Case Study: UML Framework of Obesity Control Health Information System

Majed Alkhusaili(✉) and Kalim Qureshi

College of Life Sciences, Kuwait University, Sabah Al-Salem City, Kuwait

Abstract. The aim of this paper deals with a model design through the Unified Modeling Language (UML) to present an Obesity Control Health Information System. Due to the increasing numbers of obesity around the world and for a better lifestyle of people, this design model is proposed in form of UML diagrams. To fulfilled the requirements of this paper, a complete case study scenario of the OCHIF is been considered. The outcomes of this model fill the business market needs for such a system. Besides, it will encourage the designers and developers to explore different ways of how to model an entire system using UML diagrams.

Keywords: UML diagrams · Obesity Control Health Information System · System design

1 Introduction

Model transformations play a significant role in software development systems [1–3]. Automation of repetitive and well-defined steps considered to be the key advantage in every developed transformation. It helps to shortening design time and minimizing a number of errors [2]. In an object-oriented approach, analysis, design, and implementation are the consistent signs of progress for any software development system [1].

Nowadays, many software programs support the idea of model transformations. For instance, the Unified Modeling Language (UML), and Computer-Aided Software Engineering (CASE). Both software tools proven to carry out several engineering activates, including modeling and diagraming [3]. However, CASE tools are not incompatible with UML when it comes to hard tasks. The integration tools of UML have to show better results indeed [1]. Due to that, UML has become the most widely used tool for modeling object-oriented software systems [1, 3].

UML is a set of diagram centric design notations that are loosely connected. It allows designers to experience independent views of a software system [4]. Additionally, UML provides various diagrams classified into two major levels of diagrams, i.e. structural diagrams and behavioral diagrams [5]. Structural diagrams reveal the static aspects of the system under development, while behavior diagrams address the dynamic aspects. By implementing these two features in UML, the chance of making the right decisions are more incompatible and the detection of inconsistencies will become even vital for the development of accurate software [5, 6].

K. Arai (Ed.): *Intelligent Computing*, LNNS 285, pp. 173–191, 2021.
https://doi.org/10.1007/978-3-030-80129-8_14

UML has led to various completion of electronic software models. It has brought a lot of positives feedback regarding the improvement of information management [3]. Electronic Health Information System (EHIS) is a great example to be considered as a type of electronic software model. EHIS is a collection of structures and procedures arranged to collect information that supports health care management decisions at all levels of an entire health system [7]. It has the potential to simplify work processes, make procedures more accurate, identify each case separately, reduce the risk of human errors, and improve the performance of the delivery of health services [7, 8].

In this context, Obesity Control Health Information System (OCHIS) is one of many eHealth platforms available in today's world. The idea of modeling an entire software of an obesity system has been a grey area that needs to be highlighted [4]. A method to establish such a system considers being a challenging issue that needs to be addressed.

The focus of this paper is to analyze and model an entire OCHIS. Consequently, this study will be performed using UML application tools. The rest of the paper is being organized as follows:

2 System Document Vision

This section evaluates the potential approach behind the entire system.

Problem Description
Obesity is becoming an alarming issue with implications affecting society and the health-care sector. In response, multi-professional programs with physical activity, nutritional and psychological components have been proposed [9]. Still, due to limited resources, only a small number of patients can be included in these programs. Health Information System (HIS) has the potential to tackle these challenges. Yet little is known about the design and effects of HIS in the domain of multi-professional obesity programs, in particular those tailored to children and adolescents.

Therefore, to address this problem, a HIS software prototype to support obesity interventions is necessary be built.

System Capabilities
The proposed system should have the capacity to:

- Collect and store health data and information about the patients.
- Collect data and information about food eating levels.
- Collect and store data and information about exercises.
- Keep track of health-related conditions.

Business Benefits
Deploying the proposed system will bring about the following business benefits:

- Potential to improve outcomes.
- Reduce costs of health interventions.

- Providing the possibility for self-monitoring.
- Simultaneously supporting the patients' decision making through real-time access to relevant information.

3 System Work Breakdown Structure

The framework parameters can be adjusted based on the needs and the available models as follows:

3.1 Analysis of Requirements and Specifications:

Where the essential information of the system is collected. This includes:

1. Designing proposal and presenting to board – 4 h
2. Meeting with Obesity Department manager – 2 h
3. Meeting with several patients – 5 h
4. Identifying and defining use cases – 3 h
5. Identifying and defining requirements – 3 h
6. Developing workflows – 5 h

3.2 Designing Components of the Proposed System:

Where some diagram types of the system are assigned. This cover:

1. Designing class diagrams – 2 h
2. Designing database schema – 4 h
3. Designing class diagram – 2 h
4. Designing architecture design diagram – 3 h
5. Designing entity relation diagram – 2 h
6. Designing state machine diagram – 2 h
7. Designing deployment diagram – 2 h
8. Designing package diagram – 2 h
9. Designing input and output screens – 4 h
10. Designing overall system architecture – 6 h

3.3 Implementing the Proposed System:

This is to measure whether the system is clear, easy, consistent, and simple to be followed. This include:

1. Coding Graphical User Interface (GUI) layer components – 16 h
2. Coding logical components of the system – 12 h

3.4 Testing and Debugging Proposed System:

Where the performance of the entire system is examined. This cover:

1. Performing system functionality testing – 8 h
2. Performing user acceptance testing – 10 h

3.5 Deployment of the Proposed System:

Where the interaction of the fundamental services are assessed. This include:

1. The integrating system within the organization – 10 h
2. Performing maintenance – 8 h

4 System Use Case

A system use case is considered to be a major processing step in modeling realization [10]. It is a way to avoid system defects by decrements what could go wrong earlier in stages. According to the system requirements, the study demand proposing the following descriptions:

Table 1. Defines the project use cases.

