Data Wizards: eCommerce Platform

Team Data Wizards, a group of 3rd Year Engineering students, presents an eCommerce platform project focused on Discount and Coupon Management. Our team members, Nandan(se22uari110), Vishal(se22uari126), Vandana(se22uari202), and Sidharth(se22uari160), have come together to showcase our expertise in Object-Oriented Programming (OOP) concepts and their practical application in building a robust and scalable eCommerce solution.

Project Overview

1 Key Objectives

Our eCommerce platform aims to provide efficient management of discount policies, dynamic coupon application for users, and the incorporation of fundamental OOP principles.

2 OOP Principles

We have designed our platform to leverage the power of OOP concepts, enabling flexible, scalable, and maintainable discount and coupon management functionality.

3 Platform Features

Our eCommerce platform offers a comprehensive solution for managing discounts and coupons, empowering businesses to create and apply various discount types, validate coupons, and provide personalized offers to customers.

OOP Concepts Used

Encapsulation

Wrapping data (fields) and methods into a single unit (class) and restricting access to the internals of the object.

Example: Declaring variables as private and providing public getter and setter methods.

Use in Project: In your Coupon and Discount classes, encapsulate fields like discountPercentage or couponCode. Use setters and getters to modify or access these fields securely.

Inheritance

A class (subclass) inherits the properties and behaviors of another class (superclass).

Example: class DiscountedProduct extends Product.

Use in Project: Create a base Product class. Specialized classes like DiscountedProduct or SeasonalProduct can inherit and reuse the common properties like productName, price.

Polymorphism

The ability of a method or object to take many forms.

Example: Method overloading (compile-time) and method overriding (runtime).

Use in Project: Use method overriding in classes like Coupon or Discount. For example, override a calculateDiscount() method differently for flat discounts vs percentage discounts.

OOP Concepts Used

Abstraction

Hiding implementation details and showing only the essential features.

Example: Using abstract classes or interfaces.

Use in Project: Create an abstract class Discount with an abstract method applyDiscount(). Subclasses like PercentageDiscount and FlatDiscount will provide specific implementations.

Dynamic Binding

The code to be executed for a method call is determined at runtime.

Use in Project: If you have a list of Discount objects, calling applyDiscount() will invoke the appropriate method implementation at runtime, depending on the actual object type.

Message Passing

Objects communicate with each other by invoking methods and passing data.

Use in Project: The Cart object may call methods in Product, Coupon, and Discount classes to calculate the total price after applying discounts.

OOP Concepts used

This()

Refers to the current instance of the class. It is used to differentiate between class attributes and parameters.

Use in Project: In constructors of Product, Coupon, or Discount, use this to assign parameter values to instance variables.

Class

A blueprint for creating objects. It defines properties and methods.

Use in Project: Define classes such as Product, Customer, Order, Coupon, and Discount to model the eCommerce platform.

Object

An instance of a class that contains actual values for properties and can invoke methods.

Use in Project: Create objects like Product product1 = new Product("Laptop", 1000); or Coupon coupon1 = new Coupon("SAVE20", 20);.

User Interface (UI) Using Swing in Java

Key Features of Swing

Lightweight Components: Independent of native system resources (unlike AWT).

Platform Independence: Runs consistently across all operating systems.

Pluggable Look-and-Feel: Customize the UI's appearance.

MVC Architecture: Separates data (Model), UI (View), and behavior (Controller).

Common Swing Components

JFrame: Main window.

JPanel: Container to organize components.

JLabel: Display text/images.

JButton: Clickable button.

JTextField: Editable text input.

JTable: Tabular data display.

How Swing Fits in the eCommerce Project

Purpose: Build an intuitive UI for the eCommerce platform.

Example Use: JFrame for the main window (e.g., "Shop Dashboard"). JTable to list products. JButton for actions like "Apply Coupon" or "Checkout". JTextField to enter coupon codes.

Use Case Diagram

Explanation of Use Cases:

1.Place Order: The User can initiate an order by placing items in the cart and proceeding with a purchase.

