Here is the equivalent Python code using Flask and SQLAlchemy for the provided PHP Laravel function:

```python

from flask import Flask, request, redirect, flash

from flask\_sqlalchemy import SQLAlchemy

import hashlib

import time

import random

app = Flask(\_\_name\_\_)

app.config['SQLALCHEMY\_DATABASE\_URI'] = 'sqlite:///payments.db'

db = SQLAlchemy(app)

class PayNowPayment(db.Model):

id = db.Column(db.Integer, primary\_key=True)

regnum = db.Column(db.String(50), nullable=False)

amount = db.Column(db.Float, nullable=False)

reference = db.Column(db.String(64), nullable=False)

status = db.Column(db.String(50), nullable=False)

browserurl = db.Column(db.String(200))

pollurl = db.Column(db.String(200))

hash = db.Column(db.String(64))

class StudentPackage(db.Model):

id = db.Column(db.Integer, primary\_key=True)

reference = db.Column(db.String(64), nullable=False)

paid = db.Column(db.Integer, nullable=False)

regnum = db.Column(db.String(50), nullable=False)

packageid = db.Column(db.String(50), nullable=False)

@app.route('/store', methods=['POST'])

def store():

data = request.form

regnum = data.get('regnum')

price = data.get('price')

packageid = data.get('packageid')

package = data.get('package')

myname = data.get('myname')

# Validation

if not regnum or not price or not price.isdigit() or float(price) < 1:

flash('Invalid input', 'error')

return redirect(request.url)

unique\_string = hashlib.sha256(f"{time.time()}{random.randint(0, 1000000)}".encode()).hexdigest()

desc = f"{myname} ({regnum})"

narration = f"{package} package payment for {desc}"

# Presave to get a good result URL

payment\_saved = PayNowPayment(

regnum=regnum,

amount=float(price),

reference=unique\_string,

status="created"

)

db.session.add(payment\_saved)

db.session.commit()

paynow = PayNow(

app.config['PAYNOW\_INTEGRATION\_ID'],

app.config['PAYNOW\_INTEGRATION\_KEY'],

f"{app.config['APP\_URL2']}/payconfirmation/{payment\_saved.id}",

f"{app.config['APP\_URL2']}/payment\_return"

)

payment = paynow.create\_payment(narration, 'netecsystems@gmail.com')

payment.add(narration, float(price))

try:

response = paynow.send(payment)

except InvalidIntegrationException as error:

flash('Invalid Integration, contact system admin for support', 'error')

return redirect(request.url)

if response.success():

browser\_url = response.redirect\_url()

poll\_url = response.poll\_url()

status = paynow.poll\_transaction(poll\_url)

hash\_value = response.data()['hash']

# Save the payment to DB

payment\_saved.status = status.status()

payment\_saved.browserurl = browser\_url

payment\_saved.pollurl = poll\_url

payment\_saved.hash = hash\_value

payment\_saved.reference = status.paynow\_reference()

db.session.commit()

if payment\_saved:

student\_package = StudentPackage.query.filter\_by(reference=unique\_string).first()

if student\_package:

student\_package.paid = 0

student\_package.regnum = regnum

student\_package.packageid = packageid

else:

student\_package = StudentPackage(

reference=unique\_string,

paid=0,

regnum=regnum,

packageid=packageid

)

db.session.add(student\_package)

db.session.commit()

return redirect(browser\_url)

flash('Saving payment details failed', 'error')

return redirect(request.url)

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

```

### Explanation:

1. \*\*Flask Setup\*\*: The code uses Flask to handle HTTP requests and SQLAlchemy for database operations.

2. \*\*Validation\*\*: The input validation is done using simple checks.

3. \*\*Hashing\*\*: The `hashlib` library is used to generate a unique string.

4. \*\*Database Models\*\*: `PayNowPayment` and `StudentPackage` are SQLAlchemy models representing the database tables.

5. \*\*PayNow Integration\*\*: The `PayNow` class is assumed to be a Python class that handles payment processing. You would need to implement this class or use an existing library.

6. \*\*Error Handling\*\*: Flash messages are used to display errors, and the user is redirected back to the form in case of errors.

### Notes:

- The `PayNow` class and its methods (`create\_payment`, `send`, `poll\_transaction`, etc.) are placeholders. You need to implement these or use an existing payment processing library.

- The `InvalidIntegrationException` is assumed to be a custom exception that you need to define or handle appropriately.

- The `APP\_URL2` and `PAYNOW\_INTEGRATION\_ID`, `PAYNOW\_INTEGRATION\_KEY` should be configured in your Flask app's configuration.

Here’s the equivalent Python code for the provided PHP Laravel function using Flask and SQLAlchemy:

```python

from flask import Flask, request, jsonify, abort

from flask\_sqlalchemy import SQLAlchemy

import sys

app = Flask(\_\_name\_\_)

app.config['SQLALCHEMY\_DATABASE\_URI'] = 'sqlite:///payments.db'

db = SQLAlchemy(app)

class PayNowPayment(db.Model):

id = db.Column(db.Integer, primary\_key=True)

regnum = db.Column(db.String(50), nullable=False)

amount = db.Column(db.Float, nullable=False)

reference = db.Column(db.String(64), nullable=False)

status = db.Column(db.String(50), nullable=False)

pollurl = db.Column(db.String(200))

class StudentPackage(db.Model):

id = db.Column(db.Integer, primary\_key=True)

reference = db.Column(db.String(64), nullable=False)

paid = db.Column(db.Integer, nullable=False)

regnum = db.Column(db.String(50), nullable=False)

packageid = db.Column(db.String(50), nullable=False)

@app.route('/handle\_payment\_confirmation/<int:id>', methods=['POST'])

def handle\_payment\_confirmation(id):

payment = PayNowPayment.query.get\_or\_404(id)

reference = ''

status = ''

try:

# Assuming `paynow` is an instance of a payment processing class

poll = paynow.poll\_transaction(payment.pollurl)

payment.status = status = poll.status()

reference = payment.reference

db.session.commit()

except Exception as e:

db.session.rollback()

return abort(500, str(e))

# Check if the payment is successful

if status == 'Paid':

# Update the payment status in the database

payment.status = 'Paid'

db.session.commit()

# Update the student package status

student\_package = StudentPackage.query.filter\_by(reference=reference).first()

if student\_package:

student\_package.paid = 1

db.session.commit()

return jsonify({'message': 'Payment confirmed and saved successfully'}), 200

else:

# Handle unsuccessful payment

return jsonify({'error': 'Payment was not successful'}), 400

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

```

### Explanation:

1. \*\*Flask Setup\*\*: The code uses Flask to handle HTTP requests and SQLAlchemy for database operations.

2. \*\*Database Models\*\*: `PayNowPayment` and `StudentPackage` are SQLAlchemy models representing the database tables.

3. \*\*Error Handling\*\*: The `try-except` block is used to catch exceptions during the payment polling process. If an exception occurs, a 500 error is returned with the exception message.

4. \*\*Payment Status Check\*\*: The payment status is checked, and if it is `'Paid'`, the payment and student package statuses are updated in the database.

5. \*\*Response\*\*: A JSON response is returned indicating whether the payment was successful or not.

### Notes:

- The `paynow` object is assumed to be an instance of a payment processing class that has a `poll\_transaction` method. You need to implement this class or use an existing library.

- The `poll\_transaction` method should return an object with a `status` method that provides the payment status.

- The `abort` function is used to return a 500 error with the exception message if something goes wrong.

- The `jsonify` function is used to return JSON responses.

This code should be adapted to fit your specific requirements and environment.