Use case	Description
Look up patient	Using the patient name, find health data and inform out the patient
Enter/update patient information	Enter new or update existing patient data and information
Look up contact information	Using the patient name, find contact information
Enter/update contact information	Enter new or update existing contact information
Enter/update meal plan information	Enter new or update existing meal plan information
Enter/update exercise plan information	Enter new or update existing exercise plan information

Table 1 demonstrates the task of each system use case. It represents the description of all the methods an end-user follows to perform the system. These methods are like requests of the system. It works as a joint that links the system to the user. Each use case is performed as a sequence of simple steps, starting with the user goal and ending when that goal is fulfilled.

Consequently, by looking into the provided use case diagram in Fig. 1, it can be concluded that there are two actors in a position which are the therapist and patient. These actors represent the users of the system. Each actor has a specific role to play; assign according to the requirements of the system e.g., a therapist can view and modify

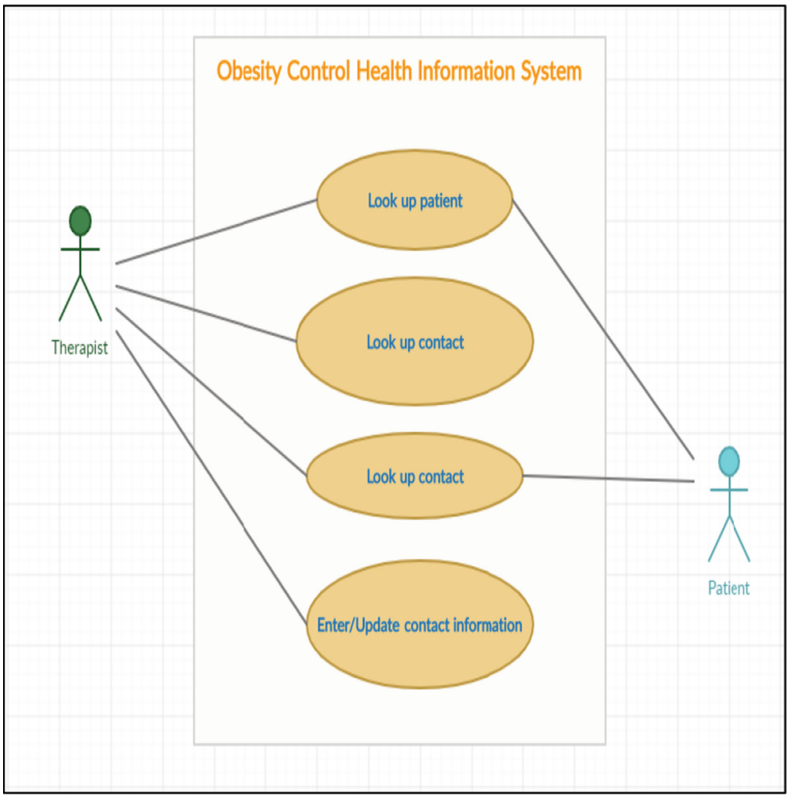

Fig. 1. System use case diagram.

all the associated tasks to the system. However, the patient is only limited to some of these tasks. The idea behind this diagram is to define the relationships of each task by their actor.

5 System Preliminary Class Diagram

A class diagram can be defined as a static type of diagram, for which it represents the static views of an application. It describes the attributes, the associated methods of the class, and the constraints imposed on the system. Besides, it works as a collection of classes, interfaces, associations, and collaborations which eventually perform a structural diagram to an entire system.

Based on the system requirements, a division of five classes has been modeled using the UML software program. All the associated classes are being mapped with each other so that the entire design system is being represented as one. The assigned numbers refer to the orders system which describes a particular aspect of the related system [4]. For instance, based on Fig. 2, at least one therapist has to examine one or more patients. Similarly, at least one therapist should provide one or more reports to the patient. On

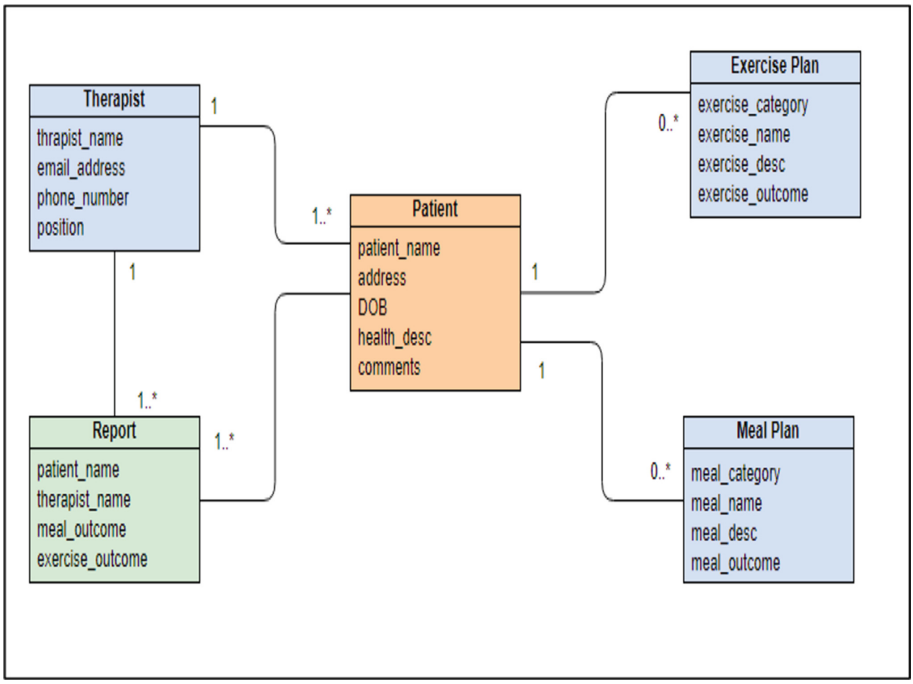

Fig. 2. A UML representation of object class diagram with attributes.

the other hand, the patient may receive either more than one exercise and meal plans or none at all. Depending on the assigned treatment plans.

