A Graduation Project entitled:

TAGERLY

presented by:

Mahmoud Fathi Rayid

mahmoudfathymf576041@gmail.com

Haneen Ali Elagamy

haneenelagamy21@gmail.com

Nour Tamer Abdelhaleem

nouryounis666@gmail.com

Hana Gamal Abu ElYazeed

hanagamal989@gmail.com

Shimaa Amr Mahmoud

shimaaamr712@gmail.com

Under the Supervisors of:

Eng: Abdelrahman Shaaban

1.1 Project Proposal

1.1.1 Overview

Tagerly is an online marketplace designed for handmade entrepreneurs, enabling artisans and creators to showcase and sell their unique products effortlessly. The platform provides advanced features such as store management, multiple payment options, and order tracking, supporting sellers in reaching a broader audience and growing their businesses.

1.1.2 Objectives

- Create a user-friendly online marketplace for handmade product sellers.
- Provide a seamless buying experience for customers.
- Implement secure payment methods and order tracking.
- Enable marketing and engagement tools for sellers.
- Ensure platform scalability and reliability

1.1.3 Scope

- Development of buyer, seller, and admin dashboards.
- Product and store management functionalities.
- Integration of multiple payment gateways.
- Implementation of smart search and filtering.
- Shipping and order tracking system.
- Customer review and support features.

1.2 Project Plan

1.2.1 Timeline (Gantt Chart)

Phase	Task	Duration
Phase 1	Requirement Gathering & Planning	1 Week
Phase 2	UI/UX Design	1 Week
Phase 3	Backend & Database Development	3 Week
Phase 4	Frontend Development	1 Week
Phase 5	Payment & Shipping Integration	1 Week
Phase 6	Testing & Debugging	1 Week

1.2.2 Milestones & Deliverables

- Requirement Documentation (End of Phase 1)
- Prototype Design (End of Phase 2)
- Functional Backend API (End of Phase 3)
- Fully Integrated Frontend (End of Phase 4)
- Working Payment & Order System (End of Phase 5)
- Final Testing & QA Reports (End of Phase 6)

1.3 Task Assignment & Roles

Role	Responsibility
Project Manager	Oversee progress, ensure deadlines are met.
Backend Developer Frontend Developer	Develop API, database, and business login Implement UI components and user interactions.
UI/UX Designer	Design intuitive interfaces and user flows.

1.4 Risk Assessment & Mitigation Plan

Risk	Impact	Mitigation Strategy
Delays in development	High	Regular progress tracking, buffer time in schedule
Security vulnerabilities	High	Implement secure coding practices, regular audits
Payment gateway failures	Medium	Use multiple payment providers
Poor user adoption	Medium	Marketing campaigns, user-friendly design

1.5 Key Performance Indicators (KPIs)

KPI	**Description**
System Uptime	Ensuring 99% uptime of the platform.
Response Time	API response time < 500ms
User Adoption Rate	Growth in new user registrations.
Order Completion Rate	Percentage of successful transactions.
Customer Satisfaction	Measured via reviews and feedback.

2. Requirements Gathering

2.1 Stakeholder Analysis:

Identifying key stakeholders and their needs:

- Buyers – Require an easy-to-use interface, secure payments, and reliable order tracking.
- Sellers – Need a store management system and marketing tool

  (create a coupon)
- Administrators – Require full control over platform operations, user management.

2.2 User Stories & Use Cases

Buyer User Stories

- As a buyer, I want to search for handmade products easily so that I can find what I need quickly.
- As a buyer, I want to securely complete my payment so that I feel safe making transactions.
- As a buyer, I want to track my orders so that I know when they will arrive.

Seller User Stories

- As a seller, I want to manage my products and inventory so that I can keep my store updated.
- As a seller, I want to analyze my sales performance so that I can optimize my business**.**

Admin User Stories

- As an admin, I want to manage user accounts so that I can maintain platform integrity.
- As an admin, I want to resolve disputes so that I can ensure smooth transactions.

2.3 Functional Requirements

- User authentication and registration.
- Product listing, categorization, and management.
- Shopping cart and checkout system.
- Payment gateway integration (Vodafone Cash, PayPal).
- Order management and tracking system.
- Admin dashboard for user, order, and dispute management**.**

2.4 Non-functional Requirements

• **Reliability**: The system should be available all time.

• **Usability**: System should be easy to use by having a simple graphical user interface.

• **Flexibility**: The system should be flexible enough to accommodate evolving data.

• **Accuracy:** Accurate results for clothing fitting on customers

---

3. System Analysis & Design

3.1 Problem Statement & Objectives

3.1.1 Use Case Diagram & Descriptions

Actors:

1. **Buyer** (End-user purchasing products)
2. **Seller** (User managing a store and products)

3. **Admin** (System administrator overseeing operations)

Use Cases & Interactions:

1. Buyer Use Cases:

- **Browse Products** → Buyers can view available products in the system.
- **Add to Cart** → Allows buyers to add selected products to their cart.
- **Checkout & Payment** (<include>) → Included within the buying process.
- **Rate & Review Products** → Buyers can leave feedback for purchased products.
- **Request Refund** (<extend> Track Orders) → Buyers may request refunds, which depends on tracking the order status.
- **Track Orders** → Buyers can monitor their orders' status.
- **Sign Up/Login** → Required for buyers to access system features.

