.env.DB_USER,
```

```
        // port: process.env.PORT,
```

```
        // password: 'TfGEHy6',
```

```
        database: process.env.DATABASE,
```

```
        charset: 'utf8mb4'
```

```
});  
  
console.log("Database connected")  
  
return db;  
}
```

```
module.exports = connection();
```

### Homepage Code

```
const database = require('./database')  
  
//DATE AND TIME  
  
const oldDate = new Date();  
  
let date = oldDate.toISOString().split('T')[0];  
  
let time = oldDate.toLocaleTimeString();  
  
class homedb {  
  constructor(){  
    global.db = database;  
  }  
}
```

```

// # DATABASE MODELS FOR NutriCo MAIN WEBSITEk

// #

// #

// ----- REGISTER SECTION-----

register(firstname,lastname,email,country,city,password,referral_code,affiliate_code,token,
callback){

    let query = 'INSERT INTO users
(firstname,lastname,email,country,city,password,referral_code,affiliate_code,token,date,ti
me) VALUES (?,?,?,?,?,?,?,?,?,?,?);'

db.query(query,[firstname,lastname,email,country,city,password,referral_code,affiliate_co
de,token,date,time], (error,response)=>{

    if(error){

        // throw error;

        return callback({

            status:false,

            message:'Tecnhical issue'

        })

    }

    else if (response.length == 0) {

        return callback({

            status: false,

```

```

        message: "Failed, Please try again.(Check email)",

    });

}

else{

    return callback({

        status:true,

        message:"You've registered successfully."

    })

}

})

}

// ----- END REGISTER SECTION-----

// #

// #

// #

// ----- LOGIN SECTION-----

get_user(email, callback) {

    let query = "SELECT * FROM users WHERE email=? AND status='ACTIVE'";

    db.query(query,[email], (err, response) => {

        if (err) {

            // throw err;

            return callback({

                status:false,

```



```

        message:'Tecnhical issue'

    })

}

if (response.length == 0) {

    return callback({

        status: false,

        message: "Don't have an account!",

    });

} else {

    return callback({

        status: true,

        message: "Login successful",

        response: response[0],

    });

}

});

}

// ----- END LOGIN SECTION-----

// #

// #

// #

// ----- CONTACT SECTION-----

```

```

add_contact(name,email,message,callback){

    let query = 'INSERT INTO contacts (name,email,message,date,time) VALUES
    (?,?,,?,?);'

    db.query(query,[name,email,message,date,time], (error,response)=>{

        if(error){

            // throw error;

            return callback({

                status:false,

                message:'Tecnhical issue'

            })

        }

        else if (response.length == 0) {

            return callback({

                status: false,

                message: "Failed, Please try again.",

            });

        }

        else{

            return callback({

                status:true,

                message:"You've sent us a message successfully."

            })

        }

    })
}

```

```

    })

}

// ----- END CONTACT SECTION-----

}

module.exports = homedb;

```

### DASHBOARD CODE

```

const database = require('./database')

//DATE AND TIME

const oldDate = new Date();

let date = oldDate.toISOString().split('T')[0];

let time = oldDate.toLocaleTimeString();

class studentdb {

  constructor(){

    global.db = database;

  }
}

```

```

// # DATABASE MODELS FOR NutriCo STUDENT DASHBOARD

// #

// #

// ----- NOTIFICATION SECTION-----

get_notification(user_id, callback){

  let query = 'SELECT * FROM notifications WHERE user_id=? AND
status="ACTIVE"'

  db.query(query, [user_id], (error,response)=>{

    if(error){

      throw error;

      return callback({

        status:false,

        message: 'Technical issue'

      })

    }else{

      let unread = response.filter((value)=> value.read_state === 'UNREAD')

      return callback({

        status:true,

        message:'Notification list requested',

        unread: unread.length,

        response: response

```

```

    })

    }

  })

}

```

```

add_notification(user_id,message, callback){

  let query = 'INSERT INTO notifications (user_id,message,date,time) VALUES
(?,?,?,?)'

  db.query(query,[user_id,message,date,time], (error,response)=>{

    if(error){

      // throw error;

      return callback({

        status: false,

        message: 'Technical issue'

      })

    }else if (response.length == 0) {

      return callback({

        status: false,

        message: "Failed, Please try again.",

      });

    }

  })

  else{

```

```

    return callback({

        status:true,

        message:"Notification sent successfully."