6 System Activity Diagram

An activity diagram is another representation of UML models, which describe the dynamic aspects of the system. It consists of flowcharts that display the flow of one activity to another. Due to the various types of activity diagram flow features, the control flow from one operation to another can be either sequential, branched, or concurrent. These types of flows assist the reader to follow and understand the order elements of each task according to the diagram easily [11]. Furthermore, the activity diagram provides tools that facilitate the use of all kinds of flow control such as fork, join, etc.

UML activity diagram in Fig. 3, explain the following processes; the process order starts when the system receives a login verification from both the patient and therapist. Then, the system provides the details as being organized. After that, the therapist shall select the patient and initiate the order. Next, at the diamond-shaped flow-node, the therapist evaluates the condition within the square brackets of the first flow. If the condition is yes, the therapist chooses that flow and move to complete the order node. Otherwise, the order will be rejected and move to add new tasks till it merged as a complete order. Once the control flow moves to the merge node, the order fulfillment is considered to be completed. In the end, a close order action is executed.

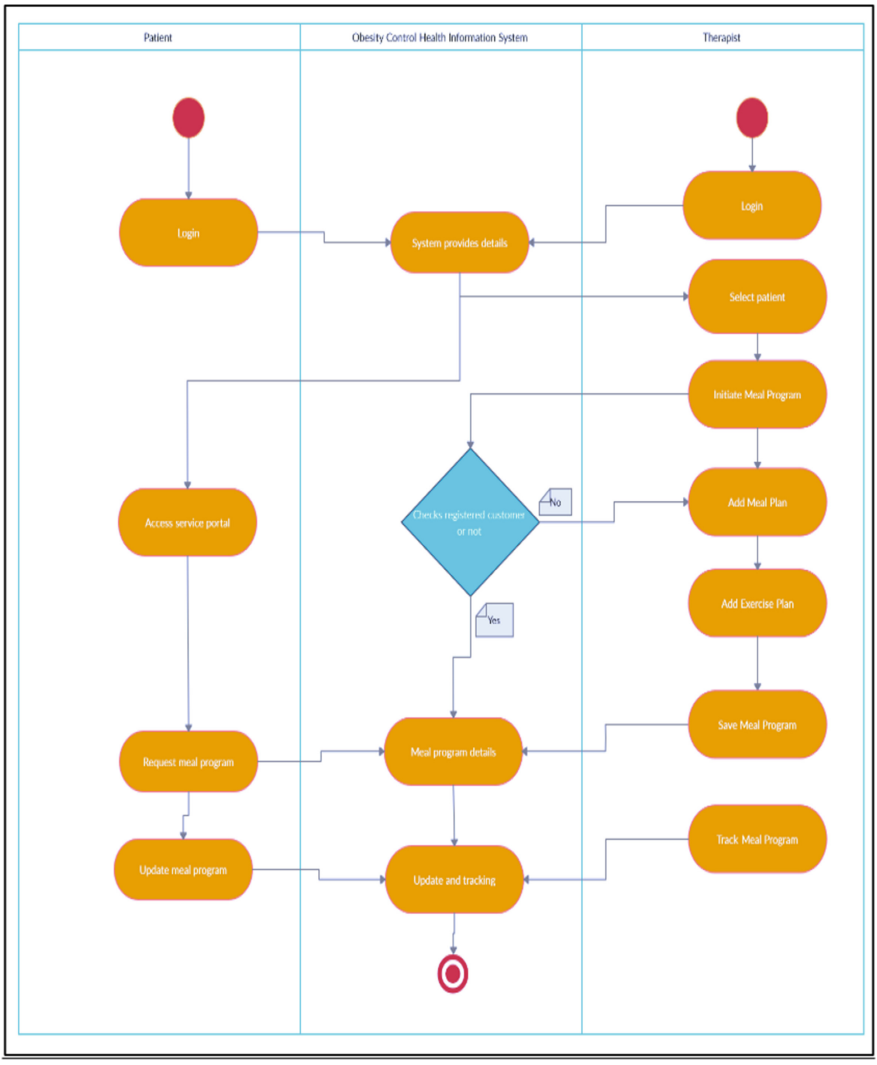

Fig. 3. System UML activity diagram.

7 System Design Database Schema

A database schema is a representation model that is usually created at the start of any system. It is a method planned out to build the infrastructure of the entire system before being implemented. For that, once the schema is implemented, it is difficult to make any changes [12]. The following Table 2 shows the database schema of the required system.

Obesity control databases have been divided into four main classes. Each class has been defined with a set of commands. These commands are been represented as attributes. The purpose of each attribute is to describe the number of databases often refers to as schema. With the help of system developers and administrators, the schema can be

Table 2. System databases schema.

Class name	Attributes
Therapist	therapist_ID: integer [PK] therapist_name: string email_address: varchar phone_number: integer position: varchar
Patient	patient_ID: integer [PK] patient_name: string address: varchar DOB: date health_desc: string comments: string
Meal plan	meal_ID: integer [PK] meal_category: vachar meal_name: string meal_desc: string meal_outcome: string
Exercise plan	exercise_ID: integer [PK] exercise_category: vachar exercise_name: string exercise_desc: string exercise_outcome: string

converted into Structured Query Language (SQL). It is a modern technology tool used to create, manage, and secure databases of a developed information system [12].

8 System Preliminary Class Diagram

Object-Oriented Programming (OOP) provides various forms of system class diagrams. The preliminary class diagram on the other hand is one of them. It is an illustration of relationships and commands defined between classes with the aid of some modeling software e.g. UML [13]. The preliminarily class diagram of the system has been modeled as follows.

