EX NO:1	E-COMMERCE WEBSITE FOR B2C
DATE	

AIM:

To prepare PROBLEM STATEMENT for any project.

ALGORITHM:

1. The problem statement is the initial starting point for a project.
2. A problem statement describes what needs to be done without describing how.
3. It is basically a one-to-three-page statement that everyone on theproject agrees with that describes what will be done at a high level.
4. The problem statement is intended for a broad audience and shouldbe written in non- technical terms.
5. It helps the non-technical and technical personnel communicate byproviding a description of a problem.
6. It doesn't describe the solution to the problem.

INPUT:

1. The input to requirement engineering is the problem statement prepared by customer.
2. It may give an overview of the existing system along with broad expectations from the new system.
3. The first phase of requirements engineering begins with requirements elicitation i.e. gathering of information about requirements.
4. Here, requirements are identified with the help of customer and existing system processes.

Problem

E-commerce platforms face challenges like inefficient user registration, inventory management, delayed order processing, and poor customer experience, often due to outdated or disjointed systems. This can result in lost sales, unfulfilled orders, and dissatisfied customers. A centralized E-commerce Management System is essential to streamline these operations and enhance user satisfaction.

Background

With the rapid growth of online shopping, traditional e-commerce systems struggle with managing user accounts, product inventories, order processing, and reporting efficiently. Manual or fragmented methods lead to delays, errors, and poor visibility into operations. A robust, centralized system integrating these functionalities can improve efficiency, reduce errors, and boost customer satisfaction.

Relevance

An efficient E-commerce Management System ensures smooth operations and timely order fulfillment, enhancing customer trust and loyalty. By automating key processes, the system reduces manual errors, optimizes inventory control, and enables real-time reporting for better decision-making, ultimately driving business growth.

Objectives

The primary objective is to build a centralized E-commerce Management System that improves operational efficiency and customer satisfaction. Specific objectives include:

1. User Registration: Simplify user sign-ups with secure account creation and management.

2. Product Management: Streamline inventory tracking, including stock levels, product details, and pricing.

3. Order Processing: Enable efficient order placement, tracking, and fulfillment.

4. Admin Tools: Provide administrators with tools to manage inventory, process orders, and oversee operations.

5. Reporting: Generate detailed sales, inventory, and customer activity reports for analysis and planning.

6. Customer Experience: Enhance the shopping experience with intuitive navigation, personalized recommendations, and seamless transactions.

Result:

EX NO:2	Write the software requirement specification document
DATE	

AIM:

To do requirement analysis and develop Software Requirement Specification Sheet(SRS) for any Project.

ALGORITHM:

SRS shall address are the following:

a) **Functionality.** What is the software supposed to do?

b) **External interfaces.** How does the software interact with people, the system‘s hardware, other hardware, and other software?

c) **Performance.** What is the speed, availability, response time, recovery time of various software functions, etc.?

d) **Attributes.** What is the portability, correctness, maintainability, security, etc. considerations?

e) **Design constraints imposed on an implementation.** Are there any required standards in effect, implementation language, policies for database integrity, resource limits, operating environment(s) etc.?

1. Introduction

1.1 Purpose

This document outlines the requirements for an E-commerce Website Management System (EWMS). The system aims to streamline user registration, product management, order processing, inventory control, and reporting to enhance operational efficiency and user experience.

1.2 Scope

EWMS will be a web-based platform catering to administrators, sellers, and customers. It will handle core e-commerce functions like managing products, tracking inventory, processing orders, and generating sales reports, ensuring smooth operations and a seamless shopping experience.

1.3 Definitions

- **EWMS**: E-commerce Website Management System
- **Admin**: Platform administrator
- **Seller**: Business owner managing products and orders
- **Customer**: End-user purchasing products

1.4 Overview

This document specifies the functionalities, interfaces, and performance requirements of EWMS, providing a development roadmap and user reference.

2. Overall Description

2.1 Product Perspective

EWMS is a centralized platform integrating essential e-commerce operations. It will ensure smooth product management, order fulfillment, and user interactions through an intuitive interface and efficient backend.

2.2 Product Functions

- **User Registration**: Simplify account creation for admins, sellers, and customers.
- **Product Management**: Manage inventory, pricing, and product details.
- **Order Processing**: Handle order placement, tracking, and fulfillment.
- **Reporting**: Generate sales and inventory reports for analysis.

2.3 User Classes

- **Admin**: Oversees platform operations.
- **Seller**: Manages products and processes orders.
- **Customer**: Purchases products online.

2.4 Operating Environment

Accessible via major browsers and compatible with desktop and mobile devices.

2.5 Constraints

- Secure handling of customer data.
- Scalable to support concurrent users.

2.6 Assumptions

- Users will have stable internet access.
- An SQL database will store data securely.

3. Specific Requirements

3.1 Functional Requirements

- **User Registration**: Secure account creation and role-based access.
- **Product Management**: Add, update, and track products in real-time.
- **Order Processing**: Enable seamless order placement and notifications.
- **Reporting**: Provide sales and inventory insights with downloadable formats.

