ChatGPT Usage Report

Throughout the project, we used GPT as an assistant to clarify design decisions, offer structural suggestions, and guide our documentation process. Below are specific examples of questions we asked and how GPT contributed to each part of our work.

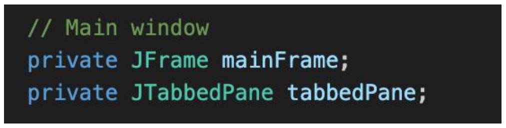
- **“Should we use inheritance or composition between Customer, Order, and Address?”**
  GPT gave a clear distinction: use inheritance when “is-a” relationships exist (e.g., Customer is a User) and composition for “has-a” relationships (e.g., Customer has an Address). This guidance helped finalize our class relationships in the UML diagram.

- **“What is the main difference in structure between a Wishlist and a ShoppingCart?”**
  ChatGPT explained that a wishlist is generally a non-committal list of products without quantities, while the shopping cart requires quantity and is transactional. It helped us define the `Wishlist` class to only contain a `List<Product>`, and `ShoppingCart` to contain `List<CartItem>` with quantity fields.

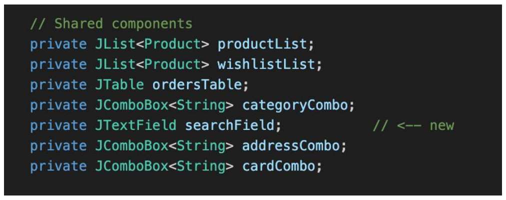
- **“How do we write a professional README file and structure our final project report?”**
  ChatGPT provided templates for both README and the

project report. For the README, it suggested including sections such as *Installation*, *Features*, *Usage*, and *Technologies Used*. For the report, it recommended dividing into *Introduction*, *UML*, *Implementation*, and *Results* with visual support.

- **"How can we build a basic user interface for product listing and shopping cart functionality?"**
  ChatGPT guided us on using `JFrame`, `JTable`, and layout managers like `BorderLayout` and `GridLayout`. It also showed how to implement `ActionListener`s to handle button clicks. While we wrote the full GUI code ourselves, we followed the structure and principles ChatGPT explained.



```
// Main window
private JFrame mainFrame;
private JTabbedPane tabbedPane;
```



```
// Shared components
private JList<Product> productList;
private JList<Product> wishlistList;
private JTable ordersTable;
private JComboBox<String> categoryCombo;
private JTextField searchField;          // <-- new
private JComboBox<String> addressCombo;
private JComboBox<String> cardCombo;
```

- **"Should we make User an abstract class, and when do we use interfaces?"**
  GPT explained the purpose of abstraction and advised us to make `User` an abstract class since we never instantiate a generic user directly. It also suggested using interfaces if we wanted multiple unrelated classes to share method contracts, which was helpful for future extensibility.