DESIGN PROPOSAL FOR AN E-COMMERCE MANAGEMENT SYSTEM

SWE5201 – ADVANCED PROGRAMMING

DONE BY:
LES PAUL RANALAN
(2230727)

TO BE SUBMITTED TO:
MARIA ABEDI

DATE OF SUBMISSION:
MARCH 26, 2024

TABLE OF CONTENTS

INTRODUCTION ---------- ***1***

AIMS & OBJECTIVES ---------- 1

UNIFIED MODELLING LANGUAGE (UML) ---------- 1

STRUCTURAL REQUIREMENTS ---------- ***2***

DATABASE SCHEMA ---------- 3

CLASS DIAGRAM ---------- 6

DATA ACCESS LAYER SPECIFICATION ---------- 9

SERVICE DESCRIPTION ---------- 11

BEHAVIORAL REQUIREMENTS ---------- ***12***

USE CASE DIAGRAM ---------- 13

SEQUENCE DIAGRAM ---------- 15

STATE TRANSITION DIAGRAM ---------- 17

ACTIVITY DIAGRAM ---------- 19

CONCLUSION ---------- ***21***

BIBLIOGRAPHY ---------- ***22***

TABLE OF FIGURES

Figure 1: Database design schema ---4

Figure 2: Class diagram--7

Figure 3: Use case diagram--- 14

Figure 4: Sequence diagram-- 16

Figure 5: State transition diagram -- 18

Figure 6: Activity diagram--- 20

INTRODUCTION

AIMS & OBJECTIVES

This report entails on the design and development of an e-commerce management system for a leading company.

The main objectives for this design proposal is to create a robust management system to **handle the company’s extensive product inventory**, **processing of orders**, **customer management**, and **company analytics**. The system also needs to be able to efficiently handle high volumes of data with it being **scalable**, **reliable**, and **maintainable**.

UNIFIED MODELLING LANGUAGE (UML)

It serves as a standardized graphical language for modeling software systems **(Visual Paradigm, 2019)**. UML provides a diverse range of notations and diagrams enabling software engineers to visually illustrate different facets of a system's architecture, design, and behavior. This facilitates clearer and more concise communication among stakeholders and software engineers within the software engineering realm.

Such diagarms in UML are: use case diagram, class diagram, sequence diagram, state diagram, etc. All of which are present within this assignment.

STRUCTURAL REQUIREMENTS

Structural requirements in the context of designing an e-commerce management system refers to the architectural elements and data structures that form the foundation of the system **(Visual Paradigm, 2019)**. These requirements focuses on **how data is organized, stored, and accessed**.

Below are the database schema, class diagram, and data access layer specification for the propsed system.

(Next page)

DATABASE SCHEMA

It is a logical blueprint that defines the structure, organization, and relationships of data stored in a database **(IBM, n.d.)**. It encompasses the layout of tables, their attributes or columns, data types, constraints, and relationships between tables. Through database schema, it serves as a foundation for data storage and retrieval, this ensures consistency, integrity, and efficiency within the system.

Tables – entities or objects

Columns – attributes of entities

Constraints – rules and integrity

Relationship – connections between entities via foreign keys

Below is the data schema for the proposed system. There are currently 8 tables, all of which serve different purposes. Customer, Staff, Supplier, and Address tables are for customer management and storing people’s details. Order, OrderDetail, and Transaction tables are for storing order details, order processing, and company analytics. Lastly, Product table is for storing the company’s inventory.

