ADDIS ABABA UNIVERSITY COLLEGE OF NATURAL SCIENCE

DEPARTMENT OF INFORMATION SCIENCE

Industrial project 1

WEB BASED COMMODITY EXCHANGE SYSTEM

Group members

1	Kirubel Asnakew	ugr/3194/12
2	Abdulmenan mohammed	ugr/2627/12
3	Anteneh Amare	ugr/1571/12
4	Biruk Amare	ugr/6400/12
5	Haileamlak waleligne	ugr/1406/12
6	Samuel Tesfaye	ugr/4893/12

Acronyms

HTML ……. Hyper Text Markup Language

CSS ……Cascading Style Sheets

AJAX ……. Asynchronous JavaScript

XML ……. Extensible Markup Language

PHP……... Hypertext Pre-processor

XAMPP……..Cross Platform Apache MariaDB PHP Perl

UML……Unified Modeling Language

WBS……. Work breakdown structure

CX ……Commodity Exchange

Contents

1. Introduction

1.1 Overview

Ethiopia has a large domestic market with a total population of about 110 million people (2020). After an intense period of the COVID-19 pandemic and a prolonged and devastating civil conflict, the Ethiopian macro economy is finding itself on a difficult war-time footing, facing immense humanitarian and security costs, as well as a heavy toll in terms of lost social and physical infrastructure.

Over the 15 years until 2019, Ethiopia's economy had been amongst the fastest growing in the world at an average of 9.5 % per year. Among other factors, growth was led by capital accumulation, in particular, through public infrastructure investments. Ethiopia's real gross domestic product (GDP) growth slowed down to 6.3% in FY 2020/21 compared to the previous year's historic growth due to COVID-19, with growth in industry and services easing to single digits. However, agriculture was not significantly affected by the pandemic according to the World Bank because around 80.0-85.0% of Ethiopians are engaged in agriculture. It is true that agriculture is the backbone of our country's economy, but even though we do not import agricultural products from abroad, the cost of these products has increased like they are imported.

Ethiopia has been characterized by high costs and high risks of transacting, forcing much of Ethiopia into global isolation. With only one-third of output reaching the market, commodity buyers and sellers tended to trade only with those they knew, to avoid the risk of being cheated or default.

Trade is done based on visual inspection because there was no assurance of product quality or quantity, this droves up market costs, leading to high consumer prices. For their part, small-scale farmers, who produce 95 percent of Ethiopia's output, came to market with little information and are at the mercy of merchants in the nearest and only market they know, unable to negotiate better prices or reduce their market risk.

We need to provide a modern, efficient, transparent, and reliable market platform and warehousing service through the adaptation of technology, excellence in innovation, and with integrity.

Our system /web-based Commodity Exchange will be a marketplace, where buyers and farmers (sellers) come together to trade, assured of quality, delivery, and payment. Our system has a mission to connect all buyers and farmers (sellers) in an efficient, reliable, and transparent market by harnessing innovation and technology, and based on continuous learning, fairness, and commitment to excellence.

We try to make the Market efficient by operating a trading system where buyers and farmers (sellers) can coordinate seamlessly based on standardized contracts and make the market transparent by disseminating market information in real time to all market players.

1.2 Statement of the problem

Currently, everything in Ethiopia is becoming more and more expensive, especially agricultural products which almost everyone relies on. Below we have classified the problems which the current market is facing into different sections.

- Performance
    - ✓ The current market situation takes time and energy
    - ✓ There is product wastage because of the loss of a comfortable warehouse and appropriate marketplace to sell products.
    - ✓ Cultural operation
- Information
    - ✓ Farmers do not have any information about where to sell their products
    - ✓ Buyers also have no information about where to buy the commodity they need
    - ✓ Both the farmer and the buyer have no information about the current price of the goods.
- Economy
- Cost of the products are much more expensive because of
    - ✓ Free market economy
    - ✓ Brokers
- Control/Security
    - ✓ Loss of product from the marketplace
    - ✓ Products are kept hidden
    - ✓ Can't keep track of the products
- Efficiency

✓ In times of need, finding a product can be challenging

- Service
    - ✓ It is difficult to get a product of preferred quality.
    - ✓ Getting the whole transaction done takes a lot of time.
    - ✓ Backward market operation

1.3 Objective of the Project

1.3.1 General Objective

The main objective of doing this project is to develop a web-based application that coordinates better, links faster, and protects the interests of both sides of the trade.

1.3.2 Specific Objective

- Make transactions simple, convenient and accessible.
- To provide farmers with user friendly interfaces that make it easy for them to sell their products so that
- Provide both traders and farmers with a simple and secure payment system.
- To provide farmers with a delivery system that will allow them to ship their products easily to consumers.
- Try to reduce the number of brokers.

1.4 Feasibility study

A feasibility study is simply an assessment of the practicality of a proposed project plan or method. This is done by analyzing technical, economic, legal, operational and time feasibility factors.

1.4.1 Operational feasibility

Operational feasibility is the ability to utilize, support and perform the necessary tasks of a system or program. It includes everyone who creates, operates or uses the system. To be operationally feasible, the system must fulfill a need required by the business. We believe that our system is operationally feasible because we are going to:

- Design a system that is easy to use after an initial training session

- Develop it per user requirements or needs
- Design it with capability to provide the end users with timely, accurate and reliable information.
- Develop it with ability to provide adequate through put and response time.

1.4.2 Economic Feasibility

Economic feasibility is also referred to as cost/benefit analysis. While developing this project we are going to use free software and use our own computers, in addition to this we are going to build our web-based application by ourselves along with the help of our advisor, so our project is economically feasible.

Cost: refers to tangible and intangible cost that is incurred to develop the system.

Tangible cost: Is the cost that is directly measured in birr.

Estimated cost for the project is listed in the table below.

Number	Items	Expected cost	Cost type
1	Advertisement	100000	recurring
2	Development	0	-------
3	Transport	500	One time
4	Deployment	2000	recurring
5	Total	102500	

Table 1.1 cost feasibility

Intangible Cost: is the cost that is not directly measured in birr.

- Loss of time and energy

Benefits

Benefits are paybacks from a business venture. This system will have tangible and intangible benefits.

Tangible benefits

Tangible benefit is an asset that produces income consistent with its fair market value, which is listed by the wed developer or organization in a quantifiable form. A benefit that we get from using this web-based application can easily measure in the following manner.

- Cost reduction and avoidance.

Intangible benefits

Intangible benefit also called soft benefits, are the gains attributable of the improvement project that are not reportable formal accounting purposes. In this case, we can see the benefits of using this web-based application that is not easy to measure in terms of money.

- Reduce the response time of activities.

1.4.3 Technical Feasibility

We have taken different courses and worked on different mini projects that allow us to do this project. We understood system analysis, design and implementation and now we are capable of writing software programs using different programming languages. Having such a team and knowledge, we believe developing the system is technically feasible.

1.4.4 Schedule Feasibility

The development process of this project is dynamic but we can complete all the sections by the time frame allocated in the project schedule in section [1.8].

1.5 Significance of the system

- Web-Based Commodity Exchange tries to benefit and modernize the way Ethiopia trades its most valuable assets, its commodities. Ethiopia needed a change from the traditional means of trading to better support the needs of all those involved in trading and production.

- CX creates trust and transparency through aggressive market data dissemination to all market actors, through clearly defined rules of trading, warehousing, payments and delivery, business conduct, and an internal dispute settlement mechanism.

- CX provides market integrity at three important levels:
  - ✓ The integrity of the product itself,
  - ✓ The integrity of the transaction, and
  - ✓ The integrity of the market actors

- Make a bank transaction and people who aren't using banks become bank customers.
- Reduce current market inflation by trying to reduce prices.
- Accessible: Purchases can be made anywhere.
- Convenient: There is no need to carry cash.
- Reliable: Due to the fact that the payment is made by a bank or tele birr, you can rest assured.

1.6 The beneficiary of the project

- Government: Inflation, cash flow, and product flow can all be easily controlled by the government.
- Farmers: will be able to find out where to sell their products without having to deal with any intermediaries
- Customers /consumers can get preferred quality when needed
- Bank: Due to the fact that the transaction is made by the bank, people who do not use the bank become customers of the bank.
- Ethio-telecom: As a result of the Tele Birr, Ethiopian telecom will benefit like banks.
- Delivery service providers.

1.7 Methodology

1.7.1 Data Collection

Systems analysis is the part of the systems development life cycle in which you determine how a current information system in an organization function. Then you assess what users would like to see in a new system.

The two parts to analysis are determining requirements and structuring. System analysts use different requirement elicitation techniques such as interviews, questionnaires, user observation, workshops, brainstorming, introspection etc.

After tuning it to our particular requirements, we decided to use at least two or all in combination for identifying the requirements of the new system from the above requirements elicitation mechanisms. These are interviewing, introspection, and document analysis.

Introspection

Introspection is the first and the most obvious method for trying to understand what properties a system should have in order to succeed. In introspection technique, requirement analyst “imagines” what kind of system is required for doing the required job or by using available equipment etc.

Reasons to use introspection

- Introspection is an easier technique to apply.
- There are no costs for implementing this technique
- It can act as a good initial step to start requirements elicitation

Document analysis

We will analyze different documents if we think it is necessary and can teach us how we should develop our system.

Reasons to use document analysis

- Document analysis is an efficient and effective way of gathering data because documents are manageable and practical resources.
- Obtaining and analyzing documents is often far more cost efficient and time efficient.
- Documents can be read and reviewed multiple times and remain unchanged.