According to the class diagram shown in Fig. 4, the classes are organized in groups which all share common characteristics. In general, a preliminary class diagram consists of a flowchart in which classes are fitted in boxes, each box containing either two or three rectangles inside. Depending on the description if it is three rectangles, the top rectangle specifies the name of the class; the middle rectangle holds the attributes of the class; the lower rectangle addresses the methods, also known as the operations of the class. However, sometimes a class diagram comes in only two boxes. In this case, the first rectangle contains the name of the class and the last rectangle describes the methods. Furthermore, a class diagram allows the usage of some arrow lines between

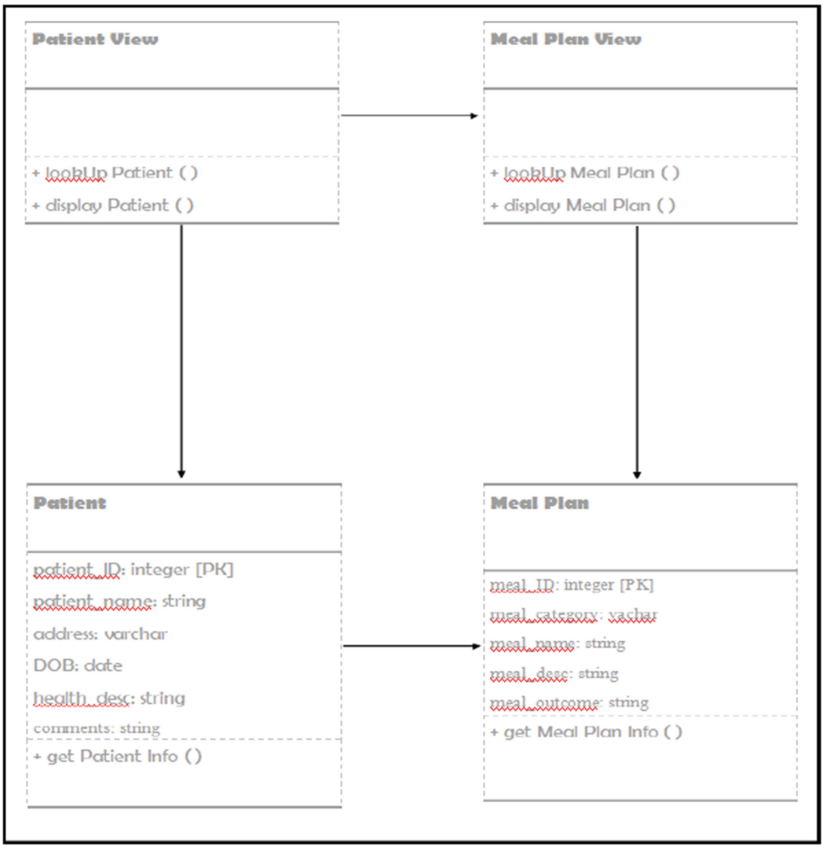

Fig. 4. Representation of system preliminary class diagram.

the classes. These arrow lines connect the boxes. Hence, It defines the relationships and associations between the classes.

9 Subsystem Architecture Design Diagram

A subsystem diagram is a part of UML diagrams, which represent independent behavioral units in the system. The role of the subsystem architecture diagram is to view large-scale components of the modeled system. It is a way to build an entire system as a hierarchy of subsystems. The following model displays the subsystem architectural diagram of the system.

A subsystem diagram normally resembles three modeling items, subsystem packages, subsystem classes, and subsystem interfaces. Based on Fig. 5, the first main package is the system name itself. Consequently, that package has been divided into two internally subsystem packages, represented as the view layer and domain layer. Each of these layers contains two independent subsystem classes. Subsequently, each subsystem classes realize one or more subsystem interfaces that define the behavioral operations of

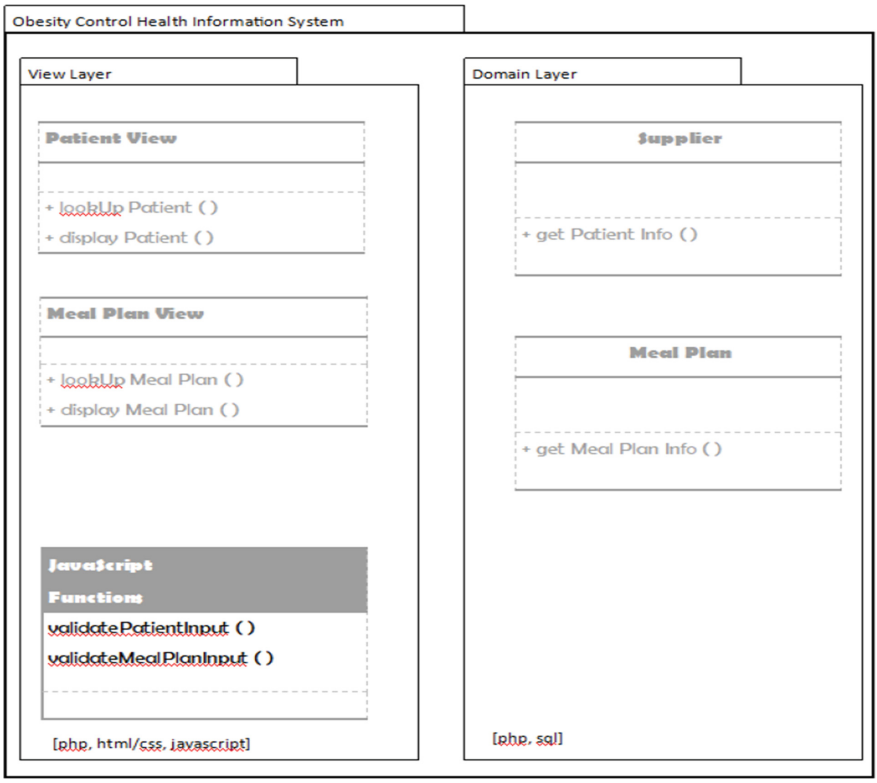

Fig. 5. Subsystem design diagram.

the subsystem. In general, the compartments of all display subsystems diagram aims to minimize the defects from the system by examining all the necessary content and roles for better system development.

10 System Functional Requirements

The paramount functional requirements of this system are:

- The system shall allow patient registration by the therapist.
- The system shall allow the therapist to review and modify patient information.
- The system shall allow report generation.
- The system shall allow communication between patient and therapist.
- The system shall allow the therapist to search for and review patient information.
- The system shall allow the patient to review their meal program and therapist instructions.

