

Czech University of Life Sciences Prague

Faculty of Economics and Management

Department of Information Engineering



Diploma Thesis

**Web Application for Local Agricultural Products in
Addis Ababa**

Aleazar Doda Gobena

© 2019 CULS Prague

CZECH UNIVERSITY OF LIFE SCIENCES PRAGUE

Faculty of Economics and Management

DIPLOMA THESIS ASSIGNMENT

Aleazar Doda Gobena, BA

Informatics

Thesis title

Web application for local agricultural products in Addis Ababa

Objectives of thesis

The purpose of this thesis is to design a type of web-based application which can be used as a platform for local farmers to make their small-scale farm products more accessible and enhancing the consumption of fresh and organic quality produces among the society. The first part of this thesis includes brief market research, literature review and requirements analysis for the web application design. The second part contains building a UML model for the web application including prototypes.

Methodology

To achieve the objectives of the thesis, a literature review on UML, software development process and databases will be carried out. Market research will be made to find out the benefits of designing the application. Reviews on other similar application will be carried out and will be analyzed. Finally, the application design will be carried out through UML modeling.

The proposed extent of the thesis

60 – 100 pages

Keywords

Object Oriented Design, Web Application, Unified Modeling Language, Organic Procut, Local farms, Database, Mysql

Recommended information sources

Booch, Grady. 2007. Object-Oriented Analysis and Design with Applications , MA: . [ed.] Addison Wesley. 3rd ed. Boston : s.n., 2007.
Date, C. J. and Darwen, Hugh. 2000. Foundation for Future Database Systems. s.l. : Addison-Wesley, 2000.
Detienne, Françoise. 2012. Software Design – Cognitive Aspect. [ed.] Frank Bott. s.l. : Springer Science & Business Media, 2012.
Dietrich, Suzanne W. and Urban, Susan D. . 2011 . Fundamentals of Object Databases. s.l. : Morgan and Claypool Publishers, 2011 .
Visual Paradigm: What is Unified Modeling Language. [Online] 2017.
<https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/>.

Expected date of thesis defence

2018/19 SS – FEM

The Diploma Thesis Supervisor

doc. Ing. Vojtěch Merunka, Ph.D.

Supervising department

Department of Information Engineering

Electronic approval: 21. 3. 2019

Ing. Martin Pelikán, Ph.D.

Head of department

Electronic approval: 21. 3. 2019

Ing. Martin Pelikán, Ph.D.

Dean

Prague on 25. 03. 2019

Declaration

I declare that I have worked on my diploma thesis titled "Web Application for Local Agricultural Products in Addis Ababa" by myself and I have used only the sources mentioned at the end of the thesis. As the author of the diploma thesis, I declare that the thesis does not break copyrights of any their person.

In Prague on March 29th, 2019

Aleazar Doda Gobena

Acknowledgement

First of all I would like to thank my family for their continuous encouragement, support and love. They have always been there for me in times of my need in all aspects of life.

I would like to thank deeply my thesis supervisor doc. Ing. Vojtěch Merunka, Ph.D. of the Faculty of Economics and Management at Czech University of Life Sciences Prague for his great support and understanding in the completion of this thesis. His irreplaceable guidance in directing me to the right ways led me to this success.

I would also like to thank my teachers, my fellow classmates and the academic staff of the Systems Engineering and Informatics of the Czech University of Life Sciences, Prague, for their support in my learning process for the last two years.

Finally, I would like to thank my friends for their willingness to support me in every way that I needed help. I could not have finished this thesis without your encouragement and your vital contributions.

Web Application for Local Agricultural Products in Addis Ababa

Abstract

In this age of technological transformation, it is inevitable for organizations to adjust and adopt new information technologies and systems rather than traditional ones. Information Technology (IT) and Information Systems (IS) are the backbones for all small to high-level organizations. IT primarily denotes the hardware, software, and telecommunications services and IS refers to the way by which individuals in an organization use technology to gather, process, communicate and store data and information. To cope up with the existing technology, organizations need to design and model a robust information system. One of the vital parts in the process of information systems is designing. Unified Modeling Language (UML) is one of the major designing tools for information systems. In this document, the basic components and terminologies of UML are defined. The latest version of UML is used to explain the comprehensive structure and workflow of a web-based application system. Additionally, the theoretical part discusses in depth the concepts of the software development process and the basic definition of database types and structures. Designing the web application for local agricultural products is used to clarify these theoretical concepts. The application is a web-based application to be implemented to support local farmers in the suburbs of Addis Ababa, Ethiopia, for a smooth transaction of their agricultural produces thereby creating an opportunity for their potential buyers to get these products with a relatively cheaper price and with higher quality. The primary focus of the application is to match the minimum price of a product that a farmer requires and the maximum price that the potential buyer is willing to pay for a certain quantity of the product. The system design is analyzed using the class model, the interaction model, and the state model and their respective UML diagrams. A prototype web application is developed to demonstrate the effect of modeling in the development process.

Keywords: UML, Web-based Application, Object-Oriented Design, Unified Modeling Language, Organic Product, Local Farmers, MySQL, Database

Návrh progresivní webové aplikace pro překladači služby

Abstrakt

V dnešní době technologických transformací je pro organizace nevyhnutelné se přizpůsobit a přijmout nové informační technologie a systémy spíše než ty tradiční. Informační technologie (IT) a Informační systémy (IS) jsou základem všech organizací. IT primárně představuje hardware, software a telekomunikační služby a IS poukazuje na způsob, kterým jednotlivci v organizaci využívají technologie ke sběru, zpracovávání, předávání a ukládání dat a informací. Aby se organizace vyrovnaly s existencí technologií, potřebují navrhnout a vytvořit silný informační systém. Jednou z nezbytných částí v postupu informačních systémů je koncipování. Jednotný modelový jazyk (UML) je jedním z hlavních projektových nástrojů pro informační systémy. V tomto dokumentu jsou vymezeny základní komponenty a terminologie UML. Nejnovější verze UML je použita k vysvětlení všeobecných struktur a pracovního postupu informačního systému založeného na webu. Teoretická část se dále důkladně zabývá konceptem softwaru vývoje procesu a základní definicí typů a struktur databáze. Koncipování webových aplikací pro místní zemědělské produkty je použito k objasnění těchto teoretických koncepcí. Tato aplikace je aplikace založená na webu určená k podpoře místních farmářů v Addis Abbeba, Etiopii, která má zajistit hladký průběh obchodování s jejich zemědělskými výrobky vytvořením příležitosti pro jejich potenciální kupující získat tyto produkty za relativně levnější cenu a s vyšší kvalitou. Hlavním bodem této aplikace je sjednotit minimální cenu produktu, kterou farmář požaduje s maximální cenou, kterou je kupující ochoten zaplatit za určité množství produktu. Systémové koncipování je analyzované pomocí vzorového modelu, interakčního modelu, státního modelu a jejich příslušných UML diagramů. Prototyp webové aplikace je vytvořen pro demonstraci efektu modelace ve vývojovém procesu.

Klíčová slova: UML, Aplikace založená na webu, Design orientovaný na předmět, Jednotný modelový jazyk, bio produkt, místní farmáři, MySQL, database

Table of Contents

1	Introduction	11
2	Objectives and Methodology	12
2.1	Objectives	12
2.2	Methodology	12
3	Literature Review	13
3.1	Web Application	13
3.1.1	Static Web Application	13
3.1.2	Dynamic Web Application	14
3.2	Software Development	14
3.2.1	Software Development Life Cycle Models	15
3.2.2	The Waterfall Model	16
3.2.3	The V-shaped Model	18
3.2.4	Evolutionary Prototyping Model	20
3.2.5	Spiral Model	20
3.2.6	Iterative and Incremental Model	21
3.2.7	Agile Development	22
3.3	UML	23
3.3.1	Origin of UML	24
3.3.2	History of UML	24
3.4	UML Diagrams	26
3.4.1	Structure Diagram	27
3.4.2	Behavior Diagrams	32
3.5	Databases	37
3.5.1	Relational Databases	38
3.5.2	Non-relational Databases	38
3.6	Web Applications as a trading platform for local farmers	39
3.7	Web Applications as a trading platform for local farmers in Ethiopia	42
4	Practical Part	43
4.1	The Web Application	43
4.2	Over view of the Application	45
4.2.1	Functional Requirements	45
4.2.2	Non – Functional Requirements	46
4.3	Analysis Model	46
4.3.1	Data Dictionary	46
4.3.2	Static Model	47

4.3.3	Dynamic Model	51
4.3.4	State Model	59
4.4	Architectural Design	60
4.4.1	Systems Disintegration	61
5	Implementation of the web application system	63
6	Results and Discussion.....	75
6.1	Results	75
6.2	Conclusion	76
7	References	77

List of Figures

Figure 1:	Software Development Life Cycle (Sami, 2017)	15
Figure 2:	Waterfall Model (Sami, 2017)	16
Figure 3	The V-Shaped Model (Sami, 2018).....	19
Figure 4:	Spiral Model (Ghahrai, 2018).....	21
Figure 5:	The Agile Software Development (Ghahrai, 2018).....	22
Figure 6:	History of UML (Ghahrai, 2018)	25
Figure 7:	UML Diagrams (Paradigm, 2017).....	26
Figure 8:	Relationships in Class Diagram (Paradigm, 2017).....	30
Figure 9:	Use Case Diagram and its interactions (Wikipedia, 2018)	33
Figure 10:	Activity Diagram (Paradigm, 2017)	35
Figure 11:	State Machine Diagram (Paradigm, 2017)	36
Figure 12:	Sequence Diagram (Paradigm, 2017)	37
Figure 13:	Mandi Trade App (Kedia, 2018)	40
Figure 14:	Rainbow app (Sharma, 2015).....	41
Figure 15:	Class Diagram of the application (Author, 2019).....	50
Figure 16:	Use Case Diagram (Author, 2019).....	52
Figure 17:	Sequence Diagram of Place Order (Author, 2019)	54
Figure 18:	Sequence Diagram of Report (Author, 2019)	55
Figure 19:	Sequence Diagram of Registration (Author, 2019)	56

Figure 20: Activity Diagram of Order (Author, 2019).....	57
Figure 21:Activity Diagram of Report (Author, 2019).....	58
Figure 22: State Diagram of Make Order (Author, 2019)	59
Figure 23: Class Customer Code in Java (Author, 2019)	64
Figure 24: PHP Code for Registration (Author, 2019).....	65
Figure 25: PHP Code for login (Author, 2019).....	66
Figure 26: Continued PHP code for login (Author, 2019)	66
Figure 27: Home Page (Author, 2019)	68
Figure 28: Register Page (Author, 2019).....	69
Figure 29:Log In Page (Author, 2019)	70
Figure 30: Add Product page (Author, 2019).....	71
Figure 31: List of Potential Buyers after product added (Author, 2019)	72
Figure 32: List of Up Coming Events (Author, 2019).....	73

1 Introduction

In today's world of technological advancement, the role of information systems in improving the quality of daily human life activities is growing highly. Recent developments in the World Wide Web and the Internet introduced a new era for data and information sharing with greater agility and cost effectively all over the world, thereby creating a conducive environment to a web-based information system to flourish.

High cost and limited functionality of traditional software systems resulted in to an increased demand of web-based information systems. Due to high need for small scale local farmers to have a system to manage their agricultural products which covers all requirements and has low cost, this document covers the design of Web-based application for local agricultural products in Addis Ababa.

The difference between Web information systems and traditional information systems is that a huge number of information are organized in a web structure which is served to the users via web pages and hyperlinks. To make this process simpler, for the development and implementation phases, UML as a core modelling language of information systems is used. The Unified Modeling Language (UML) is a standard modeling language that facilitate communication and interaction between all participants in the development process. The UML is appropriate and enough for modeling of the web application.

Web application for local agricultural products in Addis Ababa is designed to give a platform for local farmer to market and sell their products. The system will provide a list of different prices that the producers can offer based on the amount demanded by the customers and the maximum the local customers like restaurants are willing to pay for a certain amount. The system will have two different databases for the different pricings from producers and demand and maximum to pay on the side of clients. The system makes matchings between the minimum the farmers can offer with the maximum the customers are willing to pay.

2 Objectives and Methodology

2.1 Objectives

The purpose of this thesis is to design a type of web-based application which can be used as a platform for local farmers to make their small-scale farm products more accessible and enhancing the consumption of fresh and organic quality products among the society. The application will be designed to give different prices that the local farmer can offer based on the amount demanded and. Especially for restaurants as customers of these agricultural products it will help them to find the best of for their budget. The first part of this thesis includes a brief market analysis, studies of UML, reviews of SQL Database and requirements analysis for the web application design. The second part contains building a UML model for the web application including prototypes.

2.2 Methodology

To achieve the objectives of this thesis, a literature review on UML and SQL Database will be carried out. A brief market analysis using secondary data sources will take place as an attempt to analyze the benefits of the application. Reviews on other similar application (if exists) will be carried out and will be analyzed. Following this, the application design will be carried out through UML modeling. The application will have two separate Databases for different sort of prices offered by the farmers and list of the maximum budget the customers (specifically restaurants) could give to obtain a certain amount of product. The application should be able to match these two lists to show the best match that can be achieved using the available resources for restaurants and the best price for the farmers. Finally, the outcome of the thesis will be UML diagrams of the designed system and prototypes of the application.

3 Literature Review

3.1 Web Application

Web application is a client-server computer program, where there is a browser as a client (including user interface) and a web server, datum mainly stored on the server and logic being distributed among the server and the client. It is designed to run on any browser, work on desktop computers, laptops or mobile devices with established channel for data exchange between the client and server (The Journal of Systems and Software, 2019).

Web application are classified mainly in to two different major categories. These two major categories are Static Web Application and Dynamic web application. The major classification is based on how the applications show their content, their design and architectures, and how they distribute the logic in different ways between the client and server (2019).

3.1.1 Static Web Application

A static web application is a web application with a very low flexibility and is one that is usually written in plain HTML and what is in the code of the page is what is displayed to the user. Their pages are generated by a server and offer little to none interactivity. Usually, no room for personalization exists and any possible change takes effect only after a full page reload (Detienne, 2012).

Even though modifying the contents of static web apps is not easy, every page can be different if desired and the designer is free to put any special forms that a client may ask for in a unique way on different pages. It can be considered more flexible on this term. They are more suited when a very concise information is shared, and interaction is not required.

Static web applications are usually hard to maintain and the excessive amount of data they send and receive could create risks of poor performance. They are not best-suited for a mobile environment, besides, there are ongoing costs related with updating the content (Yaskevich, 2018).

3.1.2 Dynamic Web Application

Dynamic web applications are based on a much more complex web application software framework that controls web page construction and facilitates maintenance. They usually contain databases for interactively loading data and their contents are updated each time the client accesses them. Ability to connect them to databases enables easy pull of data in an organized and structured way. This in turn helps to create product pages or categories of related products sorted in a variety of different ways depending on how the user wants to view them there by enabling creation of content management system which is an interface to allow the client to manage data. They generally have an administrator who can correct or modify the application's contents (Session persistence for dynamic web applications in Named Data Networking, 2019).

The way contents are displayed on user's end in such web applications are not predetermined but rather dynamically changed by an application logic which is implemented on the server side or the client side of the application. Their use cases determine their architectural design and development approach.

PHP, C++, Python and ASP are some of many different programming languages which can be used for dynamic web application development making the structuring and upgrading of the contents very simple without even having direct access to the server. Additionally, it enables the implementation of features like forums or databases and web applications commercial sites providing a lot of information for buyers and sellers (2016).

3.2 Software Development

Software development is defined as a series of phases that provide a common understanding of the software building process including the conception, requirements, design, actual coding, documentation, testing, and bug fixing which are fundamental building blocks of creating and maintaining any sort of applications or other software components (Dietrich, 2017).

In a broader sense, software development involves all the necessary steps involved between the beginning of a required software to the final realization of the desired software in an organized, planned and structured process and it is the process of writing and maintaining the source code in a more specific way. It may involve new developments

through researching to modify existing systems and prototyping and re-engineering to reuse or maintain software products.

The three most common purposes of software are to meet requirements of a bunch of potential users, or for a specific personal use, or to address a stated requirement of a specific client. In order to achieve their specified purposes of their creations and to be able to function in a complete way, most software development processes require systems integration (Tornhill, 2018).

There are many approaches to software development process. Mainly they can be divided in two major categories. The first one is waterfall model which is more traditional approach in contrast with the second software development model; agile software development approach which is a recently innovated modern approach. These two and other similar approaches are discussed as follows:

3.2.1 Software Development Life Cycle Models

System/software development life cycle is a model and methodologies that involve the five key phases of the development process. Because it includes all the phases involved from the conception to its disposal it is called a life cycle. These phases are first requirements analysis, design, implementation and evolution (Information and Software Technology, 2019).



Figure 1: Software Development Life Cycle (Sami, 2017)

3.2.2 The Waterfall Model

Implementing waterfall model in systems development is a direct forward process due to the step by step process of the model. The steps can be slightly different for different purposes, but the major core steps include requirements, design, coding and unit test, system integration, operation and maintenance (Braude, 2014).

