# CHAPTER ONE

## Requirement Analysis (use cases)

**Requirement analysis** is a critical phase in the systems engineering, software engineering, and project management processes. It involves understanding, documenting, and managing the needs and requirements of stakeholders for a particular project or system.

As a group of 3rd-year Software Engineering students at Wachemo University, we began our web-based e-commerce app project by conducting requirement analysis to understand the needs of our university community, including students and staff. Our aim was to create a platform accessible through a web browser, enabling users to buy and sell products within the Wachemo ecosystem. We gathered requirements through informal interviews with students and staff, a focus group with five classmates. This process helped us compile a comprehensive list of functional and non-functional requirements, which we documented as a team to guide the development of our web app.

Requirement Analysis of Wachemo University E-Commerce System (Wache Mareket):

**Functional Requirement:**

1. User Registration:

* Users should be able to register an account with the web-based e-commerce system by providing basic information, such as name, email, and contact details.
* The system should ensure secure storage and handling of user credentials and personal information in MySQL.

1. Browse and Search:

* Users should be able to browse through a variety of products listed by sellers within the Wachemo community, such as textbooks or electronics.
* Users should be able to search for products using categories like “Books,” “Clothing,” or “Gadgets.”

1. Seller Profiles:

* Each seller should have a dedicated profile page displaying their listed products, transaction history, and mode status (buyer/seller).
* Users should be able to view product details from sellers, including price, description, and availability status (e.g., sold out).
* The system should indicate whether a product is still available for purchase.

1. Ordering Process:

* Users in buyer mode should be able to select products, add them to a cart, and modify their selections (e.g., quantity) before finalizing the order.
* The system should calculate the total order cost, including any applicable fees or taxes.
* Users should be able to place orders directly through the web interface.

1. Secure Payment:

* The system should offer multiple payment options via an app wallet, including Telebirr, CBE, Amole, and M-Pesa.
* Users should be able to deposit and withdraw money from their wallet using these Ethiopian payment methods.
* The system should encrypt and securely store payment information, generating transaction records in MySQL.

1. Product Management (Seller Mode):

* Sellers should be able to add new products to the web app, providing details like name, price, and category.
* Sellers should have the option to delete their product listings when no longer available.
* The system should allow sellers to view the status of their products (e.g., sold out) and update records accordingly.

1. Transaction Tracking:

* Users should be able to view their transaction history, including past purchases and sales, through the web app.
* The system should update transaction records in MySQL after each completed order or sale.

1. Account Management:

* Users should be able to manage their account settings, updating personal information, email, and payment methods.
* The system should allow users to toggle between buyer and seller modes seamlessly via the web interface.

1. System Administration:

* The system should include an administration portal to oversee user accounts, product listings, and transaction records.
* The system should validate emails, credentials, mode switches, process payments, and mark products as sold out automatically.

**Non-Functional Requirements:**

* Security: User data and transactions should be protected with encryption in MySQL.
* Security: Login sessions should be safeguarded against unauthorized access.
* Performance: Web pages should load within 3 seconds, even on university Wi-Fi.
* Performance: The system should support up to 100 simultaneous users without performance issues.
* Simplicity: The web interface should be intuitive for all users, regardless of technical skill.
* Simplicity: Key actions (e.g., adding to cart, toggling modes) should require minimal steps.
* Reliability: The system should maintain 99% uptime for consistent availability.
* Reliability: Errors (e.g., payment failures) should be handled with clear user feedback.
* Scalability: The system should accommodate an increasing number of Wachemo users over time.

## Use case diagram components

The use case diagram for the web-based e-commerce application serves as a structured representation of the system’s interactions with its users and external entities. Designed to facilitate buying and selling within the Wachemo University community, the system operates through a web browser interface and leverages MySQL for data storage. The diagram delineates the key actors, their actions, and the boundaries of the system, providing a clear framework for understanding its functional scope and behavior.

The components of the use case diagram are defined as follows:

1. **Actors:**

* Wachemo User: Represents any individual from the Wachemo University community (students, staff, or other members) who engages with the system to buy or sell products. This actor can switch between buyer and seller roles.
* Payment System: An external service (e.g., Telebirr, CBE, Amole, M-Pesa) responsible for processing financial transactions related to wallet deposits, withdrawals, and purchases.
* Admin: An entity tasked with managing user accounts, product listings, and system operations through a dedicated web portal.

1. **Use Cases:**

* Register Account: The functionality enabling a Wachemo User to create a profile and gain access to the system.
* Toggle Mode: The action allowing a Wachemo User to switch between buyer and seller roles within the web interface.
* Manage Products: The capability for a seller to add or delete product listings in the system.
* Shop Online: The process encompassing browsing products, searching by category, adding items to a cart, and placing orders.
* Process Transaction: The operation of handling payments and updating transaction records in MySQL.
* Monitor Activity: The feature permitting users to view transaction history or product status details.

