Department of Electrical & Computer Engineering

North South University

Test Plan

Project Name: Burger Wagon

Submitted by: Team12

Abdullah All Noman Abir (ID: 1521047042)

Mohammed Taufiqur Rahman (ID: 1521285042)

GitHub Link: https://github.com/nsuspring2019cse427/Group12

Course: CSE427 (Software Quality Assurance & Testing)

Section: 01

Semester: Spring 2019

Submitted to: Shaikh Shawon Arefin Shimon (SAS3)

Introduction

Burger Wagon is a favorite online delivery restaurant serving burgers. In this project we created and tested a REST backend that allows users to view the food menu online and manage orders. For unit testing purpose, we followed the Test Driven Development (TDD) method. That means- we firstly wrote the API functions only with basic signatures. Then we designed and created the unit tests which initially failed. After that we developed the API functions to pass the tests.

Testing Aspects We Implemented

We have successfully implemented following tasks as testing aspects:

1. Unit testing each API functions.
2. Input space partitioning.
3. Graph partitioning.
4. Integration testing.
5. Functionality testing.

Requirements

1. Python 3.7
2. Flask micro framework

Tools/Frameworks Used:

1. Pycharm
2. Default Python unit test module

Input Space Partitioning:

Input variables used in this project are:

- Id (integer).

- Title (String)

- Description (String)

- Price (float)

- Date created (function generated)

According to characteristics Input Space Partitioning:

Characteristics	Block 1	Block 2	Block 3
String	Alphabetic value	Numerical value	null
Float	<1.0	>=1.0	Invalid float type

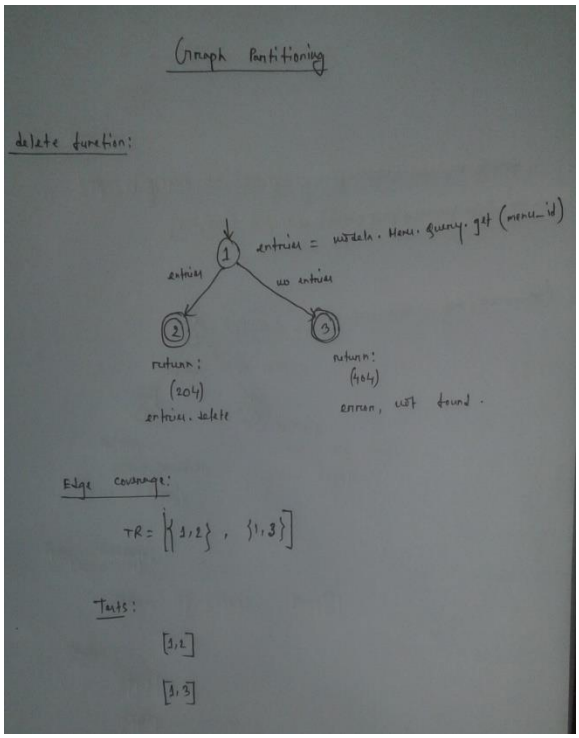
Sample values:

Characteristics	Block 1	Block 2	Block 3
String	“Apple burger”, "pet kharap hobe na 50%"	“57983479”	null
Float	0, -10.1	25 ,1200.00	"APPLE"

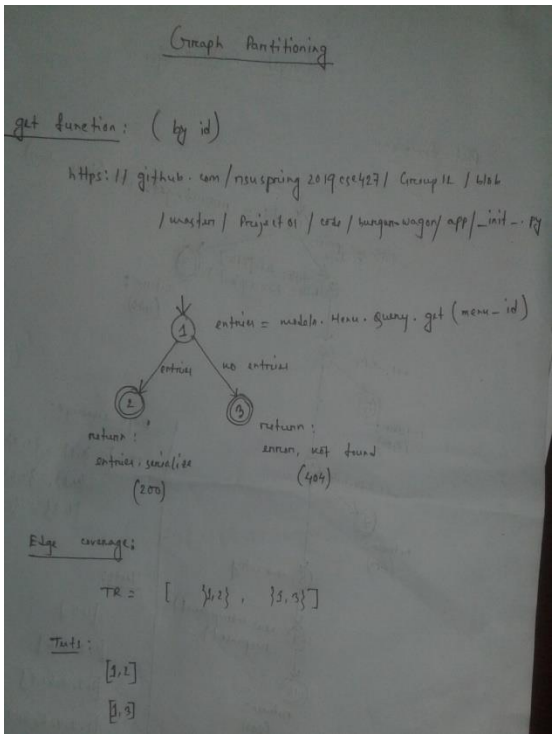
In the given github link -https://github.com/nsuspring2019cse427/Group12, under tests directory in these following files- **test_put_input_space_partitions.py** and **test_post_input_space_partitions.py**; input space partitioning codes are available in the given criterion.

