

BOUNTEES KITCHEN USE CASES:

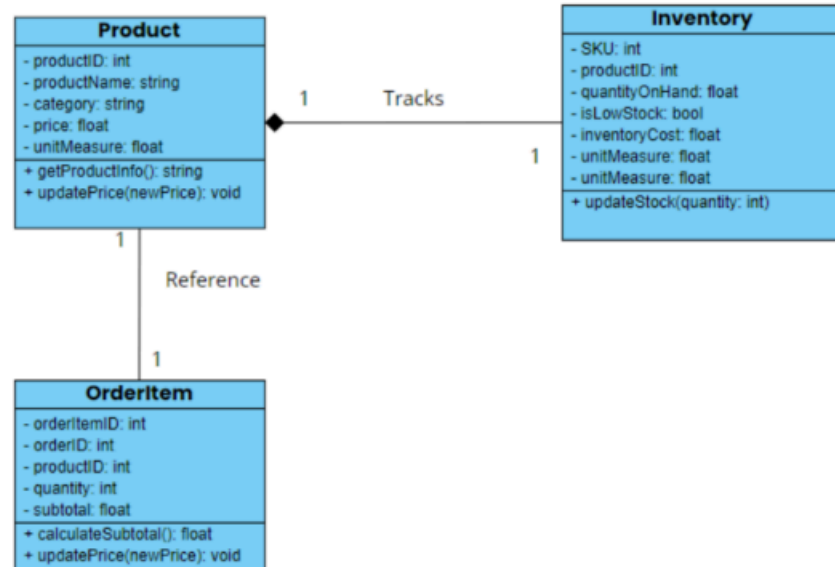
Inventory

Event	Trigger	Source	Use Case	Response	Destination
Add inventory item	New stock arrives	Owner/Staff	Add inventory item	Item added to inventory records	Inventory Database
Update inventory item	Change in item details	Staff	Update inventory item	Inventory record updated	Inventory Database
Update stock quantity	Stock used/replenished	Staff	Update stock quantity	Updated stock level recorded	Inventory Database
View inventory	Inventory inquiry request	Owner/Staff	View inventory	Current stock levels displayed	Owner/Staff
Notify low stock	Item below threshold	System	Notify low stock	Alert notification generated	Owner, Cashier

Generate inventory report	Inventory review	Owner/Staff	Generate inventory report	Inventory summary generated	Owner
---------------------------	------------------	-------------	---------------------------	-----------------------------	-------

BOUNTEES KITCHEN INVENTORY SUBSYSTEM CASE DIAGRAM:

5.2 Inventory System



BOUNTEES KITCHEN INVENTORY SUBSYSTEM OUTPUT:

Inventory Management System

Current Stock Levels

Manage your products and inventory

SKU	Product Name	Category	Quantity on Hand	Unit Measure	Price	Inventory Cost	Status	Actions
1	pandesal	food	100.0	200.0	₱3.00	₱1200.00	In Stock	<div>EditDelete</div>
2	mango juice	beverage	90.0	120.0	₱25.00	₱200.00	In Stock	<div>EditDelete</div>

Update Quantity

pandesal

Current Quantity: 100.0

100.0

Update Quantity

Cancel

127.0.0.1:5000/inventory/update-quantity/1

Inventory Management

HomeAdd InventoryGenerate Report

Update Quantity

pandesal

Current Quantity: 100.0

9

Update QuantityCancel

© 2025 Inventory Management

127.0.0.1:5000/home

Inventory Management System

Low Stock Alert

You have 1 item(s) running low on stock.

pandesal — Only 9.0 units remaining (SKU: 1)

Current Stock Levels

Manage your products and inventory

SKU	Product Name	Category	Quantity on Hand	Unit Measure	Price	Inventory Cost	Status	Actions
1	pandesal	food	9.0	200.0	₱2.00	₱1200.00	Low Stock	<div>EditDelete</div>
2	mango juice	beverage	90.0	120.0	₱25.00	₱200.00	In Stock	<div>EditDelete</div>
3	pancit canton	food	200.0	12.0	₱25.00	₱1200.00	In Stock	<div>EditDelete</div>

© 2025 Inventory Management

Acti

Category

food

Product Unit

Inventory Unit

20.0

200.0

Update Item

Inventory Management

HomeAdd InventoryGenerate Report

Update Inventory Item

Edit product and inventory values

Product Name

pandesal

Category

food

Price

₱ 20

Product Unit

200.0

Inventory Unit

200.0

Inventory Cost

₱ 1200.0

Cancel

Update Item

Inventory Management

HomeAdd InventoryGenerate Report1

"pandesal" updated successfully!