2. Seller Use Cases:

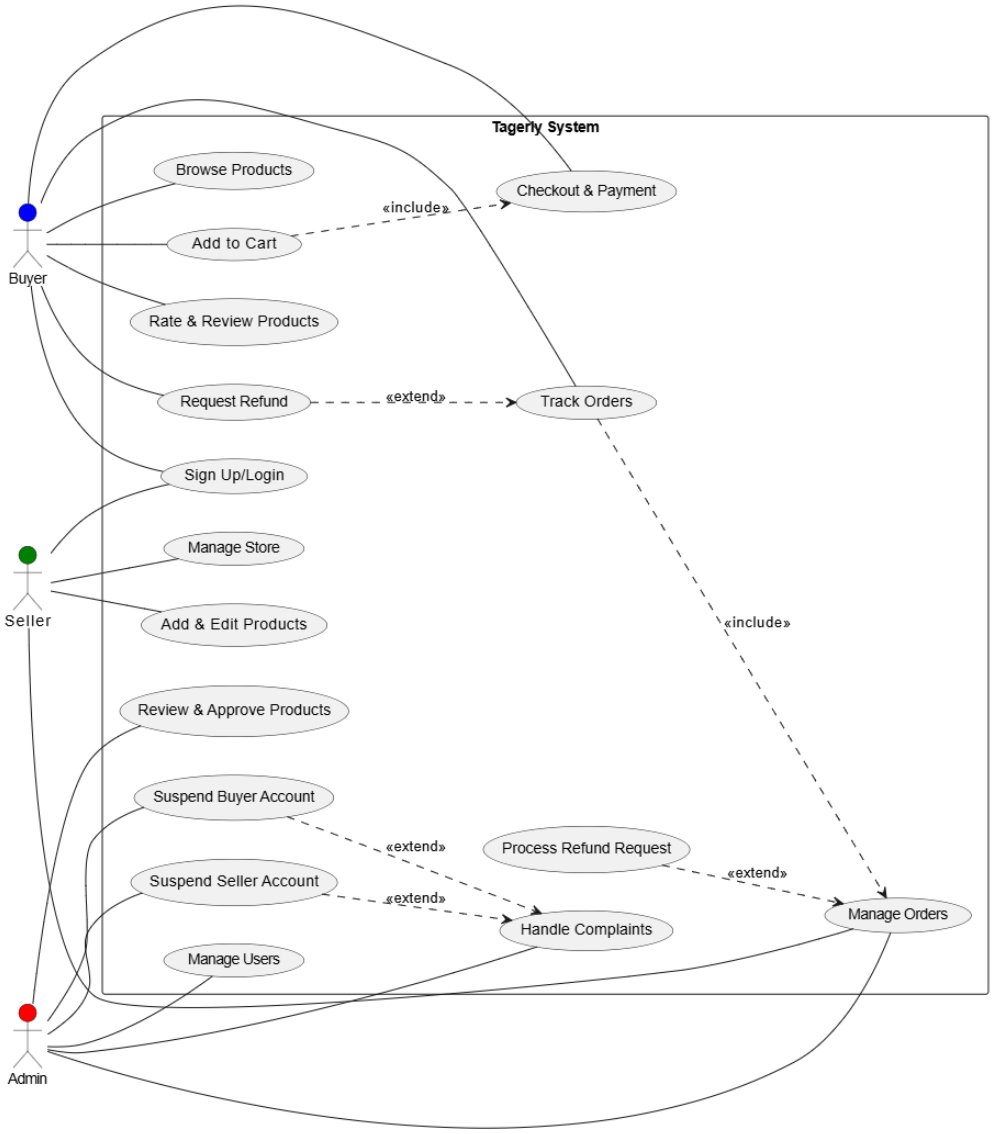
- **Manage Store** → Sellers can update their store details.
- **Add & Edit Products** → Sellers can list new products and modify existing ones.
- **Review & Approve Products** → Sellers manage their product listings.

3. Admin Use Cases:

- **Manage Users** → Admins oversee user accounts.
- **Manage Orders** → Admins track and handle system orders.
- **Handle Complaints** (<extend > Manage Orders) → Admins resolve customer disputes.
- **Process Refund Requests** (<extend > Handle Complaints) → Admins process buyer refund requests.
- **Suspend Buyer Account** (<extend > Handle Complaints) → Admins can take action against violating buyers.
- **Suspend Seller Account** (<extend > Handle Complaints) → Admins can take action against violating sellers.

Key Relationships:

- **<Include> Relationship:**
    - *"Checkout & Payment"* is a core part of the **Add to Cart** process.
    - *"Manage Orders"* includes **Order Tracking** and Admin processes.
- **<Extend> Relationship:**
    - *"Request Refund"* extends **Track Orders**.
    - *"Handle Complaints"* extends **Manage Orders**.
    - *"Suspend Buyer/Seller Account"* extends **Handle Complaints**.



- **Software Architecture:**

---

1. High-Level Design

1.1 Overview

This section describes the core system components, their interactions, and the architecture style used.

1.2 System Components

- **User Interface (UI): Interacts with users and processes inputs.**
- **Business Logic Layer: Contains business rules and coordinates operations.**
- **Data Layer: Manages storage and retrieval operations from the database.**
- **External Services: Includes APIs and third-party services.**

1.3 Component Interactions

- **Requests are passed from the user interface to the business logic layer.**
- **The business layer interacts with the data layer to fetch necessary information.**
- **Results are sent back to the user through the graphical interface.**

---

2. Architecture Styles

2.1 MVC (Model-View-Controller)

- **Model: Manages data and business rules.**
- **View: Displays data to users.**
- **Controller: Processes inputs and coordinates between model and view.**
- **Commonly used in web applications like ASP.NET MVC.**

2.2 Microservices Architecture

- **The system is divided into independent services that interact via APIs.**
- **Each service is responsible for a specific function and can be developed and deployed independently.**
- **Commonly used in large cloud-based systems like Azure Microservices.**

2.3 Layered Architecture

- **The system is divided into multiple layers such as:**
  - **Presentation Layer**
  - **Business Logic Layer**
  - **Data Access Layer**
- **Commonly used in traditional .NET applications.**

2.4 Event-Driven Architecture

- **Relies on sending and receiving events instead of direct requests.**
- **Uses message brokers like Kafka and RabbitMQ for asynchronous event processing.**
- **Commonly used in real-time processing systems.**