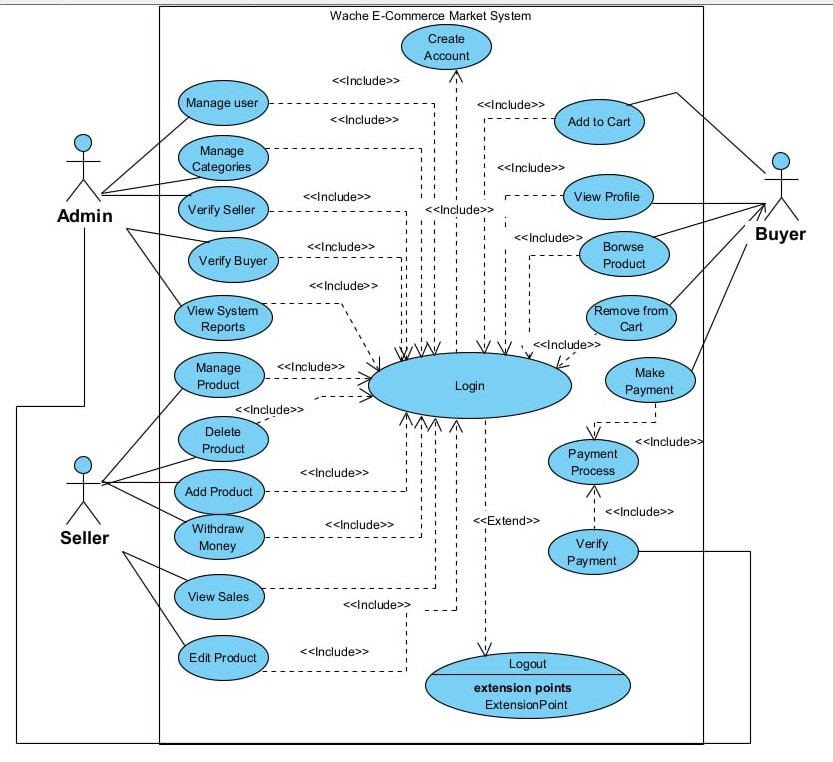
1. **Associations:**

* Wachemo User to Register Account: Indicates that Wachemo Users initiate account creation.
* Wachemo User to Toggle Mode: Reflects the ability of Wachemo Users to alternate roles.
* Wachemo User to Manage Products: Connects sellers to product management functions.
* Wachemo User to Shop Online: Links buyers to the shopping process.
* Wachemo User and Payment System to Process Transaction: Denotes the collaboration between users and the external payment service for financial operations.
* Admin to Monitor Activity: Specifies the Admin’s role in overseeing system activities.

1. System Boundary:

* Wachemo E-Commerce Web App: Encompasses all use cases (Register Account, Toggle Mode, Manage Products, Shop Online, Process Transaction, Monitor Activity) and the Wachemo User and Admin actors within a defined scope. The Payment System remains external, interfacing with the system through a payment gateway. This boundary clarifies the extent of the application’s functionality and distinguishes internal operations from external dependencies.

## Example of use case mode



## Use case Description/template

1. General use case: Create Account

|  |  |
| --- | --- |
| **Item** | **Description** |
| **Use Case Name** | Create Account |
| **Actor(s)** | Admin, Seller, Buyer |
| **Description** | Allows new users (Admin, Seller, Buyer) to create an account on the system. |
| **Precondition** | User has access to the registration page. |
| **Postcondition** | User account is successfully created and ready for login. |
| **Normal Flow** | 1. User provides registration details.2. System saves details.3. Account is created. |
| **Alternative Flow** | 1a. Invalid or missing data → system shows error.2a. User retries. |
| **Exceptions** | System error, network failure. |
| **Includes** | None |
| **Extends** | None |
| **Priority** | High |
| **Frequency of Use** | One-time per user. |

1. General Use Case: Login

|  |  |
| --- | --- |
| **Item** | **Description** |
| **Use Case Name** | Login |
| **Actor(s)** | Admin, Seller, Buyer |
| **Description** | Allows users to log in and access their dashboard and system functions. |
| **Precondition** | User has a valid account. |
| **Postcondition** | User is authenticated and logged in. |
| **Normal Flow** | 1. User enters credentials.2. System verifies.3. User is logged in. |
| **Alternative Flow** | 1a. Invalid credentials → error message.2a. Retry login. |
| **Exceptions** | Account locked, system error. |
| **Includes** | All use cases that require authentication. |
| **Extends** | Logout |
| **Priority** | High |
| **Frequency of Use** | Every session. |

