Faith Njambi Muturi

DBT 1303 Software Development

Development in Python Report

DBITNRB618618

Context/Background

You have been recently tasked with a software development project to produce a software product that enables users to monitor the security within their neighbourhood following a recent spate of muggings and house break-ins.

Justification for the chosen Agile methodology to conduct the project

Agile project management is a highly collaborative methodology that uses short development cycles called “sprints,” each of which incorporates and adapts to stakeholder and customer feedback in an effort to embrace continuous improvement and produce an expertly honed end product. The reason why agile project management has become so popular in recent times is partly due to the fast-paced nature of business today. With its focus on continued evolution and collaboration, the methodology targets organizations dealing with rapid to-market deadlines, shifting priorities, high stakeholder engagement and a need for flexibility—in other words, most businesses today! For example, rather than spend six months developing a product or service that may be outdated by the time it hits the market, a business using agile project management could release the first iteration within two weeks and continue to release updated, adaptive versions over the next six months, resulting in a much more effective, relevant and useful final deliverable. That’s why agile project management, which was originally developed for software companies, has been adopted by a wide variety of industries today, from financial services to transportation. These are reasons why I chose to use agile methodology.

A. Higher product quality. Because testing is integrated throughout the project development process, the team is able to perform regular checkups and find areas of improvement on an ongoing basis.

B. Reduced risk. Agile project management essentially eliminates the chances of absolute project failure. Working in sprints from the very beginning allows teams to develop a working product from the beginning, or fail fast and take another approach.

C. Better visibility into project performance. Agile project management allows team members to know how project progress is going. Frequent scrum meetings and sprint reviews provide increased transparency to everyone on the team.Increased project controlMore opportunities to test and adapt allow all team members to have more control throughout every stage of the project.

D. Better project predictability. Breaking up the project into shorter sprints allows project managers to better predict the exact cost, timeline, and resource allocation necessary for each sprint.

Requirements modelling by presenting a use case and class diagram for identified requirements after requirements elicitation

1. Class Diagram

2. Use Case Diagram

Explanation on your chosen software design with reference to SOLID principles

Single Responsibility Principle - The Single Responsibility Principle states that every module or class should have responsibility over a single part of the functionality provided by the software.

Open/Closed Principle - The open/closed principle states that software entities (classes, modules, functions, etc.) should be open for extensions, but closed for modification.

Liskov Substitution Principle - More generally it states that objects in a program should be replaceable with instances of their subtypes without altering the correctness of that program.

Interface Segregation Principle- In programming, the interface segregation principle states that no client should be forced to depend on methods it does not use.Do not add additional functionality to an existing interface by adding new methods. Instead, create a new interface and let your class implement multiple interfaces if needed. Dependency Inversion Principle - In programming, the dependency inversion principle is a way to decouple software modules. This principle states that

● High-level modules should not depend on low-level modules. Both should depend on abstractions.

● Abstractions should not depend on details. Details should depend on abstractions.

Phase Two

● Pseudocode of your python program

from flask import Flask, render\_template, flash, redirect, url\_for, session, request, logging

from flask\_mysqldb import MySQL

from wtforms import Form, StringField, TextAreaField, PasswordField, validators

from passlib.hash import sha256\_crypt

from functools import wraps

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

# Config MySQL

app.config['MYSQL\_HOST'] = '127.0.0.1'

app.config['MYSQL\_USER'] = 'root'

app.config['MYSQL\_PASSWORD'] = ''

app.config['MYSQL\_DB'] = 'hoodwatch'

app.config['MYSQL\_CURSORCLASS'] = 'DictCursor'

# init MYSQL

mysql = MySQL(app)

# Index

@app.route('/')

def index():

return render\_template('home.html')

# About

@app.route('/about')

def about():

return render\_template('about.html')

# ContactTracing

@app.route('/recent\_breakings')

def breakings():

# Create cursor

cur = mysql.connection.cursor()

# Get reported recent\_breakings

result = cur.execute("SELECT \* FROM reported\_breakings")

breakings = cur.fetchall()

if result > 0:

return render\_template('breakings.html', breakings=breakings)

else:

msg = 'No Responses Captured'

return render\_template('breakings.html', msg=msg)

# Close connection

cur.close()

#Single contact tracing Response

@app.route('/reported\_breakings/<string:id>/')

def breakings(id):