The waterfall model emphasizes that a logical progression of these steps be taken throughout the software development life cycle (SDLC), much like the cascading steps down an incremental waterfall. Progress flows in a downward fashion, like the way rushing water, from a height, flows downwards, hence the name "waterfall" was conferred onto this model. This means that any phase in the development process begins only if the previous phase is complete. The waterfall approach does not define the process to go back to the previous phase to handle changes in requirement (Tornhill, 2018).

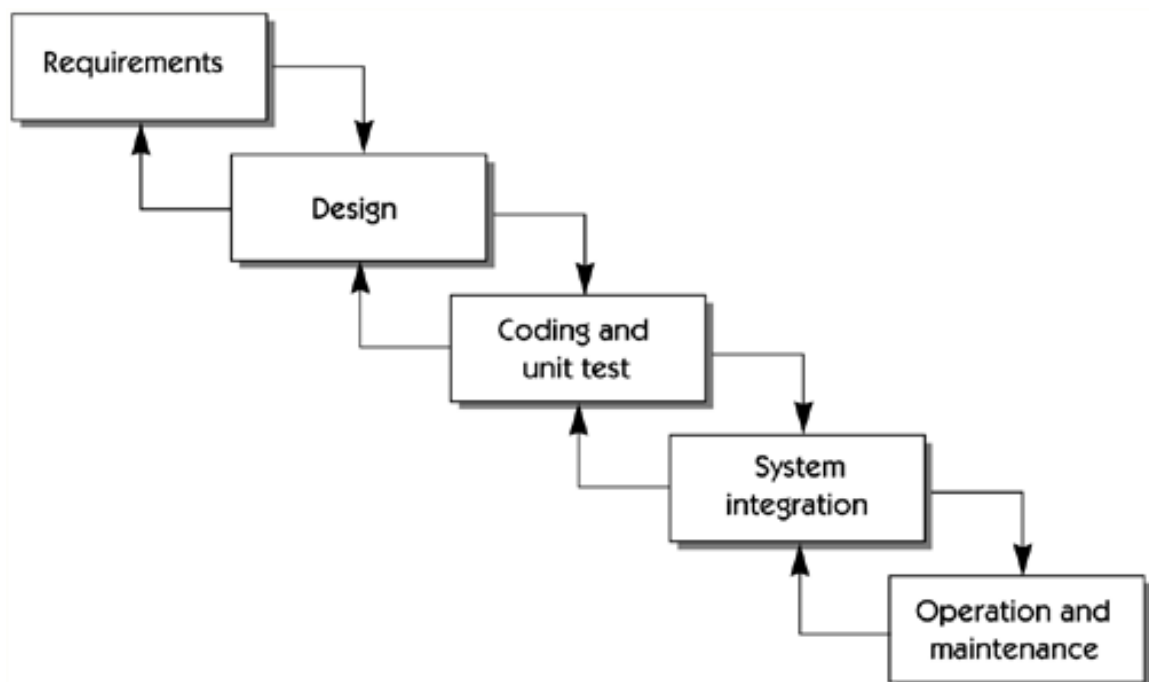


Figure 2: Waterfall Model (Sami, 2017)

Requirements:

This first phase involves understanding what it is needed to design and what its functions and purposes will be. This is a critical phase for without having a clear understanding and knowledge on what had to be designed, one cannot proceed with the development of a given project.

Here, in this phase, the requirements which the software is going to satisfy are listed and detailed to be analyzed by analysis team. Requirements are documented during this phase and clarifications can be sought. The Business Analysts document the requirement based on their discussion with the customer. A successful completion of this phase ensures a effortless working of the upcoming phases, as the programmer will not be burdened to make changes at ending phases because of changes in requirements (Dietrich, 2017).

Design:

This is the phase, where the fundamental work for actual programming and implementation is done. The requirements, that are gathered in the previous phase are broken down into logical units, so that hardware and system requirements are specified precisely defining the overall system architecture. This is the stage, when the software requirements along with the hardware requirements for every unit are identified and the software code to be written in the next stage is created. Then the designs are made to accommodate the interrelation between the various units of the software using algorithms and diagrams. This process makes it easy for implementation (Ousterhout, 2018).

Coding and Unit test:

This phase is when all the actual code is written, and the idea and flowchart of the application is physically created or materialized. It is also known as implementation phase since a software code is written and tested based on algorithms written in the previous phase. In order to check if the correct output is written, the, the system is first developed in small programs called units and each unit is tested for its functionality which is referred to as Unit Testing (Detienne, 2012).

This phase belongs to the programmers in the Waterfall method, as they take the project requirements and specifications, and code the applications based on the algorithm or flowchart designed (Dietrich, 2017).

System integration:

The system is first developed in a small program called units in the previous phase. Following the development, testing of unit is carried out to check for its functionalities and finally all the units will be integrated into a system. A proper execution of this stage ensures that the customer interested in the created software, will be satisfied with the finished product. If there are any flaws, the software development process must step back to the design phase (Braude, 2014).

Operation and maintenance:

This is the final phase and it involves installation of the application on the servers which are designated for its purpose and it involves insuring that the application will run smoothly on the server with out any down time.

As the customer will be using the developed application in this phase, some problems may arise with customers request for changes or enhancements or improper requirement might be found or mistakes in the design process could be found. Therefore, changes and modification are essentials parts of this phase (KarinaCurcio, 2019).

3.2.3 The V-shaped Model

The V- model mostly known as the validation and verification system development model is considered an extension and improvement of the traditional waterfall model for it follows linear step by step phase in the development process. The major difference between the two is in v-shaped model as the as the steps progress instead of down in a linear way the steps are fixed upwards after the coding phase to form the typical V shape (Tornhill, 2018).

The V-shaped model shows the relationships between each phase of development and the associated phase of testing. The horizontal and vertical axes represent time or project completeness and level of abstraction. Each verification phase is associated with a validation phase and the process is carrying out quality checks for all the new things a software developer adds to a project. This model, therefore, introduces a disciplined approach to software engineering.

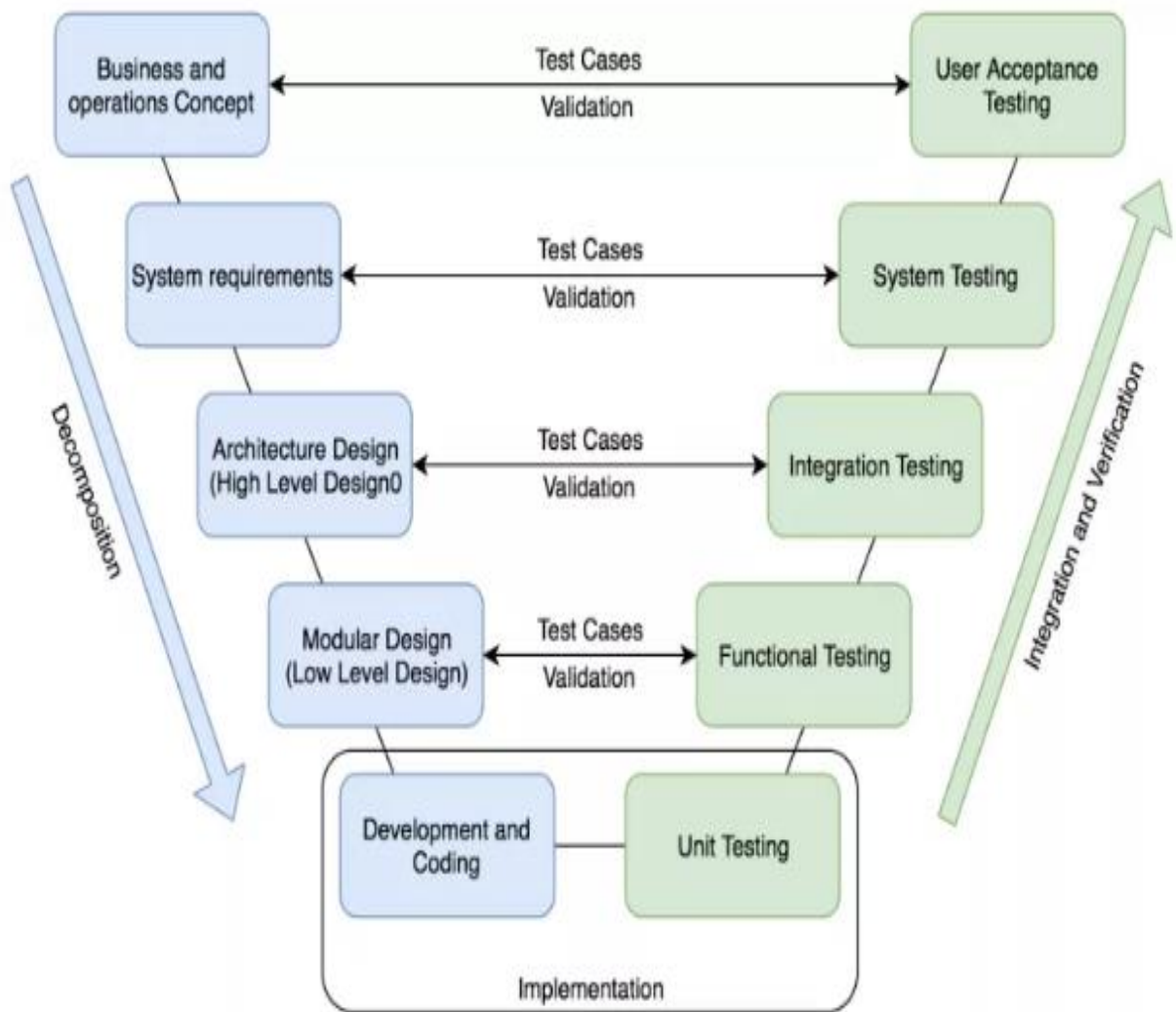


Figure 3 The V-Shaped Model (Sami, 2018)

As the same as in the waterfall any phase in the system development process begins only if the previous phase is fully completed. And on each phase, it has a corresponding related test applied against completion. The technical aspect of the project cycle is considered as a V shape starting with the business needs on the upper left and ending with the user acceptance testing on the upper right.

3.2.4 Evolutionary Prototyping Model

This model typically is an early approximation of a final system or product only including a few aspects of it, and instead of freezing the requirements before a design or coding can proceed, a throwaway prototype is built to understand the requirements though the final product could be completely different. Prototyping is mainly based on the currently known requirements of the customers (Ousterhout, 2018).

The process involves first developing, testing and refining a prototype through continuous communication with customer until a final acceptable prototype that will form the basis for developing the final product is achieved. This gives the customer an opportunity to see the product early in the development life cycle.

This model works best in scenarios where not all the project requirements are known in detail ahead of time and the process starts by interviewing the customers and developing the incomplete high-level paper model. Iteration, trial and error and strong communication between the customer and developers are the backbones of this model (Dietrich, 2017).

3.2.5 Spiral Model

This model is one of the most important Software Development Life Cycle models for it gives a means to support a great handling of risk. It is a combination of a waterfall model and evolutionary prototyping model in which phases starts with a design goal and finishes with the client reviewing the progress (Ousterhout, 2018).

It involves an iterative looping in the process of achieving requirements; taking a single requirement at a time and going through the process steps and looping all over again for the following requirements to develop a robust prototype.

After an evaluation period in the figure below, the cycle is initiated all over again with new functionalities and releasing the following prototype. This way the process continues, with the prototype growing bigger and bigger with every iteration. The theory behind this is that set of requirements are hierarchical naturally allowing additional functionalities to build on the previous efforts made. This is typically a good practice for systems in which the entire problem is well defined from the start.

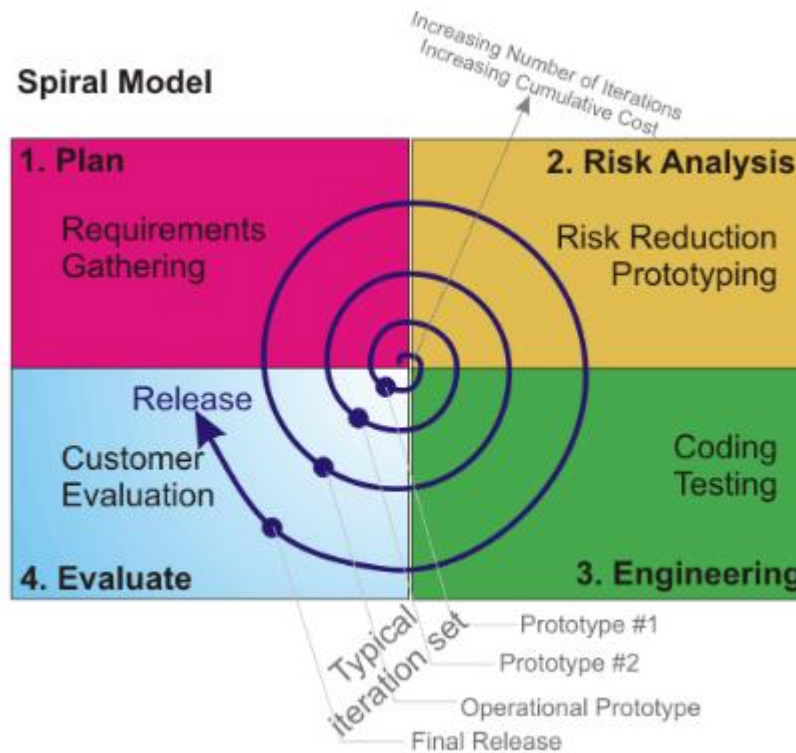


Figure 4: Spiral Model (Ghahrai, 2018)

3.2.6 Iterative and Incremental Model

The Iterative and Incremental model begins with constructing a small part of a total system with its unique functionalities so that it will be in a deliverable state. The product is decomposed into several components, each of which are intended and constructed distinctly. Increased functionality and addition of features is expected at the end of each iterative cycle which includes both development and maintenance. The product is defined as finished when it satisfies all its requirements which are set at the beginning of the project (Dietrich, 2017).

Upon completion each component will be presented to the customer. This allows partial utilization of the product and avoids a long development time there by creating good environment to deal with and fix defects, if any, on early stage of development cycle which creates a large initial capital outlay with the subsequent long wait avoided. Besides it also helps ease the traumatic effect of introducing completely new system all at once.

3.2.7 Agile Development

The agile development model refers to a group of software development methodologies involving an iterative approach to a software development that helps participating teams to deliver value to their clients with higher agility and fewer inconveniences. Teams are self-organizing and cross-functional with a natural tendency to respond to changes quickly and works are divided in to a small but consumable manner with increments. Requirements and solutions evolve through collaboration between cross-functional teams (Influence of software development agility on software firms, 2019).

Scrum, lean, Kanban, extreme programming, crystal, dynamic systems development method and feature driven development are some of the major types of the agile software development methodology. Below the figure shows how the agile methodology basically works (Dietrich, 2017).

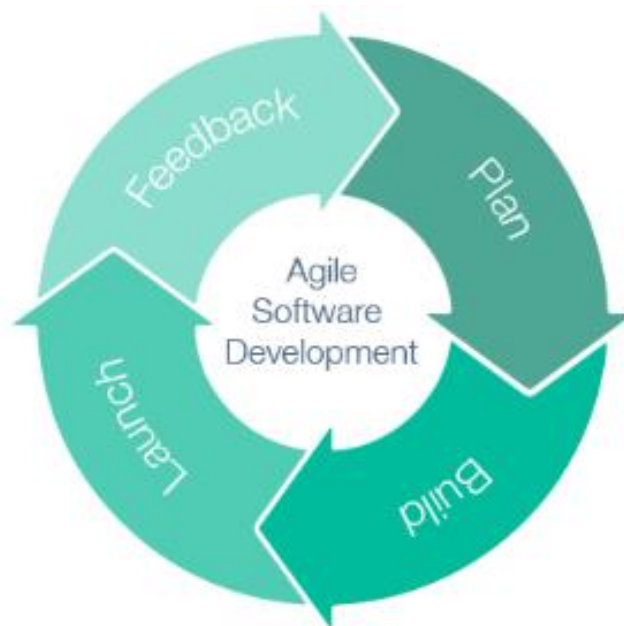


Figure 5: The Agile Software Development (Ghahrai, 2018)

3.3 UML

In this era of technological advancement, human race's life style and activities are entirely depending on established digital systems. As technology and social interaction progresses over the years, digital systems are enormously growing in their size, depth, delivery, complexity, and importance.

Though the high quantity in the growth of information systems, it is not without various quality issues. Several system software developments come to an end before even they get to the point of completion. Various reasons can be pointed out for this problem but there is a one common denominator to all and that is imprecise and/or inadequate end-user requirements and inability to completely handle them with modification on the developer's side. Most of the root sources of the problems can be up rooted by building the foundation of the systems with a predefined and strong development process. One of the most popular facility for designing a system and a tool for effective communication among stakeholders of the development process is the UML (Wazlawick, 2014).

The Unified Modeling language is all purpose modeling language for designing a system's blueprint. It is used as a standard way to clearly visualize, stipulate, and document the tools of a system. Its expressive nature and ability to give various perspectives to view all parts of a system makes appropriate to use UML for modeling systems ranging from enterprise information systems to distributed Web-based applications and even to hard real-time embedded systems (Paradigm, 2017).

The object-oriented analysis and design (OOA&D) wave in the late '80s and early '90s are the predecessor of The Unified Modeling Language (UML). Objects in UML consists of data representing the state and behavior of an object and methods which determine the characteristics of these data. Objects are the real-world entities that exist and are contained in classes in UML, which forms a hierarchical structure to model and mimic real-world systems.