11 System Non-functional Requirements

The short-term requirements of the system are:

- Security requirements.
  - o The system shall authenticate each user with a unique identification method.
- Usability requirements.
- Reliability requirements.
- Performance requirements.
- The system shall allow patients to sign in.
- The system shall allow therapists to sign in.

12 System Stakeholders

Based on the system requirements, the decision will be answered into two categories as follows:

12.1 Internal Stakeholders: Represent Direct Interaction with the System.

- Managers.
- Therapists.
- Staff

12.2 External Stakeholders: Represent Non-direct Interaction with the System.

1. Patients.
2. Parents.
3. Public.

13 System Information Collecting Techniques

Some of the collecting system information methods are:

1. Interviewing internal and external stakeholders.
2. Observing activities and processes within the organization.
3. Conducting research online.
4. Distribution and collection of questionnaires.

14 System User Goals

This section presents the tasks of each user according to the system requirements. Each user has been assigned to sets of criteria as follows.

Table 3 represents the system allocating roles according to every user's perception. Choosing the right people consider to be a primary success to every system. Initiating this step allows to organize the data flows between sub-users which leads to more system efficiency and less occurring of error.

Table 3. System user goals.

User	User goal and resulting use case
Patient	Access health information Search for a meal program View and update the current meal program
Therapist	Add/update patient information Add/update meal program information Allocate meal program to patients
Parent	View patient information Track patient information
Department manager	View therapist performance View patient reports

15 System External, Temporal, and State Events

- External event
  - o Patient attempting to login to access system at home.
- Temporal event
  - o Reports are generated at the end of every week.
- State event
  - o Therapist allocating a meal program to a specific patient.

16 System State Machine Diagram

State machine diagrams design to describe the state-dependent behavior of an object being processed in a system. It is mainly designed for objects. However, it can be used to any methods that had have behavior to other entities e.g., use cases, actors, elements, and system subsystems [14]. The following model represents the state machine diagram of the system.

Figure 6 represents the system state machine diagram. It covers the mechanical performance and state processes of the system's behavior. According to the diagram, a doctor considers being an object in the system. The state machine diagram allows the doctor to act differently from one event to another depending on the functions being processed.

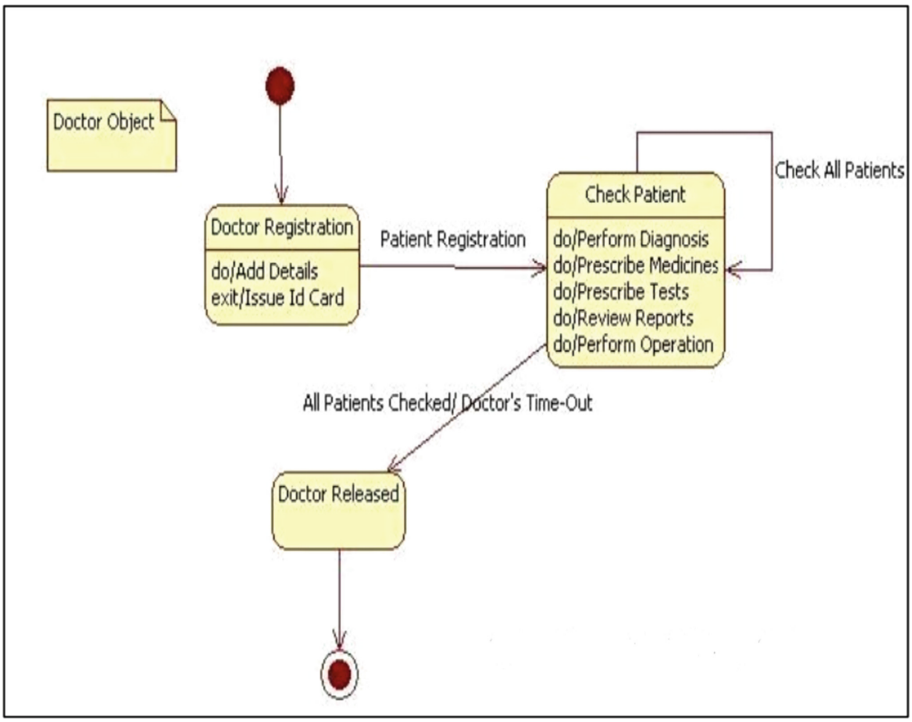

Fig. 6. System state machine diagram.

17 System Use Case Description

This section shows the system use case descriptions as represented in Table 4.

Table 4. System use case descriptions.

Use case name:	Create patient account
Scenario:	Creating a patient account for a new obese patient
Triggering event:	A new obese patient wants to start an obesity control program and visits a therapist physically
Brief description:	The therapist takes basic details and account information, including contact, address, and billing information
Actors:	Therapist, patient
Related use cases:	Might be invoked when an existing patient record needs to be modified

(*continued*)

Table 4. (*continued*)

Use case name:	Create patient account	
Stakeholders	Obesity department, hospital management	
Pre-conditions:	Patient registration sub-system must be available	
Post-conditions:	• Patient account must be created and saved	
	• The patient address must be created and saved	
	• Patient DOB must be created and saved	
	• Patient health description must be created and saved	
The flow of activities:	Actor	System
	1. The patient shows a desire to enroll and provides basic information 2. The patient provides full names 3. Patient provides address 4. Patient provides DOB 5. The patient provides a health description	1.1 System creates a new patient record 2.1 System saves the patient name 3.1 System save the s patient address 4.1 System saves patient DOB 5.1 System saves patient health description
Exception conditions:	2.1 Incomplete patient names 3.1 Invalid patient address	

18 System Sequence Diagram

A sequence diagram is one of the most fundamental UML diagrams which represent the behavioral aspect of the system under development. It is designed according to a specific use case scenario that encompasses both normal and alternative flows [15]. The role of the sequence diagram is to describe the time order of messages that have been generated between objects [5]. A complete sequence diagram for the system is been represented as follows.