Interview

Interviewing is one of the primary ways analysts gather information about an information system project.

Reasons to use Interview

- It provides information to supplement other methods of collecting data.
- It is used to gather data extensively and intensively.
- It is used to exchange ideas and experience.

1.7.2 System development methodology

System development methodologies are a standard set of steps used to develop and support information systems in organizations. We need a methodology for analyzing a problem to be solved, planning for the design of the solution and a construction method that minimizes the risk of error. We selected agile system development methodology because it employs continual planning, learning, improvement team collaboration, evolutionary development and early delivery. It encourages flexible responses to change. There are different agile system development methodologies but we will use scrum system development method.

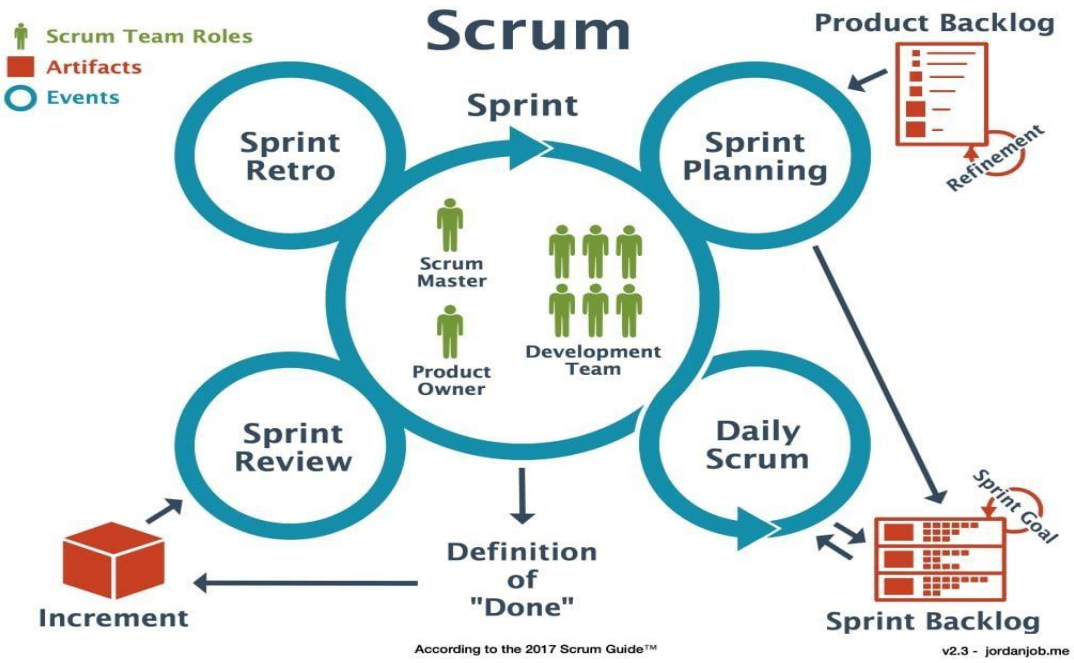


Figure 1.1 Scrum methodology structure

1.8 Development tools and technologies

1.8.1 Front end technologies

These are technologies to be used to build web pages and user interfaces for web applications.

HTML

Html (Hypertext Markup Language) is the most basic building block of the Web. It defines the meaning and structure of web content.

Advantages of using html

It is easy to use and to learn, simple to edit, light weight, user friendly, and free. In addition to these, all browsers support it.

CSS

CSS (Cascading Style Sheets) is a declarative language that

controls how webpages look in the browser.

Advantages of using CSS

It saves time, helps to make spontaneous change and consistent change, improve page loading speed, device compatible, and makes search engine better crawl web pages.

JavaScript

Java script is a programming language used most often for dynamic client-side scripts on webpages and allows you to implement complex things on web pages. It is easy to learn, debug and test, fast no need of compilation, platform independent, and has programming language capabilities.

Bootstrap

Bootstrap is a free, open-source HTML, CSS, and JavaScript framework for quickly building responsive websites. It saves time, encourages consistency, provides an excellent grid system, creates responsive website.

jQuery

jQuery is a JavaScript Library that focuses on simplifying DOM (Document Object Model) manipulation. It is open source, makes table sortable and dynamic, display easily charts, popup overs, modals, tabs, integrated or compatible with bootstrap, highly extensible, cross-browser compatible.

AJAX

AJAX (Asynchronous JavaScript and XML) allows updating parts of the DOM of a HTML webpage instead of having to reload the entire page. It is the integration of jQuery and PHP. It is easy to learn and use, allows changing content without refresh or reload webpage, makes website faster. Response time is faster so increases performance and speed.

1.8.2. Back-end technologies

These technologies used to deal with server, application and database.

PHP

PHP is a server-side scripting language. That is used to develop Static websites or Dynamic websites or Web applications. PHP stands for Hypertext Preprocessor, that earlier stood for Personal Home Pages. It is open source. PHP is manly supported by all the operating systems like windows, Unix, and Linux. It can be integrated with other programming language and database easily and there is no requirement of

redevelopment. Which means it saves a lot of effort and cost. It is simple and easy to learn and to code. It is one of the fastest languages.

Laravel

Laravel is an open-source PHP web framework for creating web applications. It is used to develop websites, APIs, and web services quickly and easily. Laravel provides an expressive and elegant syntax for writing efficient, concise code.

MySQL

MySQL is a free to use, open-source database that facilitates effective management of database by connecting them to the software. Advantages of using MySQL It is more secure and reliable, provides on demand scalability, and open source so reduces cost.

Apache

Apache is an open-source and free web server software that powers around 46% of websites around the world. The official name is Apache HTTP Server, and it's maintained and developed by the Apache Software Foundation.

Advantages of using Apache

- It is open source, reliable and stable software.
- It is also easy to configure and has easily available support in case of any problem.

XAMPP

XAMPP stands for Cross Platform Apache MariaDB (MYSQL), PHP, Perl XAMPP is a free open-source cross-platform web server solution stack package developed by Apache Friends, consisting mainly of the

Apache HTTP server, MariaDB database, and interpreters for scripts written in the PHP and Perl programming languages.

Advantages of using XAMPP

It is easy to learn and install when compared with other web servers like WAMP. It consists of Apache, and MYSQL, so there is no need of installing them separately. It is also possible to start and stop the web server and database stack with one command.

1.8.3 Documentation and modeling tools

At the end of the project, we will deliver system documentation and user documentation. Our system documentation would be the description of detail information about user requirements; systems design specifications, its internal workings and its functionalities. And our user documentation will be in the form of online help with a web connection. It will contain information about the system; how it works and how to use it.

We will use the following tools for our project's documentation:

- Lucid chart to draw UML diagrams and
- Microsoft Excel 2019 to make project schedule using Gantt chart
- Figma as prototyping tool
- Microsoft Word 2019 as word editor.

1.8.4 Deployment environment

We use **Yegarahost.com** to store the system and run it on the internet.

1.9 Scope

Scope refers to all the work involved in creating the products of the project and the processes used to create them. Defining scope of a project is extremely important to not only make sure no obstacles crop up unexpectedly but also to resolve any potential risks that may arise during its execution.

It is very important to articulate what will be included and what will be excluded in this project. In fact, we will add additional functionality if we found it very necessary to the success of the project. We may also discard functionality if we found it is not really important and affordable.

- It will only be functional in Ethiopia
- It has stakeholders like consumers, Farmers (producer) and system administrator.
- It registers different products
- It displays different products • It has delivery functionality
- It has payment system.

Exclusion:

There is no any system without any limitations. However, we believe understanding these limitations is of great importance. We have no any intention to make the system function globally.

1.10 Risks assumptions and constraints

Risks:

- The level of awareness users (especially farmers) has to technology may not be good. As a result, people may not adopt the system easily.
- We may not have sufficient time and knowledge to develop all the deliverables of the project.
- Consumers may be reluctant to the reliability of the system.

Assumptions:

- We assume consumers own and know how to use smart phones and computers.
- We assume consumers will easily understand the advert we make.
- The cost we are going to invest is enough to get the project done.
- We will have improved internet connection or at least it continues like this.
- We assume both the payment and delivery system run smoothly to make the system fully functional.

Constraint:

- Not all farmers may easily adapt the system.
- Not all of our group members may participate responsibly.

1.11 Phases and Deliverables of the project

By definition phase is the logical division of a project. Like many processes, the development of information systems often follows a life cycle. A project may be single phase or a multiphase project accordingly. Most organizations use system the standard system development life

cycle (SDLC). System experts divided SDLC into four major development phases. These phases include System Planning and Selection, System Analysis, System Design and Implementation and operation. Our new System will have seven logical phases. These are introduction, business area analysis and requirements definition, object-oriented analysis(agile), conclusion of the first three phases, object-oriented design, object-oriented implementation and conclusion of the entire project phases. This report includes up to the conclusion of the first three phases. Each phase of the project will have its own tasks and respective deliverables. The term deliverable describes a product created as part of a project. Deliverables can be product related, such as a piece of hardware or software, or process-related, such as a planning document or meeting minutes. Our project deliverables could be the set of software's and documentations. The table below shows tasks and its corresponding deliverables.