    })

}

})

}

update_not_read(user_id,not_id,callback){

    let query = 'UPDATE notifications SET read_state="READ" WHERE id=? AND
user_id=?;'

    db.query(query, [not_id,user_id], (error, response)=>{

        if(error) {

            throw error;

            return callback({

                status:false,

                message: 'Technical issue'

            })

        }

        else if (response.changedRows == 0) {

            return callback({

                status:false,

                message: 'Notification no change'

            })

        }

    })

}

```

```

    }else{

        return callback({

            status:true,

            message: 'Notification read'

        })

    }

})

}

remove_not(user_id,not_id,callback){

    let query = 'UPDATE notifications SET status="INACTIVE" WHERE id=? AND
user_id=?;'

    db.query(query, [not_id,user_id], (error, response)=>{

        if(error) {

            throw error;

            return callback({

                status:false,

                message: 'Technical issue'

            })

        }

        else if (response.changedRows == 0) {

            return callback({

                status:false,

                message: 'Notification no change'

```

```

    })

    }else{

        return callback({

            status:true,

            message: 'Notification read'

        })

    }

    })

}

// ----- END NOTIFICATION SECTION-----

// #

// #

// #

// ----- SUPPORT SECTION-----

get_support(callback){

    let query = 'SELECT * FROM faq'

    db.query(query, (error,response)=>{

        if(error){

            throw error;

            return callback({

                status:false,

                message: 'Technical issue'

            })

        }

    })

}

```



```

    }else{

        return callback({

            status:true,

            message:'Support list requested',

            response: response

        })

    }

})

}

// ----- END SUPPORT SECTION-----

// #

// #

// #

// ----- SETTINGS SECTION-----

update_image(user_id,image_path,callback){

    let query = 'UPDATE users SET image=? WHERE id=?;'

    db.query(query, [image_path,user_id], (error, response)=>{

        if(error) {

            throw error;

            return callback({

                status:false,

                message: 'Technical issue'
            })
        }
    })
}

```

```

    })
  }

  else if (response.changedRows == 0) {

    return callback({

      status: true,

      message: "Already upload this image",

    });

  }

  else{

    return callback({

      status:true,

      message: 'Profile updated successful'

    })

  }

})

}

update_name(user_id,firstname,lastname,callback){

  let query = 'UPDATE users SET firstname=?, lastname=? WHERE id=?;'

  db.query(query, [firstname,lastname,user_id], (error, response)=>{

    if(error) {

      throw error;

      return callback({

        status:false,

```

```

        message: 'Technical issue'

    })

}

else if (response.changedRows == 0) {

    return callback({

        status: true,

        message: "Already have this profile info",

    });

}

else{

    return callback({

        status:true,

        message: 'Profile updated successful'

    })

}

})

}

update_password(user_id,password,callback){

    let query = 'UPDATE users SET password=? WHERE id=?;'

    db.query(query, [password,user_id], (error, response)=>{

        if(error) {

            throw error;

            return callback({

```

```

        status:false,

        message: 'Technical issue'

    })

    }else{

        return callback({

            status:true,

            message: 'Password updated successful'

        })

    }

    })

}

get_oldPassowrd(user_id,callback){

    let query = 'SELECT password FROM users WHERE id=?'

    db.query(query,[user_id], (error,response)=>{

        if(error){

            throw error;

            return callback({

                status:false,

                message: 'Technical issue'

            })

        }else{

            return callback({

                status:true,

```

```

        message:'Support list requested',

        response: response

    })

    }

    })

}

delete_account(user_id,callback){

    let query = 'UPDATE users SET status="DELETED" WHERE id=?;'

    db.query(query, [user_id], (error, response)=>{

        if(error) {

            throw error;

            return callback({

                status:false,

                message: 'Technical issue'

            })

        }

        else if (response.changedRows == 0) {

            return callback({

                status: true,

                message: "Same status",

            });

        }

    })
}

```

```

else{

    return callback({

        status:true,

        message: 'Account deleted successful'

    })

}

})

}

// ----- END SETTINGS SECTION-----

// #

// #

// #

// ----- DASHBOARD SECTION-----

get_dashboard(user_id,callback){

    this.get_total_forms(user_id,(response)=>{

        let totalForms = response.response

        this.get_total_entries(user_id,(result)=>{

            let totalEntry = result.response

            this.get_forms_by_userid(user_id,(res)=>{

                return callback({

                    status:true,

                    message:'Dashboard',

```

```

        response:{
            totalForms,
            totalEntry,
            forms:res.response||[]
        }
    })
})

})
}

```

```

get_total_forms(user_id,callback){
    let query = 'SELECT COUNT(id) as total_forms FROM forms WHERE user_id=?;';
    db.query(query,[user_id], (error, response)=>{
        if(error){
            throw error;
        }
        return callback({
            status:false,
            message:'technical error',
            response:0

```

```

    })
  }
  else if(response.length == 0){
    return callback({
      status:true,
      response: 0
    })
  }
  else{
    let total_forms = response[0]['total_forms']
    console.log(total_forms)
    return callback({
      status:true,
      message:'total_forms',
      response: total_forms
    })
  }
})
}

get_total_entries(user_id,callback){
  let query = 'SELECT COUNT(id) as total_entry FROM entry WHERE user_id=?;';
  db.query(query,[user_id], (error, response)=>{
    if(error){

```