The UML is not a method it is rather called a modeling language because in principle all methods should consist at least both a modeling language and a process. Modeling languages mainly use graphical representations and notations to explain system designs in giving advice on steps to be taken in developing a design.

3.3.1 Origin of UML

Since UML is designed to be used for various types of applications, including distributed systems, analysis, systems design and deployment, it provides a standardized notation which will be used on all object-oriented methods of systems development. It is a notation that revolved out of a combination of OML (Object Modeling Technique), Booch (Grady Booch) and OOSE (Object-Oriented Software Engineering (Ivar Jacobson) over the years. These three modeling methods have their own area of focus and one is preferred over the other for the unique advantages they possess for system developers in 1990s. OML is used as a best tool for the purpose of analysis and data-intensive software, Booch was mostly preferred for its excellence in designing and implementation and OOSE is typically preferred for its Use Case modeling techniques (Paradigm, 2017).

The merging of the ideas between Booch and Jim Rumbaugh (creator of OMT) in 1994 as Jim joined Grady Booch laid great foundation in the development of UML for it resulted in the birth of Unified Method. Following this, in 1995, the concept of use cases introduced in to the unified method as Ivar Jacobson joined the previous two to form the full picture of what now we call the Unified Modeling Language (UML) (Paradigm, 2017).

Other notations like Mellor and Shlaer, Coad and Yourdon, Wirfs-Brock, and Martin and Odell are some of the notations in 1990s which contributed to the development of UML. Apart from these developments in the evolution of UML, it also involves new concepts which are not in the previous methods.

3.3.2 History of UML

The first Request for proposal known as RFP was initiated and catalyzed by Object Management group which paved a way for different organizations to join in the realization process in 1996. Following this, a partner's consortium was established among many organizations like HP, IBM, Microsoft and Oracle which are willing to allocate their resources towards building UML version 1.0 in its well defined, communicative, influential and generally applicable form to be submitted the object management group in 1997 (Paradigm, 2017).

In the same year IBM and 5 other partners also submitted their own versions of UML to the OMG as a distinct RFP response. All these different ideas from different

organizations working together merged together and synthesized to give rise to the revised UML version 1.1. This version is an improvement and clarification in the semantics of the previous UML version 1.0 and the process incorporating the contributions from the new partners. This version was submitted to the OMG for approval and it was accepted in the fall of 1997 and enhanced in the range of versions 1.1 to 1.5 and slowly growing in to UML 2.1 and now it ended with the current version which is UML 2.5 as it is visually explained in the figure below (Vrana, 2016).

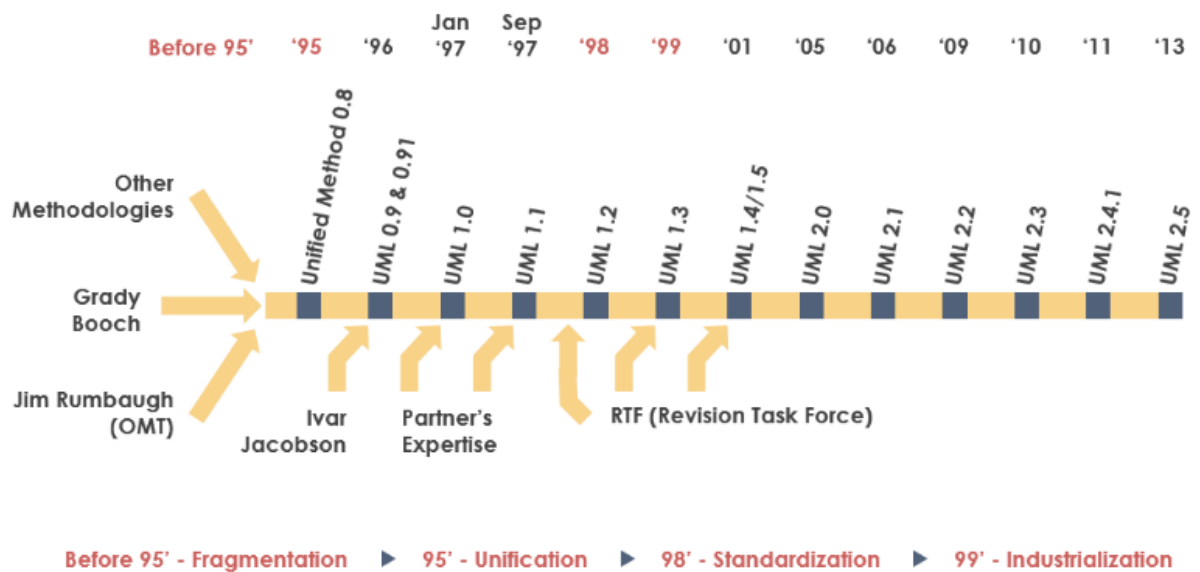


Figure 6: History of UML (Ghahrai, 2018)

One of the very early attempts in the unification of constructs of UML took place in 1994 as Coleman's fusion method even though it did not include the three main authors of the basic methods (Booch, Jacobson, and Rumbaugh) and was also quite late to market with a book explaining the approach. This method was overtaken and consumed as the sequence of events in 1994 took place with Booch and Rumbaugh come together in Rational Corporation to work on unified method (Reasoning about UML/OCL class diagrams using constraint logic programming and formula, 2019).

New Visual syntaxes were introduced with the development of UML 2. Parts of the modification are replacements and clarification for existing version's syntax and parts of it are entirely new semantics added to the language. One of the unique characteristics

of UML is provision of several means to display visually particular elements of a model which is not the same for all modeling tools (Paradigm, 2017).

Even though the fundamental principles for the most parts remain the same, UML 2 brings many syntactic improvements to UML in contrast with UML1 there by letting adoption of using UML 2 a seem less for users of the previous versions. Major changes with the new UML 2 are to its metamodel and these are not come across straight on most of modelers. This is a model of UML which itself explained in a subset of UML. Definitions of the syntax and semantics that could possibly encountered are precisely stated there by making the changes to the UML metamodel greatly about improving the precision and consistency of the UML specification.

3.4 UML Diagrams

Since a development process involves variety of stakeholders playing different roles, the unified modeling language specification version 2.5 is composed of a lot of different diagrams expressing different aspects of system development from range of different viewpoints. The diagrams can be categorized in to two broad classifications. The first one focusing on the static aspects of a project and more expressing the structural side describing objects of a system and the relations between them. The second one focusing on the dynamic aspects of a project expressing the general types of behavior of objects and how they evolve through time via different interactions among them.

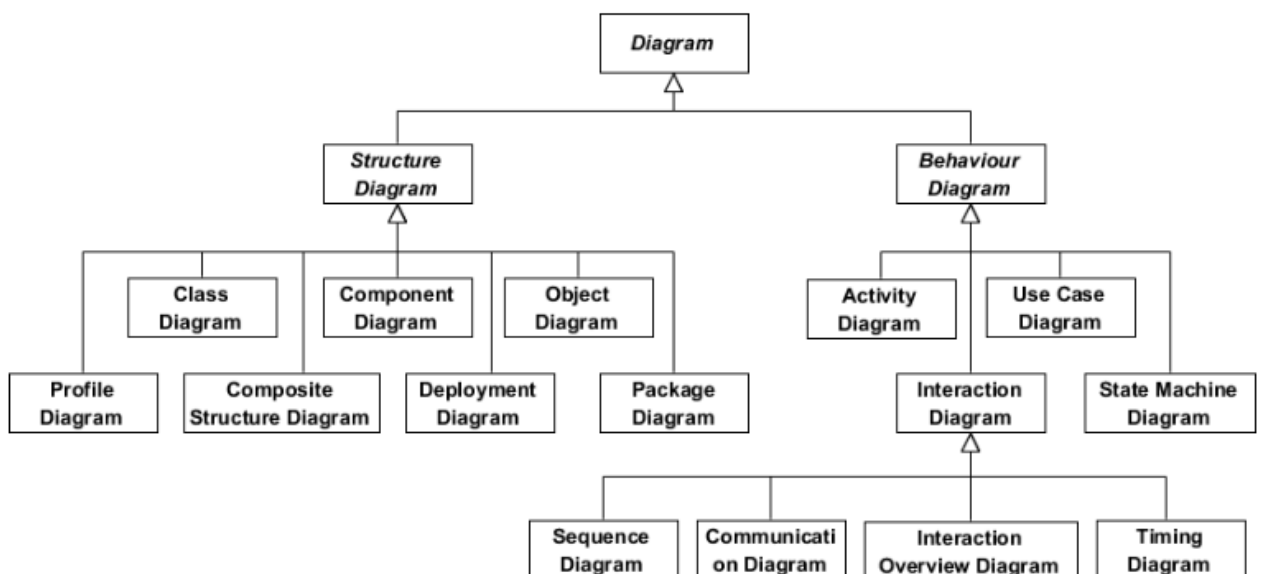


Figure 7:UML Diagrams (Paradigm, 2017)

3.4.1 Structure Diagram

These diagrams express and define a static architectural structure of a system where they are used to model elements that build up a system specification that are regardless of time changes. Object classes, interfaces and physical components are main elements that show relationships and dependencies representing the meaningful concept of a system. Structural diagrams are not designed to show specifics of behavioral dynamics but rather to show relationships of classifiers displayed through it (Paradigm, 2017). Class Diagram, Component Diagram, Deployment Diagram, Object Diagram, Package Diagram, Composite Structure Diagram and Profile Diagram are some of the main subsets of structural Diagrams and will be discussed shortly as follows, in this material.

Class Diagram

One of the most common and central diagrams among UML is the so-called Class diagram describing the kinds of objects elements in the system and several types of static relationships that is existing among them. By (Booch, 2007) class diagrams are defined as

“The descriptor for a set of objects that share the same attributes, operations, methods, relation- ships, and behavior.”

This can be summarized as a set of objects that share the same kind of behavior and attributes. It basically addresses the static structural design and process view of an information system. In a class diagram while all objects should have the same kind of attributes, the same sort of operations and the same type of operations; but the values in the attributes could be different.

All class instances i.e. objects need to follow specifications using the class as a template in the time of their creation. Every object instance in a class should have its own different values for the attributes defined within the class and should react to messages through invoking the methods stated within the class. Different objects in the same class may react in different ways to the same message depending on their current state. The major aim for class diagram is to design the static side of a system. Their ability to be associated with object-oriented languages in a straight forward manner makes them unique and widely used by developers (Dietrich, 2017).

Classes and Objects are the smallest and very basic building units in object-

oriented development model. In forming a larger system structure, classes are interrelated by different sorts of relationships. Therefore, classes by themselves might not make ample insight into how generally a system is designed. UML gives different sorts of manifesting the kind of different relationships among classes representing various ways of connections and has some limitations that are not completely addressed through UML specifications. Some of the types of relationships are discussed as follows (Vrana, 2016).

Association

Associations are tougher relationships than simple dependencies and naturally specify that one class holds a connection to another class over a stretched period. When two classes are linked via associations their lifelines are most likely not tangled together i.e. one can be demolished without inevitably destroying the other. A typical example of this relationship can be the relationship between a person working for a company and similarly, a company has several offices (Booch, 2007).

Association Class

When the relationship between two classes is a complex organizational connection, it is recommended to represent this relationship using an association class which will be a class on its own head making the connection between the two. Association class usually is a class with name and list of attributes similar with regular object classes (Booch, 2007).

Aggregation

Aggregation is a stronger form of association of object composition in object-oriented design. In contrast to association, aggregation represents a strong relationship by introducing ownership among the connected classes and this in turn resulting with dependencies of lifelines. Aggregation relationships in UML diagrams are shown in a diamond shape on the connection line next to the owning class and a solid line pointing to the owned class, as shown in class diagram visual representation below (Paradigm, 2017).

Composition

Composition signifies a very strong connection between two classes to the point where one is contained in another representing a whole part relationship. In this relationship a piece of the can only be composite part of a whole at any given time which implies a strong tie between the lifetime of instances making it impossible for the part to exist if the larger owning whole is destroyed. The only way to preserve the part piece in UML is by allowing it to be owned by a different class before the destruction of the

owning whole, though this process is an exception. Since composition relationship is always read as "...is part of...", it requires to be read the composition from the part piece to the owning whole. Composition relationship in UML diagram is represented by using a solid diamond next to the owning whole class and a solid line pointing to the owned part piece class (Vrana, 2016).

Generalization/Inheritance

A generalization relationship depicts a hierarchical relationship between classes with a nature of abstraction of child classes by the super classes. It pulls out communality among the different classifiers for sub classes. Generalizations are usually read as "...is a...", starting from the more specific class and reading toward the general class. In the UML diagram representation generalization is shown by a solid line with a solid arrow pointing from the specific sub class to the general super class (Booch, 2007).

Contrary to associations, generalizations relationships are usually without names and do not have any sort of cardinality. In UML a sub class can have more than one super class. This feature allows for multiple inheritances from different super classes with each generalization class representing a different aspect. In modern object-oriented programming languages like Java, multiple inheritance feature is not supported.

Interfaces

An interface is a classifier which can be used to support abstract specification of a system in modular development with declarations of attributes and methods but not implementations. Therefore, an interface is essentially an abstraction that encompasses a list of operations, which clients of the interface will need, and which implementations of the interface must implement. Though interfaces cannot inherit, they can be endpoints of an associational relationship from a class.

Most common use of interfaces is to group common elements between classifiers and enable an implementation which an interface must obey. Interfaces are supported by modern languages like Java, though it doesn't allow them to have properties. Unlike in the case of Java, other programming languages like C++, don't support this UML feature interfaces; rather they are typically represented as pure abstract classes. The common representation of interfaces is the standard UML classifier notation with the stereotype «interface».

Dependency

This is the weakest connection among classes. Dependency between classes means that one class uses, or has knowledge of, another class. This basically be defined as a transient relationship which means a dependent class briefly interacts with the target class but naturally doesn't retain a relationship with it for any real length of time. In UML diagram a dependency between classes is represented using a dashed line with an arrow pointing from the dependent class to the class that is used (Vrana, 2016).

Multiplicity

Because associations typically represent lasting relationships, they are often used to indicate attributes of a class. Multiplicity describes the number of objects that are involved in an association. In practice there are three common kinds of multiplicity across an association, but in general there are more options. You can express how many instances from a class are associated with how many instances from another class in a relationship. If multiplicity in a relationship is not specified a multiplicity of 1 is assumed. To show a different value, simply place the multiplicity specification near the owned class (Wazlawick, 2014).

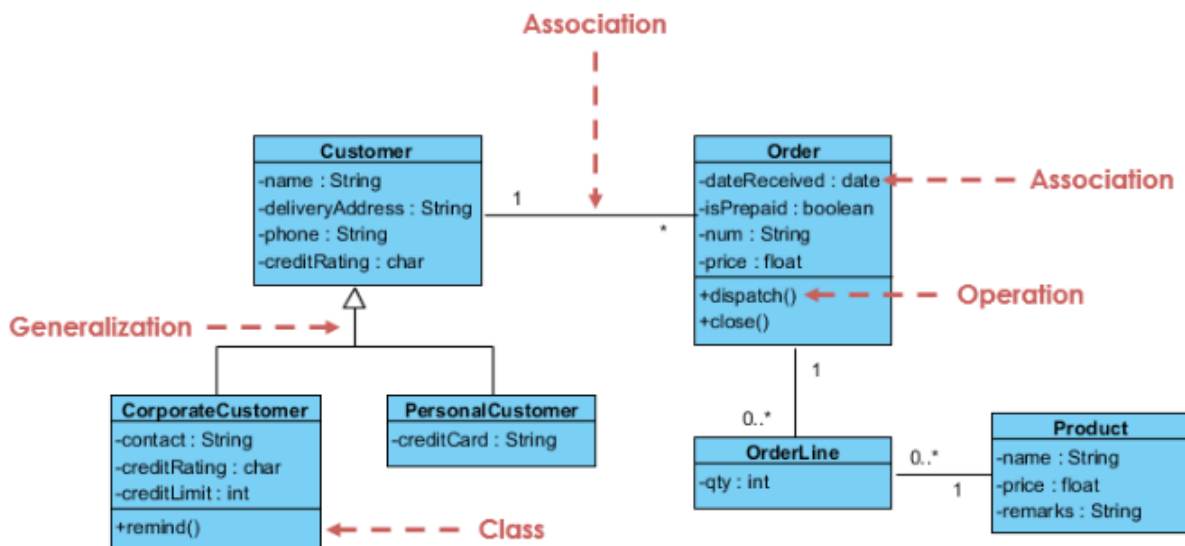


Figure 8: Relationships in Class Diagram (Paradigm, 2017)

Component Diagram

In the Unified Modeling Language, a component diagram represents how various components of a system are bound together to form bigger components or a whole software system. Basically, it demonstrates the designs of the system elements and shows the dependencies between them. These system elements compose run-time components, executable components and the source code components (Paradigm, 2017).

Deployment Diagram

The Deployment Diagram in UML aids to model the bodily aspect of an Object-Oriented information system. It is a structural diagram that displays architecture of the software as a function of distribution of software artifacts to deployment targets. Software artifacts denote tangible elements in the real world that are the result of a distribution process. It helps to design the run-time configuration in a static view and pictures the distribution of artifacts in a system application. In most cases, it involves modeling the hardware configurations together with the software components that lived on.

