Introduction

Welcome to Module 10! This week, we will learn about quality assurance and creating applications that are concerned with performance, design, reliability, and maintainability. Test-Driven development is an iterative process where you write code, test, refactor and re-test. The principal benefit is to find deficiencies (bugs) in your code early. By knowing of deficiencies early, resources in the form of time can be allocated in order to refactor and correctly implement business logic.

Topics and Learning Objectives

Topics

- Quality Assurance
- Test-Driven Development
- Refactoring
- DevOps
- Unit Testing

Learning Objectives

By the end of the module, you should be able to:

1. Apply the basic steps to test a web application.
2. Explain why testing web applications is important.
3. Set up continuous integration with 3rd party tools such as GitHub, CircleCI and Heroku.

Readings & Resources

Reading

Required

- Grinberg, M. (2018). *Flask Web Development: Developing Web Applications with Python* (2nd ed.). O'Reilly Media, Inc.
  - Chapter 15: Testing
- Facebook. (2022). Jest – Testing react apps.
- Thoughtbot, Inc. (2022). Fundamentals of TDD.

- Fowler, M. (2018). The Practical Test Pyramid.

Quality Assurance

- QA (quality assurance) is the term used to describe the systematic efforts taken to ensure that products delivered to customers meet with the contractual obligations of features such as performance, design, reliability, and maintainability.
- The core purpose of QA is to prevent mistakes and defects in development and production.
- QA includes two principles
    a. "**Fit for purpose**" – the software should be suitable for the intended purposes
    b. "**Right first time**" – mistakes should be eliminated during design and development stages

QA vs. Testing

- QA is a set of methods and activities designed to ensure that the development software conforms to all specifications (see Figure 10.1 below). It includes:
    - testing
    - quality control
    - quality assurance
- QA deals with management.
- Testing is a way of exploring the system to check how it operates and find possible defects. Various methods are used to test software.
- Testing enables stakeholders to see if the software meets their expectations on design, functionality, and maintenance.

Figure 10.1. The QA pyramid.

Source: Github, Flaskr – QATestLab Blog

Test-Driven Development

- TDD (Test-Driven Development) is an iterative development cycle that emphasizes writing automated tests before writing the actual feature or function (see Figure 10.2 below).
- The steps include:
    a. Add a test
    b. Ensure the test fails
    c. Write code
    d. Ensure the test passes
    e. Refactor
- TDD combines building and testing.
- TDD helps ensure correctness of code and indirectly evolve the design and architecture of the project.

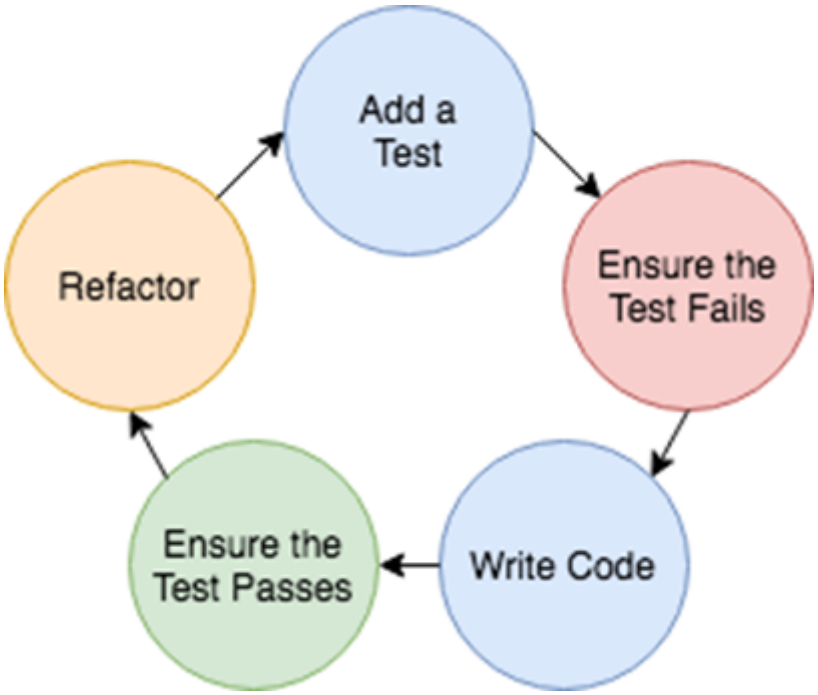

Figure 10.2. Test-Driven Development steps.