Inventory Management System

Low Stock Alert

You have 1 item(s) running low on stock.

pandesal — Only 9.0 units remaining (SKU: 1)

Current Stock Levels

Manage your products and inventory

SKU	Product Name	Category	Quantity on Hand	Unit Measure	Price	Inventory Cost	Status	Actions
1	pandesal	food	9.0	200.0	₱2.00	₱1200.00	Low Stock	<div>EditDelete</div>
2	mango juice	beverage	90.0	120.0	₱25.00	₱200.00	In Stock	<div>EditDelete</div>

127.0.0.1:5000/inventory/report

Inventory Management

HomeAdd InventoryGenerate Report1

Inventory Report

Generated on November 15, 2025 at 11:15 AM

Total Items
2

Total Inventory Value
₱28800.00

Low Stock Items
1

Low Stock Alert

SKU	Product Name	Category	Quantity on Hand	Unit Measure
1	pandesal	food	9.0	200.0

Category Breakdown

Category	Item Count	Total Quantity	Total Value
food	1	9.00	₱10800.00

1pandesalfood9.0200.0

Category Breakdown

Category	Item Count	Total Quantity	Total Value
food	1	9.00	₱10800.00
beverage	1	90.00	₱18000.00

Complete Inventory List

SKU	Product Name	Category	Quantity	Unit Measure	Price	Inventory Cost	Total Value	Status
1	pandesal	food	9.0	200.0	₱2.00	₱1200.00	₱10800.00	Low
2	mango juice	beverage	90.0	120.0	₱25.00	₱200.00	₱18000.00	In Stock

© 2025 Inventory Management

11/15/25, 11:15 AM

Inventory Management System

Inventory Management

Inventory Report

Generated on November 15, 2025 at 11:15 AM

← Back

Print

Total Items

2

Total Inventory Value

₱28800.00

Low Stock Items

1

🚨 Low Stock Alert

SKU	Product Name	Category	Quantity on Hand	Unit Measure
1	pandesal	food	9.0	200.0

Category Breakdown

Category	Item Count	Total Quantity	Total Value
food	1	9.00	₱10800.00
beverage	1	90.00	₱18000.00