1. Buyer Use Cases

|  |  |
| --- | --- |
| **Use Case Name** | **Browse Product / Add to Cart / Remove from Cart / Make Payment / View Profile** |
| **Actor(s)** | Buyer |
| **Description** | Buyer can browse products, manage their cart, make payments, and view profile. |
| **Precondition** | Buyer is logged in. |
| **Postcondition** | Actions completed (items added/removed, payment processed, profile viewed). |
| **Normal Flow** | 1. Buyer browses products.2. Adds items to cart.3. Makes payment.4. Views/updates profile. |
| **Alternative Flow** | 1a. Remove items from cart.2a. Cancel payment process. |
| **Exceptions** | Payment error, product out of stock, system error. |
| **Includes** | Payment Process, Verify Payment |
| **Extends** | Logout |
| **Priority** | High |
| **Frequency of Use** | Frequent per shopping session. |

1. Seller Use Cases

|  |  |
| --- | --- |
| **Use Case Name** | **Add Product / Edit Product / Delete Product / View Sales / Withdraw Money** |
| **Actor(s)** | Seller |
| **Description** | Seller manages product listings, views sales, and withdraws earnings. |
| **Precondition** | Seller is logged in and verified. |
| **Postcondition** | Product/sales actions completed, money withdrawn if requested. |
| **Normal Flow** | 1. Add new product.2. Edit or delete existing product.3. View sales.4. Withdraw money. |
| **Alternative Flow** | 1a. Edit fails due to invalid data.2a. Withdrawal fails due to insufficient funds. |
| **Exceptions** | System error, account not verified, product constraints. |
| **Includes** | Login |
| **Extends** | Logout |
| **Priority** | Medium to High |
| **Frequency of Use** | Regular, depending on business activity. |

1. Admin Use Cases

|  |  |
| --- | --- |
| **Use Case Name** | **Manage User / Verify Buyer / Verify Seller / Manage Categories / Manage Product / View System Reports** |
| **Actor(s)** | Admin |
| **Description** | Admin oversees users, product management, verification, and system reporting. |
| **Precondition** | Admin is logged in. |
| **Postcondition** | User, product, and system management tasks completed successfully. |
| **Normal Flow** | 1. Manage users.2. Verify buyers/sellers.3. Manage categories/products.4. View reports. |
| **Alternative Flow** | 1a. Verification fails due to missing info.2a. System reports error. |
| **Exceptions** | System restrictions, insufficient admin rights, network issues. |
| **Includes** | Login |
| **Extends** | Logout |
| **Priority** | High |
| **Frequency of Use** | Regular for system administration. |

## Tools and steps to draw Use Case

For drawing the use case diagram, we used **Visual Paradigm**, a user-friendly UML modeling tool that helps create clear and standardized diagrams.

The steps followed were:

* Open Visual Paradigm and create a new project.
* Select **Use Case Diagram** from the available diagram types.
* Add the main actors (Admin, Seller, Buyer) using the actor symbols.
* Insert the use cases (e.g., Login, Create Account, Browse Product) using ellipse shapes.
* Draw association lines between actors and use cases.
* Apply the **<<include>>** and **<<extend>>** relationships where needed.
* Arrange the diagram layout neatly for readability.
* Save and export the diagram as an image for documentation.

# CHAPTER TWO

## 2.1 High-Level Sequence Diagram

In this section, we present the high-level sequence diagram for the Wache E-Commerce Market System. This diagram shows the interactions between key system components (such as Buyer, Seller, Admin) for major use cases like product browsing, adding to cart, managing products, and completing payments.

The purpose of the high-level sequence diagram is to capture the overall flow of messages and interactions between actors and system components, without diving into detailed class-level interactions.

## ****2.2 Components of High-Level Sequence Diagram****

A high-level sequence diagram is made up of several key components that together describe how actors and system parts interact over time to accomplish a specific process or use case. Below are the main components typically found in a high-level sequence diagram:

* **Actors:** External entities that interact with the system, such as users (Buyer, Seller, Admin) or external services. They initiate or respond to system messages.
* **System Components (Objects):** The main parts of the system that handle specific tasks, such as the Product Catalog, Cart, Payment Gateway, User Account, or Order Management module.
* **Lifelines:** Vertical dashed lines extending from each actor or system component. These represent the time dimension, showing that the actor or component exists and can send or receive messages during the interaction.
* **Messages:** Horizontal arrows that represent communication between lifelines. Messages can be:
  + **Synchronous** (with a waiting reply, like a database query)
  + **Asynchronous** (without waiting, like sending a notification) These show the sequence of operations, such as “add product to cart” or “process payment.”
* **Activation Bars (Execution Occurrences):** Thin vertical rectangles drawn on lifelines, showing when a component is actively doing something (executing an operation or waiting for a reply).
* **Return Messages:** Dashed arrows showing the response or result of an earlier message, such as a confirmation or error.
* **Sequence Flow:** The top-to-bottom order of the diagram, where interactions are arranged chronologically, helping viewers understand the process flow over time.

These components together provide a clear picture of how various parts of the system work together to fulfill a high-level process, allowing stakeholders to trace the interaction steps and understand system behavior.