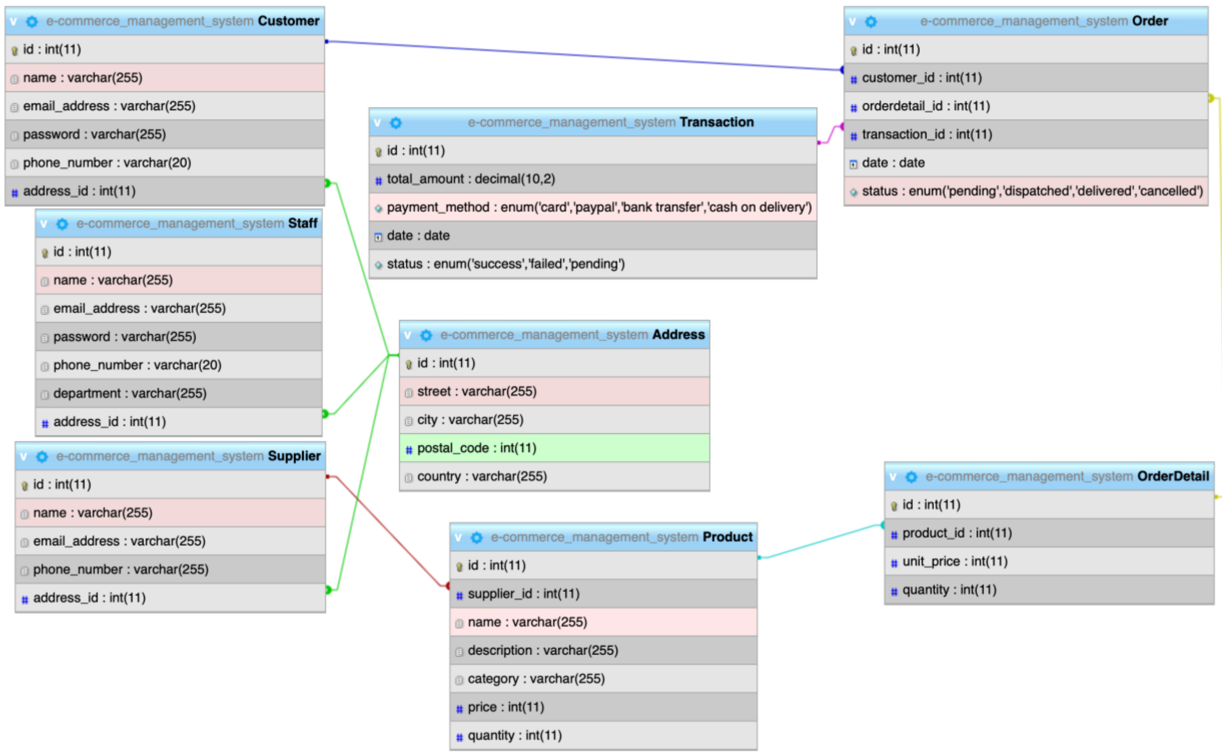
e-commerce_management_system Customer
id : int(11)
name : varchar(255)
email_address : varchar(255)
password : varchar(255)
phone_number : varchar(20)
address_id : int(11)
e-commerce_management_system Staff
id : int(11)
name : varchar(255)
email_address : varchar(255)
password : varchar(255)
phone_number : varchar(20)
department : varchar(255)
address_id : int(11)
e-commerce_management_system Supplier
id : int(11)
name : varchar(255)
email_address : varchar(255)
phone_number : varchar(255)
address_id : int(11)
e-commerce_management_system Transaction
id : int(11)
total_amount : decimal(10,2)
payment_method : enum('card','paypal','bank transfer','cash on delivery')
date : date
status : enum('success','failed','pending')
e-commerce_management_system Address
id : int(11)
street : varchar(255)
city : varchar(255)
postal_code : int(11)
country : varchar(255)
e-commerce_management_system Product
id : int(11)
supplier_id : int(11)
name : varchar(255)
description : varchar(255)
category : varchar(255)
price : int(11)
quantity : int(11)
e-commerce_management_system Order
id : int(11)
customer_id : int(11)
orderdetail_id : int(11)
transaction_id : int(11)
date : date
status : enum('pending','dispatched','delivered','cancelled')
e-commerce_management_system OrderDetail
id : int(11)
product_id : int(11)
unit_price : int(11)
quantity : int(11)

Figure 1: Database design schema

The database features normalization, via the creation of separate tables and the usage of foreign keys to connect tables, as seen in screenshot below with the lines colored in red, blue, green, yellow, and pink. Normalization is important as to ensure data integrity and efficient retrievel of information.

Below are the attributes and its functions:

NOTE: some attributes are not explained as it is self-explanatory

- `***id***` – serves as a primary key of the current table
- `***address_id***` - serves as a foreign key of the current table to point to a record in the Address table. This attribute is present in Customer, Staff, and Supplier table
- `***customer_id***` – serves as a foreign key in Order table to point to a record in the Customer table
- `***orderdetail_id***` – serves as a foreign key in Order table to point to a record in the OrderDetail table
- `***transaction_id***` – serves as a foreign key in Order table to point to a record in the Transaction table
- `***status***` – serves as a field for the status of a particular order or transaction
- `***product_id***` – serves as a foreign key in OrderDetail table to point to a record in the OrderDetail table
- `***quantity***` – serves as a field for the quantity of products in an order
- `***supplier_id***` – serves a foreign key in the Product table to point to a record in the Supplier table
- `***payment_method***` – serves as a field for the payment method that was used in the transaction
- `***total_amount***` – serves as a field for the total amount accounting all prices of products in one order

CLASS DIAGRAM

It is used to visualize the static structure of a system by representing the classes, their attributes, emthods, and the relationships between them **(Visual Paradigm, 2019)**.

