bKash Payment Gateway Integration with Flask

This guide provides a step-by-step tutorial for integrating the bKash payment gateway into a Flask application. It includes configuration setup, database model, and implementations for granting tokens, refreshing tokens, creating payments, and executing payments.

Integration Steps

1. Install Required Packages
2. Configure the Flask Application
3. Set Up the Database
4. Create the bKash Token Management Logic
5. Implement Payment Creation
6. Implement Payment Execution
7. Test the Integration
8. Deployment and Security Considerations

Step 1: Install Required Packages

Install Flask, SQLAlchemy, and requests using pip:

pip install flask sqlalchemy requests

Configuration Setup

```
# Configuration (config.py)
class Config:
    BKASH_BASE_URL = "https://api.bkash.com/checkout"
    BKASH_USERNAME = "your_username"
    BKASH_PASSWORD = "your_password"
    BKASH_APP_KEY = "your_app_key"
    BKASH_APP_SECRET = "your_app_secret"

# Ensure to load this configuration in your Flask app:
# app.config.from_object('config.Config')
```

Database Model

```
# Database Model (models.py)
from flask_sqlalchemy import SQLAlchemy
from datetime import datetime

db = SQLAlchemy()
class bkash(db.Model):
    token = db.Column(db.String(5000), primary_key=True)
    generate_time = db.Column(db.DateTime, nullable=False, default=datetime.utcnow)
```

Grant and Refresh Token

```
# Token Management (bkash.py)
import requests
from flask import current_app
from .models import bkash, db
from datetime import datetime, timedelta

def get_bkash_token():
    token_entry = bkash.query.first()
    if token_entry:
        time_diff = datetime.utcnow() - token_entry.generate_time
        if time_diff < timedelta(minutes=59):
            return token_entry.token

    url = f"{current_app.config['BKASH_BASE_URL']}/token/grant"
    headers = {
        'Content-Type': 'application/json',
        'Accept': 'application/json',
        'username': current_app.config['BKASH_USERNAME'],
        'password': current_app.config['BKASH_PASSWORD']
    }
    data = {
```

```
        'app_key': current_app.config['BKASH_APP_KEY'],
        'app_secret': current_app.config['BKASH_APP_SECRET']
    }
    response = requests.post(url, headers=headers, json=data, timeout=30)
    if response.status_code == 200:
        token = response.json()['id_token']
    else:
        raise Exception('Failed to get bKash token')

    if token_entry:
        token_entry.token = token
        token_entry.generate_time = datetime.utcnow()
    else:
        token_entry = bkash(token=token, generate_time=datetime.utcnow())
        db.session.add(token_entry)
    db.session.commit()
    return token

def refresh_bkash_token():
    return get_bkash_token()
```

Create Payment

```
def create_bkash_payment(amount, invoice_number):
    token = get_bkash_token()
    url = f"{current_app.config['BKASH_BASE_URL']}/create"
    headers = {
```

```
        'Content-Type': 'application/json',
        'Authorization': token,
        'X-APP-Key': current_app.config['BKASH_APP_KEY']
    }
    data = {
        'amount': amount,
        'mode': '0011',
        'currency': 'BDT',
        'callbackURL': 'http://127.0.0.1:5000/bkash/callback',
        'intent': 'sale',
        'payerReference': "Be",
        'merchantInvoiceNumber': invoice_number
    }
    response = requests.post(url, headers=headers, json=data, timeout=30)
    if response.status_code == 200:
        return response.json(), token
    else:
        raise Exception('Failed to create bKash payment')
```

Execute Payment

```
def execute_bkash_payment(payment_id, token):
    token = refresh_bkash_token()
    url = f"{current_app.config['BKASH_BASE_URL']}/execute"
    headers = {
        'Content-Type': 'application/json',
        'Authorization': token,
```

```
        'X-APP-Key': current_app.config['BKASH_APP_KEY']
    }
    data = {'paymentID': payment_id}
    response = requests.post(url, headers=headers, json=data, timeout=30)
    if response.status_code == 200:
        return response.json()
    else:
        raise Exception('Failed to execute bKash payment')
```

Test the Integration

Use bKash sandbox credentials to test the integration. Verify the token management and payment

Deployment and Security Considerations

- Use environment variables for sensitive credentials.
- Enable HTTPS for secure communication.
-Set a timeout for API requests to handle unresponsive servers

Conclusion

By following this guide, you can successfully integrate the bKash payment gateway into your Flask application. Ensure to handle token management securely and test the entire flow before deploying to production.