Title	**Deliverables**
System Planning and Selection	Convincing proposal document
Business Area analysis and Requirements Definition	• Existing systems documentation • Requirements documentation
Object Oriented System Analysis	• List of User Interfaces • List of Business Rules • System Use Case Diagram and Use Case Description • Class Diagram and Class Description • Sequence Diagram • User Interface Prototype

Conclusion	• Revision of the previous phases of the project and concluding statement.
Object Oriented System Design	• Class Type Architecture • Class Diagram and Description of Classes • State chart diagram • Component diagram • Collaboration diagram • Deployment diagram • Relational model • Database diagram • User Interface Flow Diagram • User interface design
Object Oriented Implementation	• Working System Program integrated to database initialized with data
Conclusion	Revision of the entire project and Concluding paragraph

Table 1.2 phases and deliverables of the project

1.12 Work-break down structure

A work breakdown structure (WBS) is a visual, hierarchical and deliverable oriented deconstruction of a project. It is a helpful diagram for project managers because it allows them to break down their project

scope and visualize all the tasks required to complete their projects. All the steps of project work are outlined in the work breakdown structure chart, which makes it an essential project planning tool. The final project deliverable, as well as the tasks and work packages associated with it rest on top of the WBS diagram, and the WBS levels below subdivide the project scope to indicate the tasks, deliverables and work packages that are needed to complete the project from start to finish.

Phases	Tasks	Responsible person
1.System planning and selection	1.1 Overview 1.3. Stating Problem statement 1.4. defining Objective of the project 1.5. Conducting Feasibility study 1.6. Analyzing Significance of the system 1.7. Defining Beneficiaries of the project	Biruk Amare Haileamlak Walelign Kirubel Asnakew
	1.8. choosing Development methodology 1.9. Specifying Development tools and technologies 1.10. Scoping of the project 1.11. Determining Risks assumptions and constraints 1.12. Determining Phases and deliverables of the project	Abdulmenan mohammed Anteneh Amare Samuel Tesfaye

Table 1.3 work-break down structure

1.13 Project schedule

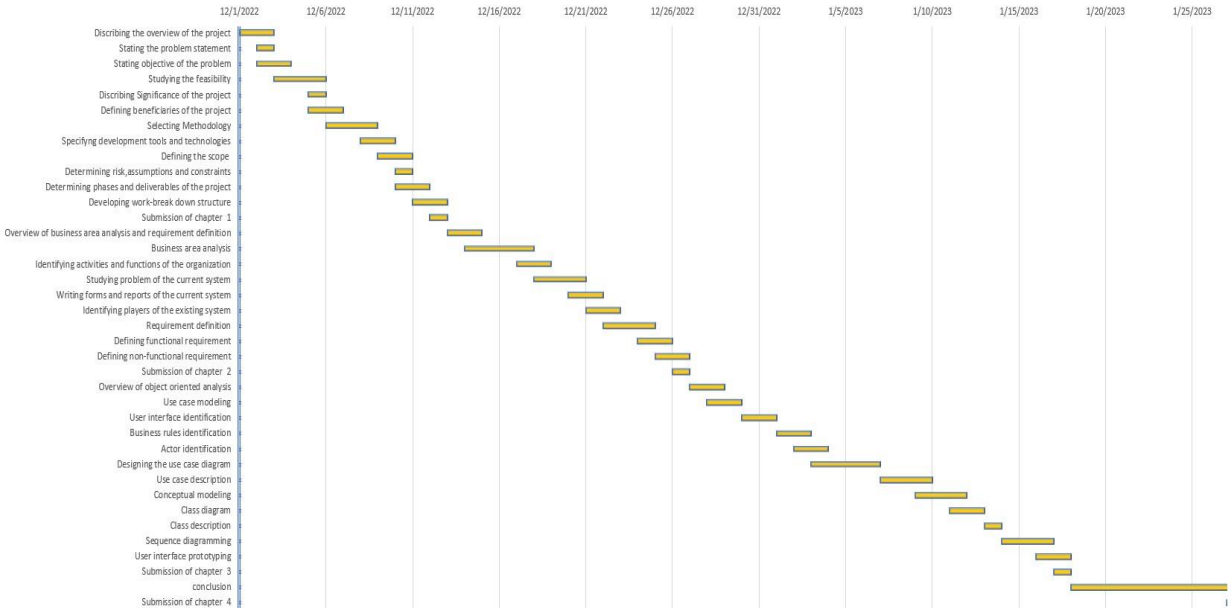


Figure 1.2 project schedule Gantt chart

2.Business area analysis and requirement definition

2.1 Overview

So far, the proposal section has covered business requirements. To create a system, we must specify more than just the business requirements. We also need to be aware of what the user requirements are. User

requirements will be covered in the following chapter. Understanding the system requirements is our top priority because they inform us of the features that our suggested system must have in order to fully satisfy the needs of our users. This chapter also looks at the reports, forms, and concerns about the current system. Finding the suggested system's functional and non-functional requirements will also be of crucial concern.

2.2 Business area analysis

In Ethiopia, people buy and sell products in different ways. The most common way of doing this is by using a traditional market where the place is less comfortable, dirty, and jostling, unlike the modern market that provides convenience for the buyer. Ethiopian Commodity Exchange is the only company providing commodity exchange services.

The Ethiopia Commodity Exchange (ECX) is a commodities exchange established in April 2008 in Ethiopia. In Proclamation 2007-550, which created the ECX, its stated objective was "to ensure the development of an efficient modern trading system" that would "protect the rights and benefits of sellers, buyers, intermediaries, and the general public.

The ECX is set up as a private company owned by a partnership of the market actors, members of the exchange, and the Ethiopian government, led by Eleni Gebremedhin a former economist for the International Food Policy Research Institute and the World Bank. As of July 2011, the physical presence of the ECX consists of 55 warehouses in 17 regional locations. It has grown from trading 138,000 tons in its first year to 508,000 tons in its third year, with nearly equal shares of coffee and oilseeds, and pulses. The value of the ECX rose 368% between 2010 and 2011 to reach US$1.1 billion. As of November 2010, the trading floor in Addis Ababa handled 200 spot contracts in commodities such as coffee,

sesame, navy beans, maize, and wheat. It was assessed in July 2011 that membership equalled 243 with clients, who trade through members, about 7,800. Farmer cooperatives represented 2.4 million smallholder farmers, which make up 12% of the membership

In spite of its remarkable results, ECX does not offer its services to the local market (it focuses solely on exports). Our project will address this challenge.

2.2.1 Activities and function of the organization

Farmers are traditionally involved in the production of their goods, and then they sell them through brokers to merchants, and merchants then sell them to customers.

2.2.2 Problems of the current system

- **Time**: the current market takes time and energy.
- **Cost**: cost of the product is much more expensive because of free market economy.
- Difficult to compare prices.
- **Information**: farmers and buyers don't have any information about where to sell and buy products.
- **Service**: Obtaining a product of the desired quality is challenging.
- **Security**: loss of the goods from the marketplace.
- **Efficiency**: It can be very difficult to find a product that you want.

2.2.3 Forms and reports of the current system

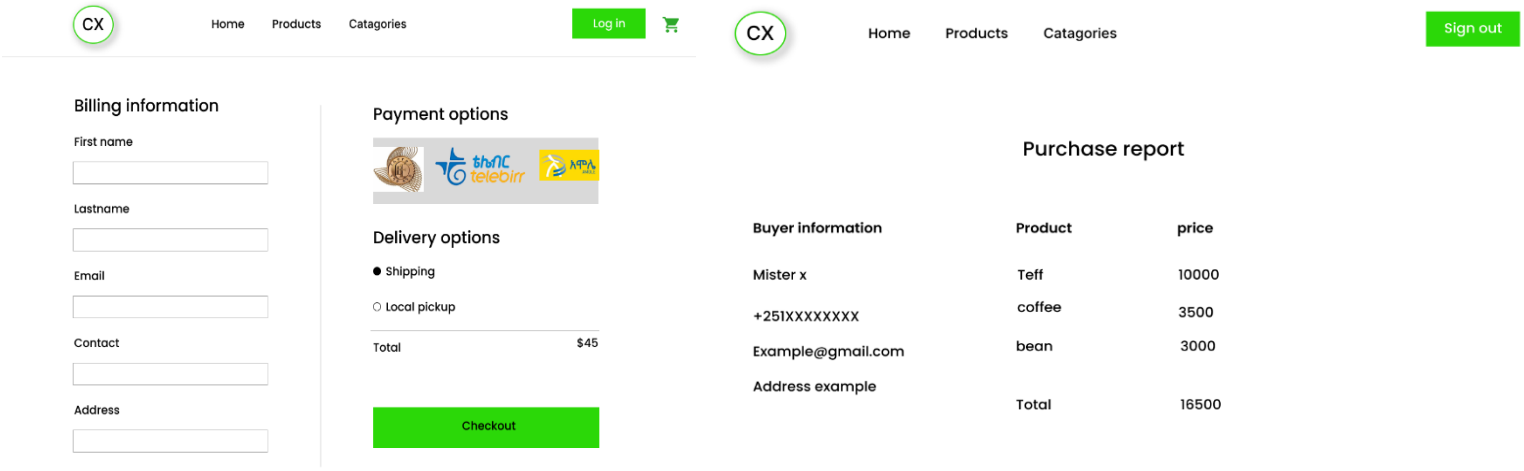


Figure 2.1 Forms and reports of the current system

2.2.4 Players of the existing system

Customer: A person responsible for purchasing goods from the merchant.
Farmer: A person responsible for producing good
Government: responsible for managing and controlling the overall market situation.
Merchant: responsible for selling goods to the customer.
Broker: Facilitate the transaction between farmer and merchant as well as customer and merchant.