Classes are depicated as rectangles with three compartments:

- **Top compartment**: name of the class
- **Middle compartment**: lists of attributes
- **Bottom compartment**: lists of methods

Relationships between classes are illustrated as using lines connecting the classes:

- **Association** – bi-directional relationship
- **Aggregation** – whole-part relationship
- **Composition** – whole-part relationship where parts are dependent on the whole
- **Inheritnance** – relationship between superclass and its subclasses
- **Dependency** – one class relies on another class

Below is the class diagram for the proposed system. Similarly to database schema, there are 8 classes, all of which serve different purposes. Each class features their own set of attributes, along with their visibility, `+` being public and `-` being private. Class diagram shows the multiplicities of each class and how they interact with each other. For instance, **1 Customer can have 1 or many (1..*) Addresses**. Similarly, **1 Customer can make 1 or many (1..*) Orders**.

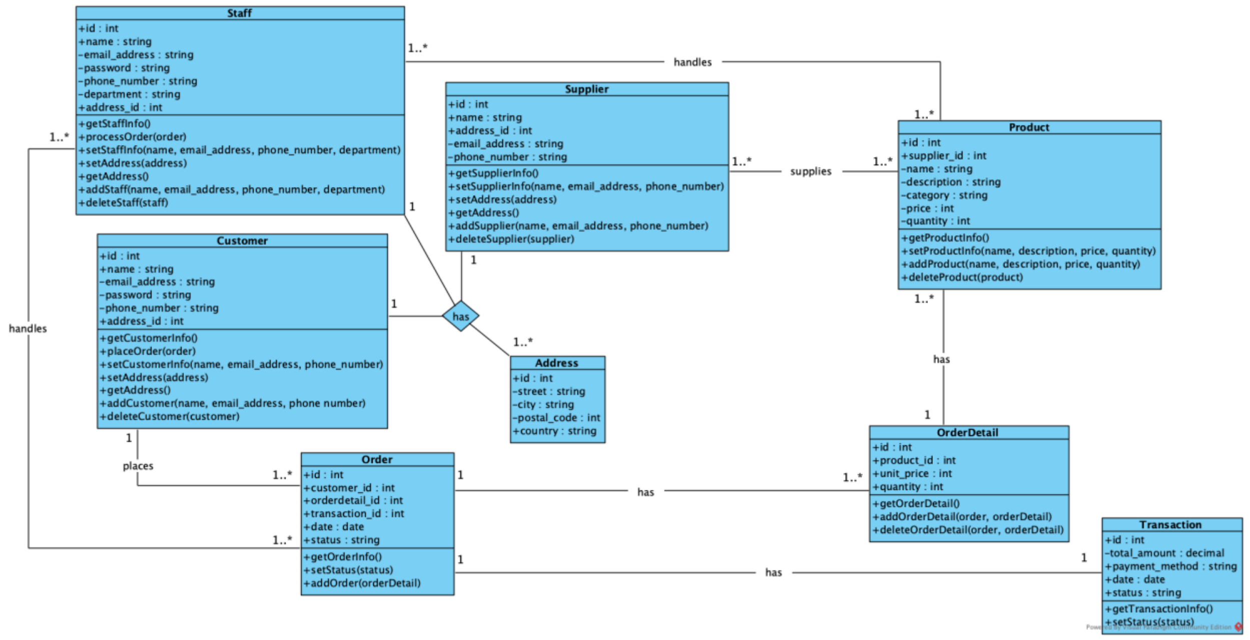

Figure 2: Class diagram

Below are the list of methods and their functions:

NOTE: “<Entity>” is a placeholder and it can be replaced with Customer, Staff, Supplier, Product, etc.

