Ecommerce Platform Security & Threat Modeling Documentation

# Introduction

## Overview

Ecommerce is a web-based platform designed to facilitate online shopping, allowing users to browse products, manage a shopping cart, and securely complete purchases. The platform emphasizes user accessibility, data privacy, and security, leveraging Djangos built-in authentication and session management. The system is designed for scalability and reliability, with a focus on protecting sensitive user and transaction data.

## Objectives of the Project

This project aims to strengthen the security of the Ecommerce platform by identifying potential threats and vulnerabilities. Using the STRIDE threat modeling approach, the project focuses on:

- Identifying potential threats to the Ecommerce platform, especially in its web and database components.  
- Evaluating vulnerabilities related to user data handling, authentication, and payment processing.  
- Enhancing the overall security posture of the Ecommerce application through early threat detection.  
- Recommending mitigation strategies to address identified security risks.  
- Testing the system to validate threats and assess the effectiveness of implemented security measures.

## Importance of Threat Modelling in System and Cloud Security

For Ecommerce platforms, threat modeling is essential due to the handling of sensitive user data (personal info, addresses, payment details) and the need for secure transactions. Key concerns addressed through threat modeling include:

- Tampering with product or order data could result in financial loss or fraud.  
- Spoofing and repudiation threats (e.g., fake accounts, denial of purchases) are mitigated with robust authentication and logging.  
- Elevation of Privilege attacks could allow unauthorized access to admin features or user data.  
- Information Disclosure (e.g., leaking user addresses or payment info) must be prevented to maintain trust and comply with regulations.

# System Overview

## Description of the System and Its Components

Ecommerce is a Django-based web application that provides a seamless shopping experience. The system includes:

- Web Front-End: Built with HTML, CSS (Tailwind), and JavaScript, providing a responsive UI for product browsing, cart management, and checkout.  
- Django Back-End: Handles business logic, user authentication, order processing, and data validation.  
- Database: Stores user accounts, product listings, orders, and cart data.  
- Static & Media Storage: Manages product images and user-uploaded files.  
- Authentication: Uses Djangos built-in authentication for secure login, registration, and session management.

## Key Features and Data Types Handled

Key Features:

1. User registration and login (with password hashing and session management).  
2. Product browsing and search.  
3. Shopping cart management.  
4. Secure checkout and order history.  
5. Admin interface for product and order management.

Data Types:

- User data: username, password (hashed), email, address.  
- Product data: name, description, price, images.  
- Order data: items, quantities, user, status, timestamps.  
- Cart data: session-based or user-linked cart items.

## System Architecture

Components:

1. Web Front-End: Handles user interactions and displays product/catalog data.  
2. Django Views & Templates: Process requests, render pages, and enforce business logic.  
3. Database (e.g., SQLite/PostgreSQL): Stores persistent data.  
4. Static/Media Storage: Serves product images and user-uploaded files.  
5. Authentication & Authorization: Manages user sessions and permissions.

Data Flow:

- Users interact with the front-end to browse products, add to cart, and checkout.  
- The back-end processes requests, interacts with the database, and returns responses.  
- Authentication ensures only authorized users can access certain features (e.g., order history, admin).

## Users and Roles

1. Customers: Register, log in, browse products, manage cart, place orders.  
2. Admins: Manage products, view orders, update order statuses.  
3. Developers: Maintain codebase, deploy updates, monitor logs.

# Methodology

## STRIDE Threat Modeling

See attached tables and diagram for STRIDE and DREAD analysis.

# Mitigation Strategies

1. Strong Authentication & Password Policy  
2. Input Validation & ORM Usage  
3. Access Control & Authorization  
4. Secure File Uploads  
5. Logging & Monitoring  
6. Session Management  
7. Regular Security Testing

# Tools Used

- Django Admin & ORM: For secure data management and access control.  
- Draw.io: For system and data flow diagrams.  
- Excel/Google Sheets: For DREAD risk scoring and threat documentation.  
- Cloud Security Tools (if deployed): Use cloud provider IAM, storage policies, and monitoring.

# Data Flow Diagram (DFD)

[See DFD\_Ecommerce.drawio for the system architecture diagram.]