2.3 Requirements definition

A requirement is simply a high-level, abstract statement of service that a system should provide or a constraint on system. These requirements reflect the needs of customers for a system that serves a certain purpose such as controlling a device, placing an order, or finding information. The process of finding out, analysing, documenting, and checking these services and constraints is called requirements engineering. Software system requirements are often classified as functional requirements or non-functional requirements.

2.3.1 Functional requirement

Functional requirements are the specifications of the product's functions (features). Simply put, functional requirements define what precisely a system must do and how the system must respond to inputs. Functional requirements define the system's goals, meaning that the system will not work if these requirements are not met.
Functional Requirements for the proposed system include:

- Register new users(farmers)
- Register products with their quality, quantity, and amount
- Store users(farmers) and product information
- Allowing a feedback mechanism
- Allow delivery system
- Allow payment system
- It allows users(buyers) to search products
- It will show products picture with brief descriptions

2.3.2 Non-Functional requirement

Non-Functional Requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviours.

Non-Functional Requirements for the proposed system include:

Usability

Usability is the degree of ease with which the user will interact with the system to achieve required goals effectively and efficiently. We will design the system to maximize the amount of time users spend on it by keeping it simple, consistent in color and texture, and easy to navigate.

Reliability

Reliability is defined as the probability of a system or system element performing its intended function under stated conditions without failure for a given period. Reliability depends on availability and Maintainability. Even if everything is going the right way if users do not perform their tasks honestly it will be a great loose to our system. Therefore, we should consider reliability while designing our system. We will take the failure rate, Repair rate, mean time to recovery, mean time to detection, and Mean time between failures measures to make our system reliable.

Performance

One of the benefits of our project is to deliver services quickly and efficiently. To achieve we believe it is necessary to set performance requirements for our system. Any search result should not exceed a maximum of 30 seconds. The system should send notifications instantly whenever it is necessary.

Availability

Availability is the probability that a repairable system or system element is operational at a given point in time under a given set of environmental conditions. Availability is another concern of our system. Our system will be with high availability when the system will be launched for the first time and will be continuously available when users need it. The system should operate a minimum of 14 hours a day and 7 days a week. We decided our system to be available from 6 a.m. to 5 p.m. This specification is based on our assumption that users will use the system more off in the daytime. However, this is not the end. Availability requirements of the system will be changed based on users' needs in the future. When the number of users is getting larger and users need higher availability of the system, we will make the system available 24/7.

Scalability

We shall design the system by considering scalability. We can make it possible because first the methodology we are using was selected considering the future scalability of the system. The selected methodology, i.e., Agile system development methodology supports scalability. Second, the system will be designed considering scalability. After, the system is designed and implemented the system administrators will add services in their respective categories and even add new categories.

Security

Information is an important asset. The more information you have at your command, the better you can adapt to the world around you. In business, information is often one of the most important assets a company possesses. Information differentiates companies and provides leverage that helps one company become more successful than another. We value our business and users' information too. Our system is a web application that has a 24/7 connection to the internet. Unless we use different security

mechanisms, we could lose our information, which will endanger our system. Therefore, we will use different security mechanisms. like: - authentication, authorization, and access control mechanisms.

3.Object-oriented analyses

3.1 Overview

In the previous chapter, we have clearly identified the problems of the existing system and we proposed an alternative solution. And also, we pointed out the functional and non-functional requirements of the new system. In this chapter, we will apply object-oriented analysis to determine the functions and requirements of the new system, including creating use case diagrams, class diagrams, sequence diagrams, UI designs, and identifying business rules. We will also build prototypes of the system and come up with a concrete visualization of how the new system will look like.

Object-oriented analysis (OOA) looks at the problem domain, with the aim of producing a conceptual model of the information that exists in the area being analyzed without considering any implementation details.

3.2 Use case modeling

Use case modeling is a software development technique that is used to facilitate the design and description of a proposed user interface. It involves representing tasks, situations, and interactions between users and the system (computer) as a two-dimensional diagram that defines how the system should respond. The model contains information regarding inputs, outputs, rules, and boundary conditions.

The use case models can help improve the accuracy of requirements analysis, by providing an understanding of what needs to be built in order to satisfy the customer's expectations. They can also assist in improving

the process of communication between developers, customers and users, by ensuring all stakeholders have a common understanding of what needs to be implemented. It also helps anticipate risks associated with specific steps within a project or software development initiative.

3.2.1 UI Identification

UI identification is a process of identifying user interface (UI) elements within an application so that they can be shown in a user-friendly way. This process involves providing descriptive names, labels, and other information that makes it easier for users to interact with the system.

ID	User interface	Description	Actors
UI_01	Home page	The first page the user or customers sees whenever they access the website	Customer or user
UI_02	About us page	The page that says some information about the system or organization	Customer or user
UI_03	Category page	page where products are organized and grouped into categories. It provides customers the ability to quickly find what they are searching for and easily browse through product offerings.	Customer or user
UI_04	Products page	A page listing the various items available for sale. It often includes product photos, descriptions, prices and purchasing options.	Customer or user
UI_05	Sign up page	A page that contains form where user create an account	Customer, admin, and system admin
UI_06	Log in page	A page in which a user can fill his log in credentials	Customer, admin, and system admin
UI_07	Cart page	A page where customers can select the items they would like to purchase and proceed to checkout.	Customer or user
UI_08	Checkout page	A page where customer review their order and submit payment information.	Customer or user

UI_09	Order confirmation page	is the final page that verifies and confirms the order details and shows payment and shipping information.	Customer or user
UI_10	Admin dashboard page	A page containing representations of business performance and overall insights into the website's activity.	admin
UI_11	Contact us page	A page that provides users with contact information and gives them a way to contact by providing a contact form or email address.	Customer or user
UI_12	Manage-category page	A page where admin manages categories	Admin
UI_13	Manage-product page	A page where admin manages Products	Admin
UI_14	Manage-order page	A page where admin manages orders	Admin
UI_15	Manage-admin page	A page where super admin manages admins	system admin
UI_16	Add-category page	A page where admin add categories	Admin
UI_17	update-category page	A page where admin update categories	Admin
UI_18	Add-product page	A page where admin add products	Admin
UI_19	update-product page	A page where admin update a product	admin
UI_20	Add-admin page	A page where super admin add admin	system Admin
UI_21	update-admin page	A page where admin update admin	Admin

Table 3.1 User interface identification

3.2.2 Business rule identification

Business rule identification is the process of discovering, documenting, and defining the business rules that govern a business process, procedure, or application. Business rules clarify what data is required, the relationships between data elements, decision logic, and other aspects of the business process. They help to establish data integrity, reduce ambiguity, and improve overall efficiency.

List of business rule

Rules	Description
R1	All customers must create an account in order to complete a checkout.
R2	All prices are listed in ETB and include taxes.
R3	Products are sold on a first-come, first-served basis.
R4	Orders will be shipped within 2-5 business days of payment.
R5	The customer will be charged at the time of purchase.
R6	Free shipping is offered on orders over 30,000 ETB.

Table 3.2 business rule

3.2.3 Actor Identification

Object-oriented system analysis involves the identification of objects and actors for a given system. Actors are users or systems outside an object, but interact with that object. Identification of actors involves studying the external interface of the system and analyzing user and other systems' interactions or communications with the object in order to determine the need for actor interaction with the object. Actors could be users, hardware, software, or even automated systems.

List of actors

- Government or super admin
- admin
- Bank
- System admin
- customers

3.2.4 Designing the use case diagram

Use case diagram is a behavioral UML diagram type and frequently used to analyze various systems. They enable you to visualize the different types of roles in a system and how those roles interact with the system. It mainly had four parts.

Actor-represents an external entity that interacts with a system
Use case: A use case represents a function or an action within the system. It's drawn as an oval and named with the function.
System: The system is used to define the scope of the use case and drawn as a rectangle.
Relationship: The relationship between actor and the system and it can be.

List of use case

- Register
- Login
- Browse products
- Browse product categories
- Search products
- Add to cart
- Choose payment method
- Choose delivery option
- Checkout
- View product
- Add product
- Update product
- Delete product
- View category
- Add category
- Update category
- Delete category
- Add admin
- View admin
- Delete admin