- ***`get<Entity>Info()`*** – returns all the entity’s information
- ***`processOrder(order)`*** – staff processes specific order
- ***`set<Entity>Info(…)`*** – changes the entity’s information
- ***`setAddress(address)`*** – changes the entity’s address
- ***`getAddress()`*** – returns the entity’s address
- ***`placeOrder(order)`*** – customer places order with products
- ***`addOrderDetail(orderDetail)`*** – adds additional order details under specific order
- ***`getOrderDetail`*** – returns order’s details about products, unit prices, and quantity
- ***`setStatus(status)`*** – changes the entity’s status

DATA ACCESS LAYER SPECIFICATION

It outlines the requirements, design, and functionality of the data access layer within a system. It serves as an interface between the application's business logic and the underlying data storage mechanism such as databases.

Below are the specifications for the proposed system and their respective methods:

- **CustomerRepo**
  - *`addCustomer(name, email_address, phone_number)*
  - *`deleteCustomer(customer)`*
  - *`getCustomerInfo()`*
  - *`setCustomerInfo(name, email_address, phone_number)`*
  - *`setAddress(address)`*
  - *`getAddress()`*
  - *`placeOrder(order)`*
- **StaffRepo**
  - *`addStaff(name, email_address, phone_number, department)`*
  - *`deleteStaff(staff)`*
  - *`getStaffInfo()`*
  - *`setStaffInfo(name, email_address, phone_number, department)`*
  - *`setAddress(address)`*
  - *`getAddress()`*
  - *`processOrder(order)`*

- **SupplierRepo**
    - *`addSupplier(name, email_address, phone_number)*
    - *`deleteSupplier(supplier)`*
    - *`getSupplierInfo()`*
    - *`setSupplierInfo(name, email_address, phone number)`*
    - *`setAddress(address)`*
    - *`getAddress()`*
- **OrderRepo**
    - *`getOrderInfo()`*
    - *`setStatus(status)`*
    - *`addOrder(orderDetail)`*
- **ProductRepo**
    - *`getProductInfo()`*
    - *`setProductInfo(name, description, price, quantity)`*
    - *`addProduct(name, description, price, quantity)`*
    - *`deleteProduct(product)`*
- **OrderDetailRepo**
    - *`getOrderDetail()`*
    - *`addOrderDetail(order, orderDetail)`*
    - *`deleteOrderDetail(order, orderDetail)`*
- **TransactionRepo**
    - *`getTransactionInfo()`*
    - *`setStatus(status)`*

SERVICE DESCRIPTION

Through this e-commerce management system, such features are implemented:

- **Efficient product inventory management** – implementing intuitive tools for organizing, updating, and tracking product inventory
- **Streamlined order processing** – designing automated workflows to expedite order fulfillment, from plaement to delivery
- **Effective customer management** – providing tools for personalized customer interactions, order tracking, and support services to enhance customer satisfaction
- **Insightful analytics** – developing sophisicated analytics modules to extract valuable insights from customer behavior, sales trends, operational metrics. Through this, it enables informed decision-making and strategic planning

BEHAVIORAL REQUIREMENTS

Behavioral requirments defines how the system should behave and interact with users or other components in different scenarios **(Visual Paradigm, 2019)**. These requirements focuses on the functional side of the system and describe the actions, events, and responses that occur during system operation.

Below are the use case diagram, sequence diagram, and state transition diagram for the proposed system.

(Next page)

USE CASE DIAGRAM

It is a graphical representation that illustrates the interactions between users and the system **(Visual Paradigm, 2019)**. Its purpose is to provide a high-level overview of the functionality of the system from the user's perspective.

Actors – individuals and other entities interacting with the system. They are depicted as stick figures.

Use cases – specific functionalities or tasks that the system or actor can perform. They are depicted as oval or ellipses.

Relationships – interactions between use cases and actors. They are depicted as lines that are connected between the use cases and the actors. Below are the types of relationships:

- **Association** – actor is interacting with the system to perform the described functionality. It is represented as a straight line.
- **Generalization** – one use case is a specialized version of another.
- **Include** – one use case includes the functionality of another.
- **Extend** – one use case extends the behavior of another

Below is the use case diagram for the proposed system. The use case diagram is mostly centered towards the customer in terms of use cases. Specific use case such as browse products have an extend relationship so customers can browse by category of by search.