Graph Partitioning: For graph partitioning we followed the edge coverage criteria. Firstly, we created graphs for all the API functions-get, post, put, delete and wrote the test requirements for covering edges. Then based on the test requirements we decided the test paths to be covered. After that we updated the existing unit test functions with graph partitioning and then wrote unit tests for the test paths which weren't covered before graph partitioning. In the given github link - https://github.com/nsuspring2019cse427/Group12, under tests directory with every unit test graph partitioning is documented through comments.

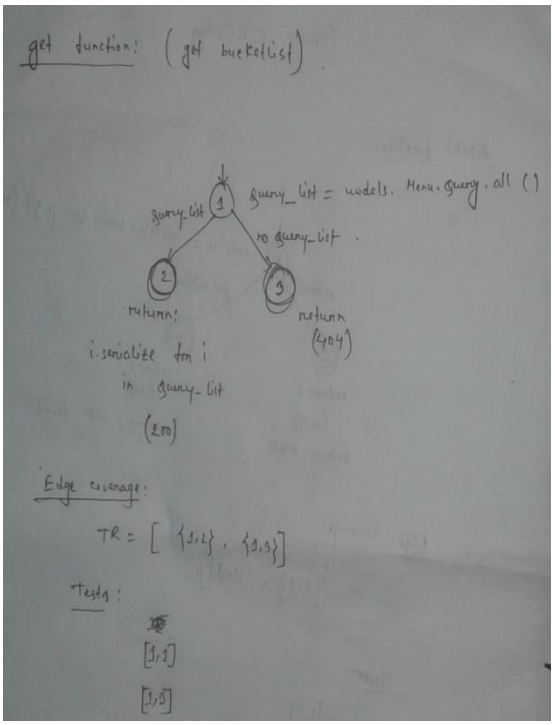
The corresponding graphs with test requirements and test path:

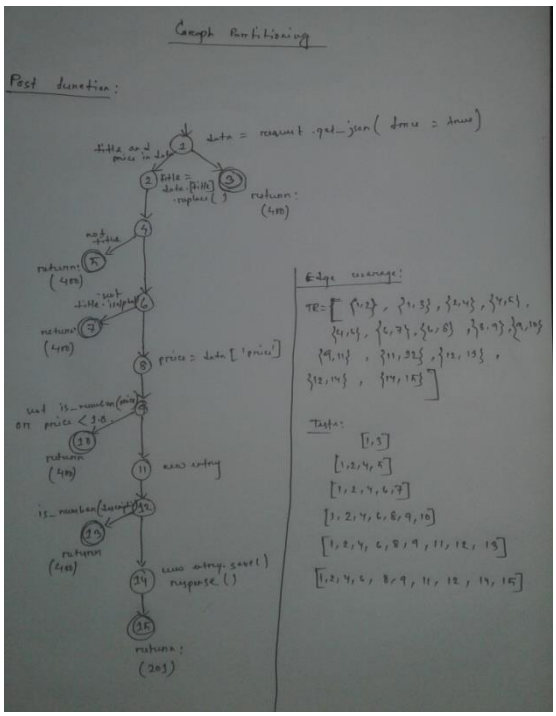


Delete function graph

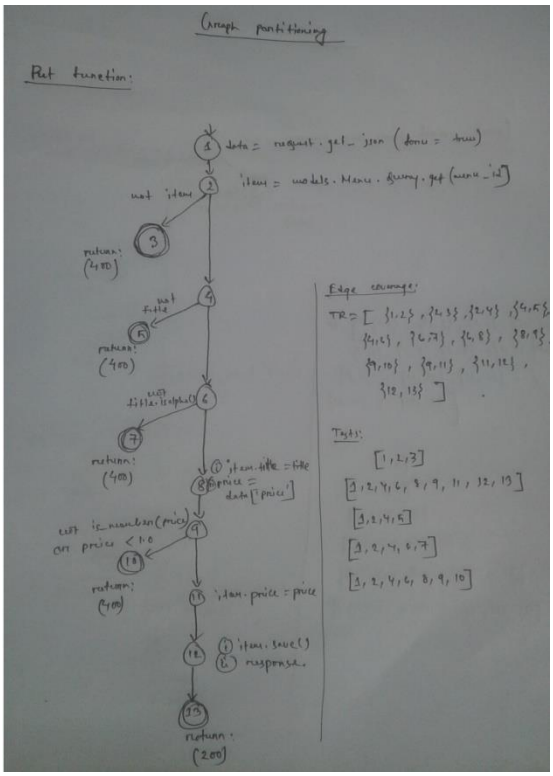


Get function(by id) graph





Get functin(for getting list) graph **Post function graph**



Put function graph

Integration Test: When we initially designed and wrote unit tests ,we already did cover integration testing. For example: to test delete function we had to post inputs, then delete it and after all that had to use get function to test wheather the item got deleted.That means in a single function we had to use these three API functions to test which is basically integration testing.For all other unit tests we wrote also comes under this rule.

Functional Test: For functional testing ,we tested these following tests:

a.Smoke test.

b.Unit test.

c. Integration test.

UI test: Initially,As we developed the backend of a website from scratch for testing purpose;we didn't developed any user interface.So, we couldn't perform any sort of UI testing.