**Vendor Billing System - User Manual**

**Table of Contents**

1. Introduction
2. Installation
3. Getting Started
4. Core Concepts
5. Advanced Usage
6. Error Handling
7. Best Practices
8. Troubleshooting
9. API Reference

**1. Introduction**

**1.1 What is the Vendor Billing System?**

The Vendor Billing System is a Python library designed to simplify multi-vendor billing processes. It provides a robust, flexible solution for managing orders, vendors, and payment distributions.

**1.2 Key Features**

* Multi-vendor support
* Flexible order management
* Automatic payment distribution
* Comprehensive error handling
* Easy integration

**2. Installation**

**2.1 Prerequisites**

* Python 3.8 or higher
* pip package manager

**2.2 Installation Methods**

**Using pip**

bash

Copy

pip install vendor-billing-system

**From GitHub**

bash

Copy

pip install git+https://github.com/yourusername/vendor-billing-system.git

**Local Development Installation**

bash

Copy

git clone https://github.com/yourusername/vendor-billing-system.git

cd vendor-billing-system

pip install -e .

**3. Getting Started**

**3.1 Basic Setup**

from billing\_system import BillingSystem, Item

*# Create billing system instance*

billing\_system = BillingSystem()

**3.2 Vendor Registration**

python

Copy

*# Register vendors*

electronics\_vendor = billing\_system.register\_vendor("Electronics Store")

clothing\_vendor = billing\_system.register\_vendor("Fashion Outlet")

**3.3 Creating an Order**

python

Copy

*# Create a new order*

customer\_order = billing\_system.create\_order()

*# Add items from different vendors*

billing\_system.add\_item\_to\_order(customer\_order,

Item(

item\_id="laptop001",

name="Gaming Laptop",

price=1200.00,

vendor\_id=electronics\_vendor

)

)

**4. Core Concepts**

**4.1 Vendors**

* Each vendor has a unique identifier
* Vendors are registered before adding their items to orders
* Vendor information is stored in-memory

**4.2 Items**

* Represents a product in an order
* Contains:
  + Unique item ID
  + Name
  + Price
  + Vendor ID

**4.3 Orders**

* Represents a customer's complete purchase
* Tracks:
  + Order ID
  + Customer ID
  + Total amount
  + Items from multiple vendors
  + Vendor payment distributions

**5. Advanced Usage**

**5.1 Custom Customer IDs**

python

Copy

*# Create an order with a specific customer ID*

custom\_order = billing\_system.create\_order(customer\_id="customer\_123")

**5.2 Payment Distribution**

python

Copy

*# Process order and get vendor payments*

vendor\_payments = billing\_system.process\_order\_payment(customer\_order)

*# Print payment details*

for vendor\_id, payment in vendor\_payments.items():

vendor\_name = billing\_system.vendors[vendor\_id]

print(f"{vendor\_name}: ${payment:.2f}")

**6. Error Handling**

**6.1 Handling Unregistered Vendors**

python

Copy

from billing\_system import VendorNotRegisteredError

try:

billing\_system.add\_item\_to\_order(order,

Item(

item\_id="invalid001",

name="Invalid Item",

price=100.00,

vendor\_id="non\_existent\_vendor"

)

)

except VendorNotRegisteredError as e:

print(f"Vendor Error: {e}")

**6.2 Common Exceptions**

* VendorNotRegisteredError
* EmptyOrderError
* BillingSystemError

**7. Best Practices**

**7.1 Vendor Management**

* Always register vendors before adding their items
* Keep vendor IDs secure and unique
* Validate vendor information before registration

**7.2 Order Processing**

* Create separate orders for different customers
* Validate items before adding to orders
* Process payments immediately after order completion

**7.3 Performance Considerations**

* Avoid creating extremely large orders
* Use unique identifiers consistently
* Handle exceptions gracefully

**8. Troubleshooting**

**8.1 Common Issues**

* **Unregistered Vendor**: Ensure vendors are registered before use
* **Payment Calculation Errors**: Verify item prices and vendor IDs
* **Order Processing Failures**: Check for empty orders

**8.2 Logging and Debugging**

* Implement proper exception handling
* Use Python's logging module for tracking events
* Review error messages carefully

**9. API Reference**

**9.1 BillingSystem Class**

* register\_vendor(vendor\_name: str) -> str
* create\_order(customer\_id: str = None) -> Order
* add\_item\_to\_order(order: Order, item: Item) -> None
* process\_order\_payment(order: Order) -> Dict[str, float]

**9.2 Item Class Attributes**

* item\_id: str
* name: str
* price: float
* vendor\_id: str

**10. Future Roadmap**

* Payment gateway integration
* Advanced reporting
* Tax and discount calculations
* Persistent storage support

**Appendices**

* Licensing Information
* Contributing Guidelines
* Version History

**Support**

For issues, feature requests, or contributions:

* GitHub Issues: <https://github.com/yourusername/vendor-billing-system/issues>
* Email: [Your Support Email]

**Version**

Current Version: 0.1.0 Last Updated: [Current Date]

[Vendor Billing and Order Management Library - Claude](https://claude.ai/chat/bb91e434-b3f6-4368-9279-2774004abf52)