```

    throw error;

    return callback({

        status:false,

        message:'technical error',

        response:0

    })

}

else if(response.length == 0){

    return callback({

        status:true,

        response: 0

    })

}

else{

    let total_entry = response[0]['total_entry']

    console.log(total_entry)

    return callback({

        status:true,

        message:'total_entry',

        response: total_entry

    })

}

```

```

    })
  }

// ----- END DASHBOARD SECTION-----

// #

// #

// #

// ----- FORMS SECTION-----

add_form(user_id,title,description,callback){

  let query = 'INSERT INTO forms (user_id,title,description,date,time) VALUES
(?,?,?,?,?)'

  db.query(query,[user_id,title,description,date,time],(error,response)=>{

    if(error){

      throw error;

      return callback({

        status:false,

```

```

        message: 'Technical error'
    })
} else if (response.length == 0) {
    return callback({
        status: false,
        message: "Failed, Please try again.",
    });
}
else{
    return callback({
        status: true,
        message: "Form added successfully."
    })
}
})
}

update_form(form_id,title,description,callback){
    let query = 'UPDATE forms SET title=?, description=? WHERE id=? '
    db.query(query,[title,description,form_id] ,(error,response)=>{
        if(error){
            throw error;
        }
        return callback({

```

```

        status:false,

        message: 'Technical error'

    })

}

}else if (response.affectedRows == 0) {

    return callback({

        status: false,

        message: "Failed, Please try again.",

    });

}

else{

    return callback({

        status:true,

        message:"Form updated successfully."

    })

}

})

}

get_form_by_id(form_id,callback){

    let query = 'SELECT * FROM forms WHERE id=?'

    db.query(query, [form_id], (error,response)=>{

        if(error){

            throw error;

            return callback({

```

```

        status:false,

        message: 'Technical issue'

    })

    }else{

        return callback({

            status:true,

            message:'Form requested',

            response: response[0]

        })

    }

    })

}

get_forms_by_userid(user_id,callback){

    let query = 'SELECT * FROM forms WHERE user_id=?'

    db.query(query, [user_id], (error,response)=>{

        if(error){

            throw error;

            return callback({

                status:false,

                message: 'Technical issue'

            })

        }else{

```

```

    return callback({

        status:true,

        message:'Form requested',

        response: response

    })

}

})

}

// -----END FORMS SECTION-----

// #

// #

// #

// ----- ENTRY SECTION-----

add_entry(user_id,form_id,given_id,relationship,firstname,lastname,house_no,gender,callback){

    let query = 'INSERT INTO entry

(user_id,form_id,given_id,relationship,firstname,lastname,house_no,gender,date,time)

VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)'

    db.query(query,[user_id,form_id,given_id,relationship,firstname,lastname,house_no,gender,date,time],(error,response)=>{

```

```

if(error){

    throw error;

    return callback({

        status:false,

        message: 'Technical error'

    })

}else if (response.length == 0) {

    return callback({

        status: false,

        message: "Failed, Please try again.",

    });

}

else{

    return callback({

        status:true,

        message:"Form added successfully."

    })

}

})

}

```

```

update_entry(entry_id,given_id,relationship,firstname,lastname,house_no,gender,callback)

{

```

```

    let query = 'UPDATE entry SET
given_id=?,relationship=?,firstname=?,lastname=?,house_no=?,gender=? WHERE id=? ;'

db.query(query,[given_id,relationship,firstname,lastname,house_no,gender,entry_id],(erro
r,response)=>{
    if(error){
        throw error;
        return callback({
            status:false,
            message: 'Technical error'
        })
    }else if (response.length == 0) {
        return callback({
            status: false,
            message: "Failed, Please try again.",
        });
    }
    else{
        return callback({
            status:true,
            message:"Form updated successfully."
        })
    }
}

```



```

    })
  }

  get_entry_by_formid(form_id,callback){

    let query = `SELECT * FROM entry WHERE form_id=? AND status='ACTIVE'`;

    db.query(query, [form_id], (error,response)=>{

      if(error){

        throw error;

        return callback({

          status:false,

          message: 'Technical issue'

        })

      }else{

        console.log('response :>> ', response);

        return callback({

          status:true,

          message:'Form requested',

          response: response

        })

      }

    })

  }

  delete_entry(entry_id,callback){

```

```

let query = 'UPDATE entry SET status="DELETED" WHERE id=?';

db.query(query, [entry_id], (error, response)=>{

  if(error) {

    throw error;

    return callback({

      status:false,

      message: 'Technical issue'

    })

  }

  else if (response.changedRows == 0) {

    return callback({

      status: true,

      message: "Same status",

    });