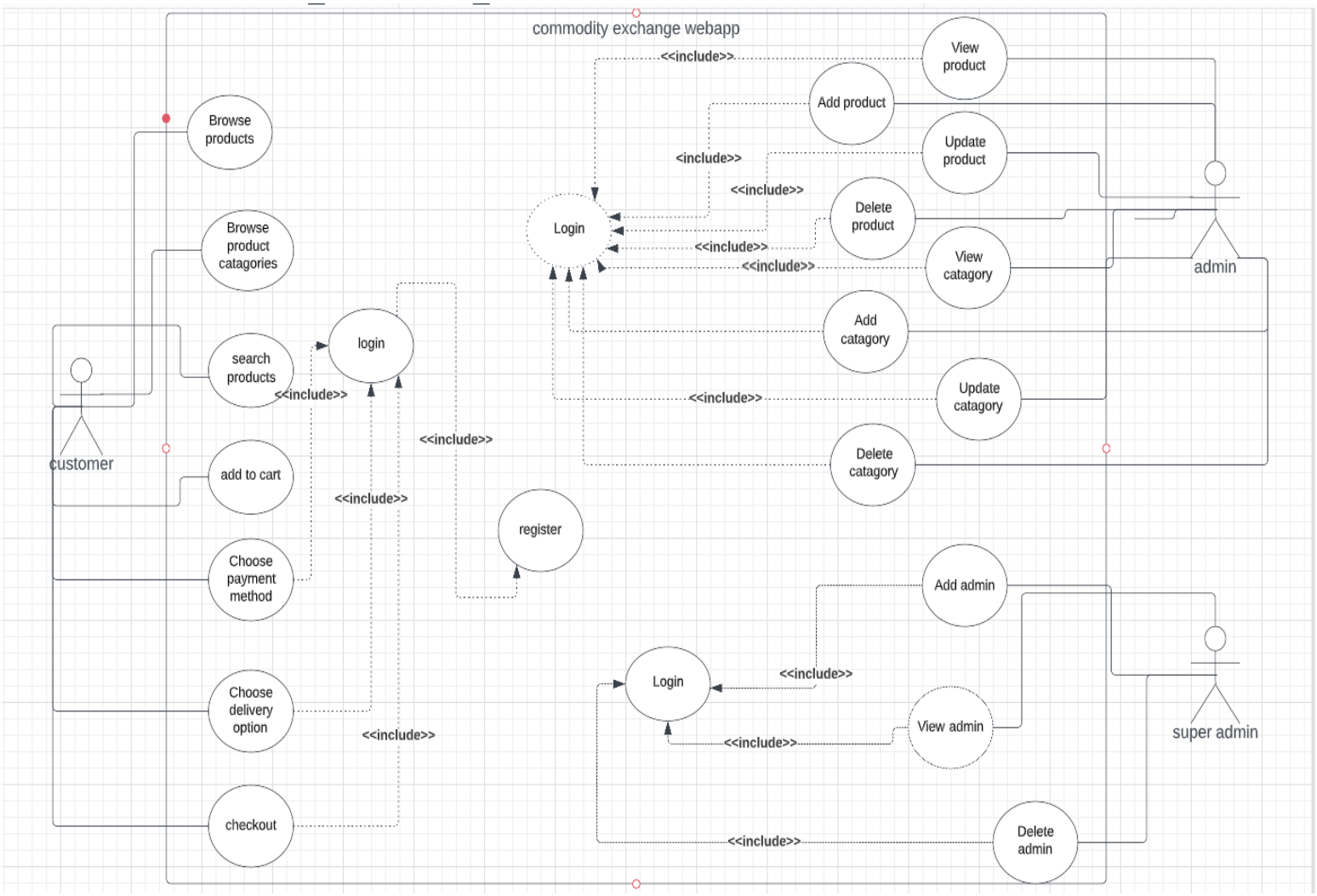


Figure 3.1 use case diagram

3.2.5 Use case description

Below are the lists of the use cases that are identified for this. The system use case description contains:

- Use case name
- Use case identification number
- Actor
- Description
- Pre-condition
- Post-condition

- Basic course of Action
- Alternative course of Action

Use case name	Register
Use case ID	UID_01
Description	it is the act of creating an account with the online store by providing personal data
Actor	Customer
Pre-condition	None for non-Customer
Post-condition	Non-Customer becomes Customer
Basic course of Action	1. The use case starts when Customer click the “Register” button 2. System display Registration form 3.Customer fill all required information 4. System checks and validate if the inputs are valid 5. Customer clicks the “register” button 6. System displays success message 7. The use case ends
Alternative course of Action	A4 System checks and validate if the inputs are valid A4.1 System checks that inputs are not valid and display error message A4.2 The use case continues at step 2

Table 3.3 Use case description for system use case register

Use case name	Login

Use case ID	UID_02
Description	It enables the user to log into the system
Actor	Customer, admin,System admin
Pre-condition	UID_01 for non - Customer
Post-condition	Customer and Admin logged in
Basic course of Action	1. The user clicks “login” button 2. The system display “Login form” 3. The user enters user-name and password, and clicks the “login” button. 4. The system verifies whether the user-name and the password is correct. 5. The user will be logged into the system. 6. The use case ends.
Alternative course of Action	A4 System checks and validate if the inputs are valid A4.1 System checks that inputs are not valid and display error message A4.2 The use case continues at step 2

Table 3.4 Use case description for system use case login

Use case name	**Browse Product**
Use case ID	UID_03
Description	Lists of commodities available to buy
Actor	Customers and non-customers
Pre-condition	None
Post-condition	View list of products/commodities
Basic Course of Action	1. The Customers or Non-customers click the “Products” button found in the home page navigation bar. 2. The system displays the available list of products

	3. The use case ends.
Alternative course of Action	None

Table 3.5 Use case description for system use case browse product

Use case name	**Browse Product category**
Use case ID	UID_04
Description	List the categories of commodities.
Actor	Customer and non-customers
Pre-condition	None
Post-condition	View a list of product categories
Basic course of Action	1. Customer or Non-customer clicks the “Category” button found in the home page navigation bar. 2. The system displays the category of the products. 3. The use case ends.
Alternative course of Action	None

Table 3.6 Use case description for system use case browse product category

Use case name	**Add to cart**
Use case ID	UID_05
Description	It enables customers to save their choice of a commodity.
Actor	Customers and non-customers
Pre-condition	None
Post-condition	List of promised commodities.
Basic course of Action	1. Customer or Non-customer clicks the “add to cart” button found in each product. 2. The system adds the products to the add-to-cart page.

	3. The system offers a process of checkout for customers. 4. The use case ends.
Alternative course of Action	None

Table 3.7 Use case description for system use case add to cart

Use case name	**Search Product**
Use case ID	UID_06
Description	Offer search options for customers to find products.
Actor	Customers and non-customers
Pre-condition	None
Post-condition	Search for every product listed in the UID_03.
Basic course of Action	1. The Customer or Non-customer clicks the bar found on the theme on the product page. 2. The system displays the required products. 3. The use case ends.
Alternative course of Action	None

Table 3.8 Use case description for system use case search product

Use case name	**Choose payment Method**
Use case ID	UID_07
Description	Offer available options for payment methods.
Actor	Customers
Pre-condition	UID_01 for Non-customer, UID_02 for Customers
Post-condition	Choose the payment option to pay the price.
Basic course of Action	1. Customers select the product from the add to cart page.

	2. The system offers or displays the available ways of payment. 3. The customer selects their choice of payment. 4. The customer buys the product/s. 5, The use case ends.
Alternative course of Action	None.

Table 3.9 Use case description for system use case choose payment method

Use case name	**Choose Delivery Option**
Use case ID	UID_08
Description	Offer available delivery options for Customers.
Actor	Customer
Pre-condition	UID_01 for Non-customers, UID_02 for Customers
Post-condition	Choose a delivery option to accept their product.
Basic course of Action	1. Customers click a delivery options button from UID_09. 2. The system offers available delivery methods 3. The customer selects their choice of delivery options. 4. Deliver the product as per the customers' choice 5. The use case ends.
Alternative course of Action	None.

Table 3.10 Use case description for system use case choose delivery option

Use case name	**Checkout**
Use case ID	UID_09

Description	Checkout is the final step in making a purchase.
Actor	Customer
Pre-condition	UID_02 then the customer should click “proceed to checkout” button in UID_05
Post-condition	verification of customer details such as address, payment method and order items before checkout to ensure accuracy.
Basic course of Action	1.the customer browse product/s 2.the customer clicks the “add to cart” button 3.the customer clicks the “proceed to checkout” button 4.the customer take the Choose payment Method and Choose Delivery Option 5.the System checks and validate if the inputs are valid 6.finally the customer clicks “Checkout” button 7.the use case ends
Alternative course of Action	5 System checks and validate if the inputs are valid A5.1 System checks that inputs are not valid and display error message A5.2 The use case continues at step 4

Table 3.11 Use case description for system use case checkout

Use case name	**View Product**
Use case ID	UID_10
Description	Allows the admin to view products
Actor	Admin
Pre-condition	UID_02 for admin
Post-condition	Admin viewed products
Basic course of Action	1. The use case starts when the admin navigates to product page 2. System displays all products with their description and operations 3. Admin view products 4. The use case ends
Alternative course of Action	None

Table 3.12 Use case description for system use case view product

Use case name	**Add Product**
Use case ID	UID_11
Description	Allows the admin to add products
Actor	Admin
Pre-condition	UID_02 and UID_10 for admin
Post-condition	Admin added products
Basic course of Action	1. The use case starts when the admin navigates to product page 2. System displays all products with their description and operations 3. Admin click on add button and navigate to add-product page 4. The system displays a form that allow the admin to add-products to the system 5. The admin fills product information and click add-product button 6. System checks and validate if the inputs are valid 7. System displays success message 8. The use case ends

Alternative course of Action	6 System checks and validate if the inputs are valid A6.1 System checks that inputs are not valid and display error message A6.2 The use case continues at 4

Table 3.13 Use case description for system use case add product

Use case name	**Update Product**
Use case ID	UID_12
Description	Allows the admin to update products
Actor	Admin
Pre-condition	UID_02 and UID_10 for admin
Post-condition	Admin updated a product
Basic course of Action	1. The use case starts when the admin navigates to product page 2. System displays all products with their description and operations 3. Admin click on update button and navigate to update-product page 4. The system displays a form that allow the admin to update product information 5. The admin update product information and click update-product button 6. System checks and validate if the inputs are valid 7. System displays success message 8. The use case ends
Alternative course of Action	6 System checks and validate if the inputs are valid

	A6.1 System checks that inputs are not valid and display error message A6.2 The use case continues at 4

