Project Report
Title: Digital Payment Book

Contents:

1. INTRODUCTION

1.1. Overview

We design a python flask web application of digital payment book.This web app helps the user for the payment of the products. In this application user dashboard display the users payment details and their payment status.And also admin maintain the payment database of the users payment details and also intimate the users of payment alert when the due date crossed by sending email alert.

1.2. Purpose

The main purpose of this application is display the payment details of the customer that helps they to make the payment as soon as possible.And also it makes useful to retailer to maintain the users payment details of the purchased products.

2. LITERATURE SURVEY

2.1. Present Problem

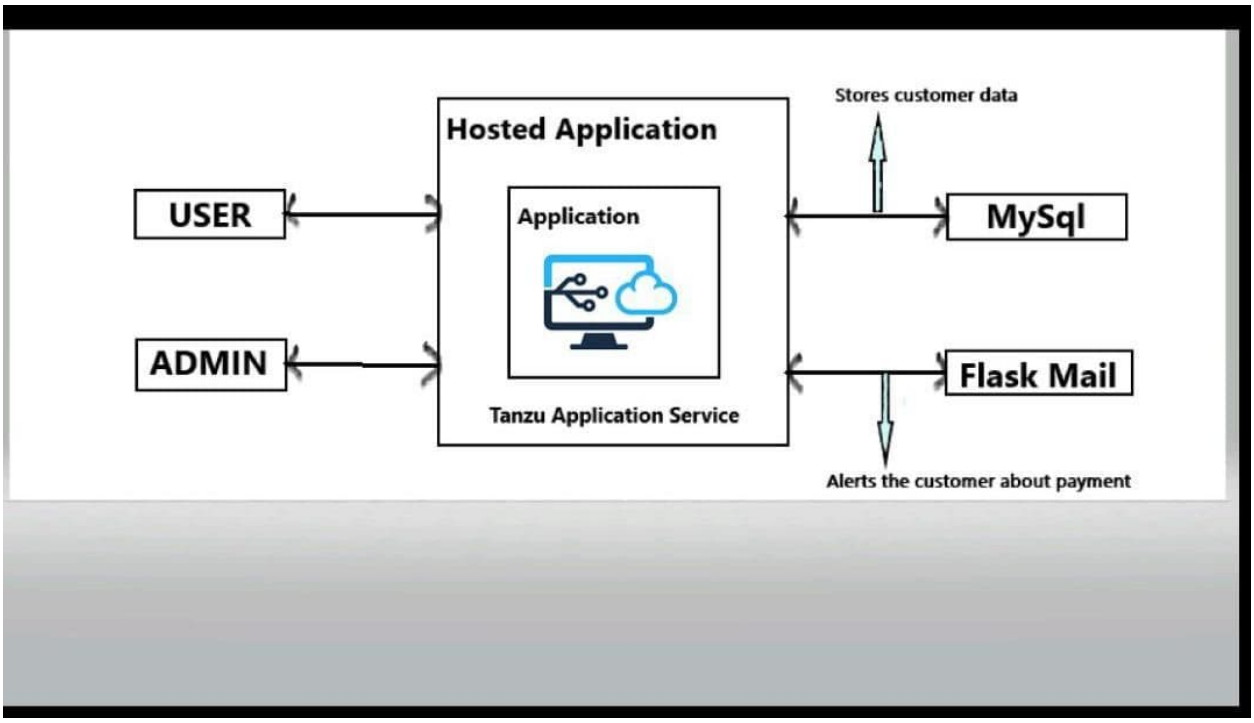
The customer has go to the shop and purchase the products and pay for the products.This process has come to end that time. The problem is the retailer has no idea about the customer liked products as a human being easily forget the purchase history and there was less bonding with customer with retailer.

2.2. Proposed Solution

This projects deals with customer purchased detailes and makes retailer to better understand the customer and also intimate the customer about payment beyond duedate.This projects make the retailer to maintain the customer purchased the data and it also used in future also.

3. THEORITICAL ANALYSIS

3.1. Block Diagram



3.2. Hardware/Software Requirements

In This projects we use python flask as a backend and HTML,CSS used as frontend development of the web application.For database connectivity, we use remotemysql.com as database service .After completing the projects make the web app into Docker image for deployment of Tanzu Application Services.

4.PROCEDURAL ANALYSIS

- Creating a flask project :

To first create python Flask app backend for user portal and admin portal by using python3 and import flask library.

- Creating Database:

Go to remotemysql.com

And sign up to create the account for database creativity and it also provide separate database host name and password for better

connectivity.

- Front End development:

Create Html amd CSS for content of the web pages which was suitable for the project. In this project we use Jinja2 template used as a user interface between frontend and backend.