---

3. Architectural Diagrams

3.1 UML Diagrams

- **Component Diagram illustrates how different parts of the system interact.**

- **Layer Diagram shows how the system is structured into layers.**

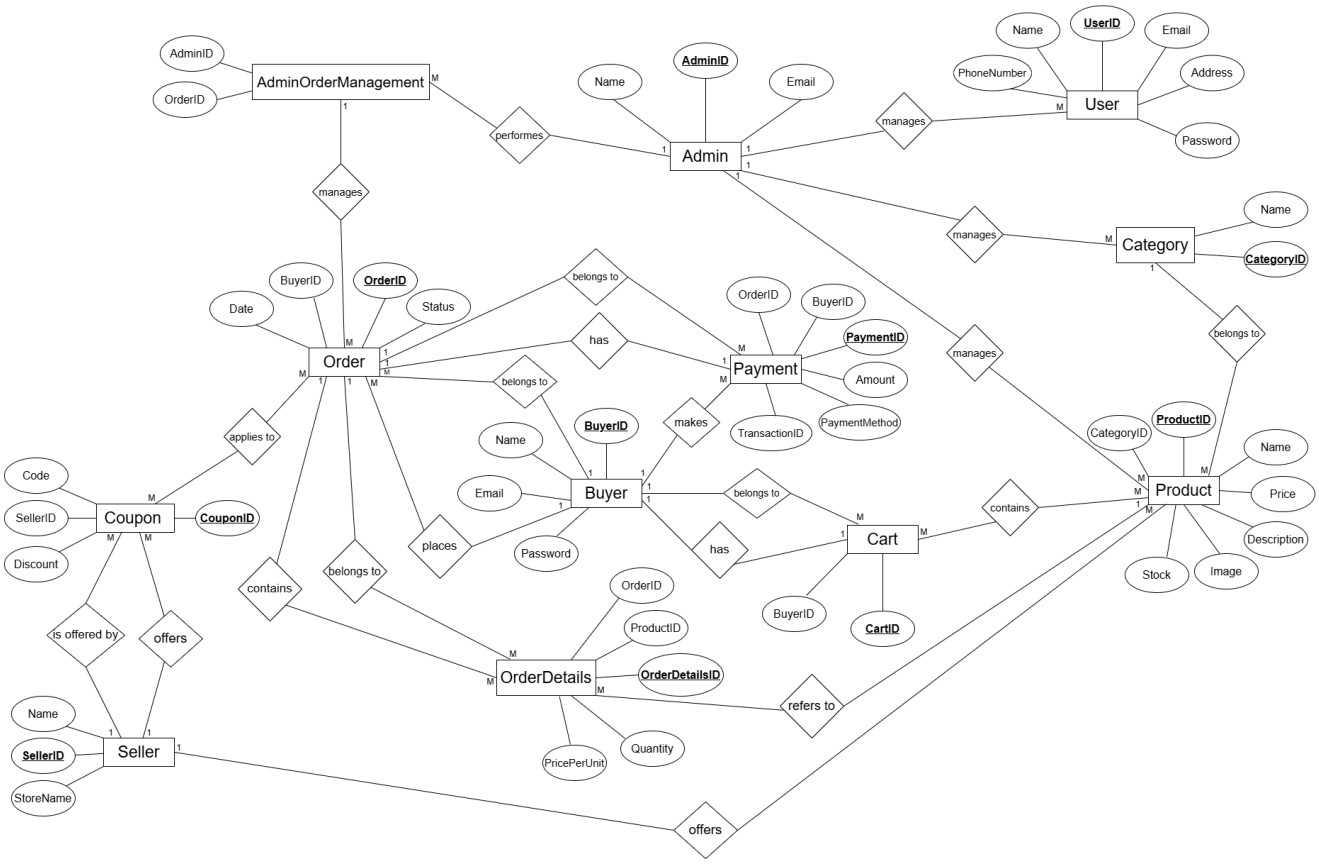
3.2 C4 Model

- **Level 1 (System Context Diagram): Shows the relationship between the system and external users.**
- **Level 2 (Container Diagram): Displays the internal components and their interactions.**
- **Level 3 (Component Diagram): Details individual components.**

3.2 Database Design & Data Modeling

3.2.1 ER Diagram (Entity-Relationship Diagram)

The following diagram represents the structure of the Tagerly database using the Relational Schema format. Entities are shown with their attributes, including primary keys (bold and underlined), foreign keys, and clearly labeled relationships with cardinality notations, specifically 1 to M and M to M.



3.2.2 Logical & Physical Schema

1. Logical Schema

This schema includes detailed descriptions of each table, attributes, primary and foreign keys, and relationships. Each table corresponds to a real-world entity involved in the e-commerce platform.

Example: Buyer Table

Attribute	Type	Description
BuyerID (PK)	INT	Unique identifier for each buyer
Name	NVARCHAR(100)	Full name of the buyer
Email	NVARCHAR(100)	Email address
Password	NVARCHAR(255)	Encrypted login password

2. Physical Schema

Below is a sample SQL CREATE TABLE statement showing data types, primary and foreign key constraints.

Example SQL for 'Buyer' table:

```
CREATE TABLE Buyer (
  BuyerID INT PRIMARY KEY,
  Name NVARCHAR(100),
  Email NVARCHAR(100) UNIQUE,
  Password NVARCHAR(255)
);
```

