# Shopping Application - Project Report

## Objective

To develop a backend-based shopping application using Python, allowing users to browse products, add them to a cart, proceed to checkout, and make payments, while providing an admin interface to manage the catalog and categories.

## Problem Statement

The project aims to build a shopping application with user and admin functionalities. Users should be able to browse products, add/remove items from their cart, and proceed to checkout with various payment options. Admins should manage product listings and categories while maintaining security constraints to prevent unauthorized actions.

## Key Requirements:

- User and admin login with session management.
- Product catalog with multiple categories.
- Ability to add, modify, and remove products (admin only).
- Shopping cart functionality (users can add/remove items).
- Secure checkout with multiple payment methods.
- Backend-only implementation without UI or database integration.

## Tools and Technologies Used:

- **Programming Language:** Python
- **Libraries/Modules:**
  - `uuid` (for session management)
  - `sys` (for handling system functions)

## Implementation Details:

1. **User Authentication & Session Management:**
   - The system initializes with predefined users (admin and general users) stored in dictionaries.
   - A login function verifies credentials and assigns a unique session ID using the `uuid` module.
   - Admin and users have separate menus with role-based access restrictions.
2. **Product Catalog Management:**
   - A dictionary-based catalog stores product details, including ID, name, category, and price.
   - Admin users can add new products with defined attributes.
   - Products can be modified or removed based on their unique product ID.
   - Users are allowed to view the catalog but cannot modify it.

3. **Shopping Cart Functionality:**
    - Each user has a dedicated cart stored in memory as a dictionary.
    - Users can add products to their cart by specifying a product ID.
    - The system validates product availability before adding it to the cart.
    - Users can remove items from their cart and view the cart contents at any time.
4. **Checkout Process:**
    - The total price of cart items is calculated dynamically.
    - Users choose a payment method from Net Banking, PayPal, or UPI.
    - Upon successful selection, the system displays a confirmation message.
    - The cart is cleared after successful checkout.
5. **Admin Functionalities:**
    - Admins can view, add, modify, and remove products from the catalog.
    - Categories can be managed by adding new ones or removing existing ones.
    - If a category is deleted, associated products must be reassigned or removed.
    - Admin actions are restricted to catalog and category management only.

**Sample Workflow:**

1. **User Login & Browsing:**
    - The user logs in using credentials.
    - Views available product categories and catalog.
2. **Adding Items to Cart:**
    - User selects products and adds them to the cart.
    - The cart maintains product IDs and quantities.
3. **Removing Items & Viewing Cart:**
    - Users can remove unwanted items.
    - The system displays all selected items with prices.
4. **Checkout & Payment:**
    - The total price is calculated.
    - User selects a preferred payment method.
    - A confirmation message is displayed.
5. **Admin Management:**
    - Admin logs in using admin credentials.
    - Admin can add/remove products and categories.
    - Modifications to existing products are allowed.
    - Admin actions are restricted to product and category management.

**Conclusion:** This project demonstrates Python's capability to handle e-commerce logic through structured backend implementation. The separation of user and admin functionalities ensures security and efficient catalog management.