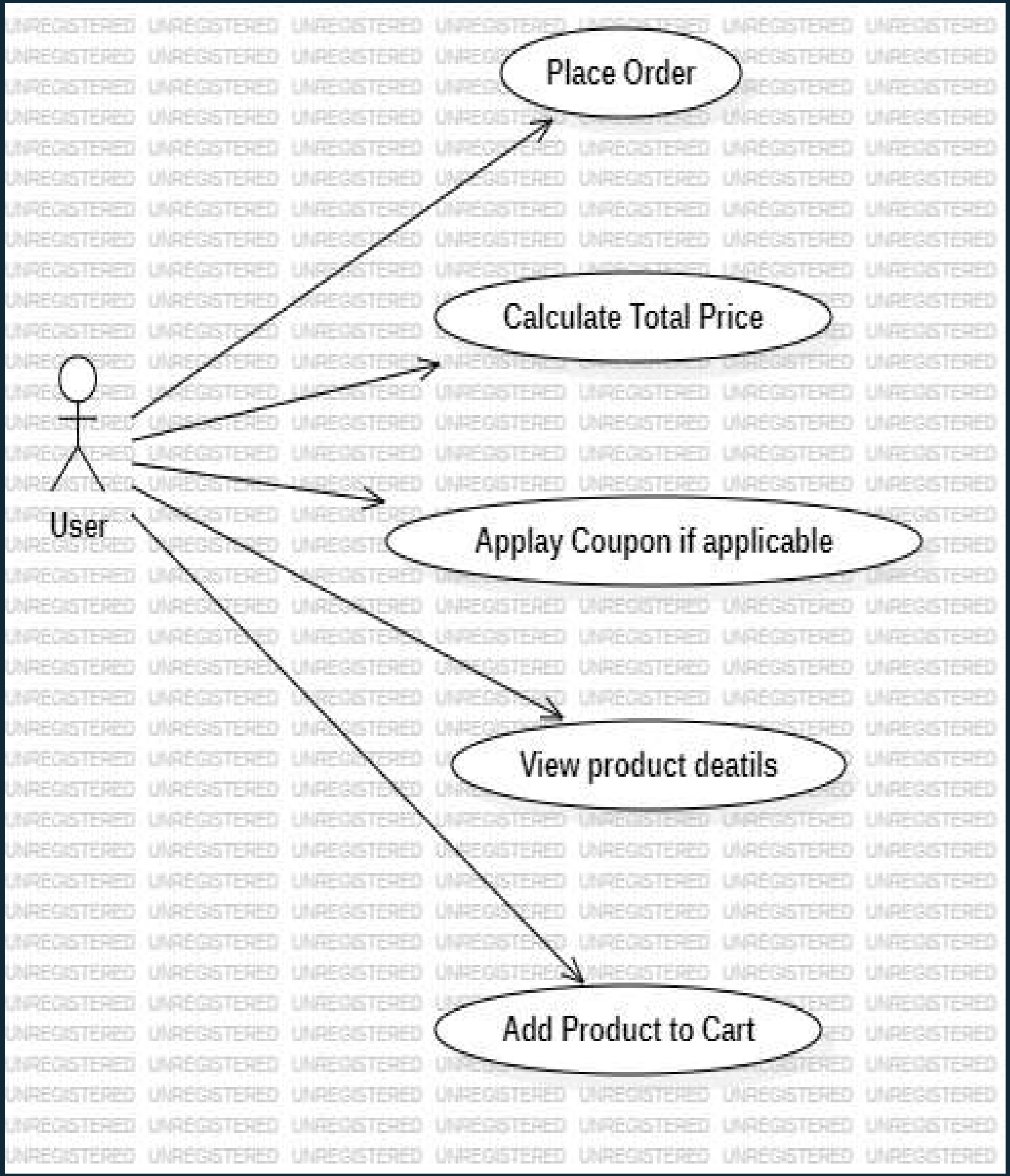
2.Calculate Total Price: The system calculates the total price of items in the cart when the user places an order.

3.Apply Coupon if Eligible: If a coupon is available, the system checks eligibility based on minimum order requirements and applies a discount if applicable.

4.View Product Details: The User can view details of a product, such as the name, description, and price.

5.Add Product to Cart: The User can add selected products to the shopping cart with specified quantities.

6.Connect to Database: The system connects to the database to store and retrieve relevant data related to users, products, orders, and inventory.