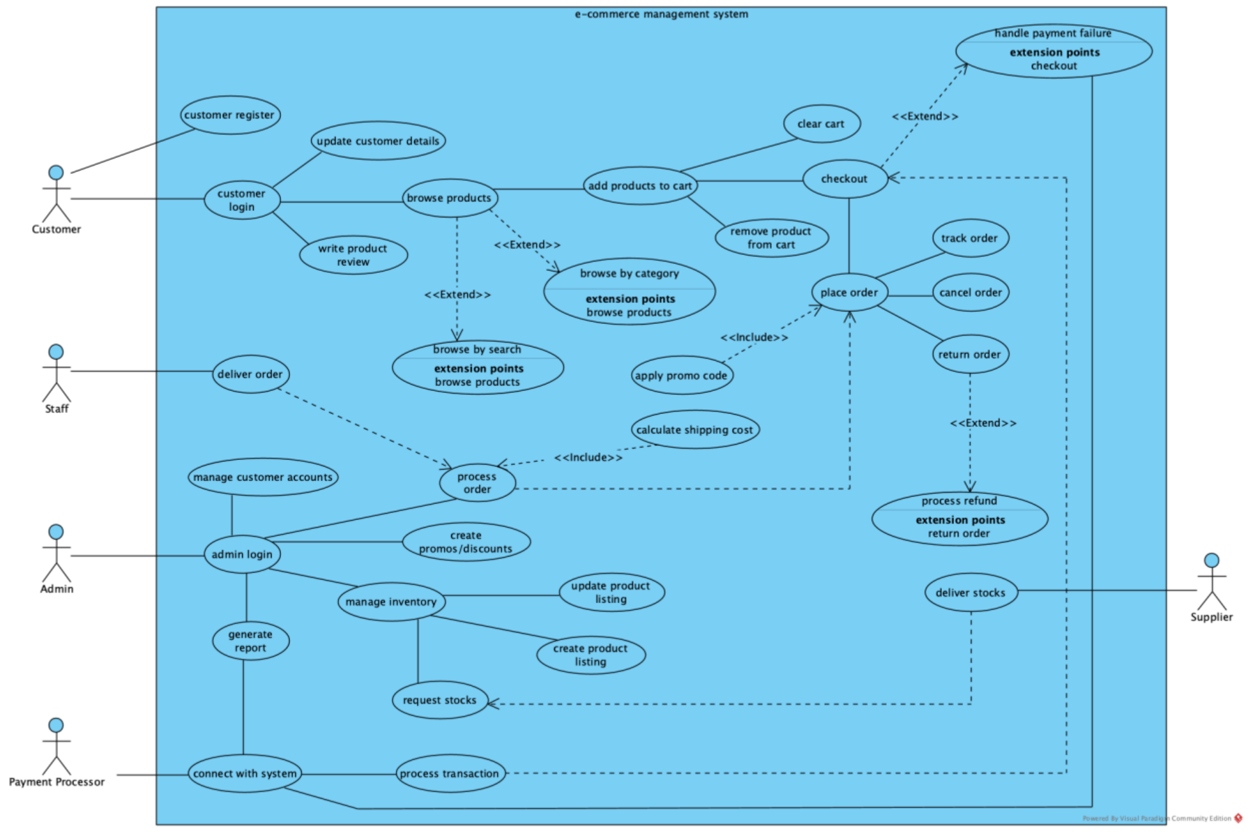

Figure 3: Use case diagram

There is also a case of dependency relationship between ***deliver order***, ***process order***, and ***place order***. This chain of events are depicated with this scenario: only if Customer places an order, Admin will be able to process it. And with that, Staff can then deliver the order. Same goes for ***request stocks*** and ***deliver stocks***: only if Admin requests stocks, supplier will deliver stocks

SEQUENCE DIAGRAM

It is used to illustrate interactions between objects or components of a system over time **(Visual Paradigm, 2019)**. It presents as a chronological sequence of messages exchanged between objects, showing the order in which interactions occur and the flow of control among them.

Lifelines – participating object. They are depicted as vertical dashed lines, and are labeled with their name inside a rectangle

Messages – interactions between lifelines. They are depicated as arrows from one lifeline to another

Activation bars – the period during which an object is active. They are depicated as vertical rectangles on top of a lifeline’s vertical dashed lines

Return/reply messages – the response of a message. They are depicated as dashed arrows

Optional elements – conditional or looping constructrs. They are depicated as boxes

Below is the sequence diagram of the proposed system. The sequence diagram is mostly based on the scenario when a customer tries to place an order. In which there are many processes that are involved such as logging in, browsing products, adding to cart, making payment, and finally placing an order, and maybe creatung a product review afterwards.