Complete Inventory List

127.0.0.1:5000/Inventory/report

1/2

1

Print

2 sheets of paper

Destination

Microsoft Print to PDF

Pages

All

Layout

Portrait

Color

Color

More settings

Print

Cancel

11/15/25, 11:15 AM

Inventory Management System

Inventory Management

Inventory Report

Generated on November 15, 2025 at 11:15 AM

Total Items

2

Total Inventory Value

₱28800.00

Low Stock Items

1

← Back

Print

127.0.0.1:5000/inventory/add

Inventory Management

HomeAdd InventoryGenerate Report

Add New Inventory Item

Create a product and add inventory record

Product Name *

pancit canton

✓

Category *

food

✓

Price

₱ 25

✓

Product Unit

12

✓

Quantity on Hand

200

✓

Inventory Unit

12

✓

Inventory Cost

₱ 1200

✓

Cancel

Add to Inventory

BOUNTEES KITCHEN INVENTORY MANAGEMENT SUBSYSTEM SCRIPT:

```
class Product(db.Model):
    __tablename__ = 'product'

    productID = db.Column(db.Integer, primary_key=True)
    productName = db.Column(db.String(255), nullable=False)
    category = db.Column(db.String(255), nullable=False)
    price = db.Column(db.Float, nullable=False)
    unitMeasure = db.Column(db.Float, nullable=False)

    def getProductInfo(self):
        """Get product information as a string"""
        return f"{self.productName} ({self.category}) - ${self.price}"

    def updatePrice(self, newPrice):
        """Update product price"""
        self.price = newPrice
```

```
class Inventory(db.Model):
    __tablename__ = 'inventory'

    SKU = db.Column(db.Integer, primary_key=True)
    productID = db.Column(db.Integer, db.ForeignKey('product.productID'), nullable=False)
    quantityOnHand = db.Column(db.Float, nullable=False)
    isLowStock = db.Column(db.Boolean, default=False)
    inventoryCost = db.Column(db.Float, nullable=False)
    unitMeasure = db.Column(db.Float, nullable=False)

    product = db.relationship('Product', backref='inventory_items')

    def updateStock(self, quantity_int):
        """Update stock quantity"""
        self.quantityOnHand = quantity_int
```

```

class OrderItem(db.Model):
    __tablename__ = 'orderitem'

    orderItemID = db.Column(db.Integer, primary_key=True)
    orderID = db.Column(db.Integer, nullable=False)
    productID = db.Column(db.Integer, db.ForeignKey('product.productID'), nullable=False)
    quantity = db.Column(db.Integer, nullable=False)
    subtotal = db.Column(db.Float, nullable=False)

    product = db.relationship('Product', backref='order_items')

    def calculateSubtotal(self):
        """Calculate subtotal based on quantity and product price"""
        if self.product:
            self.subtotal = self.quantity * self.product.price
            return self.subtotal
        return 0.0

    def updatePrice(self, newPrice):
        """Update price and recalculate subtotal"""
        if self.product:
            self.product.price = newPrice
            self.calculateSubtotal()

with app.app_context():
    db.create_all()

# LOW STOCK THRESHOLD
LOW_STOCK_THRESHOLD = 10

```

```

def check_low_stock():
    """Check all inventory items and update isLowStock status"""
    inventory_items = Inventory.query.all()
    for item in inventory_items:
        if item.quantityOnHand <= LOW_STOCK_THRESHOLD:
            item.isLowStock = True
        else:
            item.isLowStock = False
    db.session.commit()

def get_low_stock_items():
    """Get all items that are low in stock"""
    return Inventory.query.filter(Inventory.isLowStock == True).all()

# CODE FOR ADD INVENTORY ITEM
@app.route('/inventory/add', methods=['GET', 'POST'])
def add_inventory():
    """Add inventory item - creates new product and inventory entry together"""
    if request.method == 'POST':
        try:
            # Create the Product first
            product_name = request.form.get('productName')
            category = request.form.get('category')
            price = float(request.form.get('price'))
            product_unit_measure = float(request.form.get('productUnitMeasure'))

            new_product = Product(
                productName=product_name,
                category=category,
                price=price,
                unitMeasure=product_unit_measure
            )

            db.session.add(new_product)
            db.session.flush()

```

```

        quantity = float(request.form.get('quantityOnHand'))
        inventory_cost = float(request.form.get('inventoryCost'))
        inventory_unit_measure = float(request.form.get('inventoryUnitMeasure'))

        new_inventory = Inventory(
            productID=new_product.productID,
            quantityOnHand=quantity,
            inventoryCost=inventory_cost,
            unitMeasure=inventory_unit_measure,
            isLowStock=False
        )

        db.session.add(new_inventory)
        db.session.commit()

        flash(f'Product "{product_name}" added to inventory successfully!', 'success')
        return redirect(url_for('index'))

    except Exception as e:
        db.session.rollback()
        flash(f'Error adding product: {str(e)}', 'error')
        return redirect(url_for('add_inventory'))

    return render_template('add_inventory.html')

# CODE FOR UPDATING STOCK QUANTITY
@app.route('/inventory/update-quantity/<int:sku>', methods=['GET', 'POST'])
def update_quantity(sku):
    """Update quantity on hand for an inventory item"""
    inventory_item = Inventory.query.get_or_404(sku)
    product = Product.query.get(inventory_item.productID)

    if request.method == 'POST':
        new_quantity = float(request.form.get('quantityOnHand'))
        inventory_item.quantityOnHand = new_quantity

        # Check if it's low stock
        if new_quantity <= LOW_STOCK_THRESHOLD:

```

```

        inventory_item.isLowStock = True
    else:
        inventory_item.isLowStock = False

    db.session.commit()

    flash(f'Quantity updated successfully for "{product.productName}"!', 'success')
    return redirect(url_for('index'))

return render_template('update_quantity.html', inventory=inventory_item, product=product)

# CODE FOR UPDATING INVENTORY ITEM DETAILS
@app.route('/inventory/update/<int:sku>', methods=['GET', 'POST'])
def update_inventory_item(sku):
    """Update product and inventory details"""
    inventory_item = Inventory.query.get_or_404(sku)
    product = Product.query.get(inventory_item.productID)

    if request.method == 'POST':
        # Update Product details
        product.productName = request.form.get('productName')
        product.category = request.form.get('category')
        product.price = float(request.form.get('price'))
        product.unitMeasure = float(request.form.get('productUnitMeasure'))

        # Update Inventory details
        inventory_item.inventoryCost = float(request.form.get('inventoryCost'))
        inventory_item.unitMeasure = float(request.form.get('inventoryUnitMeasure'))

    db.session.commit()

    flash(f'"{product.productName}" updated successfully!', 'success')
    return redirect(url_for('index'))

return render_template('update_inventory.html', inventory=inventory_item, product=product)

```

```

# CODE FOR DELETING INVENTORY ITEM (NEW!)
@app.route('/inventory/delete/<int:sku>', methods=['POST'])
✓ def delete_inventory(sku):
    """Delete an inventory item and its associated product"""
    try:
        inventory_item = Inventory.query.get_or_404(sku)
        product = Product.query.get(inventory_item.productID)
        product_name = product.productName

        # Delete inventory first (foreign key constraint)
        db.session.delete(inventory_item)

        # Check if this product has other inventory entries
        other_inventory = Inventory.query.filter(
            Inventory.productID == product.productID,
            Inventory.SKU != sku
        ).first()

        # If no other inventory entries, delete the product too
        if not other_inventory:
            db.session.delete(product)

        db.session.commit()

        flash(f'"{product_name}" has been deleted successfully!', 'success')
    except Exception as e:
        db.session.rollback()
        flash(f'Error deleting item: {str(e)}', 'error')

    return redirect(url_for('index'))

```

```

# CODE FOR GENERATING INVENTORY REPORT
@app.route('/inventory/report')
✓ def inventory_report():
    """Generate comprehensive inventory report"""
    check_low_stock() # Update low stock status first

    inventory_items = db.session.query(Inventory, Product).join(
        Product, Inventory.productID == Product.productID
    ).all()

    # Calculate report statistics
    total_items = len(inventory_items)
    total_inventory_value = sum(inv.quantityOnHand * inv.inventoryCost for inv, prod in inventory_items)
    low_stock_items = [item for item in inventory_items if item[0].isLowStock]
    low_stock_count = len(low_stock_items)

    # Category breakdown
    categories = {}
    for inv, prod in inventory_items:
        if prod.category not in categories:
            categories[prod.category] = {
                'count': 0,
                'total_value': 0,
                'total_quantity': 0
            }
        categories[prod.category]['count'] += 1
        categories[prod.category]['total_value'] += inv.quantityOnHand * inv.inventoryCost
        categories[prod.category]['total_quantity'] += inv.quantityOnHand

    return render_template('inventory_report.html',
                           inventory_items=inventory_items,
                           total_items=total_items,
                           total_inventory_value=total_inventory_value,
                           low_stock_items=low_stock_items,
                           low_stock_count=low_stock_count,
                           categories=categories)

```

```

# THIS IS ALSO THE VIEW INVENTORY, SINCE THE INVENTORY ITEMS ARE DISPLAYED ON START OF THE PAGE
# HOME PAGE
@app.route('/')
@app.route('/home')
✓ def index(user_name=None):
    """Home page showing inventory table"""
    check_low_stock() # Check and update low stock status

    inventory_items = db.session.query(Inventory, Product).join(
        Product, Inventory.productID == Product.productID
    ).all()

    # Get low stock items for notification
    low_stock_items = get_low_stock_items()
    low_stock_count = len(low_stock_items)

    return render_template('home.html', user=user_name, inventory_items=inventory_items,
                           low_stock_count=low_stock_count, low_stock_items=low_stock_items)

if __name__ == '__main__':
    app.run(debug=True)