3.2 Non-Functional Requirements

- **Performance**: Handle 1,000+ users with a response time under 2 seconds.
- **Security**: Encrypt sensitive data and enforce role-based permissions.
- **Usability**: Ensure a simple, mobile-responsive interface.
- **Reliability**: Guarantee 99.9% uptime.

4. External Interfaces

- **User Interface**: Responsive design tailored to admins, sellers, and customers.
- **Hardware/Software Interfaces**: Compatible with standard devices, integrates email/SMS for notifications.

5. Additional Requirements

- **Privacy Compliance**: Adhere to data protection regulations.
- **Documentation**: Provide user guides and technical documentation.

Result:

EX NO:3	Draw the entity relationship diagram
DATE	

AIM:

To Draw the Entity Relationship Diagram for any project.

ALGORITHM:

Step 1: Mapping of Regular Entity Types

Step 2: Mapping of Weak Entity Types

Step 3: Mapping of Binary 1:1 Relation Types

Step 4: Mapping of Binary 1:N Relationship Types.

Step 5: Mapping of Binary M:N Relationship Types.

Step 6: Mapping of Multivalued attributes.

INPUT:

Entities

Entity Relationship Matrix

Primary Keys

Attributes

Mapping of Attributes with Entities

Result:

EX NO:4	Draw the data flow diagrams at level 0 and level 1
DATE	

AIM:

To Draw the Data Flow Diagram for any project and List the Modules in the Application.

ALGORITHM:

1. Open the Visual Paradigm to draw DFD (Ex.Lucidchart)
2. Select a data flow diagram template
3. Name the data flow diagram
4. Add an external entity that starts the process
5. Add a Process to the DFD
6. Add a data store to the diagram
7. Continue to add items to the DFD
8. Add data flow to the DFD
9. Name the data flow
10. Customize the DFD with colours and fonts
11. Add a title and share your data flow diagram

INPUT:

Processes

Datastores

External Entities

Result:

EX NO:5	Draw use case diagram
DATE	

AIM:

To Draw the Use Case Diagram for any project

ALGORITHM:

Step 1: Identify Actors

Step 2: Identify Use Cases

Step 3: Connect Actors and Use Cases

Step 4: Add System Boundary

Step 5: Define Relationships

Step 6: Review and Refine

Step 7: Validate

INPUTS:

Actors

Use Cases

Relations

Result:

EX NO:6	Draw activity diagram of all use cases.
DATE	

AIM:

To Draw the activity Diagram for any project

ALGORITHM:

Step 1: Identify the Initial State and Final States

Step 2: Identify the Intermediate Activities Needed

Step 3: Identify the Conditions or Constraints

Step 4: Draw the Diagram with Appropriate Notations

INPUTS:

Activities

Decision Points

Guards

Parallel Activities

Conditions

Result:

EX NO:7	Draw state chart diagram of all use cases.
DATE	

AIM:

To Draw the State Chart Diagram for any project

ALGORITHM:

STEP-1: Identify the important objects to be analysed.

STEP-2: Identify the states.

STEP-3: Identify the events.

INPUTS:

Objects

States

Events

Result:

EX NO:8	Draw sequence diagram of all use cases.
DATE	

AIM:

To Draw the Sequence Diagram for any project

ALGORITHM:

1. Identify the Scenario
2. List the Participants
3. Define Lifelines
4. Arrange Lifelines
5. Add Activation Bars
6. Draw Messages
7. Include Return Messages
8. Indicate Timing and Order
9. Include Conditions and Loops
10. Consider Parallel Execution
11. Review and Refine
12. Add Annotations and Comments
13. Document Assumptions and Constraints
14. Use a Tool to create a neat sequence diagram

INPUTS:

Objects taking part in the interaction.

Message flows among the objects.

The sequence in which the messages are flowing.

Object organization.

Result:

EX NO:9	Draw collaboration diagram of all use cases
DATE	

Draw collaboration diagram of all use cases

AIM:

To Draw the Collaboration Diagram for any project

ALGORITHM:

Step 1: Identify Objects/Participants

Step 2: Define Interactions

Step 3: Add Messages

Step 4: Consider Relationships

Step 5: Document the collaboration diagram along with any relevant explanations or annotations.

INPUTS:

Objects taking part in the interaction.

Message flows among the objects.

The sequence in which the messages are flowing.

Object organization.

Result:

EX NO:10	Assign objects in sequence diagram to classes and make class diagram.
DATE	

AIM:

To Draw the Class Diagram for any project

ALGORITHM:

1. Identify Classes
2. List Attributes and Methods
3. Identify Relationships
4. Create Class Boxes
5. Add Attributes and Methods
6. Draw Relationships
7. Label Relationships
8. Review and Refine
9. Use Tools for Digital Drawing

INPUTS:

1. Class Name
2. Attributes
3. Methods
4. Visibility Notation

Result:

EX NO:11	Mini Project-E-COMMERCE WEBSITE FOR B2B
DATE	

Aim:

The E-commerce Website Management System aims to streamline user registration, product management, inventory tracking, and order processing efficiently, ensuring a seamless shopping experience for customers. It optimizes operational workflows, enhances product availability, and facilitates better organization and accessibility for sellers and administrators.