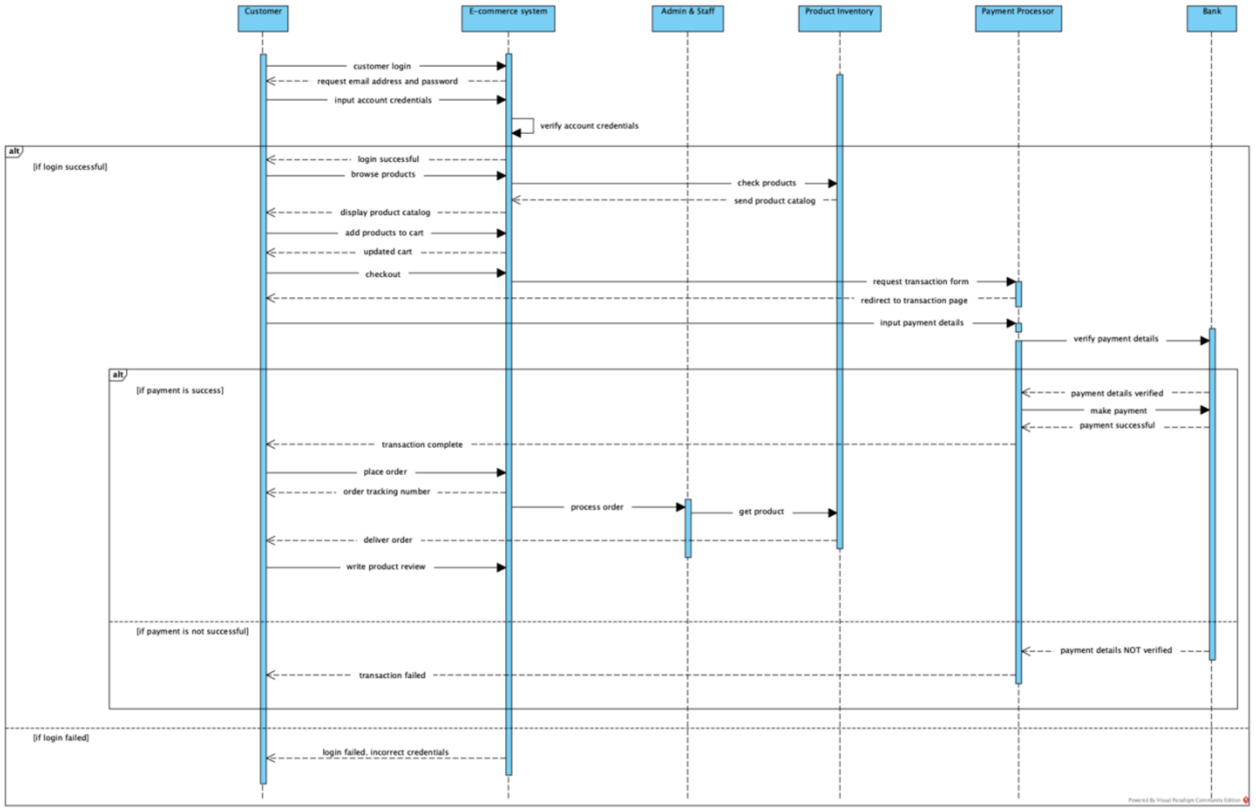

Figure 4: Sequence diagram

The sequence diagram also features a nested combined fragments for both logging in and making payments.

STATE TRANSITION DIAGRAM

It illustrates the various states an object can exist in and the transitions between those states in response to events **(Copeland, 2011)**. It mostly centers around the lifecycle of an object within a system.

- **Initial state** – starting point of the system's lifecycle. It is depictde as a solid filled circle
- **Final state** – completion of the system's lifecycle. It is depicted as a concentric circle
- **State** – specific condition of the system
- **Events** – triggers that cause transitions between states. They are labeled via text on transitions
- **Action** – activities that occur when a transition takes place
- **Transitions** – how the object moves from one state to another in response to events. They are depicted as arrows connecting the states

Below is the state diagram of the proposed system. The state transition diagram is mostly centered around the proceses when it comes to transactions, orders, and product inventory.

It first determines if transaction is in a success state or a failed state. If success, it moves into verifying order details; only if the customer decides to cancel the order, then the state of the order is cancelled. If not, it proceeds to verifying if products in the order exists in the current inventory. Through this, the state will be either be pending if some products only exists, dispatched if everything is good. And finally the delivered state, which ends the cycle.