According to Fig. 7, in a sequence diagram, all the objects are positioned at the top of the diagram along the X-axis. The new objects shall be added on the right side of the other objects as further interactions are been initiated. On contrary, messages are placed along Y-axis, dragging down to reveal time increasing. Furthermore, objects in the sequence diagram are categorized into three stereotypes, for instance, boundary, entity, and control [15]. A boundary which is also known as the interface is the object that deals with the communication between actors and system. It has been defined from the use case scenario. On the other hand, an entity object describes procession information controlled by the system, which is usually related to a special relevant concept in life. Nevertheless, a control object drives the interaction flow of one use case scenario. It works like glue between boundary and entity objects. In a sequence diagram, there are more than one boundary and entity objects. However, only one control object is recommended to be used in the use case [15].

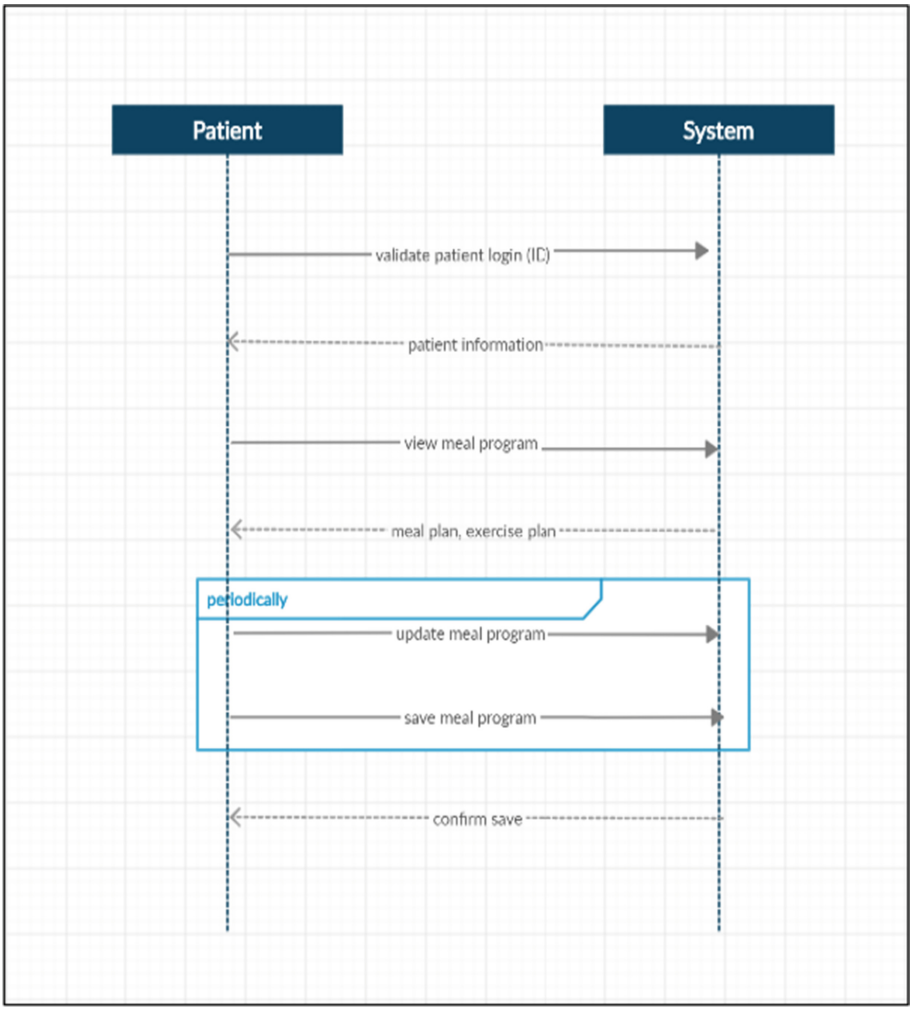

Fig. 7. System sequence diagram.

19 System Design Component Diagram

In UML, a component diagram is a static kind of diagrams. Typically, component diagrams are used to visualize the physical aspects of the system, e.g., executables, files, libraries, and documents which all reside in a node [16]. A single component diagram cannot represent an entire system. However, a collection of component diagrams are used to represent an entire system. The following model represents the component diagram of the system.

A component diagram usually resembled after the system is being designed. Particularly, when the artifacts of the system are ready, the component diagrams are used for system implementation [16]. According to Fig. 8, the user will be able to reach the