```

BOUNTEES KITCHEN INVENTORY SUBSYSTEM TEMPLATES:

add_Inventory.html

```

{% extends 'base.html' %}
{% block content %}

<div class="card mx-auto shadow-sm" style="max-width: 900px;">
  <div class="card-header bg-white">
    <h4 class="card-title mb-0">Add New Inventory Item</h4>
    <small class="text-muted">Create a product and add inventory record</small>
  </div>
  <div class="card-body">
    <form method="POST" class="needs-validation" novalidate>
      <div class="row gx-3 gy-3">
        <div class="col-12 col-md-6">
          <label for="productName" class="form-label">Product Name <span class="text-danger">*</span></label>
          <input type="text" class="form-control" id="productName" name="productName" placeholder="e.g. Rice - 5kg" required>
          <div class="invalid-feedback">Product name is required.</div>
        </div>

        <div class="col-12 col-md-6">
          <label for="category" class="form-label">Category <span class="text-danger">*</span></label>
          <input type="text" class="form-control" id="category" name="category" placeholder="e.g. Food" required>
          <div class="invalid-feedback">Category is required.</div>
        </div>

        <div class="col-6 col-md-3">
          <label for="price" class="form-label">Price</label>
          <div class="input-group">
            <span class="input-group-text">₹</span>
            <input type="number" class="form-control" id="price" name="price" step="0.01" min="0" placeholder="0.00" required>
            <div class="invalid-feedback">Enter a valid price.</div>
          </div>
        </div>

        <div class="col-6 col-md-3">
          <label for="productUnitMeasure" class="form-label">Product Unit</label>
          <input type="number" class="form-control" id="productUnitMeasure" name="productUnitMeasure" step="0.01" min="0" required>
          <div class="invalid-feedback">Enter the product unit measure.</div>
        </div>

        <div class="col-6 col-md-3">
          <label for="quantityOnHand" class="form-label">Quantity on Hand</label>
          <input type="number" class="form-control" id="quantityOnHand" name="quantityOnHand" step="0.01" min="0" required>
          <div class="invalid-feedback">Enter initial quantity.</div>
        </div>
      </div>
    </form>
  </div>
</div>

```

```

50
51         <div class="col-12 col-md-6">
52             <label for="inventoryCost" class="form-label">Inventory Cost</label>
53             <div class="input-group">
54                 <span class="input-group-text">P</span>
55                 <input type="number" class="form-control" id="inventoryCost" name="inventoryCost" step="0.01" min="0" required>
56                 <div class="invalid-feedback">Enter inventory cost.</div>
57             </div>
58         </div>
59
60     </div>
61
62     <div class="d-flex justify-content-end gap-2 mt-4">
63         <a href="{{ url_for('index') }}" class="btn btn-outline-secondary">Cancel</a>
64         <button type="submit" class="btn btn-primary">Add to Inventory</button>
65     </div>
66 </form>
67 </div>
68 </div>
69
70 <!-- Client-side Bootstrap validation helper -->
71 <script>
72 // Example starter JavaScript for disabling form submissions if there are invalid fields
73 (function () {
74     'use strict'
75     var forms = document.querySelectorAll('.needs-validation')
76     Array.prototype.slice.call(forms).forEach(function (form) {
77         form.addEventListener('submit', function (event) {
78             if (!form.checkValidity()) {
79                 event.preventDefault()
80                 event.stopPropagation()
81             }
82             form.classList.add('was-validated')
83         }, false)
84     })
85 })()
86 </script>
87
88 {% endblock %}

