Test Plan

> Goal: Implement a simple-to-use and robust testing strategy that will ensure no breaking changes will be pushed to production.

Requirements:

1. Must include test strategy, functional/component test descriptions (goals, expectations)
2. Test metrics including success criteria; tools and methodology to be used, … etc.

Test Strategy

Within our Git strategy, for every commit that is pushed to the master branch, a build will be triggered by Jenkins that will run all of the unit tests. If any of the tests has failed, the build will fail and notify all of the team members.

Our test strategy lies on three pillars: **Simplicity**, **Reliability**, **Agility**

1. **Simplicity** - test design needs to be simple enough in order to encourage developers to write unit tests for the functionality they've implemented.
2. **Reliability** - test strategy needs to be implemented with a great sense of importance, it should be working flawlessly as it doesn't exist.
3. **Agility** - each test should be responsibly for testing one thing and one thing only, test cases should be flexibly and easily modifieble.

Test Components

- Store Location Recommendation System
- Online Shopping Website
- Delivery Tracking System (DTS)
- Inventory Management System

For each of the above, we will have a superclass that will define the necessary functionality and load resources needed to be used by its subclasses. This functionality represents **Simplicity** and **Agility** principles.

Use Case Example: Developer has added functionality to the inventory management system which required an additional table in the `dev` environment. Developer will create a test class to cover this functionality by extending `InventorySystemSuite` and overrinding one of the methods to verify that the table with the same name also exists in the `prod` environment.

Online Shopping Website

> Goal: Ensure the website's availability is at 99%.

Core functionality to be tested:

- Availability of each page - should be no `404` errors
- Resources availability - pictures, banners, etc.
- Payment System functionality
- Login system with dummy credentials

Store Location Recommendaton System

> Goal: Ensure the system is producing expected results with dummy data and the API is available to the online shoppping website.

Core functionality to be tested:

- API Availability - verify with a dummy request.
- Functionality correctness - test with dummy data, system's response should match a dummy prediction.

Delivery Tracking System (DTS)

> Goal: Ensure that DTS is working as expected as a front-end feature.

Core functionality to be tested:

- Google API Availability - test whether API credentials are correct and dummy calls are succesful.

Inventory Management System

> Goal: Have a reliable and flexible inventory management system.

Core functionality to be tested:

- Tables Availability - make sure the production environment contains the exprected tables used by the backend.
- New Item functionality - verify, with dummy data, correct insertion of new item data to the database.
- Expiried item functionality - grocery items that expiried should not be displayed on the website.

Test Metrics

Success Criteria

All of the unit tests must be passed in order for the build to be considered successful.

Tools

As of now, Python's `unittest` testing framework is the tool of choice to facilitate unit testing.

Jenkins will be used to CI/CD operations.

Methology

As part of th eAgile methology, we have a dedicated team member responsible for QA. In the backlog, we will have tickets related to testing, especially before each release.

TTD (Test-driven Development) methology can be used in the Store Location Recommendaton System.