# Create cursor

cur = mysql.connection.cursor()

# Get Breakings Record

result = cur.execute("SELECT \* FROM reported\_breakings WHERE id = %s", [id])

breakings = cur.fetchone()

return render\_template('breaking.html', breakings=breakings)

# Register Form Class

class RegisterForm(Form):

name = StringField('Name', [validators.Length(min=1, max=50)])

username = StringField('Username', [validators.Length(min=4, max=25)])

email = StringField('Email', [validators.Length(min=6, max=50)])

password = PasswordField('Password', [

validators.DataRequired(),

validators.EqualTo('confirm', message='Passwords do not match')

])

confirm = PasswordField('Confirm Password')

# User Register

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

def register():

form = RegisterForm(request.form)

if request.method == 'POST' and form.validate():

name = form.name.data

email = form.email.data

username = form.username.data

password = sha256\_crypt.encrypt(str(form.password.data))

# Create cursor

cur = mysql.connection.cursor()

# Execute query

cur.execute("INSERT INTO users(name, email, username, password) VALUES(%s, %s, %s, %s)", (name, email, username, password))

# Commit to DB

mysql.connection.commit()

# Close connection

cur.close()

flash('You are now registered and can log in', 'success')

return redirect(url\_for('login'))

return render\_template('register.html', form=form)

# User login

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

def login():

if request.method == 'POST':

# Get Form Fields

username = request.form['username']

password\_candidate = request.form['password']

# Create cursor

cur = mysql.connection.cursor()

# Get user by username

result = cur.execute("SELECT \* FROM users WHERE username = %s", [username])

if result > 0:

# Get stored hash

data = cur.fetchone()

password = data['password']

# Compare Passwords

if sha256\_crypt.verify(password\_candidate, password):

# Passed

session['logged\_in'] = True

session['username'] = username

flash('You are now logged in', 'success')

return redirect(url\_for('dashboard'))

else:

error = 'Invalid login'

return render\_template('login.html', error=error)

# Close connection

cur.close()

else:

error = 'Username not found'

return render\_template('login.html', error=error)

return render\_template('login.html')

# Check if user logged in

def is\_logged\_in(f):

@wraps(f)

def wrap(\*args, \*\*kwargs):

if 'logged\_in' in session:

return f(\*args, \*\*kwargs)

else:

flash('Unauthorized, Please login', 'danger')

return redirect(url\_for('login'))

return wrap

# Logout

@app.route('/logout')

@is\_logged\_in

def logout():

session.clear()

flash('You are now logged out', 'success')

return redirect(url\_for('login'))

# Dashboard

@app.route('/dashboard')

@is\_logged\_in

def dashboard():

# Create cursor

cur = mysql.connection.cursor()

result = cur.execute("SELECT \* FROM reported\_breakings WHERE name = %s", [session['username']])

Responses = cur.fetchall()

if result > 0:

return render\_template('dashboard.html', Responses=Responses)

else:

msg = 'No Breakings Reported'

return render\_template('dashboard.html', msg=msg)

# Close connection

cur.close()

# form

class ArticleForm(Form):

name= StringField('Full Name', [validators.Length(min=1, max=200)])

address = StringField('Hood Address', [validators.length(min=1, max=200)])

description = StringField('Breaking Description',[validators.length(min=1,max=15)])

# Add report Breaking

@app.route('/report\_breaking', methods=['GET', 'POST'])

@is\_logged\_in

def add\_question():

form = ArticleForm(request.form)

if request.method == 'POST' and form.validate():

name = form.name.data

address = form.address.data

description = form.description.data

# Create Cursor

cur = mysql.connection.cursor()

# Execute

cur.execute("INSERT INTO reported\_breakings(name, address, description) VALUES(%s, %s, %s)", (session['username'], address, description))

# Commit to DB

mysql.connection.commit()

#Close connection

cur.close()

flash('Crime Response Submitted', 'success')

return redirect(url\_for('dashboard'))

return render\_template('report\_breaking.html', form=form)

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

app.secret\_key='secret123'

app.run(debug=True)

● Link to your Python program (source code) stored in Github saved using your student number then unit code.

The source codes of this software are hosted at github under This Repository

https://github.com/Fai-njambi/python-hoodwatch