Source: Flaskr – Intro to Flask, Test-Driven Development and JavaScript

Red-Green-Refactor Phases

- The red phase is the starting point of the red-green-refactor cycle. The purpose of this phase is to write a test that informs the implementation of a feature. The test will only pass when the requirements are met.
- The green phase is where you implement code to make your test pass. The goal is to find a solution without worrying about optimizing your implementation.
- In the refactor phase you are still in the green. Now you can think about how to implement the code better or efficiently.

TDD Benefits

- TDD encourages writing testable, loosely-coupled code that tends to be modular. Modular code is easier to write, debug, maintain and reuse.
- TDD has the following properties:
    - Reduce costs
    - Make refactoring and rewriting easier and faster
    - Prevent bugs and coupling
    - Improve overall team collaboration

- Increase confidence that the code works as expected
- Improve code patterns
- Eliminate fear of change

TDD Is an Iterative Process

- TDD is an iterative process.
- Writing a perfect test will take time and may not be written the first time around.
- First write a quick test to test functionality and get feedback.
- Refactor it in order to have better testing of functionality.
- Refactoring may take several steps; hence, it is an incremental or iterative process.

TDD Approaches

- There are two approaches to TDD
  1. Inside out
     - The focus is on results or state.
     - Testing begins at the smallest unit level as the architecture emerges organically.
     - This approach is easier for beginners.
     - Design happens at the refactor stage. A consequence can be large refactoring.
  2. Outside in
     - The focus is on user behaviour.
     - Testing begins at the outermost level and details emerge as you work your way in.
     - This approach relies heavily on stubbing external dependencies.
     - It is harder to learn but it ensures that code aligns with business needs.
     - Design happens at the RED stage.

Test Failures are Important

It is important to see tests fail, and tests pass in TDD.

Observing test failures demonstrates the following:

1. Validates that the new test is meaningful and unique, helping to ensure that the implemented code is not only useful but necessary as well.

2. Provides an end goal, something for you to aim for. This focuses your thinking so that you write just enough code to meet required goals.

DevOps

- DevOps is a set of practices that combines software development (Dev) and IT operations (Ops).
- DevOps aims to shorten the system development life cycle and provide CD with high software quality.
- DevOps is complementary with Agile software development.
- DevOps does not have a universal definition. The current usage is a cross-functional combination of terms and concepts for “development” and “operations.”
- A de-facto definition is “a set of practices intended to reduce the time between committing a change to a system and the change being placed into normal production, while ensuring high quality.”

CircleCI

- CircleCI is a CI/CD platform that can be used to implement DevOps practices.
- CircleCI monitors GitHub, GitHub Enterprise, and Atlassian Bitbucket repositories and launches builds for each new commit. CircleCI automatically tests builds in either docker containers of virtual machines and deploys passing builds to target environments. A dashboard and API allows tracking the status of builds and metrics related to builds.
- Review the Quickstart Guide documents for getting started.

High Level vs Low Level Test Cases

- **High Level Test Cases**
  - High level test cases define the functionality of a software in a broader way without going into deep functionality.
  - The main advantage is that a tester is not bound to follow test cases step by step and thus it gives a chance to explore more edge cases. This increases the chance to find bugs.
  - The main disadvantage is that it is difficult to work or write high level test cases.
- **Low Level Test Cases**
  - Low level test cases define the functionality of a software in deep way. These test cases generally include details like “expected results” and “test data.”
  - The main advantage is that a tester is unlikely to miss bugs and it is easy for inexperienced testers to work with these test cases as the expected results and test data is available.