3. Normalization Considerations

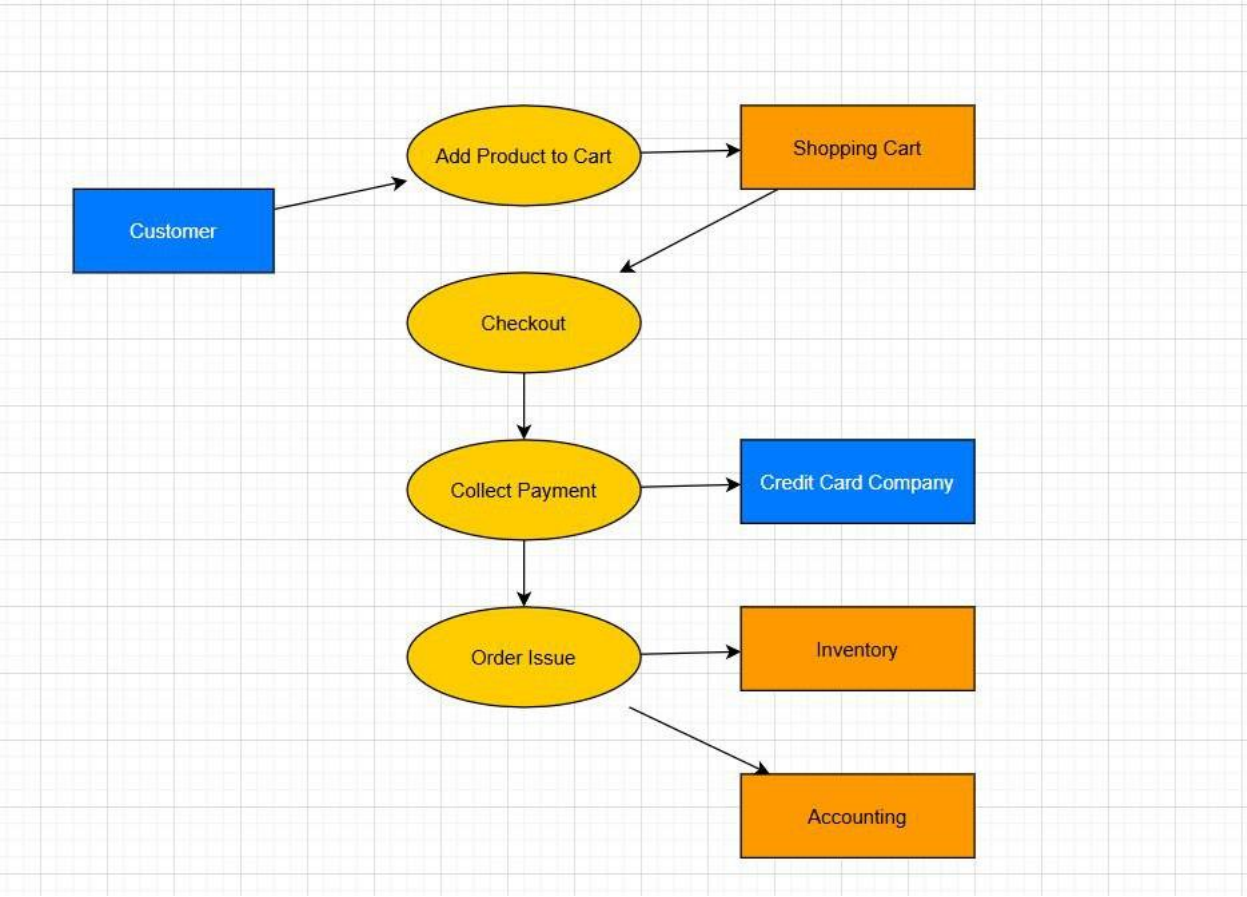
The database schema is designed to meet the requirements of Third Normal Form (3NF):

- 1NF: All attributes are atomic and there are no repeating groups.
- 2NF: All non-key attributes are fully functionally dependent on the primary key.

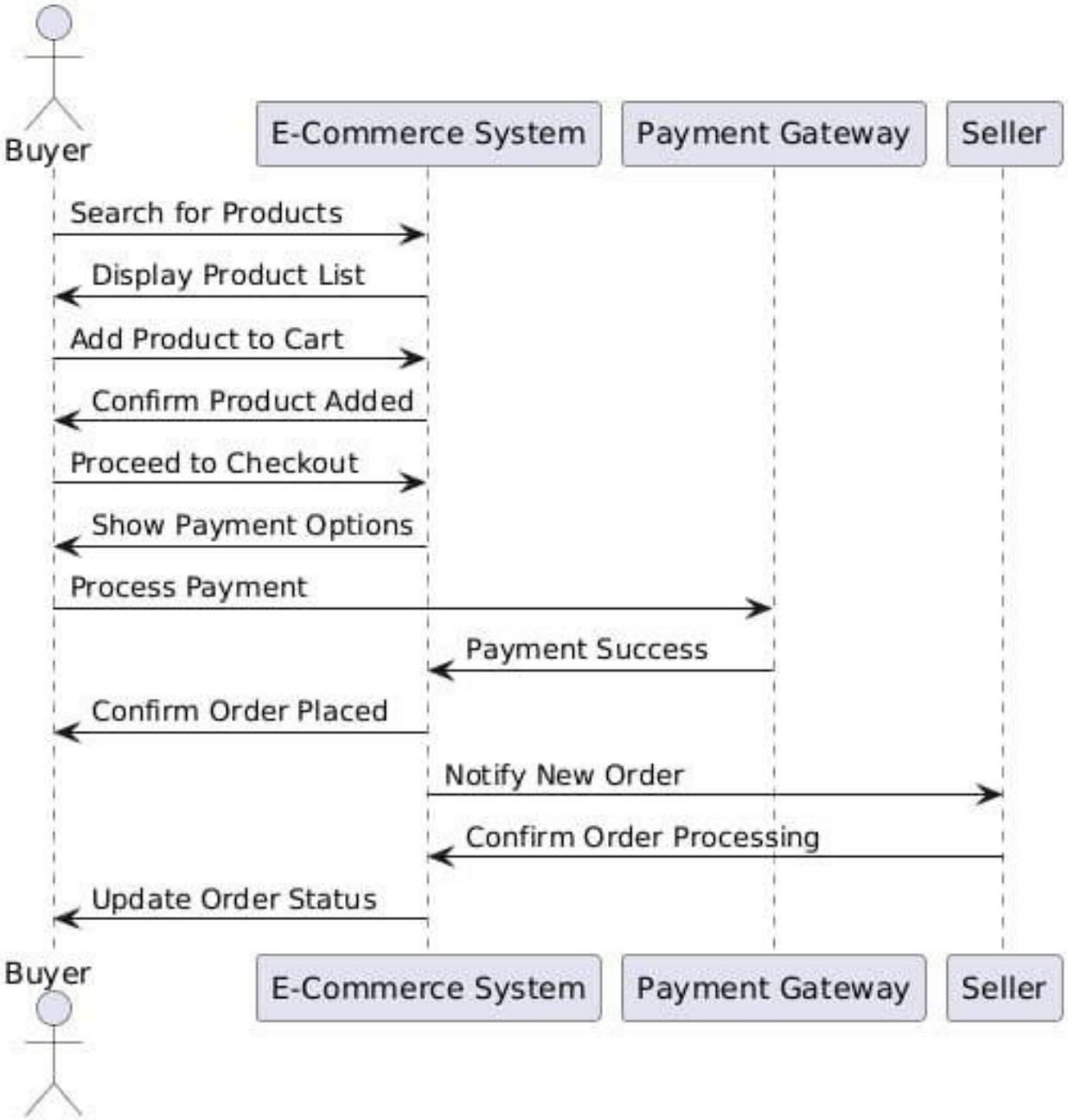
- 3NF: No transitive dependencies exist between non-key attributes.