  }

  else{

    return callback({

      status:true,

      message: 'Entry deleted successful'

    })

  }

})

}

```

```

// ----- END ENTRY SECTION-----

// #

// #

// #

// ----- DATA SECTION-----

get_data(entry_id,callback){

  let query = 'SELECT * FROM data WHERE entry_id=?'

  db.query(query, [entry_id], (error,response)=>{

    if(error){

      throw error;

      return callback({

        status:false,

        message: 'Technical issue'

      })

    }else{

      return callback({

        status:true,

        message:'Data requested',

        response: response

      })

    }

  })
}

```

```

    }

    add_data(entry_id,form_id,period,item,weight,given_time, callback){

        let query = 'INSERT INTO data

(entry_id,form_id,period,item,weight,given_time,date,time) VALUES (?, ?, ?, ?, ?, ?, ?, ?)'

db.query(query,[entry_id,form_id,period,item,weight,given_time,date,time],(error,response
e)=>{

    if(error){

        throw error;

        return callback({

            status:false,

            message: 'Technical error'

        })

    }else if (response.length == 0) {

        return callback({

            status: false,

            message: "Failed, Please try again.",

        });

    }

    else{

        return callback({

            status:true,

            message:"Data added successfully."

```

```

        })
    }
})
}

delete_data(data_id,callback){

    let query = 'UPDATE data SET status="DELETED" WHERE id=?;'

    db.query(query, [data_id], (error, response)=>{

        if(error) {

            throw error;

            return callback({

                status:false,

                message: 'Technical issue'

            })

        }

        else if (response.changedRows == 0) {

            return callback({

                status: true,

                message: "Same status",

            });

        }

        else{

            return callback({

                status:true,

```

```

        message: 'Data deleted successful'

    })

}

})

}

// ----- END DATA SECTION-----

// #

// #

// #

// ----- ALL ENTRY-DATA PLUS ENTRIES INFO SECTION-----

get_all_data(form_id,callback){

    let query = 'SELECT * FROM entry WHERE form_id=?'

    let final_data = [];

    db.query(query, [form_id], (error,response)=>{

        if(error){

            throw error;

            return callback({

                status:false,

                message: 'Technical issue'

            })

        }else if (response.length == 0) {

            return callback({

                status: false,

```

```

    message: "Failed, Please try again.",

    response:[]

  });

}

else{

  let entry_info = response.map((value)=>{

    // console.log('value :>> ', value);

    return {

      id:value.id,

      given_id:value.given_id,

      relationship:value.relationship,

      firstname:value.firstname,

      lastname:value.lastname,

      house_no:value.house_no,

      gender:value.gender

    }

  })

  entry_info.map((value,index)=>{

    this.get_data(value.id,(result)=>{

      let arrObj = result.response.map((single)=>{

        return {

          ...entry_info[index],

          item_id:single.id,

```

```

        period:single.period,

        item:single.item,

        weight:single.weight,

        given_time:single.given_time
    }

})

if(arrObj.length==1){

    final_data.push(arrObj[0])

}else{

    final_data.push(...arrObj)

}

if(index==entry_info.length-1){

    console.log('last ', final_data)

    callback({

        status:true,

        message:'woring',

        response:final_data.sort((a,b)=> b.item_id-a.item.id)

    })

}

})

})

```



```

    }

  })
}

// ----- END DATA SECTION-----

// #

// #

// #

// ----- CONVERT SECTION-----

converter(entry_id,callback){

  let convertedData = [];

  this.get_data(entry_id, (response)=>{

    let entryData = response.response

    // console.log('entryData :>> ', entryData);

    let data = entryData.map((value,index)=>{

      // console.log('value :>> ', value);

      let item = value.item;

      this.get_nutrient(item,(result)=>{

        let nutrients = result.response

        let obj = {

          period:value.period,

```

```

        given_time:value.given_time,

        item:value.item,

        weight:value.weight,

        protein:this.calculate(value.weight,nutrients.protein),

        carbohydrate:this.calculate(value.weight,nutrients.carbohydrate),

        minerals:this.calculate(value.weight,nutrients.minerals),

        vitamins:this.calculate(value.weight,nutrients.vitamins)

    }

    convertedData.push(obj)

    console.log('result :>> ',obj);

    if(index == entryData.length-1){

        console.log('final')

        setTimeout(=>{

            console.log('TIMEOUT')

            return callback({

                status:true,

                response:convertedData

            })

        },2000)

    }

})

})

```

```

    })
  }

  calculate(weight,constant){

    let new_weight = parseFloat(weight);

    let new_constant = parseFloat(constant);

    let multiply = new_weight * new_constant

    let answer = multiply/100

    // console.log('answer :>> ', answer);

    return answer.toFixed(2);