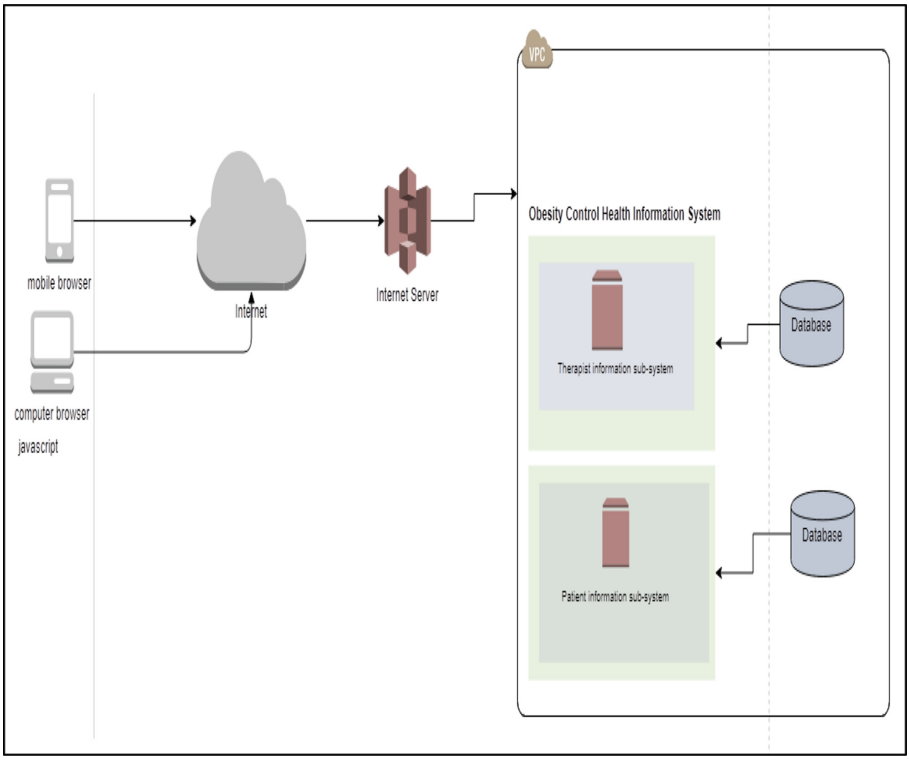

Fig. 8. System component diagram.

system internet server through the internet network. Then, the user will be able to access the system through a secure Virtual Private Cloud (VPC). The role of VPC is to help organizations protect the confidential details of their customers, which are known to be the users, and their system assets [16]. Furthermore, in a sequence diagram, it is known that the artifacts are the files. Consequently, there are two identified artifacts represented as the therapist information subsystem and the patient information subsystem. Both of these artifacts are being linked into two different dependent databases which work as an engine to sort, change, or serve the information within the system.

20 System Deployment Diagram

A deployment diagram is another representation of a static diagram, which describes the deployment view of the system. It is usually modeled to give an overview look at the hardware components of the system. Precisely, both the component diagram and deployment diagram are somehow related to each other. The component diagram defines the components and the deployment diagram shows how these components are been deployed in system hardware [17]. The next diagram represents the deployment diagram of the system.

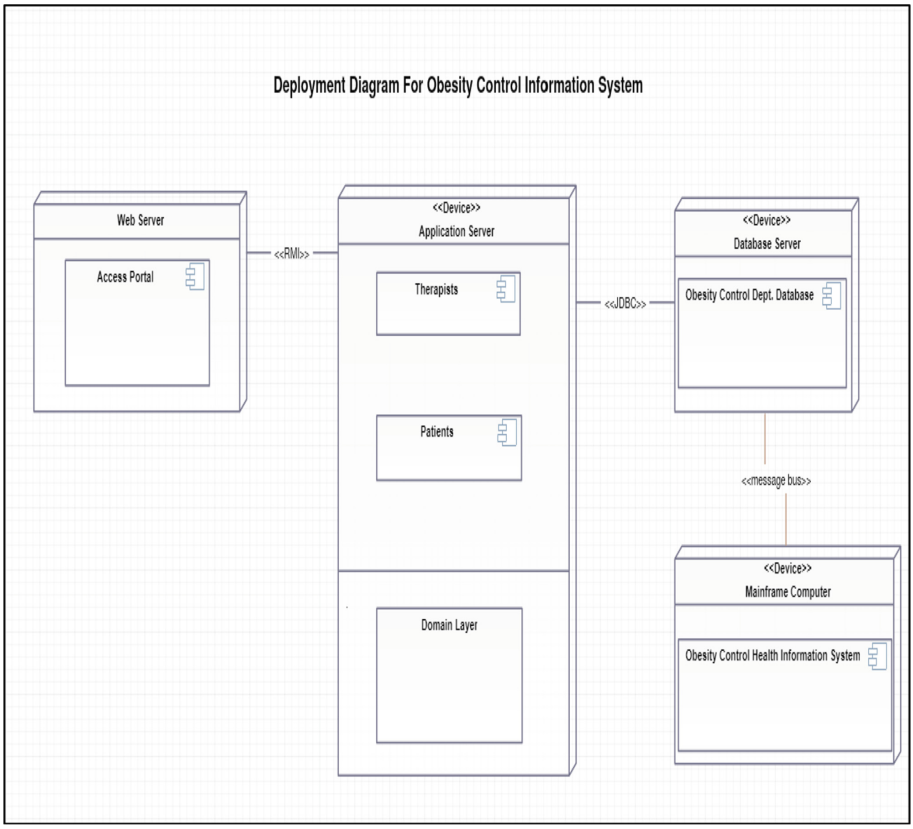

Fig. 9. System deployment diagram.

Deployment diagram is mainly used to control various engineering parameters, e.g., system performance, scalability, maintainability, and portability [17]. A simple deployment diagram has been utilized in Fig. 9, based on the system requirements. This system uses a different type of nodes which are represented as the portal layout, available devices such as the application and computer mainframe, and servers. The system is assumed to be a web-based program, for which it will be deployed in a lumped environment to get the benefits of web server, application server, and database server. All the connections will be processed using the internet network. The control flows from servers back to the environment in which the software components reside.

21 System Interaction Diagram

An interaction diagram is used to describe the dynamic nature of a system by defining various types of interaction among different elements in a system. Sometimes visualizing the interaction could be difficult to process. However, the developers might use different types of models to capture different aspects of the interaction [18]. According to Fig. 10,

the dynamic aspect in this situation is the therapist. In this way, the system flow messages will be described along with the structural organization of the interacting objects. The following representation explains the interaction diagram of the system.

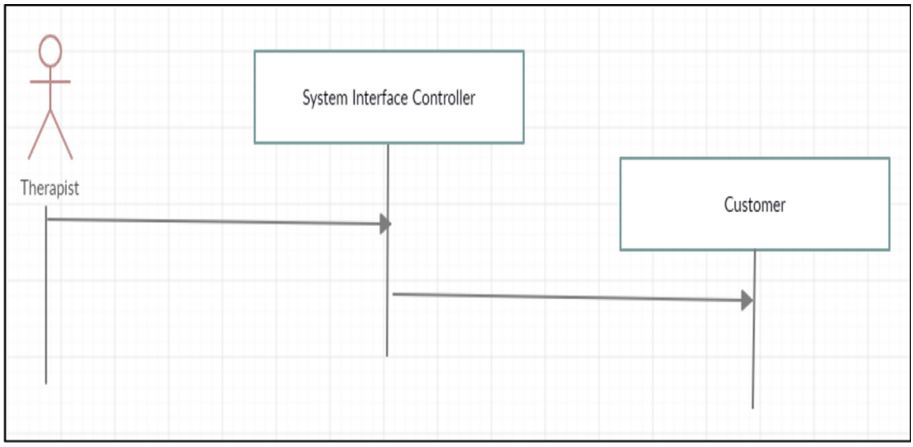

Fig. 10. System interaction diagram.