- Create Docker Image:

Download Docker tool ao hub.docker.com

Create account at hub.docker.com

Then create Dockerfile for converting Docker image

Use command "**docker build -t docker image name .**" to build the docker image.

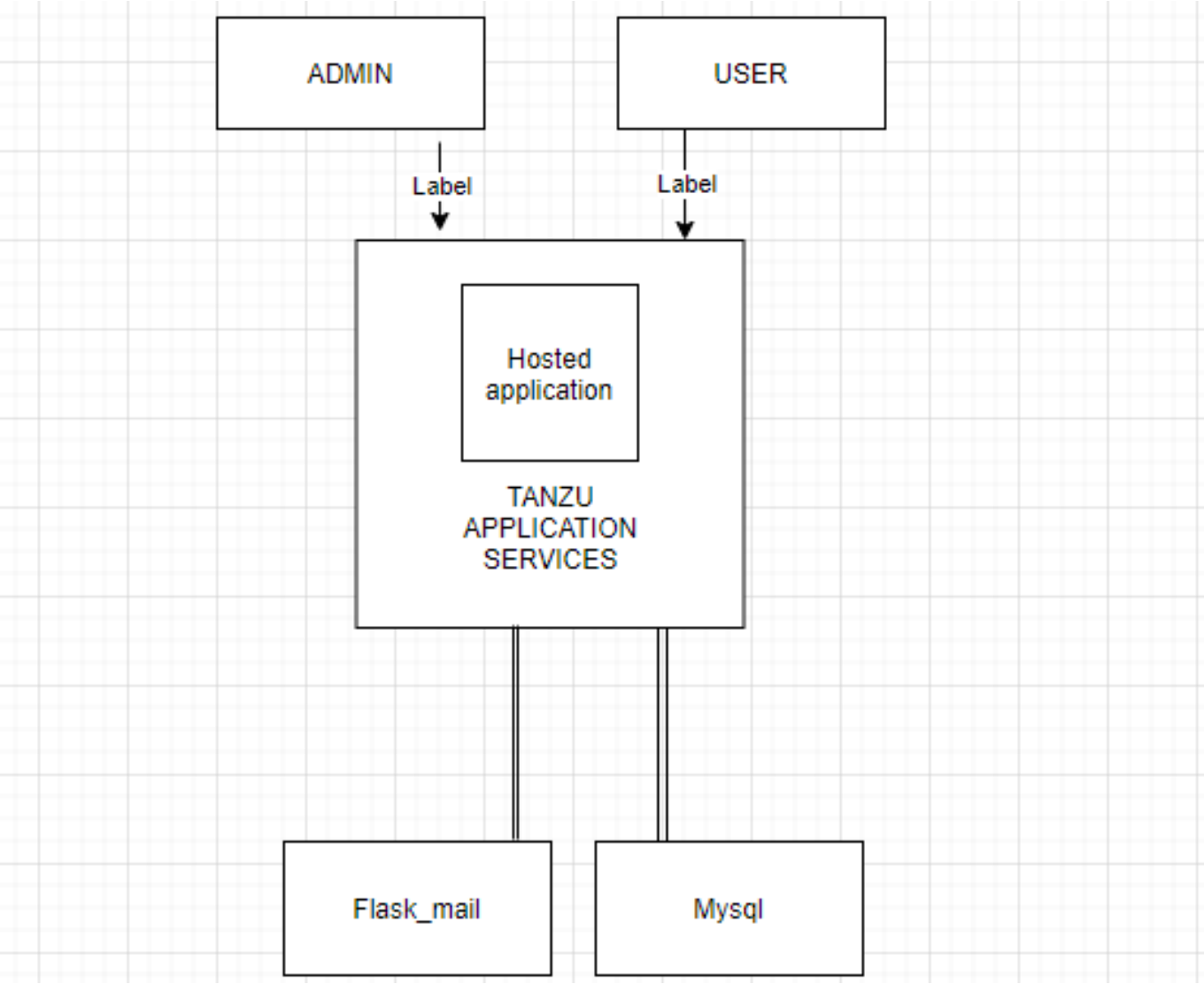
- Docker image Deployment at TANZU application services:

After Docker image converting , download clf tool at cloud native foundation for Tanzu deployment.

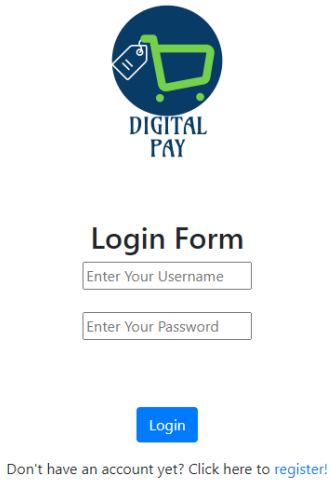
open the command prompt put api url name and username password for tanzu service accesss.