  }

  get_nutrient(item, callback){

    let query = 'SELECT * FROM nutrients WHERE item=?;'

    db.query(query,[item],(error,response)=>{

      if(error){

        throw error;

        return callback({

          status:false,

          message:'Technical error'

        })

      }else{

        return callback({

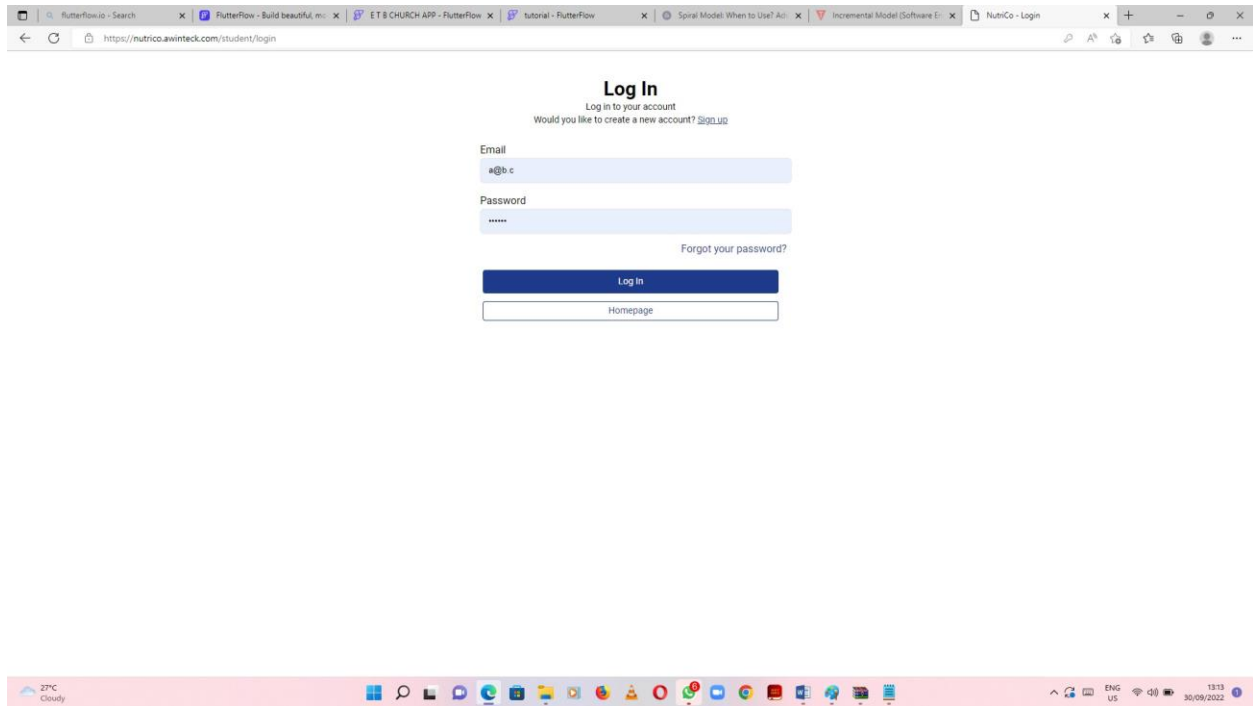
          status:true,

```

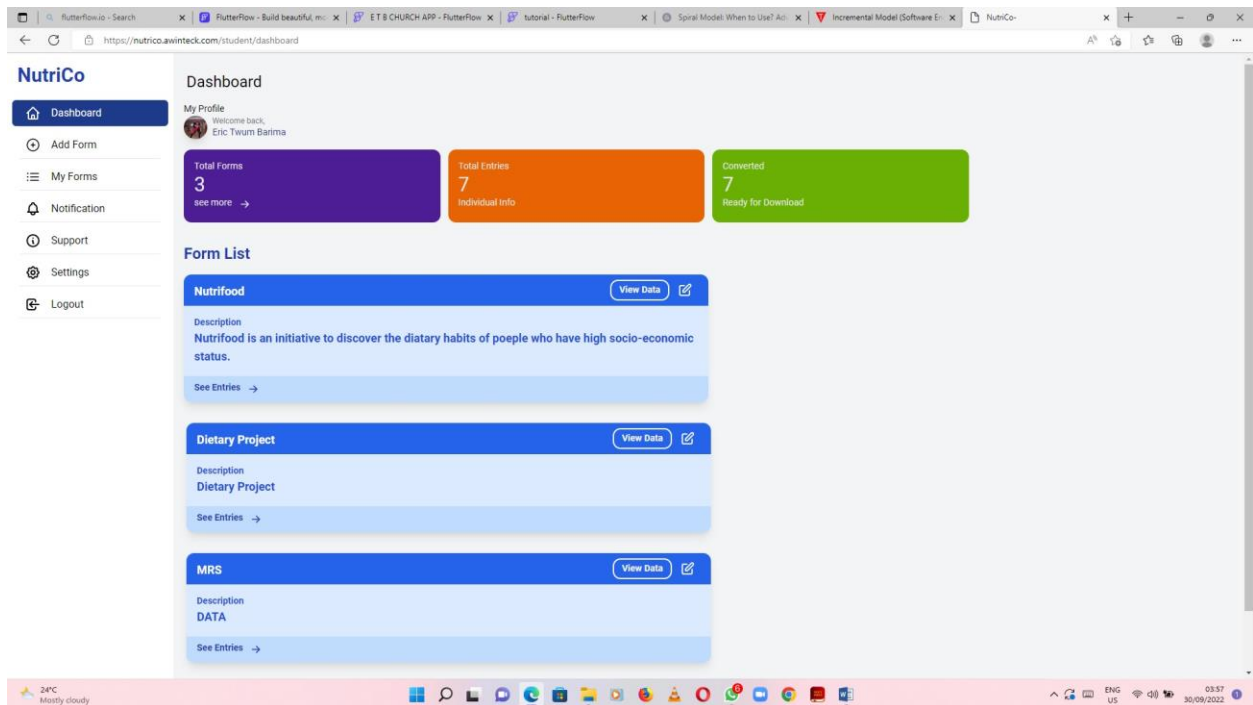
```
        message:'Nutrient requested',  
        response:response[0]  
    })  
}  
})  
}  
  