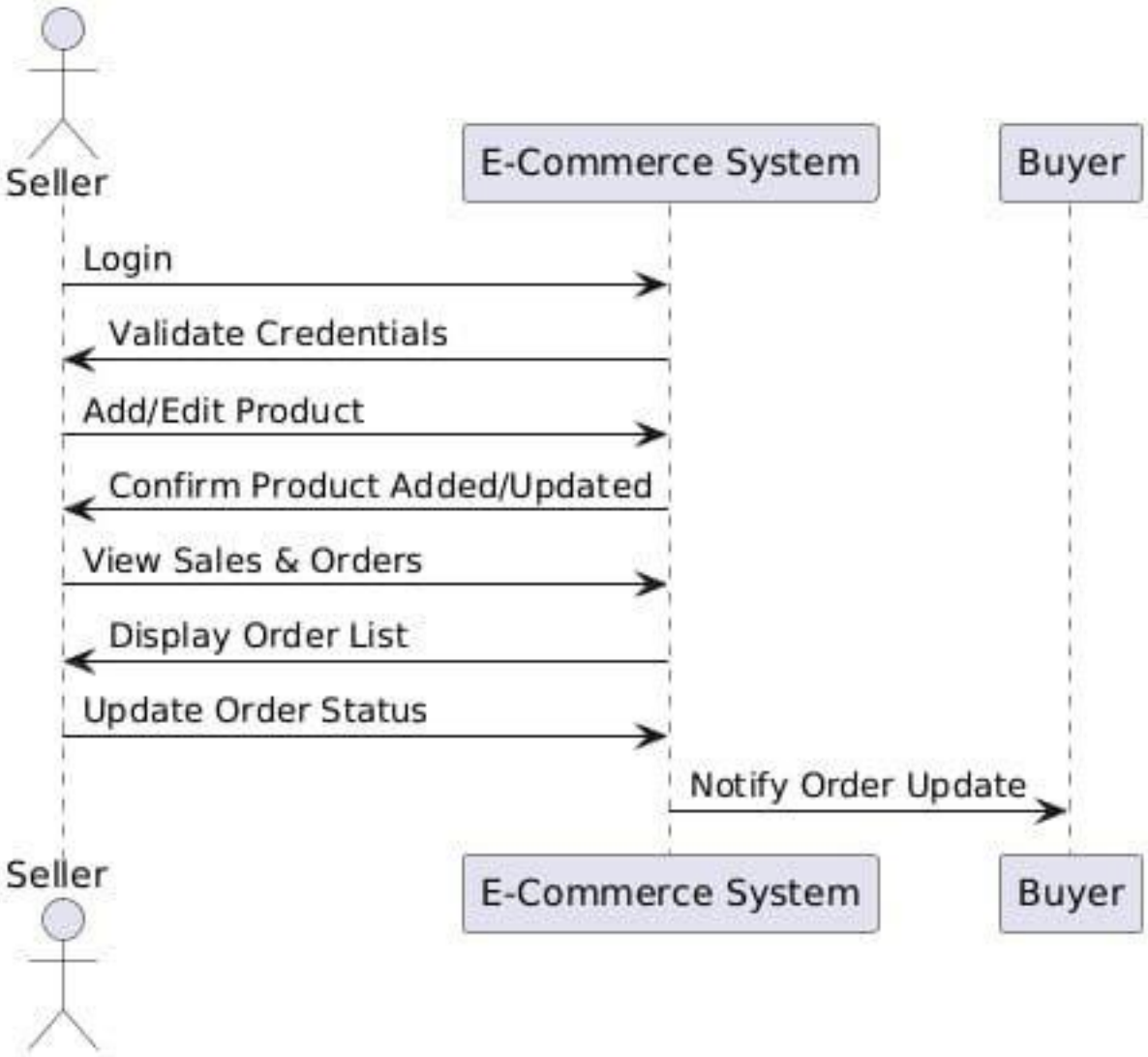
3. Data Flow & System Behavior

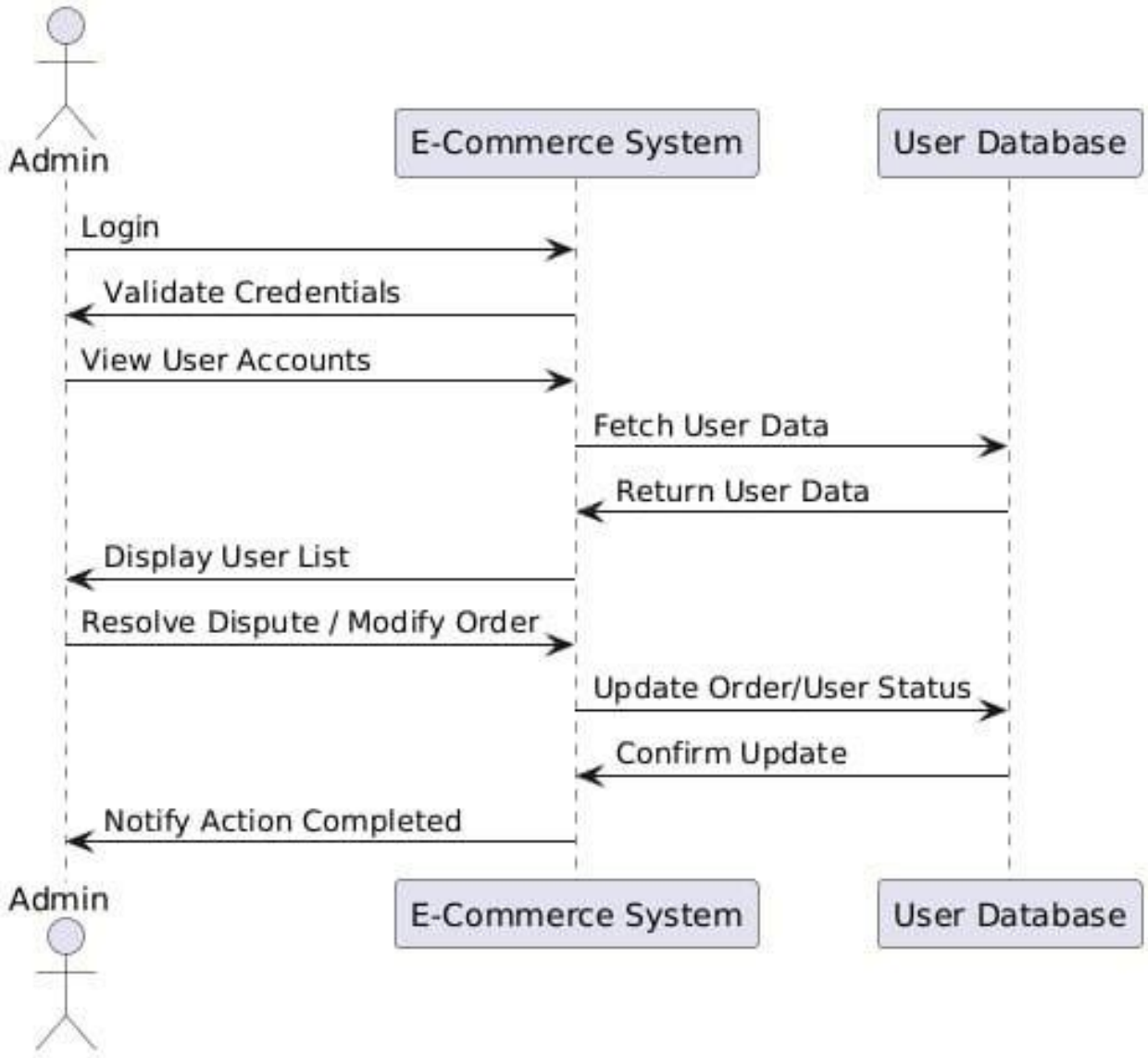
3.1 DFD (Data Flow Diagram)