Algorithm:

1. **User Registration: Collect and verify user details for customers, sellers, and administrators, ensuring accurate role-based access.**
2. **Product Management: Allow sellers to add, update, and manage product listings, including descriptions, pricing, and availability.**
3. **Inventory Management: Monitor and update product stock levels; alert sellers and admin when stock is low.**
4. **Order Requests: Accept and process customer orders, ensuring product availability before confirmation.**
5. **Appointment Scheduling (Optional): Provide scheduling options for services (e.g., product consultations or in-store pickups).**
6. **Order Processing: Approve or deny orders based on product stock, notifying customers of the status and tracking updates.**
7. **Generate Reports: Summarize data on sales, inventory, and pending orders for sellers and administrators.**

Program:

SQL CODE:

```
CREATE DATABASE ecommerce;


USE ecommerce;


CREATE TABLE users (
   id INT AUTO_INCREMENT PRIMARY KEY,
   username VARCHAR(100) NOT NULL,
   password VARCHAR(100) NOT NULL,
   email VARCHAR(100) NOT NULL UNIQUE
);
```

```
CREATE TABLE products (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    description TEXT,
    price DECIMAL(10, 2) NOT NULL,
    stock INT DEFAULT 0
);

CREATE TABLE orders (
    id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT,
    product_id INT,
    quantity INT DEFAULT 1,
    total_price DECIMAL(10, 2),
    order_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(id),
    FOREIGN KEY (product_id) REFERENCES products(id)
);

-- Insert sample data
INSERT INTO products (name, description, price, stock)
VALUES
('Laptop', 'A high-performance laptop.', 1200.00, 10),
('Smartphone', 'Latest smartphone with advanced features.', 800.00, 15),
('Headphones', 'Noise-cancelling headphones.', 200.00, 20);
```

PYTHON CODE:

```
import streamlit as st
import mysql.connector
from mysql.connector import Error


 Database connection
def create_connection():
    try:
        conn = mysql.connector.connect(
            host="127.0.0.1",
            user="root",
            password="surya0306",
            database="ecommerce"
        )
        return conn
    except Error as e:
        st.error(f"Error connecting to database: {e}")
        return None


 Fetch products from database
def fetch_products():
    conn = create_connection()
    if conn:
        cursor = conn.cursor(dictionary=True)
        cursor.execute("SELECT * FROM products")
        products = cursor.fetchall()
        conn.close()
```

```
        return products
    return []


Place an order
def place_order(user_id, product_id, quantity):
    conn = create_connection()
    if conn:
        cursor = conn.cursor()
        cursor.execute("SELECT price, stock FROM products WHERE id = %s", (product_id,))
        product = cursor.fetchone()
        if product and product[1] >= quantity:  # Check stock
            total_price = product[0] * quantity
            cursor.execute(
                "INSERT INTO orders (user_id, product_id, quantity, total_price) VALUES (%s, %s, %s, %s)",
                (user_id, product_id, quantity, total_price),
            )
            cursor.execute(
                "UPDATE products SET stock = stock - %s WHERE id = %s",
                (quantity, product_id),
            )
            conn.commit()
            st.success("Order placed successfully!")
        else:
            st.error("Insufficient stock!")
        conn.close()
```

```
 Streamlit App
def main():
    st.title("E-Commerce Website")

     Login/Signup placeholder
    st.sidebar.header("User Section")
    username = st.sidebar.text_input("Username")
    password = st.sidebar.text_input("Password", type="password")
    if st.sidebar.button("Login"):
        st.sidebar.success("Logged in as " + username)

    st.header("Products")
    products = fetch_products()
    if products:
        for product in products:
            st.subheader(product["name"])
            st.text(product["description"])
            st.write(f"Price: ${product['price']}")
            st.write(f"Stock: {product['stock']}")
            quantity = st.number_input(f"Quantity ({product['name']})", min_value=1,
max_value=product["stock"], step=1)
            if st.button(f"Buy {product['name']}"):
                place_order(user_id=1, product_id=product["id"], quantity=quantity)  # Static user_id
for demo
    else:
        st.error("No products available!")
```

```
if __name__ == "__main__":

    main()
```

Conclusion :

The E-Commerce Website developed using Streamlit and SQL provides a seamless platform for managing product listings, user accounts, and order transactions. With its interactive interface and efficient database connectivity, the system ensures real-time updates on product availability and order status, offering users a smooth shopping experience. This project demonstrates the capability of combining modern web development frameworks with robust backend support to create scalable and user-friendly e-commerce solutions. It serves as a solid foundation for future enhancements like personalized recommendations, advanced payment integration, and AI-driven analytics to improve user engagement and business growth.