Table 3.14 Use case description for system use case update product

Use case name	**Delete Product**
Use case ID	UID_13
Description	Allows the admin to delete products
Actor	Admin
Pre-condition	UID_02 and UID_10 for admin
Post-condition	Admin deleted a product
Basic course of Action	1. The use case starts when the admin navigates to product page 2. System displays all products with their description and operations 3. Admin click on delete button to delete specific product 4. System displays success message 5. The use case ends
Alternative course of Action	none

Table 3.15 Use case description for system use case delete product

Use case name	**View Category**
Use case ID	UID_14
Description	Allows the admin to view category
Actor	Admin
Pre-condition	UID_02 for admin
Post-condition	Admin viewed categories
Basic course of Action	1. The use case starts when the admin navigates to category page

	2. System displays all categories with their description and operations 3. Admin view categories 4. The use case ends
Alternative course of Action	None

Table 3.16 Use case description for system use case view category

Use case name	**Add Category**
Use case ID	UID_15
Description	Allows the admin to add categories
Actor	Admin
Pre-condition	UID_02 and UID_14 for admin
Post-condition	Admin added categories
Basic course of Action	1. The use case starts when the admin navigates to category page 2. System displays all categories with their description and operations 3. Admin click on add button and navigate to add-category page 4. The system displays a form that allow the admin to add-categories to the system 5. The admin fills category information and click add-category button 6. System checks and validate if the inputs are valid 7. System displays success message 8. The use case ends
Alternative course of Action	6 System checks and validate if the inputs are valid A6.1 System checks that inputs are not valid and display error message A6.2 The use case continues at 4

Table 3.17 Use case description for system use case add category

Use case name	Update Category
Use case ID	UID_16
Description	Allows the admin to update category
Actor	Admin
Pre-condition	UID_02 and UID_14 for admin
Post-condition	Admin updated category
Basic course of Action	1. The use case starts when the admin navigates to category page 2. System displays all categories with their description and operations 3. Admin click on update button and navigate to update-category page 4. The system displays a form that allow the admin to update category information 5. The admin update category information and click update-category button 6. System checks and validate if the inputs are valid 7. System displays success message 8. The use case ends
Alternative course of Action	6 System checks and validate if the inputs are valid A6.1 System checks that inputs are not valid and display error message A6.2 The use case continues at 4

Table 3.18 Use case description for system use case update category

Use case name	Delete Category
Use case ID	UID_17
Description	Allows the admin to delete categories

Actor	Admin
Pre-condition	UID_02 and UID_14 for admin
Post-condition	Admin deleted category
Basic course of Action	1. The use case starts when the admin navigates to category page 2. System displays all categories with their description and operations 3. Admin click on delete button to delete specific category 4. System displays success message 5. The use case ends
Alternative course of Action	none

Table 3.19 Use case description for system use case delete category

Use case name	**Add Admin**
Use case ID	UID_18
Description	The System Admin is able to add an admin to the system
Actor	System Admin
Pre-condition	UID_02 for System Admin
Post-condition	Admin added to the system
Basic course of Action	1. The use case stats when System Admin clicks login button 2. System displays Super Admin page 3. System Admin clicks “Add admin “button 4. System Displays Registration Form 5. System-Admin fill all required information 6. System checks and validate if the inputs are valid 7. Member/super-Admin clicks the register button. 8. System displays a success message. 9. The use case ends
Alternative course of Action	6.System checks and validate if the inputs are valid A6.1 System checks that inputs are not valid and display error message A6.2 The use case continues at step 4

Table 3.20 Use case description for system use case add admin

Use case name	**View Admin**
Use case ID	UID_19
Description	The system Admin is able to view and track admins on the system
Actor	System Admin
Pre-condition	UID_02 for System Admin
Post-condition	System Admin view any changes done by admins
Basic course of Action	1. The use case stats when System Admin clicks login button 2. System displays System Admin page 3. System Admin clicks “View admin “button 4. System Displays list of admins with their detailed information. 5. System-Admin view, manage track 6. Use case ends
Alternative course of Action	None

Table 3.21 Use case description for system use case view admin

Use case name	**Delete Admin**
Use case ID	UID_20
Description	The System Admin is able to Delete admins Account on the system
Actor	System Admin
Pre-condition	UID_02 and UID_19 for Super Admin
Post-condition	Admin account is deleted from the system
Basic course of Action	1. The use case stats when Super Admin clicks login button 2. System displays Super Admin page 3. System Admin clicks “View admin “button

	4. System Displays list of admins with their detailed information. 5. System-Admin clicks delete button to delete an admin
Alternative course of Action	None

Table 3.22 Use case description for system use case delete admin

3.3 Conceptual modeling

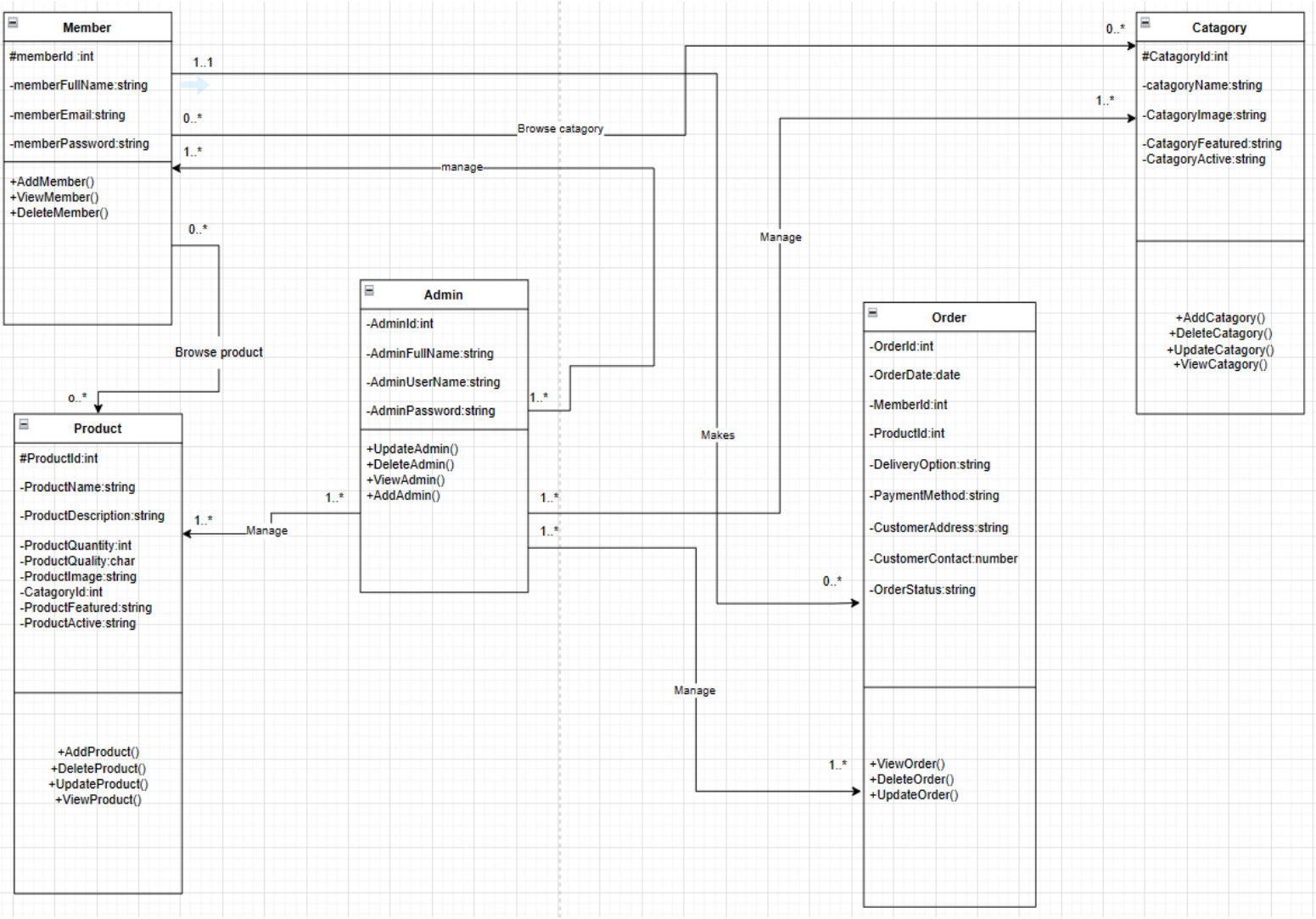
A conceptual model is a representation of a system. It shows how people, places, and things interact. They show the real-world features and interactions of your design idea. In other words, a conceptual model is an abstraction of a piece of the real world or a design you plan to bring into the real world.

Class diagram contains:

- The name classes
- The interrelationships (including inheritance, aggregation, and association)
- the operations
- attributes of the classes

3.3.1 class diagram

A class diagram is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects. Class diagrams are widely used to describe the types of objects in a system and their relationships. Class diagrams model class structure and contents using design elements such as classes, objects, attributes, operations and



relationships. They can also be used for visualizing, documenting and constructing components of software or hardware systems.