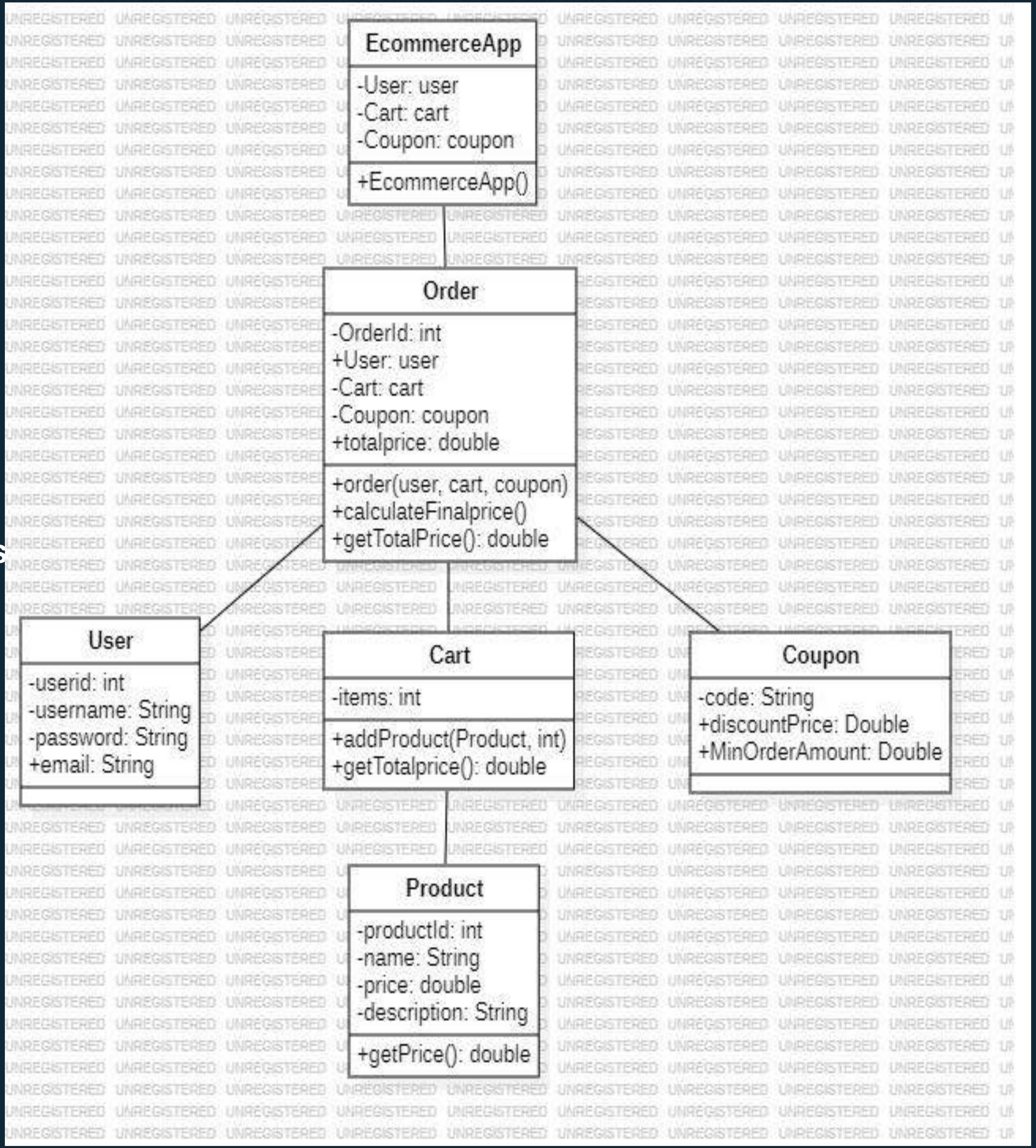
Class Diagram

Explanation:

1. ECommerceApp: It is the main class (a JFrame) that creates instances of User, Cart, Product, and Coupon and initializes the application's UI.
2. Order: It aggregates User, Cart, and Coupon to place an order.
3. User: Holds the user data.
4. Cart: Contains a collection of Product objects with their quantities and calculates the total price of items.
5. Coupon: Defines a discount coupon that applies a discount if certain conditions are met.
6. Product: Represents a product with attributes such as ID, name, price, and description.
7. DatabaseConnection: Provides a static method to connect to a database.

Relationships:

- Aggregation: Order aggregates User, Cart, and Coupon.
- Association: ECommerceApp uses User, Cart, and Coupon.
- Composition: Cart has a composition relationship with Product, as Product instances are added directly to Cart.