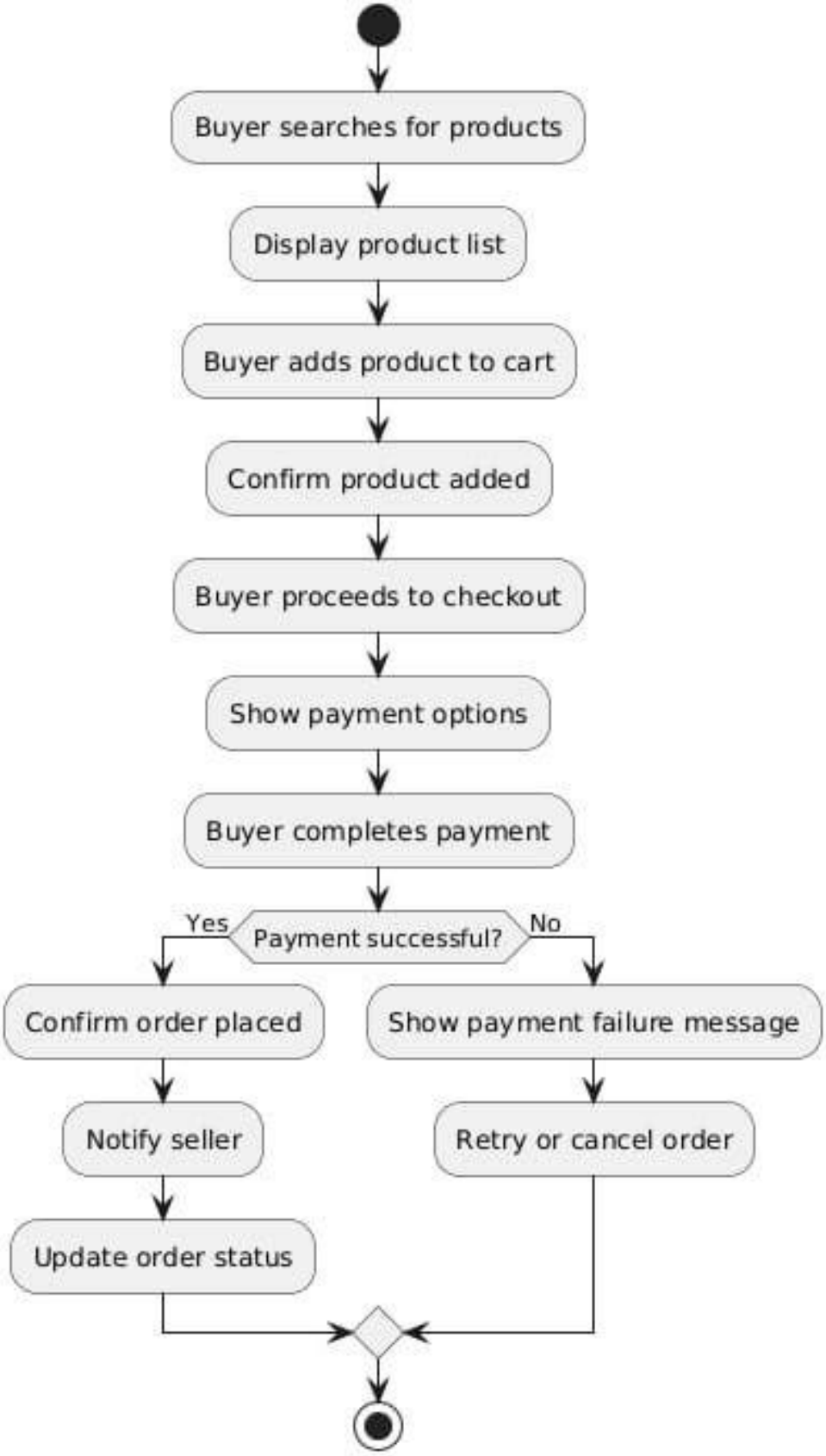
3.2 Sequence Diagrams

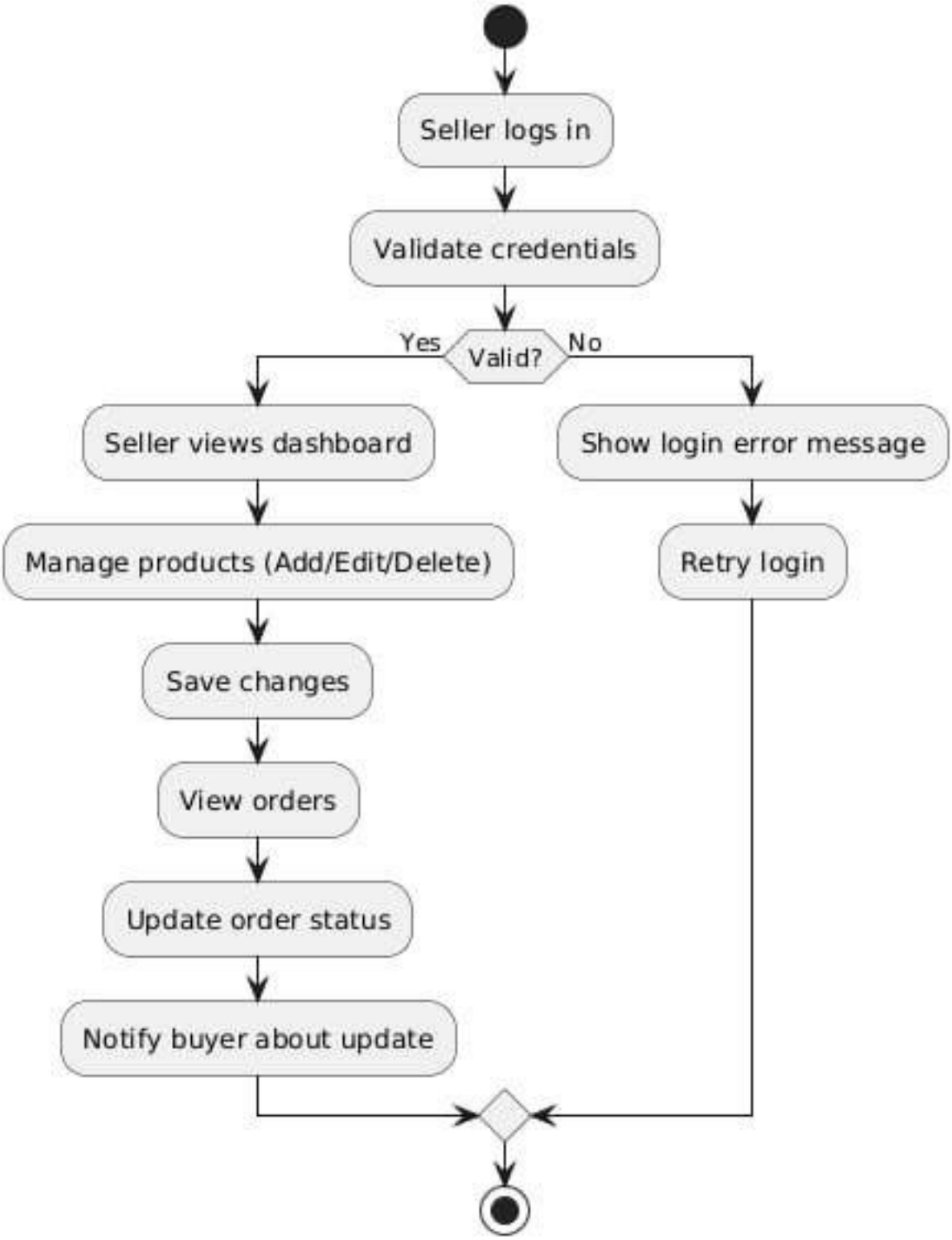



Seller
E-Commerce System
Buyer
Login
Validate Credentials
Add/Edit Product
Confirm Product Added/Updated
View Sales & Orders
Display Order List
Update Order Status
Notify Order Update
Seller
E-Commerce System
Buyer