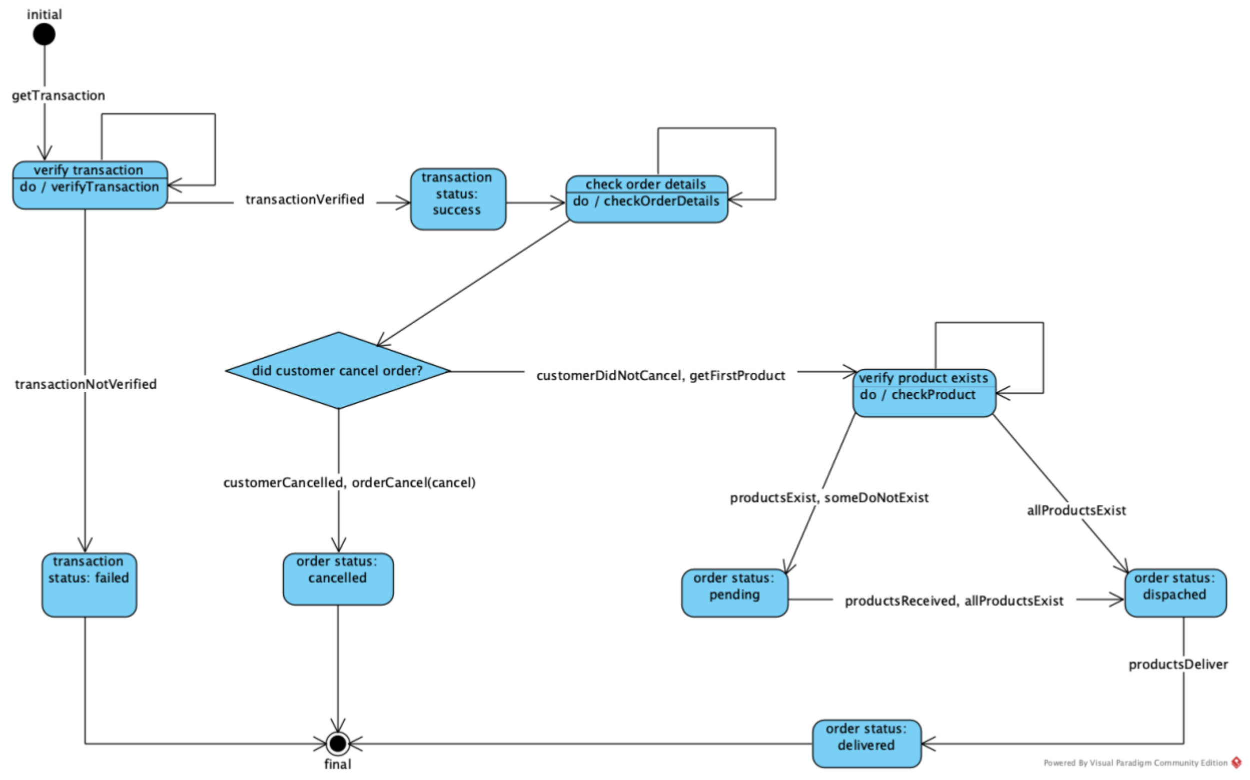

Figure 5: State transition diagram

ACTIVITY DIAGRAM

It represent the workflows, showing the sequence of activities and their interactions within a system **(Visual Paradigm, 2019)**. It is commonly used in software engineering, business process modelling, and system analysis. Through this, it is used to visualize the flow of activities from start to finish

- **Activities** – tasks that are performed within a system. They are depicted as rounded rectangles
- **Decision points** – different paths can be taken based on conditions. They are depicted as diamonds
- **Initial state** – starting point of the system's lifecycle. It is depictde as a solid filled circle
- **Final state** – completion of the system's lifecycle. It is depicted as a concentric circle

Below is the activity diagram for the proposed system. This activity diagram is mostly centered around the customer placing an order by using the system.

Initial node is that the customer opens the system. It then features decision points with with either customer logging in or registering. And thernm, how the customer will be able to view the product, whether via browsing all products, searching, or by category. Similarly there are decision points in the cart actions; whether customer still wants to do more shopping or not, and the payment detail events; if it fails or succeeds. Lastly, final state is when a customer's order is already placed.