```

Inventory_report.html

```

1  {% extends 'base.html' %}
2  {% block content %}
3
4  <div class="d-flex justify-content-between align-items-center mb-3">
5      <div>
6          <h1 class="m-0">Inventory Report</h1>
7          <small class="text-muted">Generated on {{ now.strftime('%B %d, %Y at %I:%M %p') if now else 'N/A' }}</small>
8      </div>
9      <div class="d-flex gap-2">
10         <a href="{{ url_for('index') }}" class="btn btn-outline-secondary">← Back</a>
11         <button onclick="window.print()" class="btn btn-primary">Print</button>
12     </div>
13 </div>
14
15 <div class="row g-3 mb-4">
16     <div class="col-md-4">
17         <div class="card">
18             <div class="card-body">
19                 <h6 class="card-subtitle mb-2 text-muted">Total Items</h6>
20                 <div class="h4">{{ total_items }}</div>
21             </div>
22         </div>
23     </div>
24     <div class="col-md-4">
25         <div class="card">
26             <div class="card-body">
27                 <h6 class="card-subtitle mb-2 text-muted">Total Inventory Value</h6>
28                 <div class="h4">P{{ "%.2f"|format(total_inventory_value) }}</div>
29             </div>
30         </div>
31     </div>
32     <div class="col-md-4">
33         <div class="card">
34             <div class="card-body">
35                 <h6 class="card-subtitle mb-2 text-muted">Low Stock Items</h6>
36                 <div class="h4 text-danger">{{ low_stock_count }}</div>
37             </div>
38         </div>
39     </div>
40 </div>
41
42 {% if low_stock_count > 0 %}
43     <h4 class="text-danger">⚠ Low Stock Alert</h4>

```

```

44     <div class="table-responsive mb-4">
45         <table class="table table-striped">
46             <thead>
47                 <tr>
48                     <th>SKU</th>
49                     <th>Product Name</th>
50                     <th>Category</th>
51                     <th>Quantity on Hand</th>
52                     <th>Unit Measure</th>
53                 </tr>
54             </thead>
55             <tbody>
56                 {% for inventory, product in low_stock_items %}
57                 <tr>
58                     <td>{{ inventory.SKU }}</td>
59                     <td>{{ product.productName }}</td>
60                     <td>{{ product.category }}</td>
61                     <td class="text-danger">{{ inventory.quantityOnHand }}</td>
62                     <td>{{ inventory.unitMeasure }}</td>
63                 </tr>
64                 {% endfor %}
65             </tbody>
66         </table>
67     </div>
68 {% endif %}
69
70 <h4>Category Breakdown</h4>
71 <div class="table-responsive mb-4">
72     <table class="table table-bordered">
73         <thead class="table-light">
74             <tr>
75                 <th>Category</th>
76                 <th>Item Count</th>
77                 <th>Total Quantity</th>
78                 <th>Total Value</th>
79             </tr>
80         </thead>
81         <tbody>
82             {% for category, data in categories.items() %}
83             <tr>

```

```

88         </tr>
89     {% endfor %}
90 </tbody>
91 </table>
92 </div>
93
94 <h4>Complete Inventory List</h4>
95 <div class="table-responsive">
96     <table class="table table-striped table-hover">
97         <thead class="table-dark">
98             <tr>
99                 <th>SKU</th>
100                <th>Product Name</th>
101                <th>Category</th>
102                <th>Quantity</th>
103                <th>Unit Measure</th>
104                <th>Price</th>
105                <th>Inventory Cost</th>
106                <th>Total Values</th>
107                <th>Status</th>
108            </tr>
109        </thead>
110        <tbody>
111            {% for inventory, product in inventory_items %}
112            <tr>
113                <td>{{ inventory.SKU }}</td>
114                <td>{{ product.productName }}</td>
115                <td>{{ product.category }}</td>
116                <td class="{{ 'text-danger fw-bold' if inventory.isLowStock else '' }}">{{ inventory.quantityOnHand }}</td>
117                <td>{{ inventory.unitMeasure }}</td>
118                <td>P{{ '%.2f'|format(product.price) }}</td>
119                <td>P{{ '%.2f'|format(inventory.inventoryCost) }}</td>
120                <td>P{{ '%.2f'|format(inventory.quantityOnHand * inventory.inventoryCost) }}</td>
121                <td>{% if inventory.isLowStock %}<span class="badge bg-danger">Low</span>{% else %}<span class="badge bg-success">In Stock</span>{% endif %}</td>
122            </tr>
123            {% endfor %}
124        </tbody>
125    </table>
126 </div>
127
128 {% endblock %}

