PHENIKAA UNIVERSITY
PHENIKAA SCHOOL OF COMPUTING

SOFTWARE ARCHITECTURE

Lab 6: Introducing the API Gateway Pattern

Instructor: Vũ Quang Dũng
Course: Software Architecture – Class N02
Team Members: Group 11
Nguyễn Trọng Thái – 23010457
Nguyễn Minh Quang – 23010419

Tháng 1 năm 2026

Activity Practice 1: Project Setup and Dependencies

Goal: Create the Gateway project and install the necessary libraries to handle reverse proxying.

1. Create Gateway Directory:

Bash

```bash
# Ensure you are outside the shipment_service directory
mkdir api_gateway
cd api_gateway
python -m venv venv
source venv/bin/activate # Windows: venv\Scripts\activate
pip install Flask requests
touch gateway.py
```

2. Define Service Configuration:

We will point the Gateway to the **Shipment Service** running on port 5001.

File: gateway.py (Configuration)

Python

```python
# Configuration for QuickShip Backend Services
SHIPMENT_SERVICE_URL = 'http://127.0.0.1:5001/api/shipments'
GATEWAY_PORT = 5000
```

Activity Practice 2: Security and Routing Implementation

Goal: Implement centralized token validation and request forwarding logic.

1. Implement Security and Routing:

The Gateway will check for a Bearer token. Only admins can update shipment statuses, while regular users can track their packages.

File: gateway.py (Full implementation)

Python

```python
from flask import Flask, request, jsonify, make_response
import requests

app = Flask(__name__)

# --- SECURITY STUB ---
def validate_token(auth_header):
    """Simulates checking an Authorization token."""
    if not auth_header:
        return False, "Authorization header missing"

    token = auth_header.split("Bearer ")[-1]
    # Acceptable tokens for QuickShip
    if token in ("quickship-admin-key", "quickship-user-key"):
        return True, None
    return False, "Invalid or expired token"

def is_admin(auth_header):
    """Checks if the token belongs to an admin/courier."""
    return auth_header and "quickship-admin-key" in auth_header

# --- ROUTING LOGIC ---
@app.route('/api/shipments', defaults={'path': ''}, methods=['GET',
'POST', 'PATCH'])
@app.route('/api/shipments/<path:path>', methods=['GET', 'POST',
'PATCH', 'DELETE'])
def route_shipment_service(path):
    # 1. SECURITY CHECK
    auth_header = request.headers.get('Authorization')
    is_valid, error_msg = validate_token(auth_header)

    if not is_valid:
        return jsonify({"error": "Unauthorized", "details": error_msg}),
401

    # Admin/Courier check for modifying shipments
(PATCH/POST/DELETE)
    if request.method in ['POST', 'PATCH', 'DELETE'] and not
is_admin(auth_header):
        return jsonify({"error": "Forbidden", "details": "Only
Couriers/Admins can modify shipments"}), 403

    # 2. FORWARDING LOGIC
    # Construct the URL for the Shipment Service from Lab 5
    target_url = f'http://127.0.0.1:5001/api/shipments/{path}'
    if request.query_string:
        target_url += f'?{request.query_string.decode("utf-8")}'

    try:
        response = requests.request(
            method=request.method,
            url=target_url,
            headers={k: v for k, v in request.headers if k.lower() != 'host'},
            data=request.get_data(),
            timeout=5
        )

        # 3. RESPONSE HANDLING
        gateway_response = make_response(response.content,
response.status_code)
        for key, value in response.headers.items():
            if key.lower() not in ['content-length', 'connection']:
                gateway_response.headers[key] = value
        return gateway_response

    except requests.exceptions.RequestException as e:
        return jsonify({"error": "Service Unavailable", "details":
f"Shipment Service is down: {e}"}), 503

if __name__ == '__main__':
    print(f"QuickShip API Gateway running on port
{GATEWAY_PORT}...")
    app.run(port=GATEWAY_PORT, debug=True)
```

Activity Practice 3: Testing the QuickShip Gateway

Goal: Verify the Gateway acts as a secure proxy.

Prerequisites

1. **Start Shipment Service:** Go to shipment_service folder and run python app.py (Port 5001).
2. **Start API Gateway:** Go to api_gateway folder and run python gateway.py (Port 5000).

Test Cases (using cURL):

1. **Test Unauthorized Access (No Token):**
   - **Command:** curl -i -X GET http://127.0.0.1:5000/api/shipments
   - **Expected:** 401 Unauthorized.
2. **Test Authorized Tracking (User Token):**
   - **Command:** curl -i -H "Authorization: Bearer quickship-user-key" http://127.0.0.1:5000/api/shipments/1
   - **Expected:** 200 OK with JSON data for Shipment #1.
3. **Test Forbidden Modification (User attempting Admin task):**
   - **Command:** curl -i -X POST -H "Authorization: Bearer quickship-user-key" http://127.0.0.1:5000/api/shipments
   - **Expected:** 403 Forbidden (User cannot create shipments).
4. **Test Resiliency (Service Down):**
   - **Action:** Stop the Shipment Service (Port 5001).
   - **Command:** curl -i -H "Authorization: Bearer quickship-user-key" http://127.0.0.1:5000/api/shipments
   - **Expected:** 503 Service Unavailable.