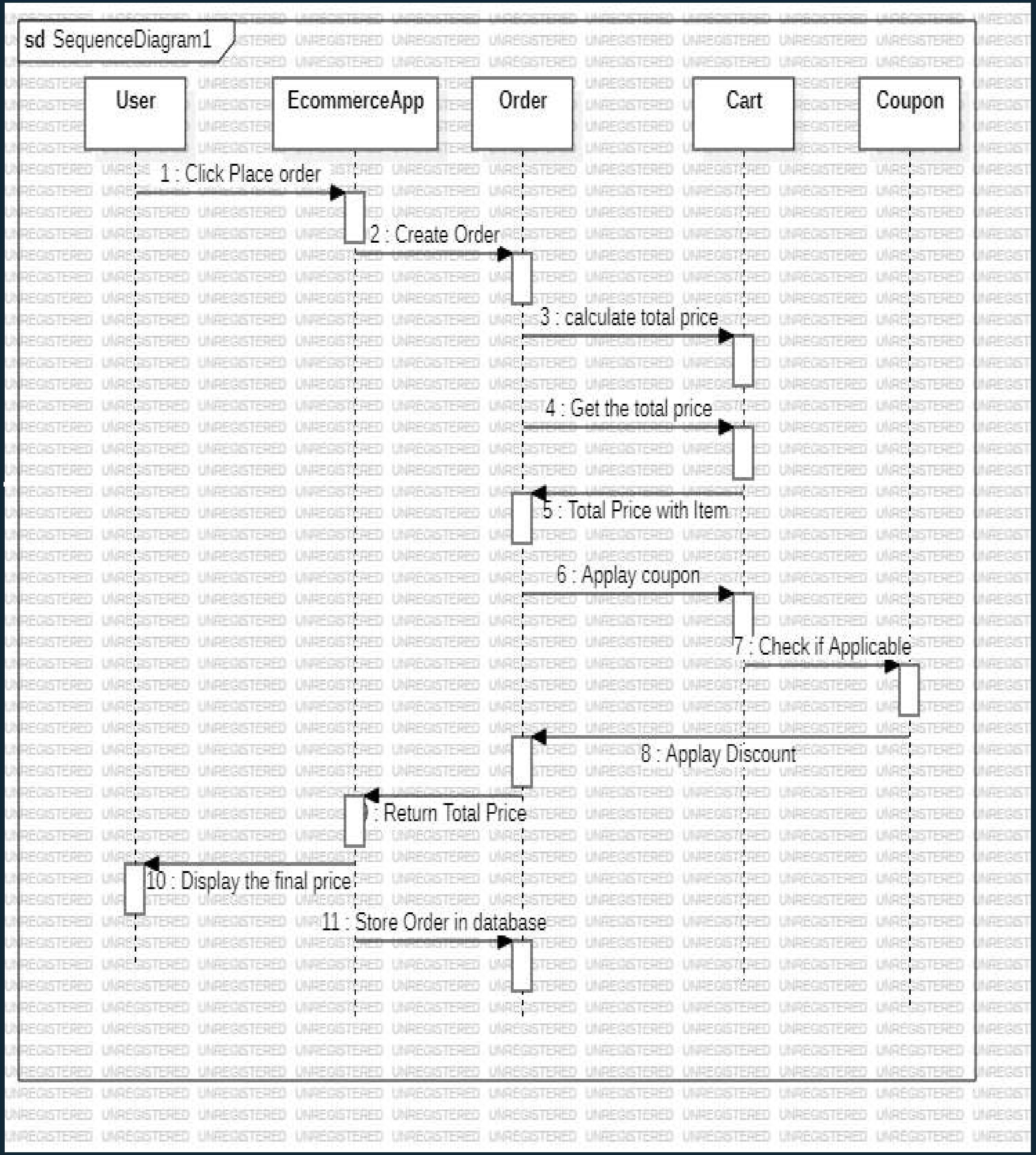


Sequence Diagram

Explanation of steps:

1.Click Place Order: The User initiates the order by clicking the "Place Order" button.

2. Create Order: The ECommerceApp creates a new Order object.
3. Calculate Total Price: Order requests the total price from the Cart.
4. gettotalprice: The Cart calculates the total price of items.
5. Apply Coupon: If a Coupon is present, Order checks if the coupon is applicable.
6. Check if applicable: The Coupon class verifies if the order amount meets the minimum requirement.
7. Apply Discount: If applicable, the discount is applied to the total price.
8. Return Final Price: The discounted price is returned to the Order.
9. Display Final Price: ECommerceApp displays the final price to the User.
10. Store Order in Database: ECommerceApp saves the order details to the database.
11. Connect to Database: DatabaseConnection establishes a connection to store the order data.



Activity Diagram

Explanation of Activities:

Start: The process begins when the user initiates the order by clicking the "Place Order" button.

Create Order Object: The ECommerceApp creates an Order instance.

Calculate Total Price: The application calculates the total price of the cart items.

Check if Coupon is Applicable: The application checks if any available coupon is applicable based on the cart's total price.