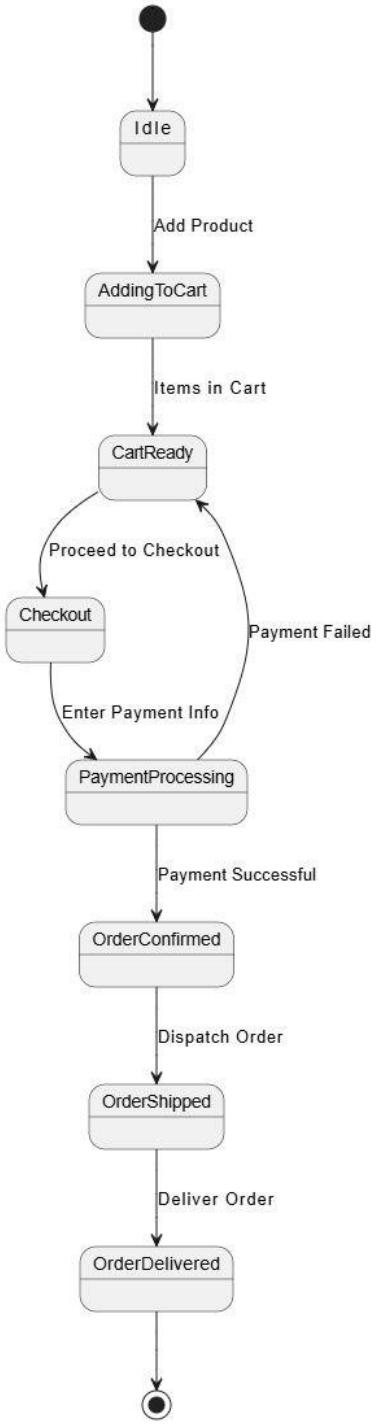


3.3 Activity Diagram

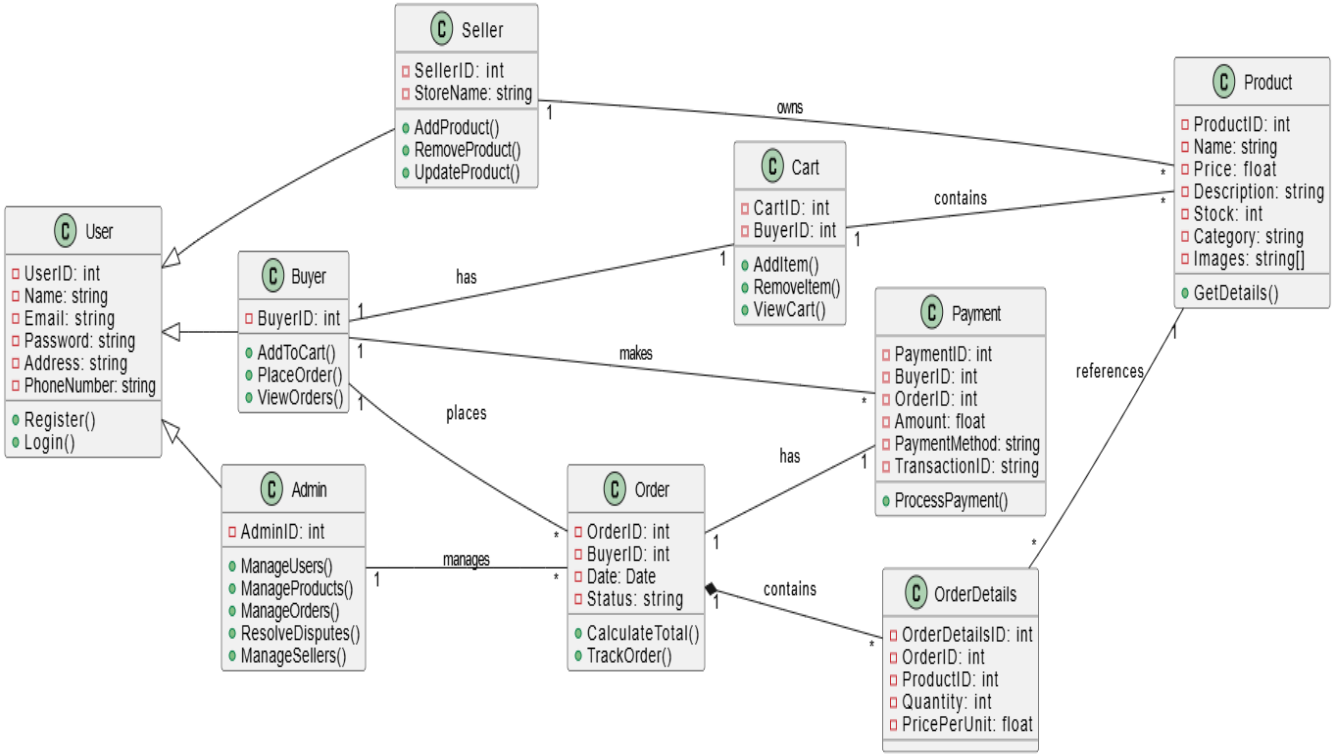

Buyer searches for products
Display product list
Buyer adds product to cart
Confirm product added
Buyer proceeds to checkout
Show payment options
Buyer completes payment
Payment successful?
Yes
No
Confirm order placed
Notify seller
Update order status
Show payment failure message
Retry or cancel order


Seller logs in
Validate credentials
Valid?
Yes
No
Seller views dashboard
Show login error message
Manage products (Add/Edit/Delete)