Object Diagram

Objects and data values are the main components of an object diagram. A static object diagram is a graphical representation of an instance of a class diagram; it shows a snapshot of the detailed state of a system at a point in time. The main difference between a class diagram and an object diagram is that a class diagram signifies an abstract model consisting of classes and their relationships in contrast, an object diagram signifies an instance at a specific moment in time, that is solid in nature. The practice of object diagrams is justly limited (Paradigm, 2017).

Package Diagram

Package diagram is one of UML structure diagrams that displays packages and relational dependence among the packages. Model diagrams allow to show different views of a system, for example, as multi-layered (aka multi-tiered) application - multi-layered application model (Paradigm, 2017).

Composite Structure Diagram

Composite Structure Diagram is one of the new artifacts added to UML 2.0. It resembles to a class diagram and component diagram slightly differing for it will often be applied for designing a software at micro point of view depicting individual parts instead

of whole classes. It is a type of static structure diagram that shows the internal structure of a class and the collaborations that this structure makes possible (Ousterhout, 2018).

This diagram can contain interior portions, ports through which the parts interrelate with each other or through which instances of the class interact with the parts and with the external world, and connectors among parts or ports. A composite structure is a set of interrelated elements that cooperate at runtime to accomplish some purpose. Each element has some defined role in the collaboration.

Profile Diagram

A profile diagram allows one to create domain and platform specific typecasts and define the associations amongst them. It is possible to create stereotypes by drawing stereotype shapes and relating them with composition or generalization through the resource-centric interface. It is also possible to define and visualize marked values of stereotypes.

3.4.2 Behavior Diagrams

Dynamic aspects of system in UML are covered and modeled by Behavior Diagram. There are two major aspects of behavioral modeling. The first one is an interaction model that is a characteristic that encompasses an exchange of messages between a set of objects in a specific situation to achieve a definite purpose. The second one is a state model containing sequence of several states that can be obtained by an object or an interaction throughout its lifetime. In state model change of states occurs as a result of different events which are triggered by different operations in a class (Wazlawick, 2014).

Behavioral Diagrams include Use Case Diagrams, State Machine diagrams, Activity Diagrams, Sequence Diagram, Communication Diagrams, Interaction overview diagrams and Timing diagrams. Basically, these diagrams focus more on different happenings in a software or in any business process by describing the functionalities among the systems and their inner workings. Some of behavior diagrams are discussed in this material as follows (Booch, 2007).

Use Case Diagrams

System requirements and functionalities in UML are captured by use case diagram which is a combination of list of actors invoking different activities, defined pieces of functionalities in a system that are called use cases and object elements which are

responsible for implementation of the use cases. An actor in use case diagram is represented with a stick figure and the entire system is represented as a rectangular box and a use case is represented by an ellipse inside the rectangular box. Use cases are representations of a separate part of functionalities of a system or a component. Use cases need to have a name which is unique and composed of a few words unfolding the essential functionality commonly represented in an oval with the name of the use case at the center of it (Booch, 2007).

The actions of a use can be explained by the interaction diagrams i.e. sequence and collaboration diagrams. Additionally, not only Activity diagrams and state diagrams but natural languages and texts are also ways of explaining the interaction.

Use case is a high-level explanation of how an information system is supposed to function aiming to capture requirements of the system, which implies that a use case signifies a user communication with various possibilities to describe user interactions (Wazlawick, 2014).

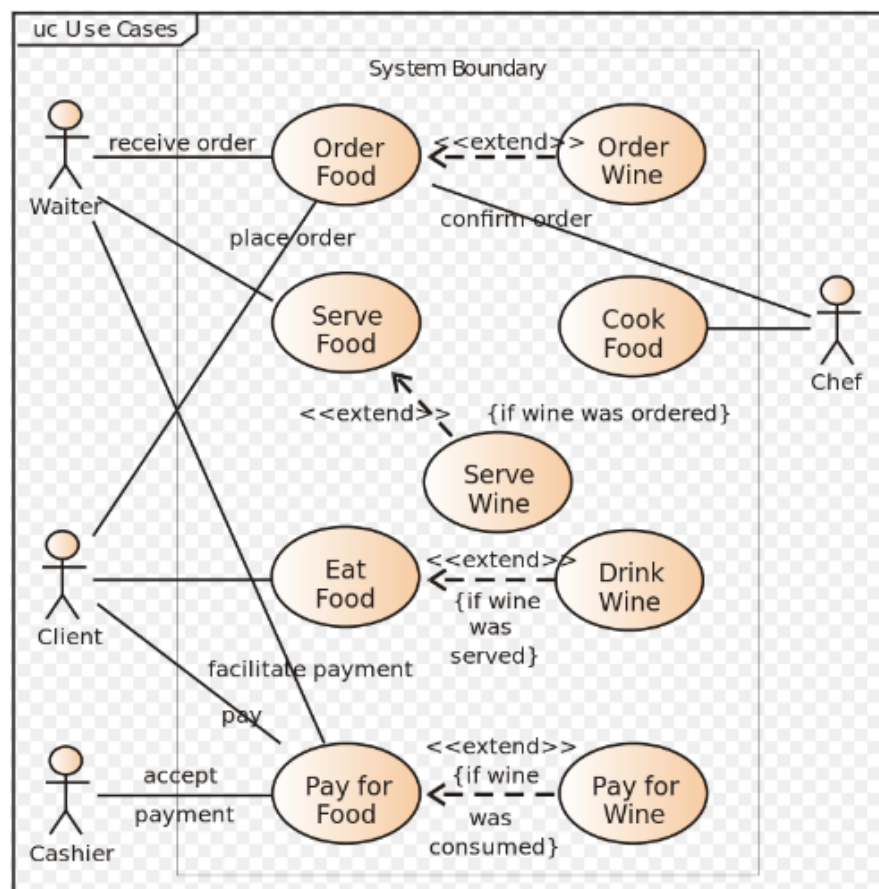


Figure 9: Use Case Diagram and its interactions (Wikipedia, 2018)

There are three different relationships in a use case diagram; Include, Extend and Generalization. An include relationship among use cases represent the ability of a use case to include and use other use cases in a clearly defined place in its explanation. In contrast, an extend connection among use cases means that the base use case is extended with additional behavior by the extending use case. A diagrammatic explanation of the use case diagram and the sort of relationships in it is given on the figure below (Paradigm, 2017). This use case on the example is with the scenario of a restaurant.

Activity Diagrams

Activity diagrams are visual explanations of flow of work in a step by step fashion and they represent activities and actions and contain support for choosing iteratively and concurrently. It shows control flow of a given system, such as the discovering intricate business rules and processes. The modeling of both computational and organizational processes is the main goal of activity diagrams in UML (Paradigm, 2017).

The ability to allow modelling of a process as plain steps consisting of a collection of nodes connected by edges makes activity diagrams often to be considered as "object-oriented flowcharts" that can be used in modeling any sort of elements for the purpose of explaining their behavior.

Usually these diagrams are easy to be understood by partakers of the development process. The similarities between the most common flow charts and activity diagrams is the main reason why it is easy to understand them there by making them a great, simple and easy to understand tool of communication. With the new UML version 2, activity diagrams obtained completely new semantics giving rise to a clear distinction between activity and state machine UML diagrams and provision of greater flexibility of modeling options (Booch, 2007). An example of activity diagram is given below.

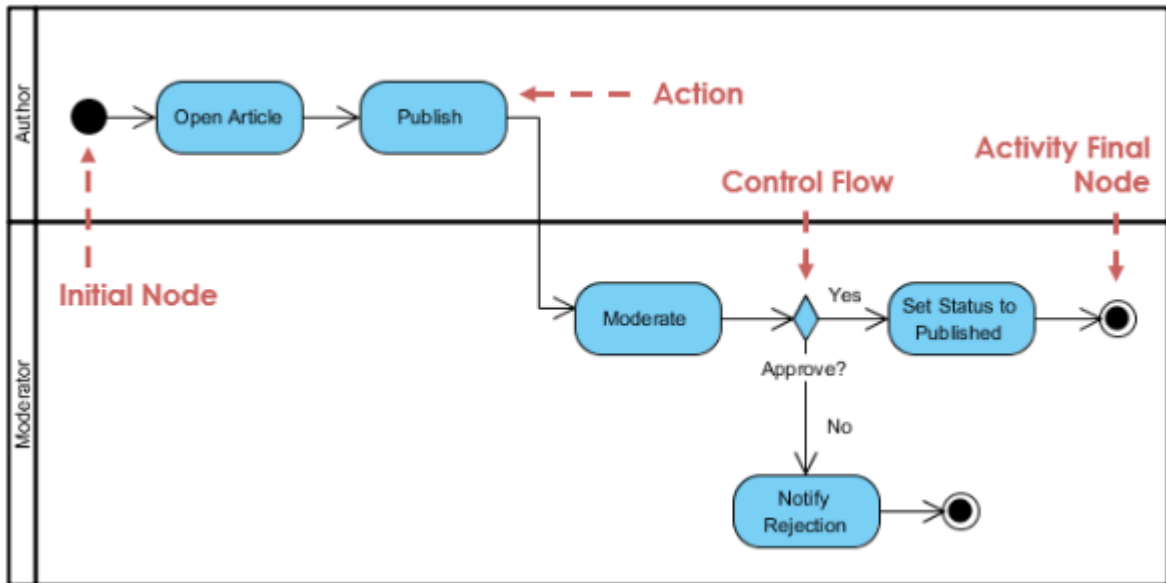


Figure 10: Activity Diagram (Paradigm, 2017)

State Machine Diagrams

State Machine diagrams contain the behavioral aspects of an information system. They are often used to model behavior of system elements at a different level of hierarchy such as at class, a subsystem, or an entire application. A state in UML is represented by using the basic rectangular notation with the name printed at the very top compartment of the rectangle. A state can only have one of the only two states at a time; either it will be active, or it will be inactive. As transactions fire, states can be entered and as soon as a state is entered, it is considered active. Similarly, a state is considered inactive right after the state is left. A transition in a state diagram means the relationship among separate states representing the definite change in the arrangement of a state as it flows from one state to the other. Sometimes transition can be protected by a guard condition that indicates if the transition should be taken in consideration. An activity that leads execution of the transition is called a trigger (Vrana, 2016). Visual representation of the State Machine diagram is as shown below.

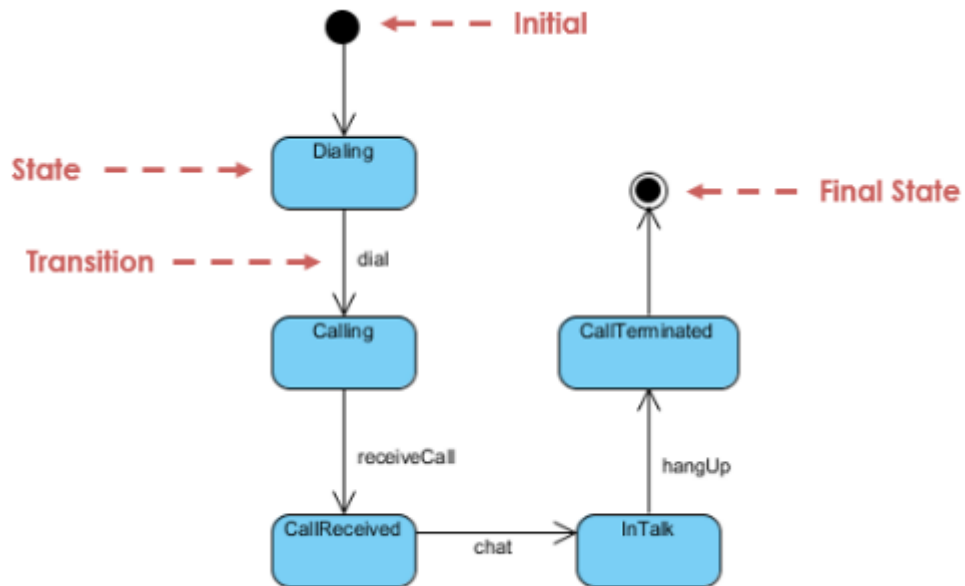


Figure 11: State Machine Diagram (Paradigm, 2017)

Sequence Diagrams

One of the Interaction Diagrams in UML is Sequence Diagram model emphasizing the interactions among objects and not on the data operations linked with those interactions based on time sequence. It expresses how the objects interrelate with others in a scenario of a given use case.

Participants in an interaction of Sequence Diagrams are represented using a solid rectangular bar called a lifeline. When they interact, their interactions are represented with a dashed line with a rectangle at the end of the line pointing the related life line. Communication among participants can take a range of several forms such as method calls, signals, new instance etc. which can generally be categorized as messages. These relationships and the whole sequence diagram are illustrated in the figure below (Vrana, 2016).

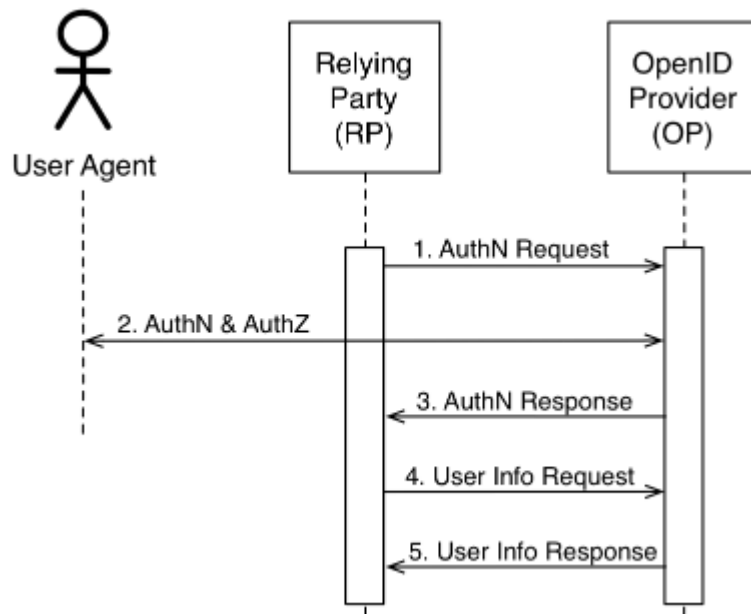


Figure 12: Sequence Diagram (Paradigm, 2017)

3.5 Databases

Generally, a database (DB) can be defined as list of information which is arranged in orderly fashion for sake of easy management of accessing and updating. Information in a database is manipulated through digital systems. Collection of files and records in a database are typically kept and sorted by rows and columns of a data table. Rows in a table are always indexed for easy identification of relevant information. As new information is added in to a raw existing data will be either deleted or updated. To this end databases need to give their users a means to access information, read or write and enable them to generate a report. Size wise, databases could range from bulky mainframe systems to minor scattered workstations. Most modern and large databases use a database management system (DBMS) to facilitate the interaction between the database and its users (Brown, 2001).

Database Management Systems (DBMS) are standardized software tools which are used in storing, organizing and manipulating records in a database. Several standard queries and algorithms are used to execute deletion, updating, creation of a record. Even though, there are several types of database management systems, mainly they can be divided in two major categories. These are, relational databases and non-relational

databases. These two major categories are discussed in this document briefly (Dietrich, et al., 2011).

3.5.1 Relational Databases

Relational databases are first founded and introduced by E.F. Codd in 1970 at International Business Machines. This is the most commonly used database management system. Relational database is a tabular arrangement and organization of data. Data tables are fitted into predefined classifications and every table needs to have at least one data classification in a column and each row needs to have a firm data record of the classifications. In relational database data are saved in a table with each row having a unique identifier called a key that enables it to connect with other related rows in other data tables (Williams, 2015).

Major behavioral properties that are relate to the characteristics of all relational databases are as follows:

- Atomic values
- Unique row
- Columns consisting of the same sort of data
- Insignificant row order
- Columns share a common name

The most common standard for application and users to interface in relational databases management system is the structured query language. Modifying records in relational database using SQL (Structured Query Language) is easy to expand since additional category can be created without requiring modification of all the existing systems. SQL is a domain specific language mainly used to handle structured data in relational database where there exists a connection among various parts of the data. Some of the major relational database management systems include Microsoft SQL Server, MySQL, IBM DB and Oracle Database (Date, et al., 2000).

3.5.2 Non-relational Databases

Non-relational database is different from relational databases in a way that they do not track the traditional database management system they are rather designed to overcome the limitations in the relational databases. They usually are referred as a NoSQL since they use a different query language other than SQL and their popularity is rising in recent years

since 2000 as usage of big data system grows. MongoDB, HBase, Ne04j and Cassandra are the major ones among the non-relational database management systems (Harrington, 2013).

The major distinction between relational and non-relational database management systems is, non-relational databases are more mountable and flexible than relational databases. They provide a data retrieval and management technique that is completely different from the traditional tabular models. For the following reasons, we can state non-relational databases are improvements to relational data model (Fortier, 1999).

First, their simplicity in design, since joining of several data tables is not required, makes them preferable over the relational databases. Second, this data models are flexible in updating and handling changes. Third, scaling horizontally is much easier. Fourth, designed to accommodate unstructured data. Fifth, better availability especially when servers are being added or removed. Sixth, most of the non-relational database models are open source. Seventh, suitable to handle big data. Eighth, less costly. Finally, better speed. These are the advantages of non-relational databases over the relational (Geoffrey, et al., 1994).