// ----- END CONVERT SECTION-----  
  
}  
  
module.exports = studentdb;
```

# SCREENSHOTS OF APPLICATION

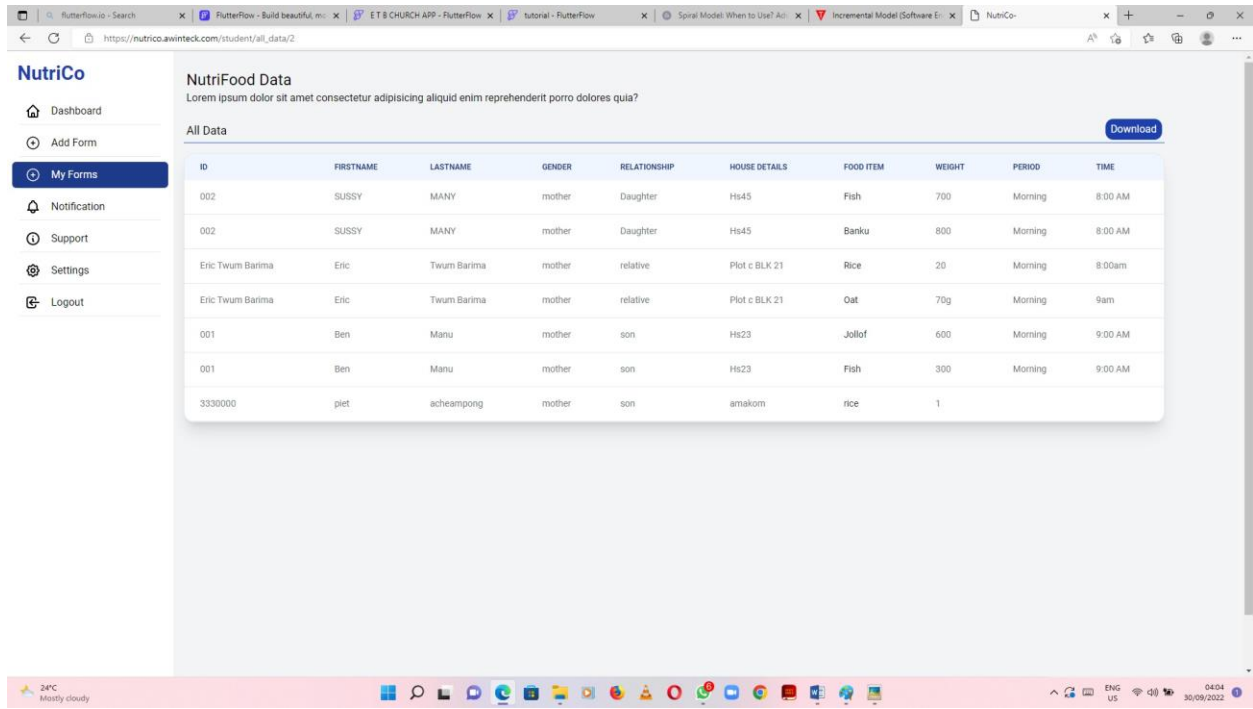
## LOGIN PAGE



## HOMEPAGE



## DATA COLLECTION PAGE



**NutriCo**

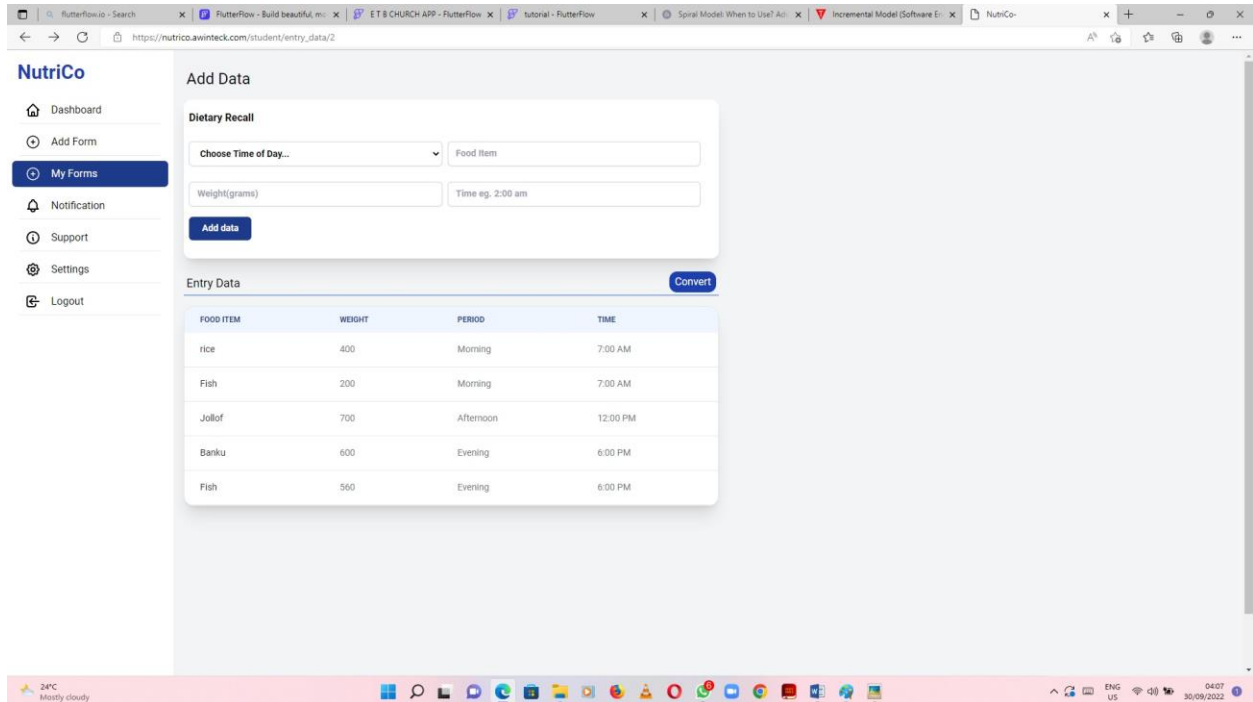
- Dashboard
- Add Form
- My Forms**
- Notification
- Support
- Settings
- Logout

**NutriFood Data**  
Lorem ipsum dolor sit amet consectetur adipiscing elit...

All Data Download

ID	FIRSTNAME	LASTNAME	GENDER	RELATIONSHIP	HOUSE DETAILS	FOOD ITEM	WEIGHT	PERIOD	TIME
002	SUSSY	MANY	mother	Daughter	Hs45	Fish	700	Morning	8:00 AM
002	SUSSY	MANY	mother	Daughter	Hs45	Banku	800	Morning	8:00 AM
Eric Twum Barima	Eric	Twum Barima	mother	relative	Plot c BLK 21	Rice	20	Morning	8:00am
Eric Twum Barima	Eric	Twum Barima	mother	relative	Plot c BLK 21	Oat	70g	Morning	9am
001	Ben	Manu	mother	son	Hs23	Jollof	600	Morning	9:00 AM
001	Ben	Manu	mother	son	Hs23	Fish	300	Morning	9:00 AM
3330000	piet	acheampong	mother	son	amakom	rice	1		

## DIETRY ENTRY DATA PAGE



**NutriCo**

- Dashboard
- Add Form
- My Forms**
- Notification
- Support
- Settings
- Logout

**Add Data**

**Dietary Recall**

Choose Time of Day...

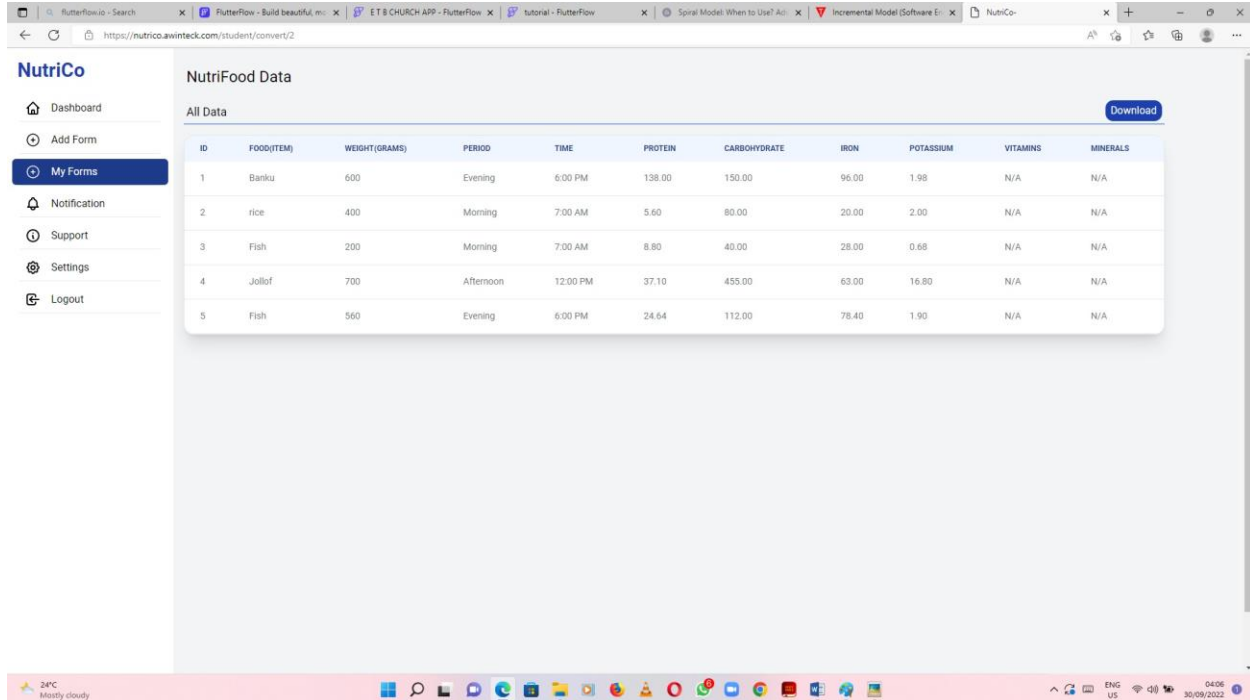