Figure 3.2 class diagram

3.3.2 Class description

Class	Member	It includes all registered members
	memberId	Type:int Visibility: Protected
	memberFullName	Type: string

Attribute		Visibility: private
	memberEmail	Type:string Visibility: private
	memberPassword	Type:string Visibility: private
Method	Add Member ()	Visibility: public Parameters: memberId, memberFullName, memberEmail, memberPassword
	ViewMember ()	Visibility: public
	DeleteMember()	Visibility: public

Table 3.23 member class description

Class	**Admin**	**It includes all registered admins**
Attribute	AdminId	Type:int Visibility: private
	AdminFullName	Type:string Visibility: private
	AdminUserName	Type:string Visibility: private
	AdminPassword	Type:string Visibility: private
Method	UpdateAdmin()	Visibility: Public Parameters: AdminId, AdminFullName, AdminUserName, AdminPassword
	ViewAdmin()	Visibility: Public
	DeleteAdmin()	Visibility: Public
	AddAdmin()	Visibility: Public Parameters:AdminId, AdminFullName, AdminUserName, AdminPassword

Table 3.24 admin class description

Class	**Product**	**It includes all registered Products**

Attribute	ProductId	Type:int Visibility: Protected
	ProductName	Type:string Visibility: private
	ProductDescription	Type:string Visibility: private
	ProductQuantity	Type:int Visibility: private
	ProductQuality	Type: char Visibility: private
	ProductImage	Type:string Visibility: private
	CatagoryId	Type:int Visibility: private
	ProductFeatured	Type:string Visibility: private
	ProductActive	Type:string Visibility: private
Method	AddProduct()	Visibility: public Parameters: ProductId, ProductName, ProductDescription, ProductQuantity, Product Quality, ProductImage, CatagoryId, ProductFeatured, ProductActive
	DeleteProduct()	Visibility: public
	UpdateProduct()	Visibility: public Parameters: ProductId, ProductName, ProductDescription, ProductQuantity, Product Quality, ProductImage, CatagoryId, ProductFeatured, ProductActive
	ViewProduct()	Visibility: public

Table 3.24 product class description

Class	Category	It includes all registered categories
Attribute	CatagoryId	Type:int Visibility: protected
	CatagoryName	Type:string Visibility: private
	CatagoryImage	Type:string Visibility: private
	CatagoryFeatured	Type:string Visibility: private
	CatagoryActive	Type:string Visibility: private
Method	AddCatagory()	Visibility: public Parameters: CatagoryId, CatagoryName, CatagoryImage, CatagoryFeatured, CatagoryActive
	ViewCatagory()	Visibility: public
	DeleteCatagory()	Visibility: public
	UpdateCatagory()	Visibility: public Parameters: CatagoryId, CatagoryName, CatagoryImage, CatagoryFeatured, CatagoryActive

Table 3.24 category class description

Class	Order	It includes all orders made by member
Attribute	OrderId	Type:int Visibility: private
	OrderDate	Type: date Visibility: private
	MemberId	Type:int Visibility: private
	ProductId	Type:int Visibility: private
	DeliveryOption	Type: string Visibility: private
	PaymentMethod	Type:string Visibility: private

	CustomerAddress	Type:string Visibility: private
	CustomerContact	Type:number Visibility: private
	OrderStatus	Type:string Visibility: private
Method	ViewOrder()	Visibility: public
	DeleteOrder()	Visibility: public
	UpdateOrder()	Visibility: public Parameters: orderStatus

Table 3.25 order class description

3.4 sequence diagramming

Sequence diagramming is a type of UML (Unified Modeling Language) diagram that shows the order that activities, events, and operations occur in a system or process. It typically employs a vertical timeline that connects multiple lifelines representing different objects or components involved in the system, as well as arrows indicating message/signal flows between them. Sequence diagrams are used to illustrate complex interactions among objects, and can help identify logic errors when developing software systems.

sequence diagram consists of the following component

class Roles or Participants- Class roles describe the way an object will behave in context. Use the UML object symbol to illustrate class roles, but don't list object attributes.

Activation or Execution Occurrence-Activation boxes represent the time an object needs to complete a task. When an object is busy executing a process or waiting for a reply message, use a thin gray rectangle placed vertically on its lifeline.

Messages-Messages are arrows that represent communication between objects.

- **Synchronous Message:** A synchronous message requires a response before the interaction can continue. It's usually drawn using a line with a solid arrowhead pointing from one object to another.
- **Asynchronous Message:** Asynchronous messages don't need a reply for interaction to continue. Like synchronous messages, they are drawn with an arrow connecting two lifelines; however, the arrowhead is usually open and there's no return message depicted.
- **Reply or Return Message:** A reply message is drawn with a dotted line and an open arrowhead pointing back to the original lifeline.
- **Self-Message:** A message an object sends to itself, usually shown as a U-shaped arrow pointing back to itself.

Lifelines- Lifelines are vertical dashed lines that indicate the object's presence over time.

Destroying Objects- Objects can be terminated early using an arrow labeled "<< destroy >>" that points to an X. This object is removed from memory.

Loops- A repetition or loop within a sequence diagram is depicted as a rectangle. Place the condition for exiting the loop at the bottom left corner in square brackets.