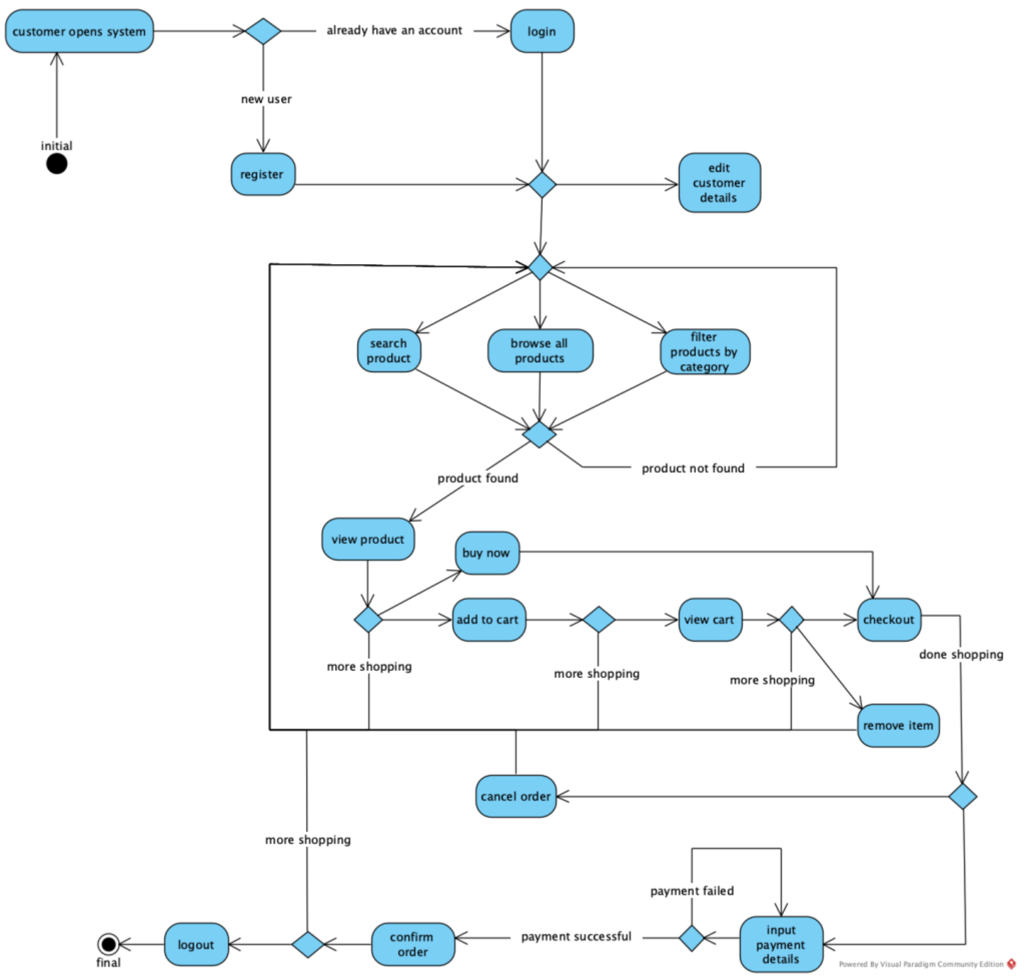

Figure 6: Activity diagram

CONCLUSION

The services that are being offered brings a comprehensive solution for e-commernce management systems, addressing both structural and behavioral requirements to meet the diverse needs of leading companies. Through clear communication and precise modeling, this system ensures the development of scalable, reliable, and maintanable systems that makes businesses grow and thrive in the market

BIBLIOGRAPHY

Copeland, L. *(2011). State-Transition Diagrams. [online] StickyMinds. Available at: https://www.stickyminds.com/article/state-transition-diagrams.*

IBM *(n.d.). What is a database schema? | IBM. [online] www.ibm.com. Available at: https://www.ibm.com/topics/database-schema.*

Visual Paradigm *(2019a). UML - Behavioral Diagram vs Structural Diagram. [online] Visual-paradigm.com. Available at: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/behavior-vs-structural-diagram/.*

Visual Paradigm *(2019b). What is Activity Diagram? [online] Visual-paradigm.com. Available at: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-activity-diagram/.*

Visual Paradigm *(2019c). What is Class Diagram? [online] Visual-paradigm.com. Available at: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-class-diagram/.*

Visual Paradigm *(2019d). What is Sequence Diagram? [online] Visual-paradigm.com. Available at: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/.*

Visual Paradigm *(2019e). What is Unified Modeling Language (UML)? [online] Visual-paradigm.com. Available at: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/.*

Visual Paradigm *(2019f). What is Use Case Diagram? [online] Visual-paradigm.com. Available at: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/.*