Weight(grams)

Add data

**Entry Data** Convert

FOOD ITEM	WEIGHT	PERIOD	TIME
rice	400	Morning	7:00 AM
Fish	200	Morning	7:00 AM
Jollof	700	Afternoon	12:00 PM
Banku	600	Evening	6:00 PM
Fish	560	Evening	6:00 PM

## CONVENTION PAGE



NutriCo

Dashboard

Add Form

**My Forms**

Notification

Support

Settings

Logout

NutriFood Data

All Data [Download](#)

ID	FOOD(ITEM)	WEIGHT(GRAMS)	PERIOD	TIME	PROTEIN	CARBOHYDRATE	IRON	POTASSIUM	VITAMINS	MINERALS
1	Banku	600	Evening	6:00 PM	138.00	150.00	96.00	1.98	N/A	N/A
2	rice	400	Morning	7:00 AM	5.60	80.00	20.00	2.00	N/A	N/A
3	Fish	200	Morning	7:00 AM	8.80	40.00	28.00	0.68	N/A	N/A
4	Jollof	700	Afternoon	12:00 PM	37.10	455.00	63.00	16.80	N/A	N/A
5	Fish	560	Evening	6:00 PM	24.64	112.00	78.40	1.90	N/A	N/A

24°C Mostly cloudy

ENG US 04:08 30/09/2022

## 5.2 CONCLUSION

The main aspect behind NUTRITION DATA COLLECTION AND CONVENTION SYSTEM is that it enabled us to bring out the new ideas that were sustained from implementation come to light to support the nutrition departments across. This project offers nutritionist the opportunity to easily and swiftly record and convert food entry data`s of their clients through this web application. Data collection is also made easy by the Nutrition data collection and convention system since is just a matter of querying the database used by Developing a good system is critical to the success of the system to prevent system failures and to gain wide acceptance as the

best method available. A good NDCNC system requires ten characteristics which this system already has. These are:

Accuracy Convenience Reliability Verifiability Flexibility Consistency Democracy Mobility Social Acceptance Privacy In analyzing, designing, implementing, and maintaining standards, we considered these characteristics as the foundation. These standards were made national.

### **5.3 RECOMMENDATION**

After my research and my finalization of this project, I highly recommend that the this software system) serves to be the best to be put in use especially in the 21st century where human beings are embracing technology and where there is malicious struggle of good Dieting all over the world. I therefore recommend that various hospitals and nutrition departments should put this project or software technology at practice to phase out some of the problems they go through during data collection.