As a matter of statistics, among the many non-relational databases, MongoDB ranks out number one with number of downloads over 10,000,000 and multiple dispositions. It is recently named as a front-runner database management system covering both relational and non-relational models leading all NoSQL database goods (Fortier, 1999).

Based on different aspects, databases also can be further divided in seven different categories. Hierarchical, Network, Relational, Object-oriented, Graph, Documentational and ER model.

3.6 Web Applications as a trading platform for local farmers

From time to time our dependence on digital technologies is rising. Almost all our day to day activities are supported by some sort of information system application. Trading is not an exception in this list of human activities.

Trading is one of the early human activities since ancient times. As trading evolved through time, it paved a way for mankind to prefer trading over subsistence. This in turn lead to unrestricted movement of humans beyond geographical boundaries resulting in

advancement of human lives in general and economic advancement in specific. As a result, a profound economic productivity was brought in the scene.

In the early eras of human evolution trading in its basic form was practiced as an exchange of agricultural products and simple tools used for hunting. Even now in the modern time trading of agricultural products still considered one of the major trading items between countries.

In this time of digital modernization, trading is greatly advanced with the support of information technologies. It improved further the speed and the quality of trading by enabling an exchange of massive information between trading parties. Currently all over the world there are plenty of different platforms to support trading in local markets to take advantage of the benefits related with it. For instance, applications like Mandi Trades helps local farmers and consumers of their product in a great way (Kedia, 2018). It helps farmers to sell their agricultural products locally to the consumers without involvement of any mediators. And, it provides digital solutions for farmers in boosting their harvests in India.



Figure 13:Mandi Trade App (Kedia, 2018)

Basically, the app serves as a platform linking farmers and customers in the agrarian value chain by enabling both parties to access and manage market data and fast communication between them. This improves the bargaining power in making an informed decision for the farmers and further it arranges means of loan from financial institutions if needed (Kedia, 2018).

Similarly Rainbow Agri is an internet of farmers for it allows to have a direct communication with customer and other farmers in India. It enables farmers to sell and customers to buy locally produced agricultural produces even by providing a live change in prices. Other than individual customers local restaurants and supermarkets are also able to use this app in the facilitation of their trading. One can find and associate even with farmers in the neighborhood to see a real time pictures of their produce. This app enables the users to make a customized order, get discounts and get detailed information on the orders made (Sharma, 2015).

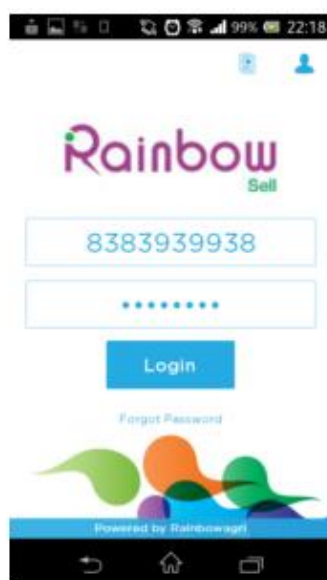


Figure 14: Rainbow app (Sharma, 2015)

Similarly, Agro-Hub is an application designed to help farmers trade in Cameroon. The application was developed by a business vendor in cooperation with Cameroonian telecom services to facilitate the growth of local farmer's income by trading their produces for good prices both in local and international markets. In achieving these goals agro – hub established a collaborative effort of the communities through SMS mobile messages and the internet in broadcasting and managing any information in relation to existing agricultural situations.

Therefore, we can state that different web applications and mobile applications are used to promote trading and to ensure the benefits of local farmers in the developing

economies are secured. These applications are used in a more agricultural intensive and growing economies than in advanced developed economies.

3.7 Web Applications as a trading platform for local farmers in Ethiopia

Background

Ethiopia is a nation located in the horn of Africa. It is a land locked country with a total area covering about 1.1 million kilometer-square. The economy is one of the fastest growing economies in Africa and its mainly agriculture based. This sector is the largest contributor to the countries annual GDP followed by the service sector and manufacturing sector. Even though the developments in spreading the services sector is encouraging there is still a lot of work to do regarding this. The only internet and other telecom service in the country is Ethio Telecom. This monopolistic approach led the country's telecom service quality to downgrade and rank as one of the worse in the world.

Recent changes in the political situations of the country forced Ethio Telecom to resolve some issues in relation to the availability and affordability of internet and telecom services though profound problem of monopolistic approach remains unchanged. These changes enable the low-income classes of the society to be able to afford buying some of the services by opening a window of opportunity to introduce new information technology-based application.

Current Situation

For the past few years not only the prevalence of telecom services over the country was very limited and low quality but it was extremely expensive to the point were more than half of working force couldn't afford to have 3G internet. These negatively affected by hindering the growth of information technology-based application there by depriving the society from benefiting from them. But as the recent political transformation takes place through social revolutions, the prices are significantly lowered since the past few months.

Considering the history of telecom services in the whole country as a general and the capital Addis Ababa specifically there were no web applications to support farmers to trade their agricultural produces. Farmers living in the suburbs of Addis Ababa, use the traditional market to sell their produces which is not cost effective, fast and modern.

4 Practical Part

4.1 The Web Application

This web application is designed to help local farmers living in the suburbs of Addis Ababa, Ethiopia. The application facilitates smooth transaction between the customers and the farmers without involving a middle man. This benefit both the farmers and the customers. Since these farmers are local and small scale, accessing the market for commercial purposes is limited. Accessibility of produces of these farmers is one of the major roles of the application. Therefore, for the local farmers it will be a way to access the market without needing to take so many bureaucratic processes on their side. Similarly, the benefit for customers also great. First off, it will be a good place to get a fair price to buy agricultural products, not only that but it will be a source of organic food which is more health compared to mass produced commercial products.

The application allows the farmers to post their produces with the amount quantity they can supply. They can also post different prices for different quantity demanded on the customers side for a certain product. The customers mostly are comprised of individual residents of Addis Ababa and Restaurants in the city. Customers will have an option to search for a specific product that they want. They must put the name of the product that they are looking for and the amount they want before they hit the search button.

Typically following the search, the customers will be provided by a list of the best prices they can get based on the quantity they demand. Based on these search results the customer will decide and place the order. Before they complete placing the order, they will be prompted by the system to choose if their preferred delivery mechanism. Either they can choose to use delivery services or a personal pick up. If they choose the delivery services, the price will be included in the order to be placed.

As soon as the customer placed the order it will be a pending order. Confirmation of the order will take place, after the full settlement of payment for the order within 30 min from the time the order was made. Payments can be settled online or in person. Otherwise, the pending reservation will expire. As soon as the payment is settled the order will be marked as confirmed by the system. A notification for the relevant supplier (farmer) and delivery service provider will be sent following the confirmation of the order. The farmer can view the details of the order and prepare the order.

For this to take place smoothly in the system, different parts of the system should work harmoniously. There will be a systems integration with the bank system, delivery service providers and the application. Many customers are expected to use this application since the residents of Addis Ababa are 3 million. The application uses a real-time data of the orders and updates of posts of products.

For these and other related reasons the application needs to be a web-based application and the design process should be made carefully. UML one of the best tools to design any information systems that are web base. Therefore, it is also used in the design of this web application to cover all the vital parts of the system design.

This application is a web-based application intended and developed to assist farmers and customers to exchange efficiently and effectively, which involves several farmers and customers living in and around Addis Ababa. To this end, this web application will have several different mechanisms and functionality to support the on-going activities. These functionalities include online registration system, customer management system, payment system, delivery management, employee management, reporting and analytics and other extra custom features.

Based on the level of importance they have for the overall system some of the key components of the system are discussed in this document as follows.

Online Registration system: This allows farmers and their potential customers to register online using internet. Both users can register and have an account.

Customer Management system: is a method to manage the interaction of farmers with current and potential customers. It uses data analysis of customers' history with the suppliers to progress business relationships with customers, giving a more emphasis on customers satisfaction.

Payment System: This is designed to handle all issues related with the billing of customers. Refund requests and online payments related with services and goods.

Employee Management: is an information system that supports employees involved in running this web application smoothly.

Reporting and Analytics: Reporting is the act of shaping a raw data into a meaningful information. It helps to see how the different aspects of the application are performing. On the other hand, analytics is the procedure of extracting meaningful insights that are useful in understanding and improving the business to a better level.

4.2 Over view of the Application

The anticipated application will be able to display all necessary information for both customers and farmers. It will be designed with high robustness to be available and accessed online at any time, anywhere.

4.2.1 Functional Requirements

Functional requirement specifies what a specific result would be. It also states definition for the functions that the whole system will have in general and particularly of components which constitute it. Functional requirement describes the communication among the application system and the users regardless of its execution. The following are the main functional requirements of the web application:

- Record both customers and farmers information,
- Register restaurant requirement information,
- Execute online payment according to the payment method,
- Display available products with their full information. For instances, their prices, availability and so on,
- Downloadable reports in pdf format,
- A search functionality,
- The application should provide navigation to a different category of the system,
- The application should be able to authenticate different users and their assigned privileges,
- The application should give the ability for all users for them to be able to change password,
- The application should have a function to show the possible buyers for the farmers,
- The application should enable for the customers to find the minimum price in the market for their desired products,
- The application should allow the farmers to add new products with their specified details
- The application should display information on upcoming market events
- The application should display the administration page menu according to the

assigned user's right and so on.

4.2.2 Non – Functional Requirements

Non-functional requirements of the system are requirements which enable one to judge the quality of operation of the system. Unlike functional requirements it doesn't focus on specific behaviors of the system. They are often referred as quality attributes of a system application.

The following are some of the non-functional requirements of the system from both the front end and the back end of the system:

- The application on the front it should allow users who only have a username and password,
- The system should be able to work in a networked environment,
- The application in the back end should be able to validate data entries,
- The application should respond to different web requests in a rational period which refers that the system should respond at least before session expiration,
- The application should provide only valid result. It shouldn't have to crash in case no data is found, instead it should be able to handle this kind of situations,

4.3 Analysis Model

In this section of the document, the necessary system design for the application is discussed with different UML diagrams to the best visualization. Class diagram is used to show the static behavior of the system and interactions between entities of the system are described using the dynamic models of unified modeling language.

4.3.1 Data Dictionary

- **Customer:** is a person or a restaurant who desires to buy a product from the local farmers in the market.
- **System Amin:** is an employee who performs clerical work related to orders of customers.
- **Order:** is the placed requirement of the customers in the system. And it consists of the chosen product of their interest and the means of delivery.
- **Farmer:** is a person who supplies the agricultural products and enters them in the

system with their specified prices and quantity.

- **Product:** is a product posted in the system by the suppliers. Product should include at least two different prices, one for individual customers which will be stated as a retail price and one for restaurants as customers which will be stated as a business price. Business prices should be a discounted price than the retail prices.
- **Restaurant:** is a customer who will be interested to buy a product and offers a maximum price affordable for the quantity demanded.
- **Pending Order:** a temporary placement of an order.
- **Confirmed Order:** is a confirmed order. As soon as the customers makes a payment on a pending order it will change to a confirmed order.
- **Delivery:** involved in the system to give the service of delivery.
- **Receipt:** is a confirmation of payment. Receipt can be either an electronic one which can be downloaded from the system or a paper one.
- **Refund Request:** is a request for the payment already made on an order. It can be made by customers if the product wasn't as it was promised on the app or for any reason if the customer doesn't like it.

4.3.2 Static Model

Class Diagram

Figure 15 below reveals the over-all arrangement of putting all the elements i.e. objects and classes and shows the relationship among them in a class diagram. In each class we can see the objects with their related attributes and operation/methods. All the relationships between the classes their links, association and generalizations were recognized before designing the diagram. The class diagram is design taking these in to consideration. As shown in Figure 15 the classes and their relationships are designed by understanding the project description of the web application. We can see here classes of order, person, customer, restaurant, product, delivery, system admin and so on. The connections among the classes and their respective multiplicities are also identified in the diagrams.

As we can see on the below figure 15 the class Person generalizes three classes i.e. class Farmer, class Customer and class System Admin. Class person abstracts the common attributes from these child classes. These classes inherit the attributes and the

operations from their super class Person. Therefore, these three child classes have a generalization relationship with their super class Person. Even though, these three classes inherit their common attributes from the super class, they still have their own unique operation which makes them different from one another. It is also evident that child classes have their own unique attributed other than their inheritance, in this case we don't see a unique attribute for all child classes.

In this generalization relationship, even though, the child classes inherit the operations from their super class, the way they implement it could be different on their own respect because of polymorphism. This lets the same operation on a given time to be executed in various ways resulting different activities in the child classes. Likewise, we see a generalization relationship between class Order, super class and class Pending Order and class Confirmed Order. All the behaviors and characteristics discussed for the previous generalization are also applicable to this generalization.

The other relationship we will see on figure 15 below is a relationship of association. Association relationship is characterized by multiple instance of relationships between the associated classes. In this class diagram class Restaurant and class Person have association relationship. Their cardinalities are stated on the figure. A person can own non or multiple restaurants at a given time. Therefore, the multiplicity to the association from class Person to class Restaurant is zero to many. Likewise, a restaurant also can be owned by different persons at a given time. Therefore, the multiplicity from class Restaurant to class Person is one to many which is slightly different from the previous multiplicity. This is because a restaurant needs to have at least one owner at a given time.

Similarly, in Figure 15 we can see that there is an association relationship between classes farmer and product, classes system admin and order, classes customer and order, and classes customer and refund request.

The other type of relationship we see the class diagram on Figure 15 is a relationship of aggregation. We can see these relationships between class Order and class Delivery. When the customers make order, they can include a delivery service or a personal pickup. If the customer chose to use the delivery service, this service will be the integral part of their order. It depicts a part whole relationship. Therefore, in this aggregation relationship class order is the whole and class delivery in the part. In aggregation relationship it is not a must that the part is included in the whole. Similarly, we see the same type of relationship between class restaurant and class customer.

The special type of aggregation is called a composition relationship. Composition relationships are more stronger relationships than aggregation relationships because they represent ownership from the whole to the part. Destroying the whole effectively leads to destroying the part. On figure 15 these relationships can be seen between class order and class product. An order must have a product in it. It is not optional like in aggregation relationship. Therefore, destroying class order will effectively destroy the product in it. The same kind of relationships can be observed between classes confirmed order and receipt and classes refund request and confirmed order.

As we can see in the figure, both aggregation relationship and composition relationship can have different cardinalities.

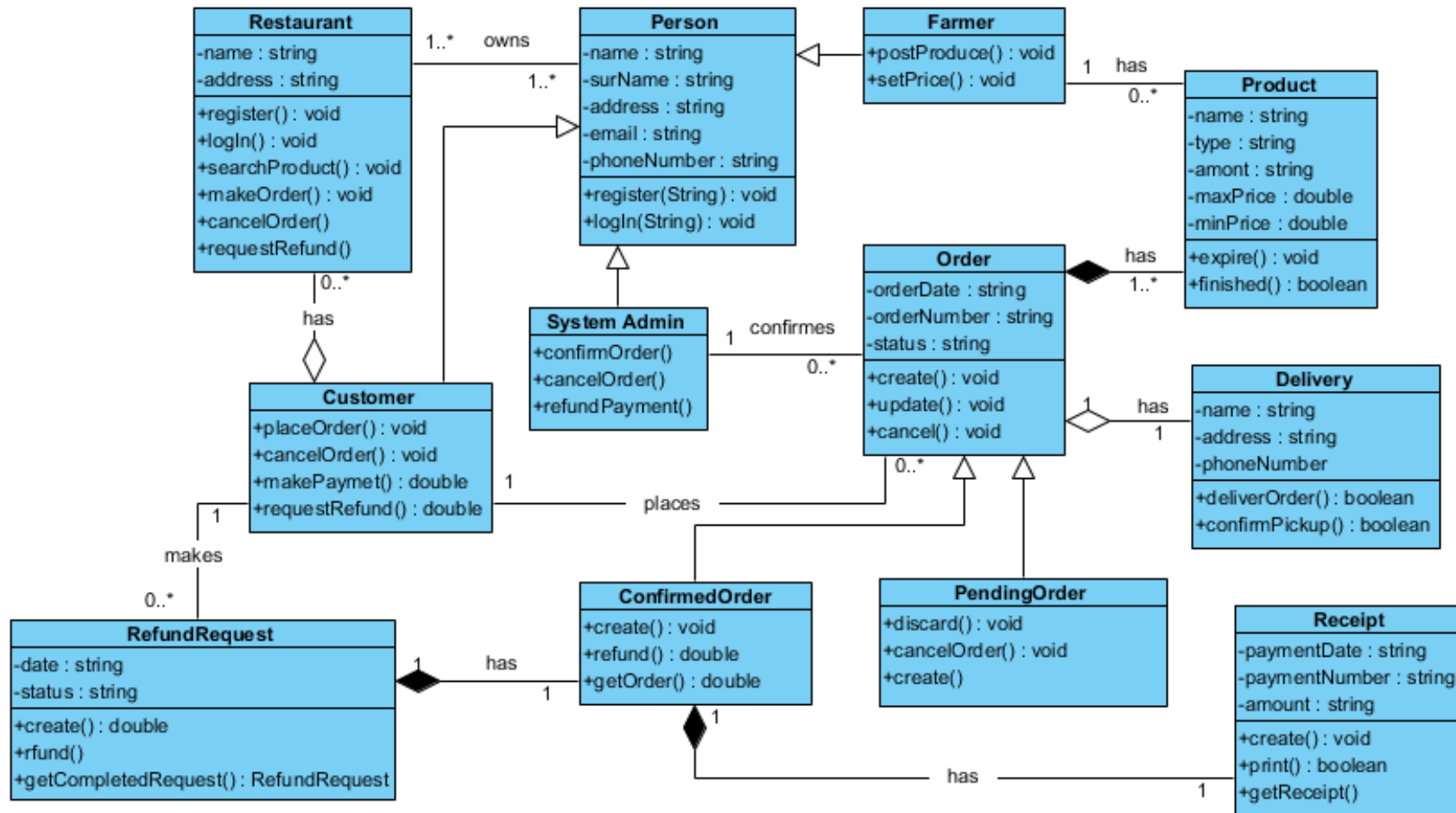


Figure 15: Class Diagram of the application (Author, 2019)

4.3.3 Dynamic Model

Use Case Diagram

As we can see in the figure 16 below the use case login is used by all the actors in the system. Customers, system admin and farmers can register to the web application system gateway and make their desired inquiries. The login use case includes the register use case and extends help in case any of the users demand support in the area.