```

update_inventory.html

```

1  {% extends 'base.html' %}
2  {% block content %}
3
4  <div class="card mx-auto" style="max-width: 800px;">
5      <div class="card-header bg-white">
6          <h4 class="card-title mb-0">Update Inventory Item</h4>
7          <small class="text-muted">Edit product and inventory values</small>
8      </div>
9      <div class="card-body">
10         <form method="POST" class="needs-validation" novalidate>
11             <div class="row gx-3 gy-3">
12                 <div class="col-12 col-md-6">
13                     <label for="productName" class="form-label">Product Name</label>
14                     <input type="text" class="form-control" id="productName" name="productName" value="{{ product.productName }}" required>
15                     <div class="invalid-feedback">Product name required.</div>
16                 </div>
17
18                 <div class="col-12 col-md-6">
19                     <label for="category" class="form-label">Category</label>
20                     <input type="text" class="form-control" id="category" name="category" value="{{ product.category }}" required>
21                     <div class="invalid-feedback">Category required.</div>
22                 </div>
23
24                 <div class="col-md-4">
25                     <label for="price" class="form-label">Price</label>
26                     <div class="input-group">
27                         <span class="input-group-text">P</span>
28                         <input type="number" class="form-control" id="price" name="price" value="{{ product.price }}" step="0.01" min="0" required>
29                         <div class="invalid-feedback">Enter a valid price.</div>
30                     </div>
31                 </div>
32
33                 <div class="col-md-4">
34                     <label for="productUnitMeasure" class="form-label">Product Unit</label>
35                     <input type="number" class="form-control" id="productUnitMeasure" name="productUnitMeasure" value="{{ product.unitMeasure }}" step="0.01" min="0" required>
36                 </div>
37
38                 <div class="col-md-4">
39                     <label for="inventoryUnitMeasure" class="form-label">Inventory Unit</label>
40                     <input type="number" class="form-control" id="inventoryUnitMeasure" name="inventoryUnitMeasure" value="{{ inventory.unitMeasure }}" step="0.01" min="0" required>
41                 </div>
42
43                 <div class="col-md-6">
44                     <label for="inventoryCost" class="form-label">Inventory Cost</label>
45                     <div class="input-group">
46                         <span class="input-group-text">P</span>
47                         <input type="number" class="form-control" id="inventoryCost" name="inventoryCost" value="{{ inventory.inventoryCost }}" step="0.01" min="0" required>
48                     </div>
49                 </div>
50
51             </div>
52
53             <div class="d-flex justify-content-end gap-2 mt-4">
54                 <a href="{{ url_for('index') }}" class="btn btn-outline-secondary">Cancel</a>
55                 <button type="submit" class="btn btn-primary">Update Item</button>
56             </div>
57         </form>
58     </div>
59 </div>
60
61 {% endblock %}

```

update_quantity.html

```
1  {% extends 'base.html' %}
2  {% block content %}
3
4  <div class="card mx-auto" style="max-width:480px;">
5    <div class="card-body">
6      <h3 class="card-title mb-2">Update Quantity</h3>
7      <p class="text-muted">{{ product.productName }}</p>
8
9      <form method="POST">
10        <div class="mb-3">
11          <label for="quantityOnHand" class="form-label">Current Quantity: {{ inventory.quantityOnHand }}</label>
12          <input type="number" class="form-control" id="quantityOnHand" name="quantityOnHand" step="0.01" min="0" value="{{ inventory.quantityOnHand }}" required>
13        </div>
14
15        <div class="d-flex gap-2">
16          <button type="submit" class="btn btn-success">Update Quantity</button>
17          <a href="{{ url_for('index') }}" class="btn btn-outline-secondary">Cancel</a>
18        </div>
19      </form>
20    </div>
21  </div>
22
23  {% endblock %}
24
```