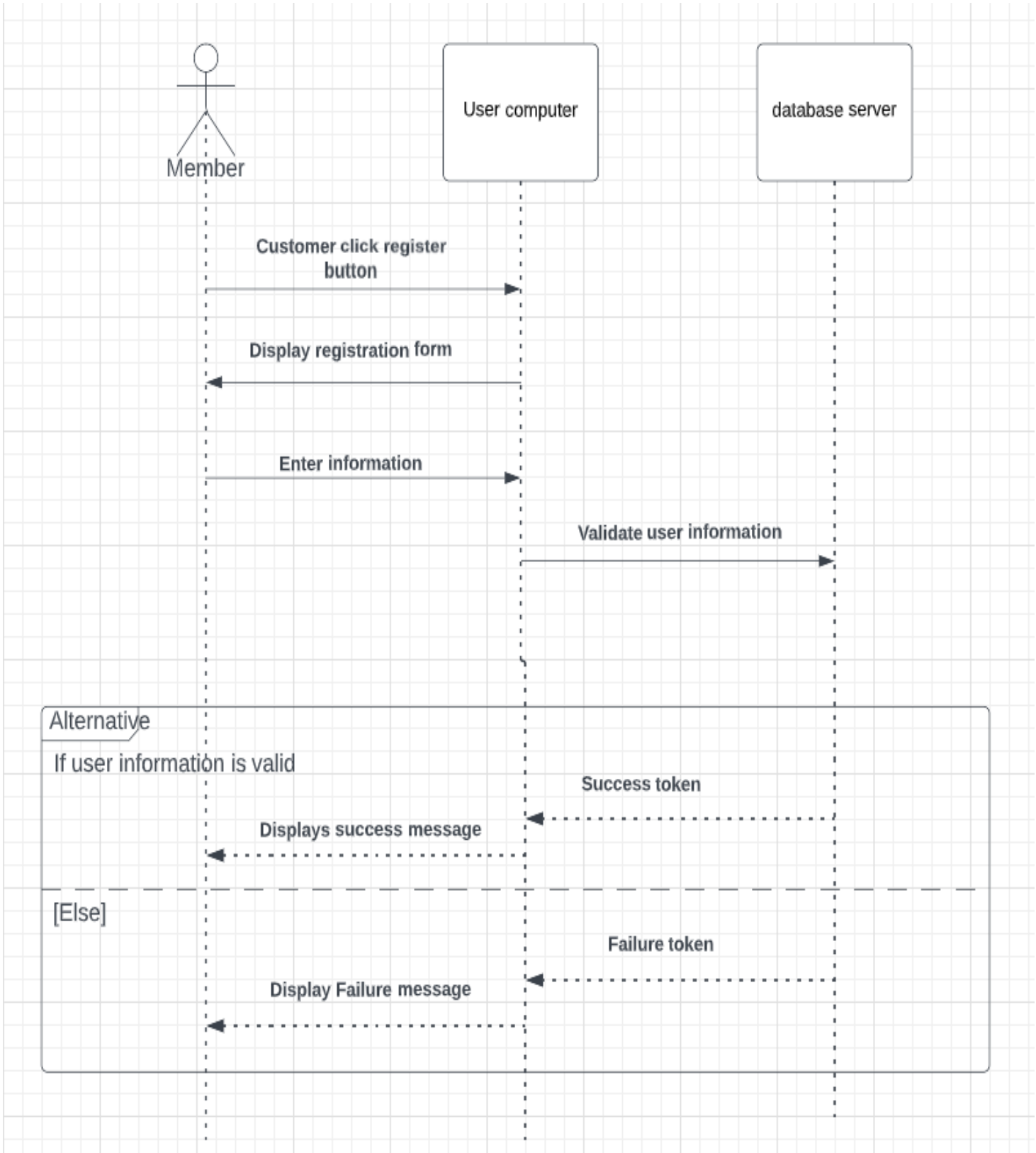


Figure 3.2 sequence diagram for register

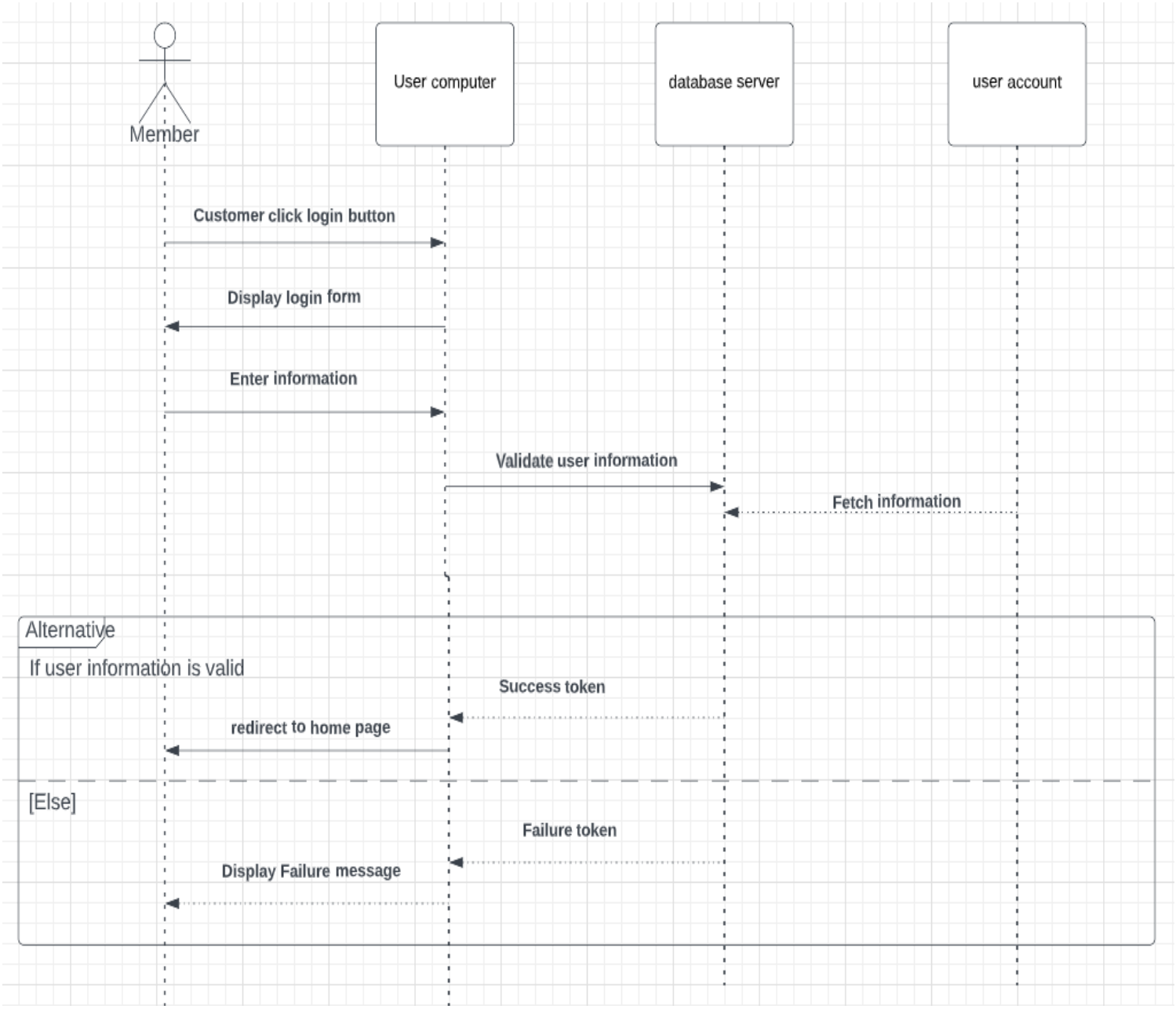


Figure 3.3 sequence diagram for login

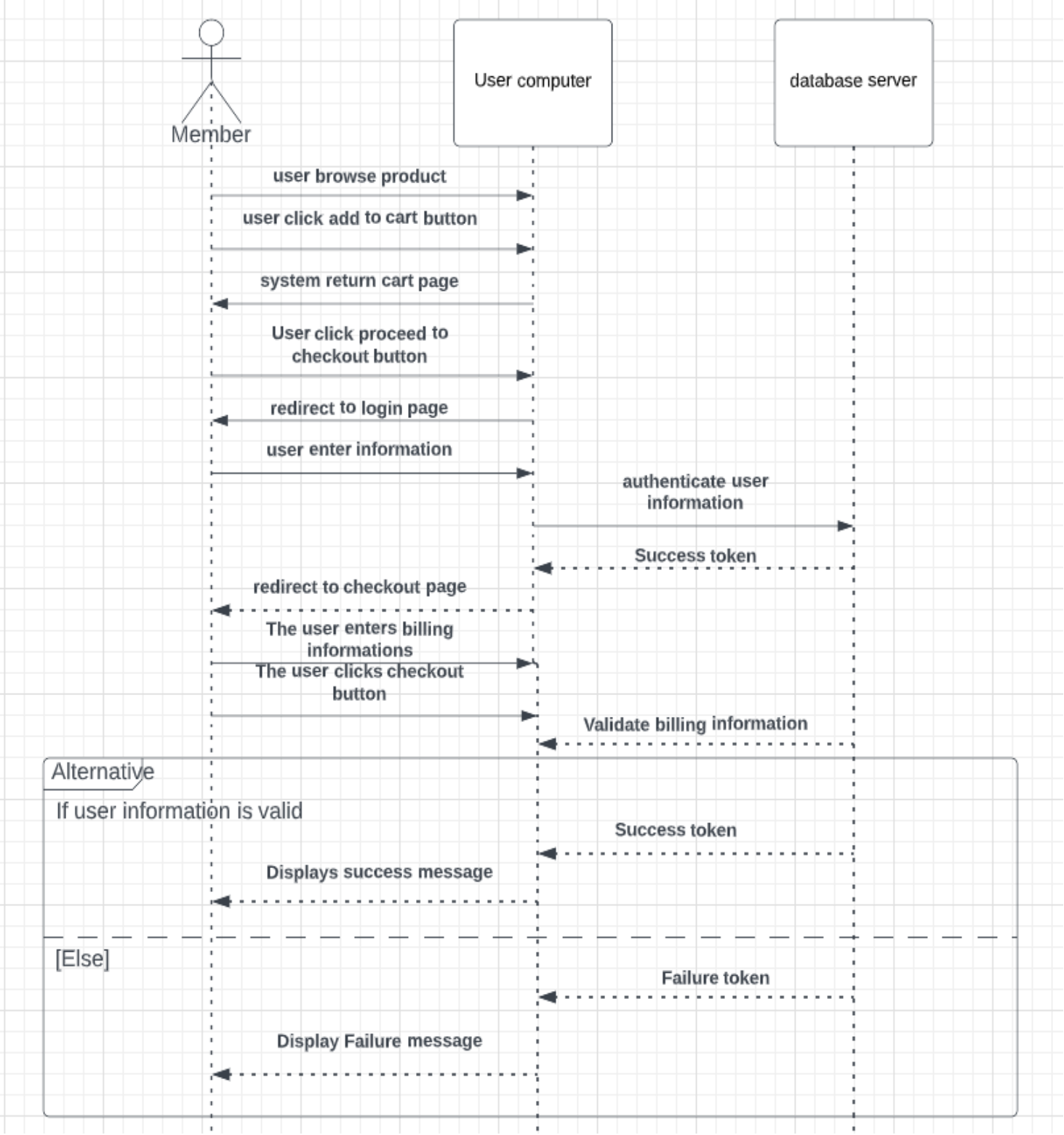


Figure 3.4 sequence diagram for product order

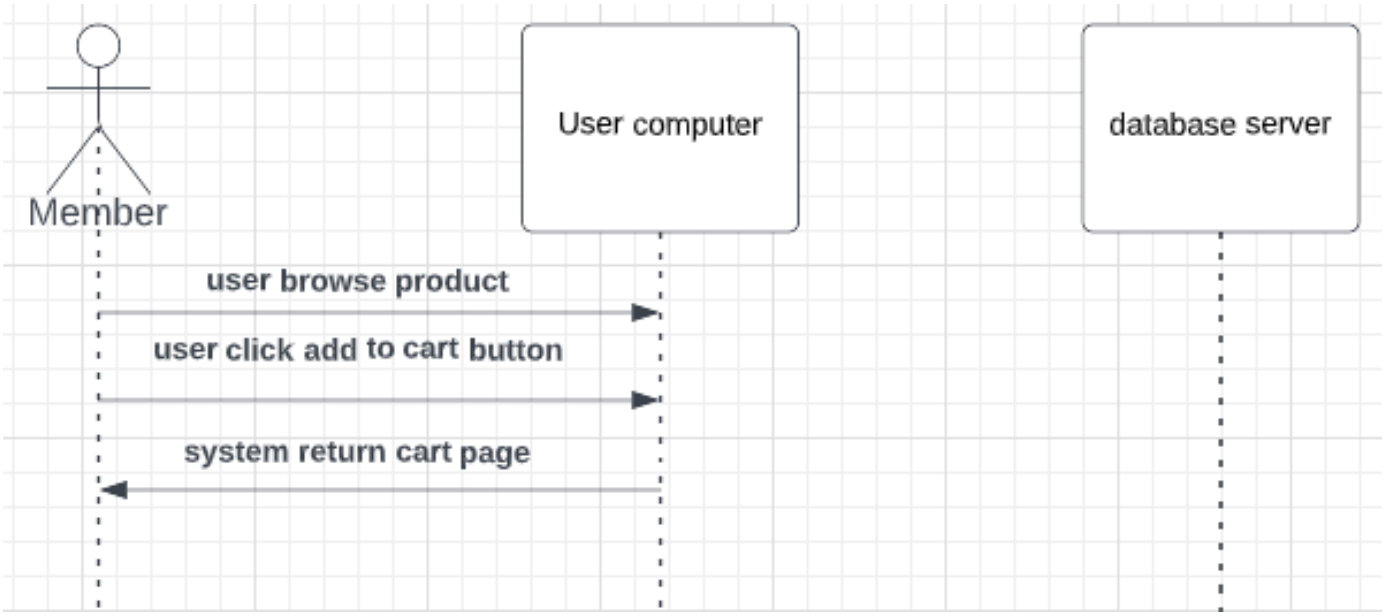


Figure 3.5 sequence diagram for add to cart