Farmers as users of the system can add a product to it. They will be required to fill in the fields in the user interface which gives them access to fill their data on the database table. One product to be added to the system it at least requires fulfilling four necessary fields. The first one, is the name of the product. Second, the minimum price that the farmer can offer to individual customers as a retail price. The third one is the minimum price that the farmer can offer to restaurants as a business price and the fourth is the amount of the product that the farmer has in his/her stock.

After registered and log in the customer will have access to place order as it is shown in the use case on figure 16. Placing order starts with a process of searching for desired product. Then the system will give them a list of minimum price offers from the farmers. When they see the desired product, they can add it to the cart and then select their delivery service if they need and proceed with the payment. As soon as the complete the payment they will get a confirmation. The use case place order includes use case make payment and use case search and use case pending order are extension points to it.

System admin the other actor in the system as shown on figure 16. The system admin check until payment is made on the pending orders and as soon as the payment is made, he/she will confirm the order and the system sends a message to the delivery and the farmer to prepare the order and it also confirms the customer. Similarly, the system admin is also the actor for the use case reports. He/she can download a report from a system regarding every aspects of the system.

All the users in the system will have a different type of accounts. During the registration, they will be prompted by the system to choose the type of the account they will use and access to activities will be provided based on that. Especially, individual customers and restaurants will be treated differently when it comes to price of the products.

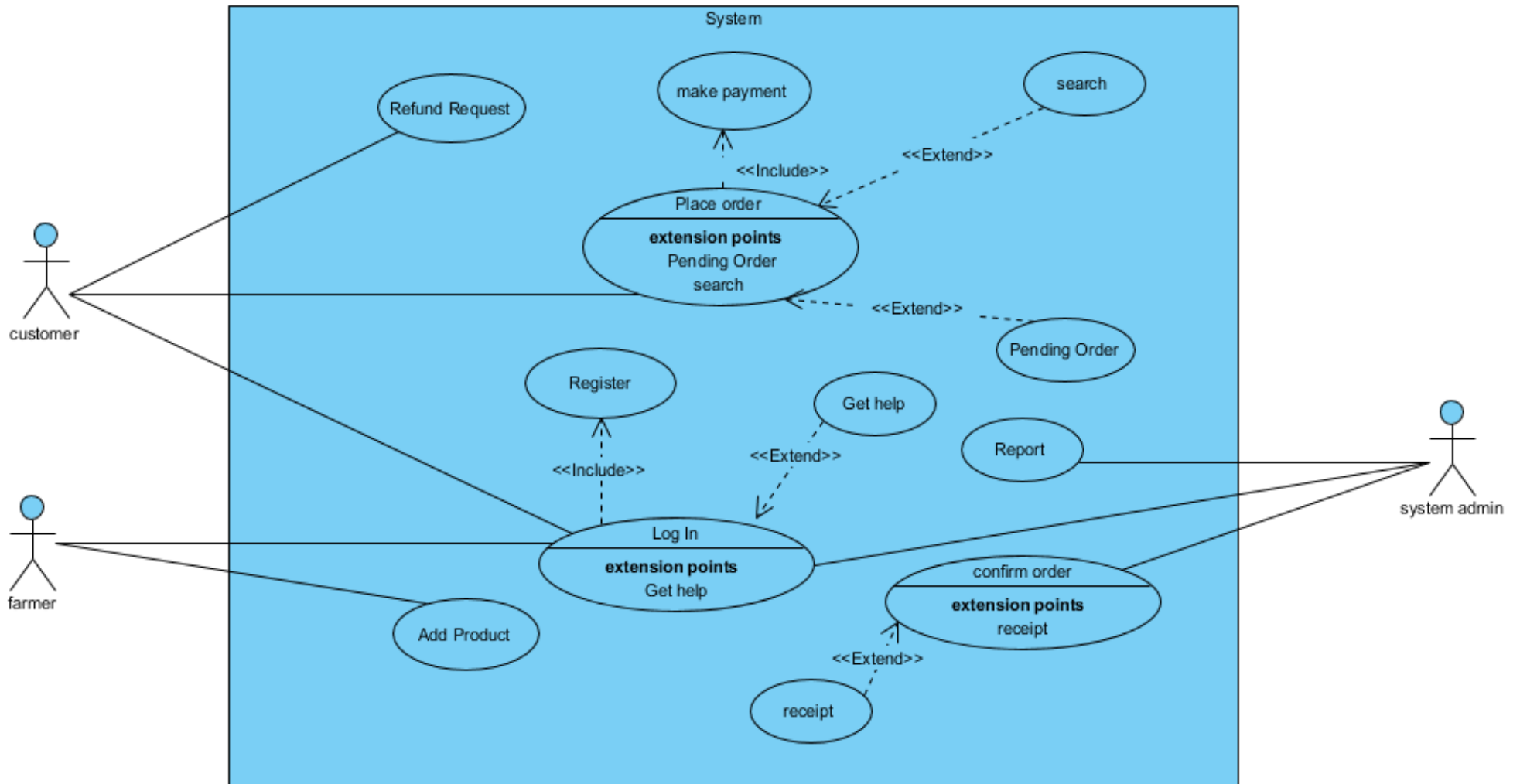


Figure 16: Use Case Diagram (Author, 2019)

Sequence Diagram

A sequence diagram is related with the above use case objects on figure 16. It mainly depicts the communication and interaction among involving objects in a single use case. Even though, sequence diagrams must be represented by a graphical form, a descriptive textual support is used to explain the model. This textual explanation is called a scenario. Below selected sequence diagrams of the system are discussed.

Sequence Diagram of Place Order

Figure 17 shows a sequence diagram for Place Order.

Scenario: Customer makes the order.

The customer opens the website of the system to make order. As the customer searches and places the order, the system checks the availability of the requested goods and services and the quantity demanded. Following the result, the customer decides the best option that suits him.

Sequence Diagram of Report

Figure 18 shows a sequence diagram for Report.

Scenario: System admin will draw general report.

System admin logs in to the web application using her/his credentials. The system admin goes through the system to identify among pending orders which are already paid. He/she will discard the expired pending reservations. And finally download the report and email it to the management.

Sequence Diagram of Report

Figure 19 shows a sequence diagram for Registration.

Scenario: User is on the web application web page.

As the user clicks on register option the system should prompt him/her to fill a registration form. After the completion of the registration form, the system will create an account for the user base on the input data. The user will access the other pages in the website depending on the kind of user he/she is. There will also be an option for the user to register in the application through social medias.

Sequence Diagram of Place Order

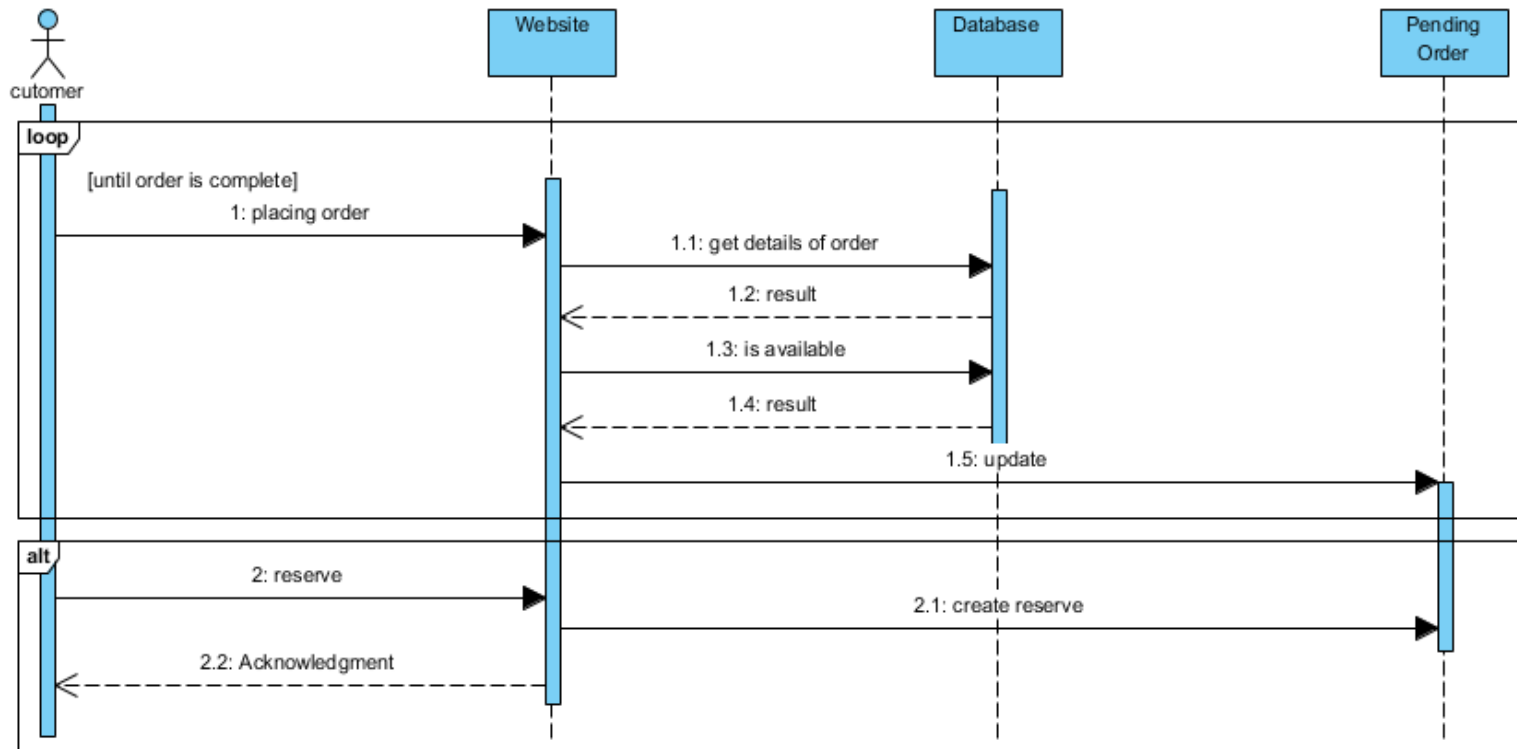


Figure 17: Sequence Diagram of Place Order (Author, 2019)

Sequence Diagram of Report

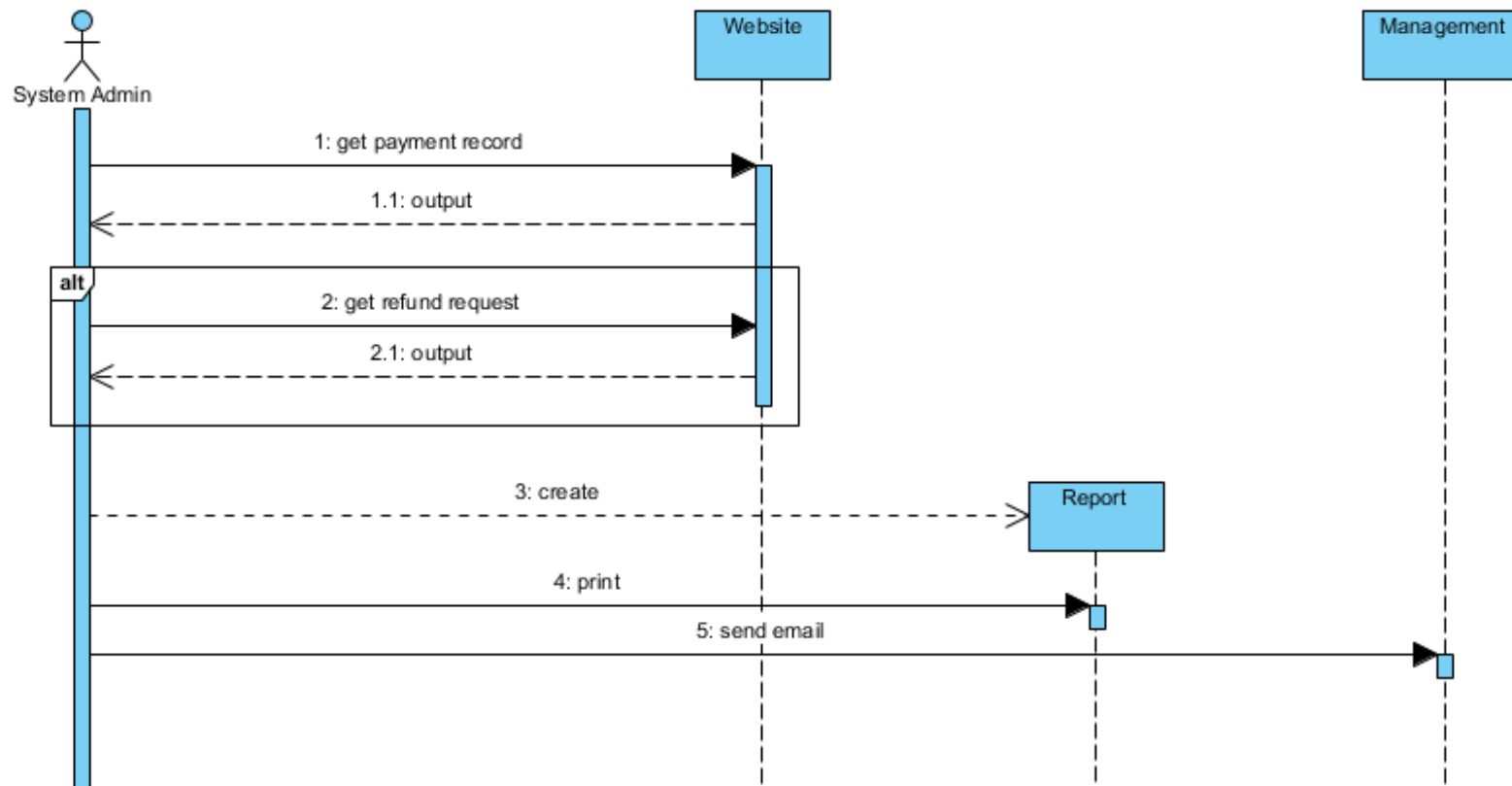


Figure 18: Sequence Diagram of Report (Author, 2019)

Sequence Diagram of Registration

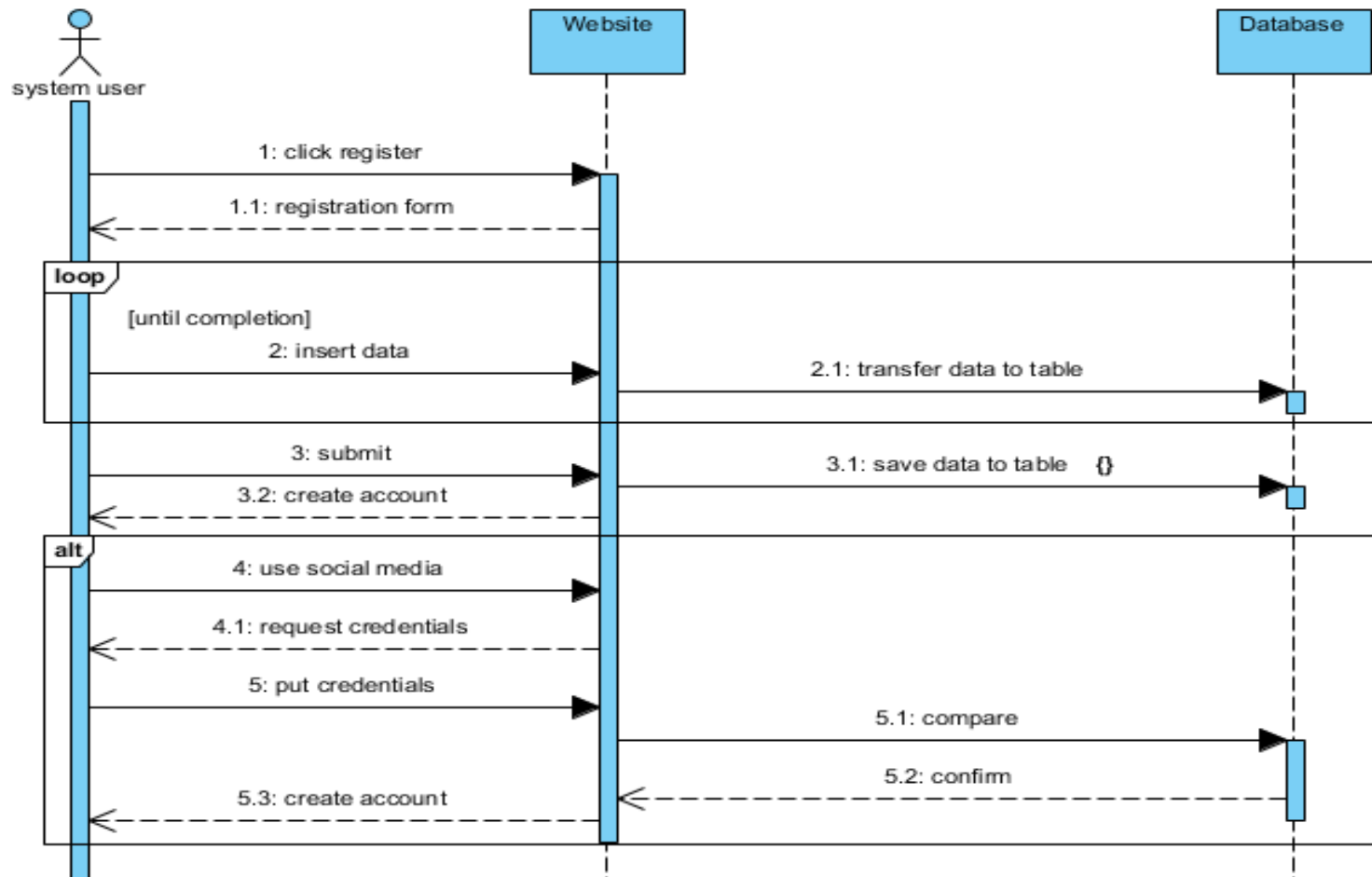


Figure 19: Sequence Diagram of Registration (Author, 2019)

Activity Diagram

Activity Diagram of Make Order

Figure 19 shows an activity diagram for making order. The customer can make a pending order and make payment later. The order will be reserved for 12 hours before it expires. The payment must be made within these 12 hours.