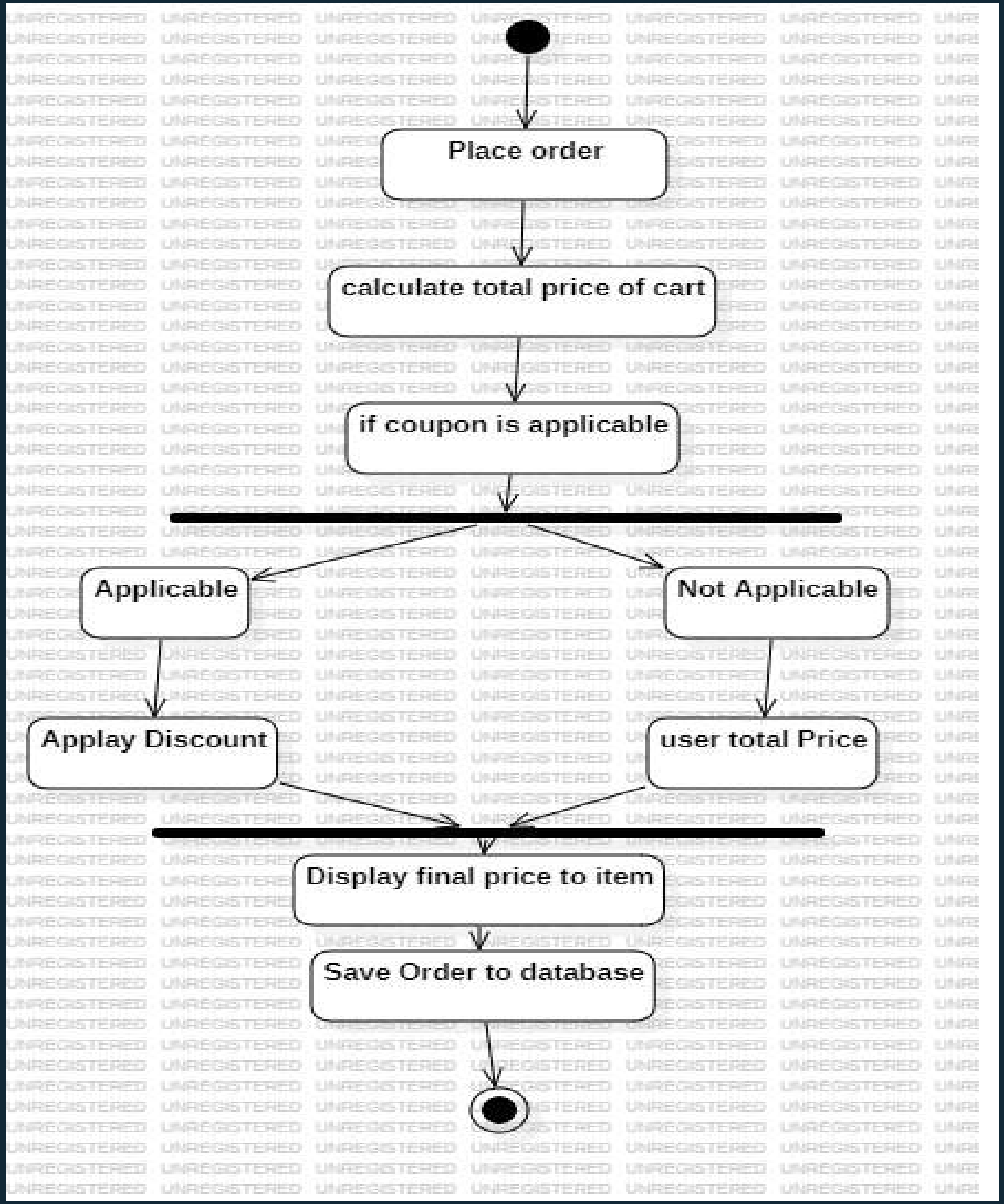
If Applicable: If the coupon meets the minimum order amount, it applies the discount to the total price.

If Not Applicable: If the coupon does not meet the requirements, it skips the discount and uses the total price.

Display Final Price to User: The application displays the final (discounted or regular) total price to the user.

Save Order to Database: If applicable, the application saves the order details to the database.

End: The "Place Order" activity concludes.



Addressing Discount Management Challenges

Complex Validation Rules

Efficient coupon validation is key. Rules like expiry dates and minimum purchase requirements are handled with robust validation methods.

Multiple Discount Types

Flexible and extensible design for different discount types. This includes percentage-based and flat-rate discounts.

Platform Extensibility

Encapsulating discount logic for seamless integration of new discount models. This ensures the platform can adapt to evolving needs.

*Forward engineering
*Backward engineering