Retry login
Save changes
View orders
Update order status
Notify buyer about update

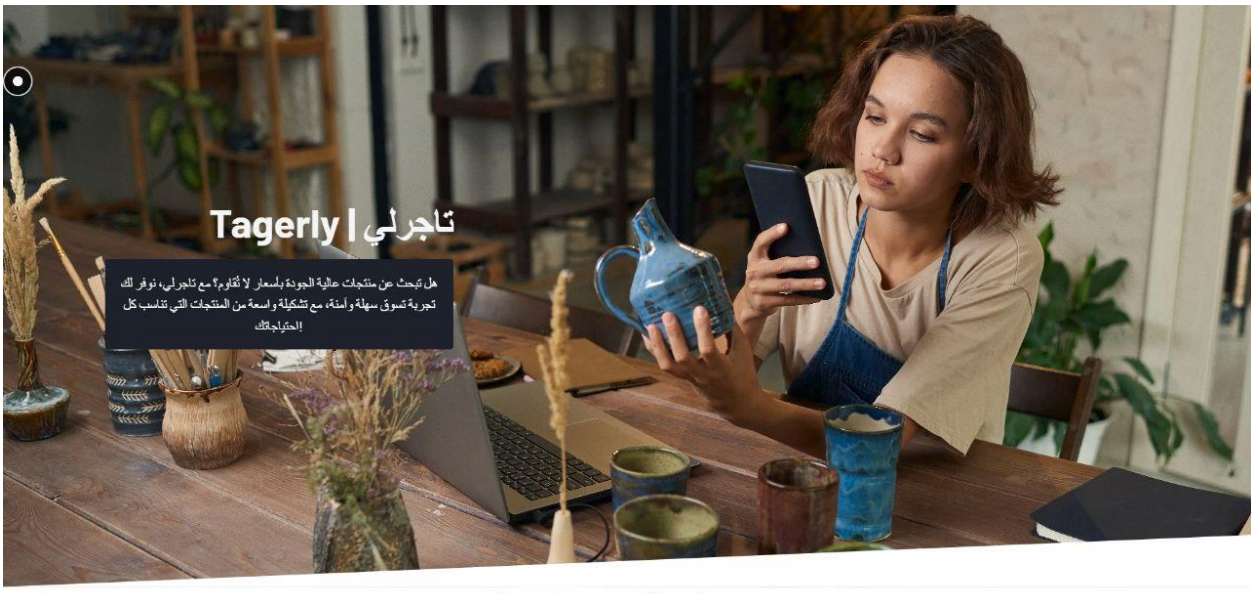
3.4 State Diagram



3.5 Class Diagram



4. UI/UX Design & Prototyping



Customer Reviews

About Us

نحن نؤمن بتقديم أفضل تجربة تسوق إلكتروني لعملائنا، مع منتجات عالية الجودة وخدمة متميزة. هدفنا هو تلبية احتياجاتك بسهولة وأمان

We strive to provide the best online shopping experience with high-quality products and excellent service. Our goal is to meet your needs efficiently and securely.