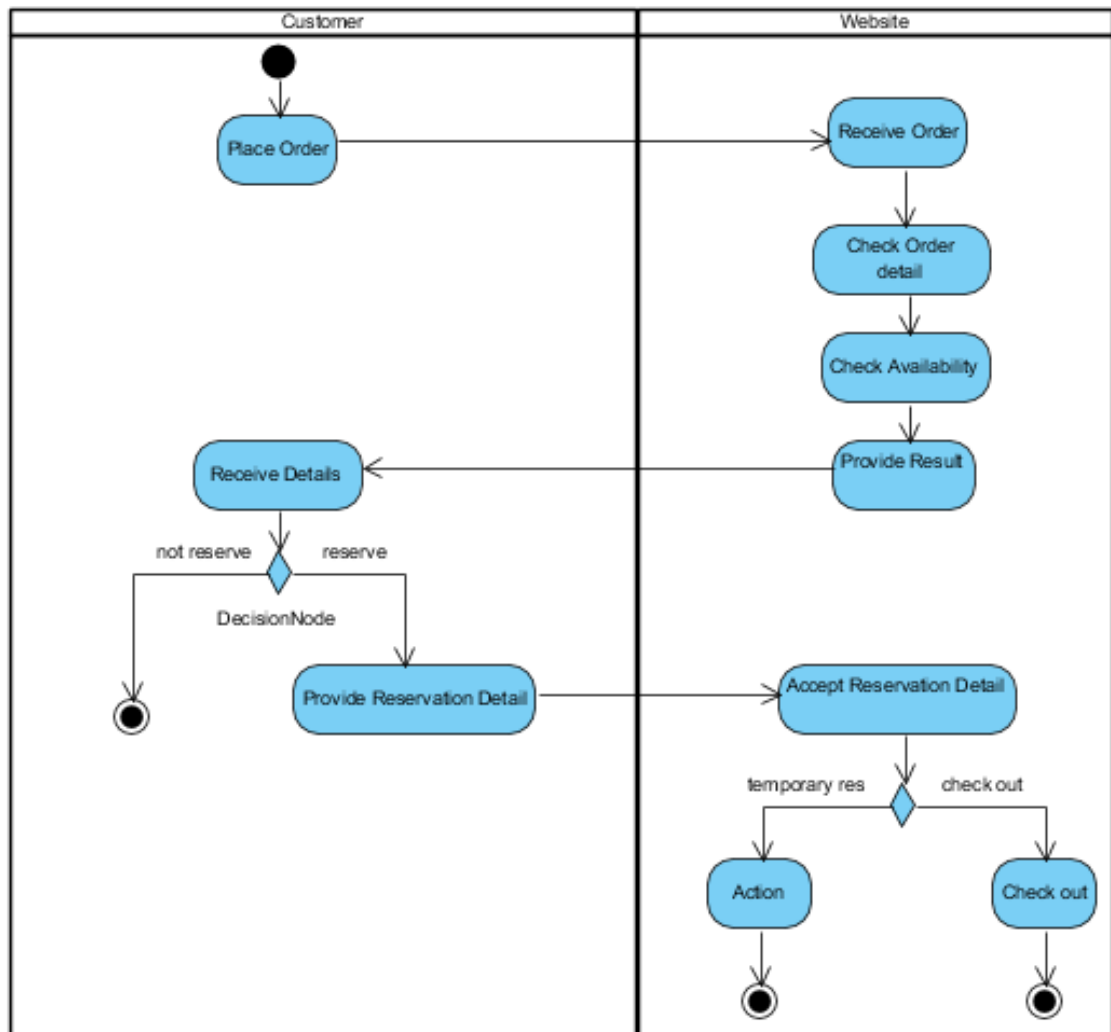


Figure 20: Activity Diagram of Order (Author, 2019)

Activity Diagram of Report

As it can be seen on Figure below, Activity diagram for report is shown. The system admin collects the necessary information specifically focused on payments and refund requests. The system admin has a responsibility of collecting this information and summarizing it. The final action will be reporting the summarized report to the management team.

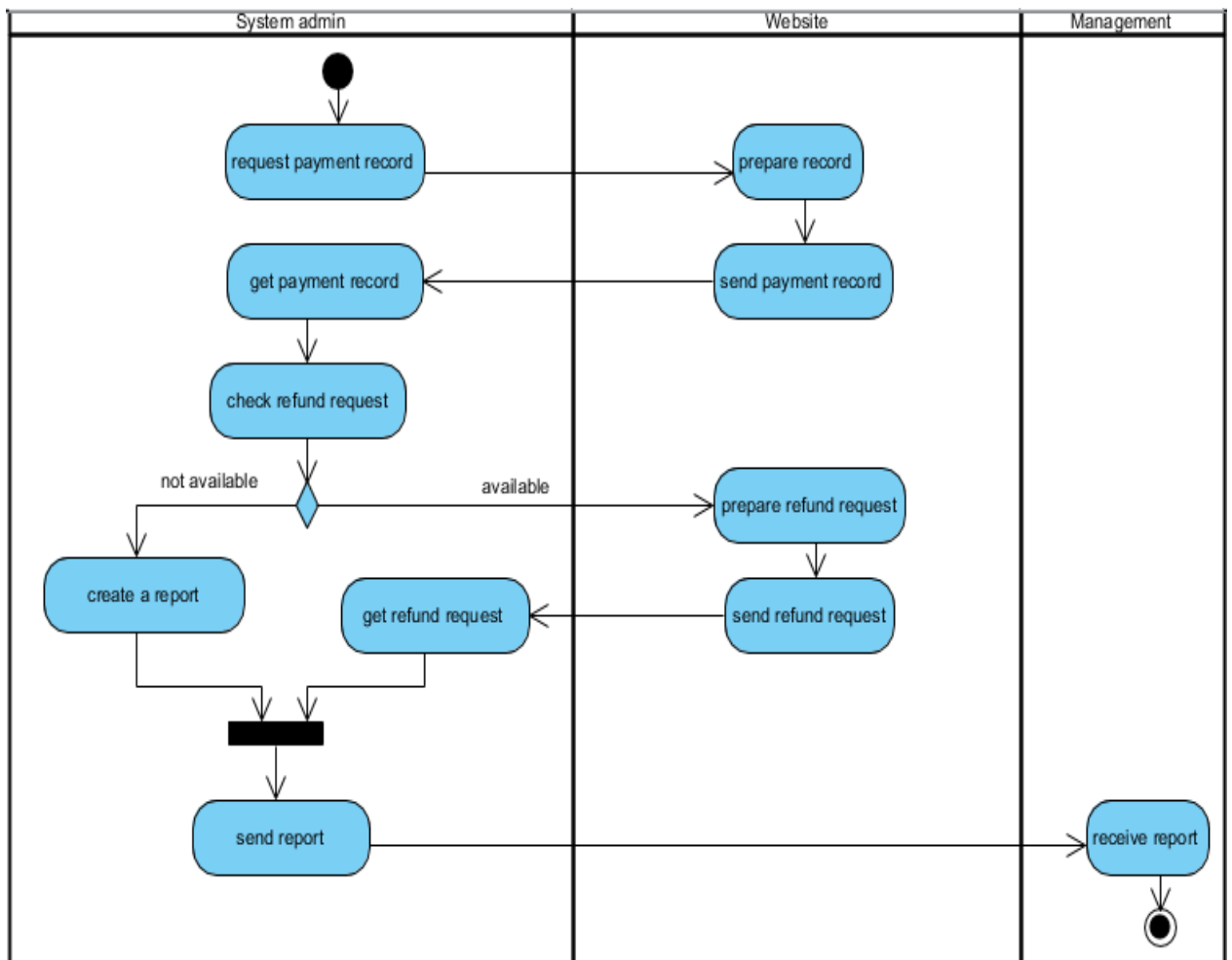


Figure 21:Activity Diagram of Report (Author, 2019)

4.3.4 State Model

State Diagram

State machine diagram describes the behavior of a system with a graphical representation. The foundation to state machine diagram is different state a system can be in at a given time of activities and interactions. It depicts and shows the states and their respective transitions and events which trigger an action to fire. In this document state diagram for making order is explained on figure22 below.

State Diagram of Make Order

As we can see on figure 22, the state starts with the customer logged in to the application and is on the main menu. Then they system will display the necessary information and the customer searches and adds the desired products and services in to the cart. An order in a cart can be canceled, temporarily reserved or can be checked out and paid. As we can see with every activity taking place in the application the state for order will change.

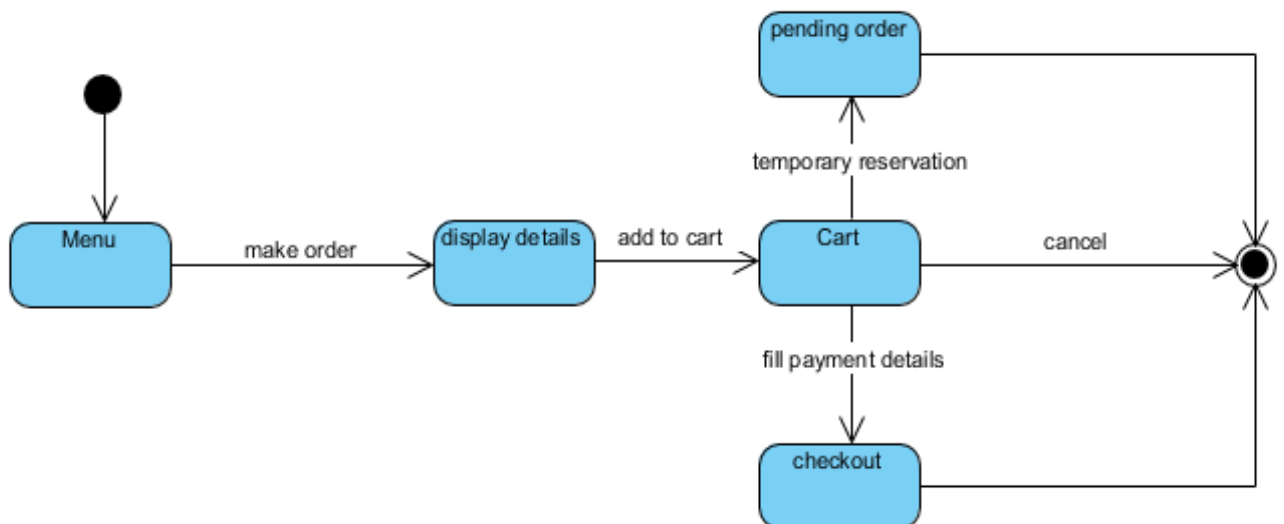


Figure 22: State Diagram of Make Order (Author, 2019)

4.4 Architectural Design

In the process of designing a web application one must strictly follow standard architectural designing procedures. These procedures are followed in the development of several software development and are widely accepted by the system development community. The main parts of these criterion are discussed as follows:

Performance:

The first criterion is the criteria of fast response. It must perform fast in a high throughput. This means a customer should get a response from the system with 1 minute when he/she places an order. Retrieving details on order should also be delivered quickly.

Reliability:

This refers to a backup plan that a system should have in order to tackle failures. Failures in any systems, could happened and therefore there should be a way to cover these failures with a least down time possible. It must also be able to recover any kind of losses in data at any given time.

Availability:

The web application must be available 24/7 all throughout the year. To make the uptime the highest and to avoid down times. A host with high percentage of SLA must be used for the web application. This makes the recovery time quicker with a very small down time when undesired things happened with the system.

Maintenance:

In order to maximize the performance and functionality of the web application, it should be in a state of easy maintainability. There will be a periodical procedure to check up and maintain the application.

Security:

Multiple security procedures must be in place to secure the system. Starting from its physical safety to an air tight protection of the data. Users will be managed to have access based on their needs and security issues in the application only be reserved for authorized system admins. There should also be ability to detect suspicious activities like several attempted login failures. When these activities detected, the IP address must be black listed.

End User:

A responsive easily understandable graphical user face is required. For instance,

fields in any form in the system should be self-explanatory. Button and menus should have a simple and descriptive name.

Implementation of the proposed system architecture will have a great level of availability. It will be structured in a structure. To this end, the database servers will be in clusters and continuously up and running. This will also help the system by giving more capacity and performance to handle when unexpected odds happen in the system. Similarly, it allows performing systematic and steady maintenance of the infrastructure. The architecture is expected to be robust with a minimum downtime and unavailability.

4.4.1 Systems Disintegration

The proposed architecture consists of five major subsystems. Specializations of performed tasks in the system are the basis for the categorization of the sub systems. The disintegration of the system in to different sub systems will guarantee a high level of coherence. High level of data exchange between all subsystems is required to ensure the smooth running of the whole web application system. Here all the sub systems i.e. search, resources, customers, employee, and security are discussed as follows.

Search

Its main concern is dealing with the global search functionality of the application. In addition, it deals with new information taken from the users through the system to improve the interactivity of the web application. It is designed to enable the system users to search for the product of their interest which are provided by the web application for local agricultural products. The farmers as users also should be able to search for a better price.

Resources

Resources are collections of several types of agricultural products with their related business prices, retail prices and their quantities which are available for the customers to choose from. Customers will be able to select a product and the quantity that they demand.

Customers

All data records in relation to customers interactions with the web application and their activities are dealt with this subsystem. It basically improves the services delivered to the customers by opening a way to understand the customer's behavior. Customers can be categorized in to two different account types which are business and individual. The

services that they get will be different base on the type of the account a user will have.

Employee

All activities and interaction of employees with the web application system are contained in this subsystem. Its main intention is to monitor the activities of employees and differentiate between the active and the passive ones. Additionally, things in relation to vacancies and new positions are also dealt with this.

Security

It needs to be designed with the high level of structural design since security issues are critical in the performance and existence of an organization. Security is a major component of many business architectures and needs to be considered seriously. Any failure with security could cause major problem that threaten the existence of a business. Therefore, it must be air tight and robust.

5 Implementation of the web application system

Implementation can take two major ways. Either it will be creating an improvement for existing system or introducing a completely new system. In this project an attempt is made to create a new system. Creating a new system from scratch requires to establish and introduce the new features of the application for employees. Employees must be trained to the point where they are able to operate the system on a surface level and deeper level. All possible ways of malfunctioning of the system should be identified and countering mechanisms and troubleshooting techniques must be communicated with them likewise. There should be a prepared manual to guide the users through this troubleshooting steps. The implementation will be partial creating prototypes.

A prototype web-based application for local agricultural products is implemented using Java, PHP, JS, HTML and CSS. UML modeling language is used in the designing ideas and concepts of the web application system which enables different stakeholders to have the same view of the system in the development process. It smooths the communication and fast implementation among coders, designers and testers of the system. It can easily also be implemented on various software development paradigms.

As UML is a language to specify the web application precisely, unambiguously and completely, it is used to design the system. Though it is not used in this implementation, using some traditional programming languages like PHP, it is possible to generate some code from UML directly. This document mainly focuses of discussing the process of how the farmers add their products in the system and get list of their potential buyers.