After authenticating deploy the Docker image toTANZU application services

5. FLOWCHART


ADMIN
USER
Label
Label
Hosted application
TANZU APPLICATION SERVICES
Flask_mail
Mysql

6. RESULTS



7. ADVANTAGES AND DISADVANTAGES:

Advantages:

- Multiple customer Handling possible
- 24/7 availability
- Easily understandable

Disadvantages:

- Payment not process on online mode

8. APPLICATION:

Used as in education, medical filed, other business involvement.

9. CONCLUSION:

Thus , by using this application make retailer do remember the customer purchase details and make use for future business also.Customer has also remember the purchase details and payment date to pay for the purchased products.

10. FUTURE SCOPE:

The customer accurate details are displayed are future scope.

And make retailer manage the customer payment are also future scope of the app.

11. BIBLIOGRAPHY:

Reference links:

- https://code.tutsplus.com/tutorials/creating-a-web-app-from-scratch-using-python-flask-and-mysql--cms-22972
- https://blog.miguelgrinberg.com/post/designing-a-restful-api-with-python-and-flask
- https://ckraczkowsky.medium.com/building-modern-user-interfaces-with-flask-23016d453792
- https://www.askpython.com/python-modules/flask/flask-mysql-database
- https://www.geeksforgeeks.org/dockerize-your-flask-app/
- https://drive.google.com/file/d/1Y7V9XBm1iCsC37Fjv2UO6fnyGPZJcE6r/view?usp=sharing

APPENDIX

SOURCE CODES:

User portal:

```
from flask import Flask, render_template, request, redirect, url_for, session, flash
import re
import MySQLdb.cursors
from flask_mysqldb import MySQL



app = Flask(__name__)

app.secret_key = 'kolaaa'


app.config['MYSQL_HOST'] = 'remotemysql.com'
app.config['MYSQL_USER'] = 'FBYfzJrlLk'
app.config['MYSQL_PASSWORD'] = '8G5IXLpLrY'
app.config['MYSQL_DB'] = 'FBYfzJrlLk'
mysql = MySQL(app)


@app.route('/')
def home():
```

```
    return redirect(url_for('login'))

@app.route('/login', methods =['GET', 'POST'])
#users login
def login():
    global userid
    msg = ''


    if request.method == 'POST' :
        username = request.form['username']
        password = request.form['password']
        cursor = mysql.connection.cursor()
        cursor.execute('SELECT * FROM users WHERE username = % s AND password = %
s', (username, password ))
        account = cursor.fetchone()
        print (account)
        if account:
            session['loggedin'] = True
            session['id'] = account[0]
            userid=  account[0]
            session['username'] = account[1]
            msg = 'Logged in successfully !'

            msg = 'Logged in successfully !'
            return redirect(url_for('dashboard', msg=msg))
        else:
            msg = 'Incorrect username / password !'
    return render_template('login.html', msg = msg)

#users registeration
@app.route('/register', methods =['GET', 'POST'])
def register():
    msg = ''
    if request.method == 'POST' :
        username = request.form['username']
        email = request.form['email']
```

```
        password = request.form['password']


        cursor = mysql.connection.cursor()
        cursor.execute('SELECT * FROM users WHERE username = % s', (username, ))
        account = cursor.fetchone()
        print(account)
        if account:
            msg = 'Account already exists !'
        elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):
            msg = 'Invalid email address !'
        elif not re.match(r'[A-Za-z0-9]+', username):
            msg = 'name must contain only characters and numbers !'
        else:
            cursor.execute('INSERT INTO users VALUES (NULL, % s, % s, % s)', (username,
email,password))
            mysql.connection.commit()
            msg = 'You have successfully registered !'

    elif request.method == 'POST':
        msg = 'Please fill out the form !'
    return render_template('register.html', msg = msg)
#users dashboard
@app.route('/dashboard', methods =['GET'])
def dashboard():
    print(session["username"],session['id'])

    cursor = mysql.connection.cursor()
    cursor.execute('SELECT * FROM pay WHERE userid = % s', (session['id'],))
    account = cursor.fetchone()
    print("accountdislay",account)


    return render_template('dashboard.html',account = account)
```

```
  #contact our services
@app.route('/contact')
def contact():

    return render_template('contact.html')
#logout
@app.route('/logout')
@app.route('/logout')
def logout():
   session.pop('loggedin', None)
   session.pop('id', None)
   session.pop('username', None)
   return render_template('login.html')


if __name__ == '__main__':
   app.run(host='0.0.0.0',debug = True, port = 8080)
ADMIN PORTAL:
#app0.py
from flask import Flask, render_template, request, redirect, url_for, flash
from flaskext.mysql import MySQL
import pymysql
import smtplib
from flask_mail import Mail, Message

app = Flask(__name__)
app.secret_key = "admin"

mysql = MySQL()

# MySQL configurations
app.config['MYSQL_DATABASE_USER'] = 'FBYfzJrlLk'
app.config['MYSQL_DATABASE_PASSWORD'] = '8G5IXLpLrY'
app.config['MYSQL_DATABASE_DB'] = 'FBYfzJrlLk'
app.config['MYSQL_DATABASE_HOST'] = 'remotemysql.com'
mysql.init_app(app)
```

```
@app.route('/')
def home():
    return redirect(url_for('admin_login'))
#admin_login page
@app.route('/admin_login', methods=['GET', 'POST'])
def admin_login():
    error = None
    if request.method == 'POST':
        if request.form['username'] != 'admin' or request.form['password'] != 'admin':
            error = 'Invalid Credentials. Please try again.'
        else:
            return redirect(url_for('admin_panel'))
    return render_template('admin_login.html', error=error)
#admin_dashboard
@app.route('/admin_panel')
def admin_panel():
    conn = mysql.connect()
    cur = conn.cursor(pymysql.cursors.DictCursor)

    cur.execute('SELECT * FROM pay')
    data = cur.fetchall()

    cur.close()
    return render_template('admin_panel.html', pay = data)

@app.route('/add_contact', methods=['POST'])
#add user payement details
def add_pay():
    conn = mysql.connect()
    cur = conn.cursor(pymysql.cursors.DictCursor)
    if request.method == 'POST':
        username = request.form['username']
        products = request.form['products']
        cost = request.form['cost']
        duedate = request.form['duedate']
        payment_status = request.form['payment_status']
        cur.execute("INSERT INTO pay (username, products, cost, duedate,
```

```
payment_status) VALUES (%s,%s,%s,%s,%s)", (username, products, cost, duedate,
payment_status))
        conn.commit()
        flash('Payment Added successfully')
        return redirect(url_for('admin_panel'))

 #edit user payment details
@app.route('/edit/<userid>', methods = ['POST', 'GET'])
def get_pay(userid):
    conn = mysql.connect()
    cur = conn.cursor(pymysql.cursors.DictCursor)

    cur.execute('SELECT * FROM pay WHERE userid = %s', (userid))
    data = cur.fetchall()
    cur.close()
    print(data[0])
    return render_template('edit.html', pay = data[0])
 #update user payment details
@app.route('/update/<userid>', methods=['POST'])
def update_pay(userid):
    if request.method == 'POST':
        username = request.form['username']
        products = request.form['products']
        cost = request.form['cost']
        duedate = request.form['duedate']
        payment_status = request.form['payment_status']
        conn = mysql.connect()
        cur = conn.cursor(pymysql.cursors.DictCursor)
        cur.execute("""
            UPDATE pay
            SET username = %s,
                products = %s,
                cost = %s,
                duedate = %s,
                payment_status = %s
            WHERE userid = %s
        """, (username, products, cost, duedate, payment_status, userid))
```

```
        flash('Payment Updated Successfully')
        conn.commit()
        return redirect(url_for('admin_panel'))
 #delete the user payment details
@app.route('/delete/<string:userid>', methods = ['POST','GET'])
def delete_pay(userid):
    conn = mysql.connect()
    cur = conn.cursor(pymysql.cursors.DictCursor)

    cur.execute('DELETE FROM pay WHERE userid = {0}'.format(userid))
    conn.commit()
    flash('Payment Removed Successfully')
    return redirect(url_for('admin_panel'))

app.config['MAIL_SERVER']='smtp.gmail.com'
app.config['MAIL_PORT'] = 465
app.config['MAIL_USERNAME'] = '<youremail>@gmail.com'#put your eamil account
app.config['MAIL_PASSWORD'] = '' #put your email password
app.config['MAIL_USE_TLS'] = False
app.config['MAIL_USE_SSL'] = True
app.config['MAIL_DEFAULT_SENDER'] = '<youremail>@gmail.com'
mail = Mail(app)
#send message to users for due_payment
@app.route('/sendmail', methods=['GET', 'POST'])
def sendmail():
    if request.method == 'POST':
        recipient = request.form['recipient']
        msg = Message('Digitalpay', recipients=[recipient])
        msg.body = ('Dear customer, please pay the pending payment with '
                    'Dgitalpay')
        msg.html = ('<h1>Digitalpay</h1>'
                    '<p>Dear customer, please pay the pending payment with '
                    '<b>Digitalpay</b>!</p>')
        mail.send(msg)
        flash(f'A test message was sent to {recipient}.')
        return redirect(url_for('sendmail'))
    return render_template('sendmail.html')
```

```
# starting the app
if __name__ == "__main__":
    app.run(host='0.0.0.0',debug = True, port = 8080)
```