Test Plan

Describes the testing framework for the Concrete Jungle Tours webapp. On delivery the app should include two types of testing: unit testing, and automated E2E testing. The automated segment must include regression testing. The developers will be writing the unit tests as they deliver their user stories aiming for 100% coverage of the business logic and validation logic layers.

# Tasks

The unit tests for the Python with Flask backend will done in Pytest and are expected to perform isolated tests from the database using a mock fixture. The backend unit tests will make use of Pytest mocker and gauge the coverage metric using Pytest Cover library. The E2E testing will represent the result of the BDD approach using a JAVA powered automation test framework, that uses Maven as a dependency manager. The tests will be written in Cucumber directly using the test case lingo and automated using Selenium. The final deliverable will contain the necessary setup of the continuous integration/continuous deployment and delivery (CI/CD) using Jenkins automation server.

# SCOPE

## General

The test suite will include unit tests on the backend and end-to-end tests on the frontend.

### Testing Strategy

The traceability matrix will allow the testing for all requirements and include a Business Requirement ID Number, a Requirement Description, a Test Case ID, and an Implementation Status. The test cases document will include Test Case ID, Test Case Scenario, Test Case Steps, Test Case Data, Expected Results, Actual Results, Status, Defect ID, and Comments.

### Unit testing

For every test case in every user story there must be a unit test. The results will be included in the test case document and in case of defects it must include the Defect ID and Comments categories. Unit tests will target atomic portions of the code responsible for user validation values and feature flow control.

### Methodology

Unit tests will be written in Python using pytest-mock and pytest-cover libraries.

### End-to-End Testing

Every test case will have a correspondent End-to-End black box testing of the frontend.

### Methodology

The automated test framework will be written in Java. Maven will be used for managing dependencies and plugins. The test automation framework will make use of the Surefire plugin to implement TestRunner classes which will bind the glue code inside the steps definitions containing Selenium instructions and Page Object Model objects with the Gherkin scenario descriptions in the feature files.

## Features to be tested

### MVP Features

### Login

### Registration

### Display Tours

### Display Guide Tours

### Add Tour

### Update Tour

### Delete Tour

### View Guides

#### Feature: Login

1. Scenario: Successful login
2. Scenario: Invalid email
3. Scenario: Invalid password
4. Scenario: Blank email
5. Scenario: Blank password
6. Scenario: Invalid email and password

#### Feature: Registration

1. Scenario: Signing up successfully
2. Scenario: Blank email
3. Scenario: Blank password
4. Scenario: Blank first name
5. Scenario: Blank last name
6. Scenario: Blank phone
7. Scenario: Email already taken
8. Scenario: Email invalid format
9. Scenario: Name invalid format
10. Scenario: Phone invalid format
11. Scenario: Password invalid format

#### Feature: Display Tours

1. Scenario: Signing up successfully with a user account and loading the landing page

#### Feature: Display Guide Tours

1. Scenario: Signing up successfully with a guide account and loading the landing page

#### Feature: Add Tour

1. Scenario: Add a tour using valid data
2. Scenario: Add a tour without selecting a valid number of points of interest
3. Scenario: Blank price
4. Scenario: Blank title

#### Feature: Update Tour

1. Scenario: Update successfully using valid data.
2. Scenario: Update a tour without selecting a valid number of points of interest
3. Scenario: Blank price
4. Scenario: Unselected tour status

#### Feature: Delete Tour

1. Scenario: Click the delete option

#### Feature: View Guides

1. Scenario: Click the Our Guides on any page

### Resources, Roles, and Responsibilities

Every team member is responsible for setting up their own test environment and handle their respective share of tests.

### Deliverables

Test plan, test case document, and traceability matrix.

### Tools

IntelliJ, Pycharm, Postman, Pytest, Cucumber, Selenium, AWS, Jenkins.