The development process of web application for agricultural products is dependent on the source codes and a tangible user interface. The user interface helps the users to easily use the functionalities of the system and be able to read information with the right font and colors. User interfaces for registration process, log in and add product functions of the system are discussed in this document.

Some screen shots of the implementation code are shown below. The implementation is based on Java and PHP on the back-end server side and html and CSS and java scrip are used to create the user interface on the front end and to make it responsive.

Figure 23 below shows definition of customer class. The attributes are defined. A constructor is made to provide a template for the class and the appropriate getters are also

provided to make sure that the fields are not accessible. As we can see all the attributes are private to access them, we need to through the getters. Directly.

```
public class Customer {

    private String name;
    private String surName;
    private String address;
    private String phoneNumber;
    private String email;

    public Customer(String name, String surName, String address, String phoneNumber, String email){

        this.name = name;
        this.surName = surName;
        this.address = address;
        this.phoneNumber = phoneNumber;
        this.email = email;

    }

    public String getName() {
        |   return name;
    }
    public String getSurName() {
        |   return surName;
    }
    public String getAddress() {
        |   return address;
    }
    public String getPhoneNumber() {
        |   return phoneNumber;
    }
    public String getEmail() {
        |   return email;
    }
}
```

Figure 23: Class Customer Code in Java (Author, 2019)

Customer class has several methods but two of them are explained two of the methods; register and login are discussed below on figure 24, 25 and 26 as follows. Figure 24 shows the PHP code and the user interface code to the method of registration. The values from the fields of the user interface are collected and inserted in to the database table. Database connection needs to be established there by creating a business logic to pass the values into the tables.

```
<h5>Register</h5><br />
<div class="row">
  <div class="col">
    <input type="text" class="form-control" placeholder="First name">
  </div>
  <div class="col">
    <input type="text" class="form-control" placeholder="Last name">
  </div>
</div>
<br />
<div class="form-group">
  <input type="string" class="form-control" id="address" aria-describedby="emailHelp" placeholder="Address">
</div>
<div class="form-group">
  <input type="string" class="form-control" id="phone-number" aria-describedby="emailHelp" placeholder="Phone">
</div>
<div class="form-group">
  <select class="custom-select" id="inputGroupSelect01">
    <option selected>Choose account type...</option>
    <option value="1">Supplier</option>
    <option value="2">Business</option>
    <option value="3">Individual</option>
  </select>
```

Figure 24: PHP Code for Registration (Author, 2019)

Figure 25 and 26 similarly describe the codes used for login. From the website text fields, the necessary data i.e. username and passwords are passed to the php code running on the back end. Using the established connection matching between the entered user name and password will be made with the user name and password stored in the database. If successful user will be directed to home page and will have all the additional access of the system which are only reserved for users with accounts. If the attempt is unsuccessful the fields will be cleared, and the user is prompted to try again.

```

<?php
if (isset($_POST["btn_login"])){
    $user_email = $_POST["user_email"];
    $user_password = $_POST["user_password"];
    // $user_id = $_POST["user_id"];

    if($user_email == "doyealeazar@gmail.com" && $user_password == "1234"){
        $_SESSION['is_logged_in'] = true;
        // $_SESSION['user_id'] = $user_id;
        header("location: addProduct.php");
    }
    else {
        // code...
        echo "<div class='text-center'><h5> Your user name or password did not match </h5></div>";
    }
}
}
?>

```

Figure 25: PHP Code for login (Author, 2019)

```

<div role="main" class="container about-txt">
  <div class="row">
    <div class="col-sm-6">
      <div class="card social-login">
        <h5>Login with social site</h5><br />
        <a href="#"></a><br />
        <a href="#"></a>
      </div>
    </div>
    <div class="col-sm-6">
      <div class="card">
        <form method="post">
          <div class="form-group">
            <label for="exampleInputEmail1">Email address</label>
            <input type="email" class="form-control" name="user_email" id="user_email" aria-describedby="emailHelp" />
          </div>
          <div class="form-group">
            <label for="exampleInputPassword1">Password</label>

```

Figure 26: Continued PHP code for login (Author, 2019)

The web application for local agricultural products has different navigation menus which are linked with related functionalities and services of the web site. The major menus i.e Home, About, Events, Add Product and login are discussed as follows:

Home:

The home page is first page users see when they open the web site. No matter on what page they exit the system users when they login from different locations in the city the system takes them to this home page. The home page and an introductory well come page for all. It has a global search engine which only appears on the page when the user is logged in. The users can type in the text field to search for anything that they want in the application and they will be provided with a list of results.

About:

The about page is a page in the web application system which provides the necessary information about the company running this website. It states the background history of the company briefly. It also covers the company's visions and mission. The organizational structure and other similar issues like location are raised in this page.

Events:

The Events page gives information about an upcoming event if any. Events are exhibitions and special holiday or weekend markets in which both customers and suppliers can be participating to their respective benefits. This events also will be a good way to introduce the web application to the general society. Mostly small-scale farm products will be exhibited by the farmers. Additionally, handmade traditional house hold products are also can be exhibited and traded.

Add Product:

This is a page where farmers will be able to add products and detail information about them including their This page can only be accessed by a logged in account user. And the account type must be a supplier account type. As soon as the supplier enters the details of his/her product, he/she will get a list of potential buyers with their respective addresses.

Register:

Register is a page on which customer will put their details to create an account which appropriates their interest. All types of users are required to have registered and their accounts chosen and detail on their accounts must be saved to the database. Unless users are not able to use the different functionalities of the web application and even some menus like add product and search will only appear after they logged in.

Login:

This is a page which allows all users of the system to log in. The system administrators will have a full right to access and have control over the system. Access rights if needed can be granted by this user and reports also can be printed out by them. All users are required to put their saved user name and password to log in to the system and it is a must to do this before the use the system.

Some of the prototype pages for the web application are shown in the following consecutive pictures.

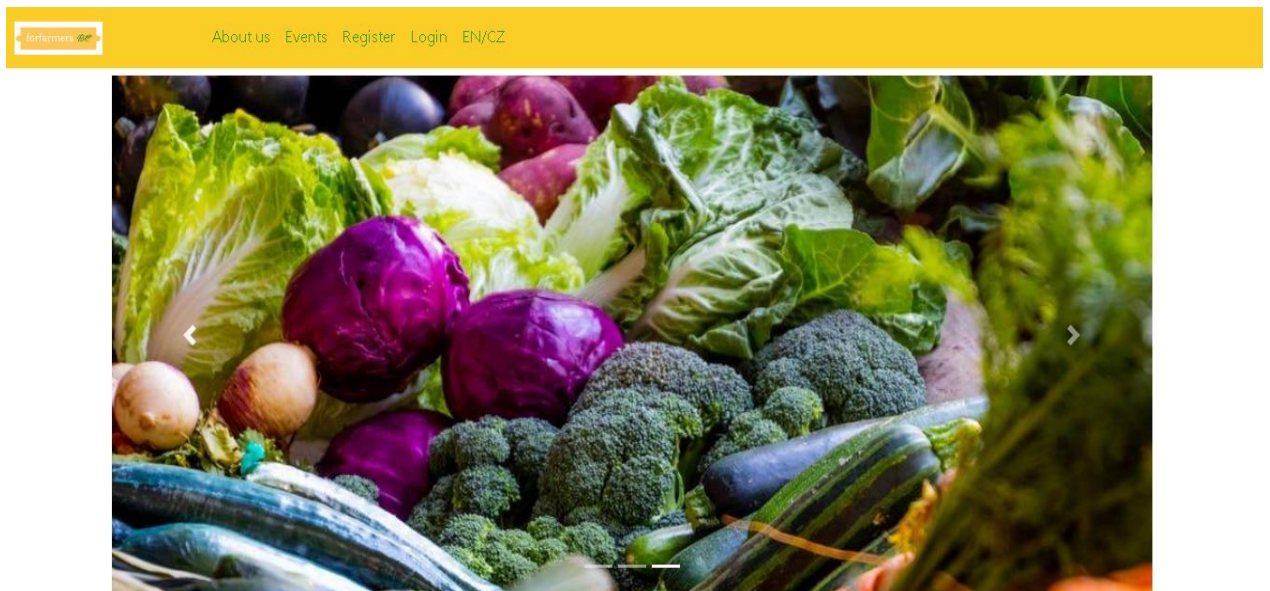




Figure 27: Home Page (Author, 2019)

Register with social site


 Sign up with Facebook


 Sign up with Google

Register

Figure 28: Register Page (Author, 2019)

Login with social site

 Login with Facebook

 Login with Google

Email address

Password

[Forget your password?](#)

[Contact us](#)

[Try Our Mobile Apps](#)

Figure 29:Log In Page (Author, 2019)

Add Product

Potential Buyers

Restaurant	Product	Qty	Offers	Phone
------------	---------	-----	--------	-------

Contact us



Try Our Mobile Apps

Figure 30: Add Product page (Author, 2019)

Please fill all the values

Add Product

Product name Product Quantity

Business price Retail price

Potential Buyers

Restaurant	Product	Qty	Offers	Phone
200 Habesha	Lettuce	29	10	+251912131415
Kategna	Lettuce	25	9	+251911553272
Youd Abbysinia	Lettuce	50	9	+251911554342
Lewi	Lettuce	40	8	+251916863165
Tsege Shiro	Lettuce	60	8	+251910897656

Contact us



Try Our Mobile Apps

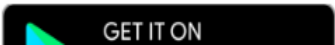


Figure 31: List of Potential Buyers after product added (Author, 2019)

Upcoming Events

Market Events



- Hayahulet Mazori Open Market (2019-12-12)
- Abuware Open Market (2019-12-20)

University Events



- Addis Ababa University (2019-12-13)
- Unity University College (2018-12-16)

Figure 32: List of Up Coming Events (Author, 2019)

In this part of the document all the necessary partial implementations of the web application for local agricultural products are discussed and illustrated with the screen shots of the prototype web site. MySQL database is used to save all the related data of the server of the application. Using UML in the system design made it easier to implement i.e. executing the necessary codes and building the user interface. In the following section of the document result of the project will be discussed.

6 Results and Discussion

6.1 Results

On the third chapter of this thesis document a literature review is carried out. The unified modelling language, software development life cycle and types of databases are discussed in detail. Similarly, an examination is made to find if there exist similar application elsewhere in the world and to assess related benefits of using them. As the findings show similar application are being used in India and Cameroon. These applications highly contributed in improving the effective trading of agricultural products, increasing the bargaining power of farmers and creating an opportunity for a better price.

Furthermore, a background study on Addis Ababa region in Ethiopia is carried out to examine if there is a conducive environment for the application to be implemented. Based on the research conducted the current political, social and technological revival in the country opens a window of opportunity to the successful implementation. On the other hand, our findings show that there is no similar application in use in the area currently.

These two major topics discussed above can be taken as a premise to argue that the web application for local agricultural products has a great potential to change the traditional trading ways in the area there by improving effective information exchange between the farmers and their potential customers. This has a greater implication on the improvement of both the suppliers of these products and the consumers.

Following this background study, the web application for local agricultural products design is carried out in unified modelling language. Both static and dynamic elements of the system are addressed using appropriate static and dynamic UML diagrams. Similarly, architectural design of the system is also carried out. Both designs are discussed in detail on the fourth chapter of this document.

Since, there was a time and resource limitation on the researcher's side, it wasn't possible to carry out the full implementation of the web application for local agricultural products. Therefore, a partial implementation and prototyping is made according to the system design, architecture, and functional and non-functional requirements. To this effect a prototype user interface with basic chosen functionalities of the system are implemented. Details of the partial implantation and prototyping is discussed on the fifth chapter of this document.

6.2 Conclusion

The purpose of this research includes designing a prototype of web application for local agricultural products in Addis Ababa, Ethiopia and to describe the benefits of the application in the day to day activities of the farmers living in the suburbs of Addis Ababa and their potential customers as well as to understand if the designed application can serve as a trading platform enhancing the effective transaction of agricultural goods. The second chapter of this document addresses in detail the objectives of this research and the methodologies carried out to obtain the desired outcomes of this process.

In the third chapter of this document a literature review is made on three main topics i.e. the software development process, the unified modelling language and database structures and types. These topics are discussed in relation to the web application for local agricultural products. In addition, a research is made on similar applications in India and the effects of the applications are also examined. Furthermore, a background study of the country in relation to this technology and the status quo is examined to look if similar applications are existing in the generally in the country and in Addis Ababa region specifically.

In the practical part of this research the web application system implementation is carried out. The system designing is carried out using the unified modelling language. Class diagram is used to describe the main objects of the system and their relations. Furthermore, UML dynamic diagrams are used to show the interactions of the objects in the system. Following this a prototype of the web application is carried out. Some of major functionalities of the system and a user interface is built.

Based on the results of the study, it can be argued that the web application for local agricultural products has a potential to be successful in improving the agricultural market of Addis Ababa region there by improving the availability of local organic agricultural products for the society residing in this region. Since this cannot be a concluding statement, our recommendation is to make a further research to make a more complete conclusion and develop the web application as a strong trading platform for the farmers.

7 References

- Booch, Grady. 2007.** *Object-Oriented Analysis and Design with Applications*, MA: . [ed.] Addison Wesley. 3rd ed. Boston : s.n., 2007.
- Braude, Eric J. 2014.** *Software Design*. Los Angeles : J. Wiley, 2004, 2014.
- Brown, Paul. 2001.** *Object-relational Database Development*. s.l. : Prentice Hall, 2001.
- Date, C. J. and Darwen, Hugh. 2000.** *Foundation for Future Database Systems*. s.l. : Addison-Wesley, 2000.
- Detienne, Françoise. 2012.** *Software Design - Cognitive Aspect*. [ed.] Frank Bott. s.l. : Springer Science & Business Media, 2012.
- Dietrich, Erik. 2017.** *Developer Hegemony*. 3rd. s.l. : LeanPub, 2017.
- Dietrich, Suzanne W. and Urban, Susan D. . 2011 .** *Fundamentals of Object Databases*. s.l. : Morgan and Claypool Publishers, 2011 .
- Fortier, Paul J. 1999.** *SQL-3: Implementing the Object-relational Database*. Michigan : McGraw-Hill enterprise, 1999.
- Geoffrey, Roderic and Cattell, Galton . 1994.** *Object Data Management: Object-Oriented and Extended Relational Database Systems*. Michigan : Addison-Wesley Publishing Company, 1994.
- Ghahrai, Amir. 2018.**
- Harrington, Jan L. 2013.** *Relational Database Design Clearly Explained*. [ed.] Second. 2013.
- Influence of software development agility on software firms.* **FarzanaSadia. 2019.** 2019.
- Information and Software Technology.* **SarahBeecham. 2019.** 2019.
- KarinaCurcio. 2019.** *Computer Standards & Interfaces*. s.l. : RodolfoSantana, 2019.

Kedia, Shruti. 2018. This mobile-based IoT platform aims to double farmers' income by 2022. *This mobile-based IoT platform aims to double farmers' income by 2022.* [Online] March 2018. <https://yourstory.com/2018/04/mandi-trades>.

2019. MobiDev. *MobiDev.* [Online] Mobi Dev, 2019. https://mobidev.biz/blog/3_types_of_web_application_architecture.

Ousterhout, John K. 2018. *A Philosophy of Software Design.* s.l. : John K Ousterhout, 2018, 2018.

Paradigm, Visual. 2017. Visual Paradigm. *What is Unified Modeling Language.* [Online] 2017. <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/>.

Reasoning about UML/OCL class diagrams using constraint logic programming and formula. **Pérez, Beatriz and Porres, Ivan. 2019.** s.l. : Science Direct, 2019.

Sami, Mohamed. 2017.

—. **2018.**

Session persistence for dynamic web applications in Named Data Networking.

Junliang Chen. 2019. January 2019, Journal of Network and Computer Applications.

Sharma, Radhika. 2015. 3 Agri Apps That Are Helping Farmers Buy/Sell Produce Effectively. *3 Agri Apps That Are Helping Farmers Buy/Sell Produce Effectively.* [Online] June 2015. <http://www.networkedindia.com/2015/06/03/3-agri-apps-that-are-helping-farmers-buysell-produce-effectively/>.

The Journal of Systems and Software. **Abdullah, Muhammad. 2019.** Doha : Science Direct, 2019, The Journal of Systems and Software.

Tornhill, Adam. 2018. *Software Design X-Rays.* 1st. s.l. : Andy Hunt, 2018.

Vrana. 2016. *Projecting of Information Systems with UML.* Prague : Czechia, 2016.

Wazlawick, R. S. 2014. *Object-Oriented Analysis and Design for Information Systems: Modeling with UML.* Boston, : Elsevier, 2014.

Wikipedia. 2018.

Williams, Robert. 2015. *Upgrading Relational Databases with Objects*. [ed.] 2nd. New York : Printes hall, 2015.

Yaskevich, Anastasia. 2018. DZone. *Types of Web Applications*. [Online] 2018.
<https://dzone.com/articles/types-of-web-applications-from-a-static-web-page-t>.

2016. YeePLY. *yeePLY*. [Online] April 2016. <https://en.yeePLY.com/blog/6-different-kinds-web-app-development/>.