22 Conclusion

From the above work, it is obvious that a complete model of an entire system can be generated using UML applications. With the aid of these promising software tools, OCHIS has been successfully designed. By applying various UML techniques, nearly ten diagrams have been generated and analyzed, and more than three tables along with research definitions have been collected and classified as the current body of evidence regarding the approach of this paper. The result of this study can be used to encourage the designers and developers to explore the different ways of how to model an entire system using UML diagrams.

The object of this paper was to analyze and design an entire OCHIS using some sort of modern software programs such as UML. The outcomes of this research have been achieved and they can be represented as suitable tool-chains to fulfilled business needs.

References

1. Massago, M., Colanzi, E.: An assessment of tools for UML class diagram modeling: support to adaptation and integration with other tools. In: SBQS 2019: Proceedings of the XVIII Brazilian Symposium on Software Quality, pp. 10–19. https://doi-org.kulibrary.vdiscovery.org/10.1145/3364641.3364643 (2019). Accessed 19 Oct 2020.
2. Technicznej, B.: Construction of UML class diagram with model-driven development. Wojskowa Akademia Techniczna, Redakcja Wydawnictw WAT, ul. gen. S. Kaliskiego 2, 00-908 Warszawa, 65(1), 111–129 (2016)

3. Sergievskiy, M., Kirpichnikova, K.: Optimizing UML class diagrams. ITM Web Conf. **18**, 03003 (2018)
4. George, R., Samuel, P.: Fixing class design inconsistencies using self-regulating particle swarm optimization. Inform. Softw. Technol. **99**, 81–92 (2018). https://doi.org/10.1016/j.infsof.2018.03.005. Accessed 20 Oct 2020
5. OMG: OMG Unified Modeling Language (OMG UML) Version 2.5.1. Object Management Group (2017)
6. Torre, D., Labiche, Y., Genero, M., Elaasar, M.: A systematic identification of consistency rules for UML diagrams. J. Syst. Softw. **144**, 121–142 (2018). https://doi.org/10.1016/j.jss.2018.06.029. Accessed 20 Oct 2020
7. Khumalo, N., Mnjama, N.: The effect of ehealth information systems on health information management in hospitals in Bulawayo, Zimbabwe. Int. J. Healthc. Inform. Syst. Inform. **14**(2), 17–27 (2019). https://doi.org/10.4018/ijhisi.2019040102. Accessed 20 October 2020
8. Phichitchaisopa, N., Naenna, T.: Factors affecting the adoption of healthcare information technology. Exp. Clin. Sci. Int. J. **12**, 413–436 (2013)
9. Pletikosa, I., Kowatsch, T., Maass, W.: Health information system for obesity prevention and treatment of children and adolescents. In: Conference: European Conference on Information Systems, Tel Aviv, Israel (2014)
10. Niepostyn, S.: The sufficient criteria for consistent modelling of the use case realization diagrams with a new functional-structure-behavior UML diagram. Przegląd Elektrotechniczny **1**(2), 33–37, (2015). https://doi.org/10.15199/48.2015.02.08. Accessed 22 Oct 2020
11. Ahmad, T., Iqbal, J., Ashraf, A., Truscan, D., Porres, I.: Model-based testing using UML activity diagrams: a systematic mapping study. Comput. Sci. Rev. **33**, 98–112 (2019). https://doi.org/10.1016/j.cosrev.2019.07.001. Accessed 22 Oct 2020
12. Link, S., Prade, H.: Relational database schema design for uncertain data. Inform. Syst. **84**, 88–110 (2019). https://doi.org/10.1016/j.is.2019.04.003. Accessed 24 Oct 2020
13. Faitelson, D., Tyszberowicz, S.: UML diagram refinement (focusing on class - and use case diagrams). In: ICSE 2017: Proceeding of the 39th International Conference on Software Engineering, pp. 735–745 (2017). https://dl-acm-org.kulibrary.vdiscovery.org/doi/10.1109/ICSE.2017.73. Accessed: 25 Oct 2020
14. Khan, M.: Representing security specifications in UML state machine diagrams. Procedia Comput. Sci. **56**, 453–458 (2015). https://doi.org/10.1016/j.procs.2015.07.235. Accessed 26 Oct 2020
15. Kurniawan, T., Lê, L., Priyambadha, B.: Challenges in developing sequence diagrams (UML). J. Inform. Technol. Comput. Sci. **5**(2), 221 (2020). https://doi.org/10.25126/jitecs.202052216. Accessed 27 Oct 2020
16. Ermel, G., Farias, K., Goncales, L., Bischoff, V.: Supporting the composition of UML component diagrams. In: SBSI 2018: Proceedings of the XIV Brazilian Symposium on Information Systems, Article No. 5, 1–9 (2018). https://doi-org.kulibrary.vdiscovery.org/10.1145/3229345.3229404. Accessed 27 Oct 2020
17. Barashev, I.: Translating semantic networks to UML class diagrams. Procedia Comput. Sci. **96**, 946–950 (2016). https://doi.org/10.1016/j.procs.2016.08.085. Accessed 28 Oct 2020
18. Mani, P., Prasanna, M.: Test case generation for embedded system software using UML interaction diagram, J. Eng. Sci. Technol. **12**(4), (2017). https://doaj.org/article/9bcd7013a5874abdb52c7fc402c86e30. Accessed 28 Oct 2020