System Analysis for Online Shopping System

1. Introduction

The Online Shopping System is a Python-based desktop application using Tkinter for the interface. It allows users to browse products, add them to a cart, and checkout with delivery fees based on their governorate. Administrators can manage products and apply discounts. Data is stored in JSON files.

2. System Analysis

2.1 Requirements

Functional Requirements

- **User**:
    - Register with details: name, phone, email, gender, governorate, password, age, national ID.
    - Login using email and password.
    - Browse categories and view products.
    - Search products by name (O(log n) using binary search).
    - Sort products by price (ascending/descending using bubble sort).
    - Add items to cart and view cart.
    - Checkout with total price (non-iterative) and delivery fees based on governorate.
- **Admin**:
    - Login with fixed credentials (admin@gmail.com, admin123).
    - Add, update, or delete products in categories.
    - Apply discounts to categories.
    - View all users and all products.
- **System**:
    - Support navigation with a "Back" button using a stack.
    - Save user and product data in JSON files.

Non-Functional Requirements

- **Performance**: Fast search (O(log n)) and sorting without built-in functions.
- **Usability**: Simple GUI with dark theme and hover effects on buttons.
- **Reliability**: Show error messages for invalid inputs.
- **Maintainability**: Modular code using classes (User, Administrator, Item, Category, Store).

2.2 Stakeholders

- **Customers**: Browse and purchase products.
- **Administrators**: Manage products and discounts.
- **Developers**: Maintain and update the system.

3. System Design

3.1 High-Level Design

The system uses a GUI (Tkinter) that interacts with backend classes (User, Administrator, Item, Category, Store). Data is saved in JSON files (user.json, categories.json).

3.2 Detailed Design (UML Diagrams)

3.2.1 Use Case Diagram

- **Actors**:
  - Customer: Register, Login, Browse, Search, Sort, Add to Cart, Checkout.
  - Admin: Login, Manage Products, Apply Discounts, View Users/Items.
- **Use Cases**:
  - Customer: Browse -> Search/Sort -> Add to Cart -> Checkout.
  - Admin: Add/Update/Delete Item, Apply Discount, View Data.

3.2.2 Class Diagram

- **Classes**:
  - **User**: Attributes (name, phone, email, gender, governorate, password, age, nationalID); Methods (to_dict, from_dict).
  - **Administrator**: Inherits User; Methods (add_item, update_item, delete_item, apply_discount, view_all_users, view_all_items).
  - **Item**: Attributes (name, price, brand, model_year); Method (to_dict).

    - **Category**: Attributes (name, items); Methods (binary_search, bubble_sort, add_item, update_item, delete_item).
    - **Store**: Attributes (users, categories, cart); Methods (add_user, save, load).

3.2.3 Sequence Diagram (Example: Customer Shopping)

1. Customer enters email/password.
2. System validates via Store.
3. If valid, show Home page with categories.
4. Customer selects category, views products.
5. Customer searches/sorts products, adds to cart.
6. Customer views cart and checks out.

3.2.4 Data Flow Diagram (DFD)

- **Level 0**: Customer/Admin <--> System <--> JSON Files.
- **Level 1 (Customer)**: Login -> Browse Categories -> Search/Sort Items -> Add to Cart -> Checkout.
- **Level 1 (Admin)**: Login -> Manage Products -> Apply Discounts.

3.2.5 Entity-Relationship (ER) Diagram

- **Entities**:
    - User (PK: email; attributes: name, phone, etc.).
    - Admin (inherits User).
    - Category (PK: name; attributes: items).
    - Item (PK: name in category; attributes: price, brand, model_year).
    - Cart (attributes: items, total_price).
- **Relationships**:
    - User 1--1 Cart
    - Cart -- Item
    - Category 1--* Item

4. Implementation Notes

- **GUI**: Tkinter with frames for pages (Login, Register, Home, Cart, Admin).
- **Data Structures**: Dictionaries (users, categories), lists (items, cart), stack (navigation history).
- **Algorithms**: Binary search for searching, bubble sort for sorting.
- **Storage**: JSON files for users and categories.

5. Testing

- **Unit Tests**: Validate binary_search, bubble_sort, add_user.
- **Integration Tests**: Test full flow (e.g., Login -> Browse -> Checkout).
- **GUI Tests**: Ensure buttons and navigation work correctly.

6. Conclusion

The system provides a simple, efficient solution for online shopping with clear user and admin functionalities. Future enhancements could include a database or web interface.