- The main disadvantage is that low level test cases are tedious and time consuming to work with.

Performing High Level Test cases using RESTer

- A single high level test case may be equivalent to many low level test cases. In small it is desirable to use high level test cases due to lack of resources.
- RESTer is a REST client for any web service. RESTer is a plugin for Firefox or Chorme web browsers.
- RESTer may be used to implement high level test cases.
- RESTer may be used to invoke a URI endpoint in your web service. Furthermore, you can submit different parameters to your URI endpoint in order to test standard cases, edge cases, error cases and impossible cases.
- Postman is a service that fulfills the same task as RESTer.

Flask-Testing

- Flask-Testing is an extension that provides unit testing for Flask. Flask-Testing may be installed with pip.

```
pip install Flask-Testing
```

- Flask-Testing uses OOP.
- You write test cases by subclassing TestCase.

```
from flask_testing import TestCase
class MyTest(TestCase):
        pass
```

Writing a Flask Unit Test

A valid unit test requires a create_app() method. If this method is missing a NotImplementedError will be raised.

```
from flask import Flask
from flask_testing import TestCase

class MyTest(TestCase):
def create_app(self):
    app = Flask(__name__)
    app.config['TESTING'] = True
    return app
```

Testing Live Servers

- If you want to test done via Selenium or other headless browsers like PhantomJS you can use the LiveServerTestCase.
- Note the url, http://localhost:8943.

```
import urllib2
from flask import Flask
from flask_testing import LiveServerTestCase

class MyTest(LiveServerTestCase):

    def create_app(self):
    app = Flask(__name__)
    app.config['TESTING'] = True
    # Default port is 5000
    app.config['LIVESERVER_PORT'] = 8943
    # Default timeout is 5 seconds
    app.config['LIVESERVER_TIMEOUT'] = 10
    return app

    def test_server_is_up_and_running(self):
    response = urllib2.urlopen(self.get_server_url())
    self.assertEqual(response.code, 200)
```

Automate Unit Testing

- It is recommend you put all your tests into one file so that you can use the unittest.main() function.
- unittest.main() function will discover all your test methods in your TestCase classes. Remember, the names of the test methods and classes must start with test (case-insensitive) so that they can

be discovered.

- Create a file named test.py with the following code. You can run this file to run all unit tests.

```
import unittest
import flask_testing

# your test cases

if __name__ == '__main__':
unittest.main()
```

Video

Please watch the following video.

Introduction to Quality Assurance
TMU Video

This video demonstrates how to unit test your web application.

Summary

Quality assurance focuses on creating applications that are concerned with performance, design, reliability, and maintainability. Test-Driven development is an iterative process where you write code, test, refactor, re-test, and repeat the previous steps till the test passes. The principal benefit is to find deficiencies (bugs) in your code early. By knowing deficiencies early, resources in the form of time can be allocated in terms of personnel and time in order to correctly implement business logic.

Assessments

Each week, you'll find here reminders of assignments or labs that you should be working on. For Week 10:

- Lab 10: Automated testing for web applications (5% of the final grade) is due in 7 days, i.e., the end of day Friday of Week 10.

Reminder

You have a weekly Zoom session. The date/timing for your Zoom session can be found in the Course Schedule in the Course Outline. The link to the Zoom session will be provided separately to you by your instructor. The Zoom session will consist of 20–30 minutes of lecture, followed by questions and answers (Q&A), followed by a lab period. You may ask questions about the previous or current week's content or labs.

Please bring any questions to the weekly Zoom session. However, if you have any questions during the week outside of the Zoom session, you may post them in the Discussion Board. Your instructor may respond in the Discussion board or during the weekly Zoom session.

Click-n-reveal: **Where is the Discussion board?**

Click on the "Communication" area at the top main course menu, and select Discussions from the dropdown menu.

References

None for this week.