Test Plan Document

OCTOBER 10, 2022

BBB (Better Business Bureau)
Authored by: Data Divas (Medhanit Asrat Bekele, Mohammed Ahnaf Khalil, Wengel Tsegaselassie, Wen Sun)

TABLE OF CONTENTS

1. Introduction

Our project consists of improving the data quality by automating the process of finding and correcting problems in BBB's database. Our client provided us with a CSV file that contains more than 150,000 business along with their necessary information. Since that file is too large, we decided to take a tranche of the data to test our code. Our tranche consists of 100 rows which were extracted from the CSV file. We chose 100 rows because it was the ideal size to cover most test cases without impeding execution. To test our code, we implemented a series of functions that we will go more in depth in this document.

2. Objectives and Tasks

The objective of writing a test plan is to evaluate the code to determine whether it does what we expect it to do. Our test plan documents the testing strategies that will be used to ensure testing activities are effectively implemented within the Data Divas team. The plan defines the testing scope, testing cases, testing strategy, testing tools & environment that support the project deliverables, team members, and contributors.

3. Scope

3.1. Features to be tested

SyntaxChecker.py

Checks the syntax of each URL and attempts to correct it:

1) **check_syntax(url)**

   This function uses regex to check whether a URL is syntactically correct or not. Our function takes a URL as a parameter and returns true if the URL matches the regex pattern and returns false otherwise. We will be testing whether our function successfully recognizes URLs with incorrect syntax or not.

2) **fix (url)**

   This is a function that takes in the URL with incorrect syntax and tries to fix it. It utilizes re to fix some errors that are detected within the URL.

3) **verify(url)**

   This method checks if the URLs that have gone through the fixing process have actually been fixed. It first checks if the URL is valid, if not it tries to fix it and checks if it actually has been fixed. It returns true and the URL if the URL is valid in the first place, returns true and the fixed URL if the URL has been fixed and is now valid, and returns false and the URL if the fixing process didn't work and the URL is invalid.

StatusCodeChecker.py

Checks the status code of a give URL.

1) **get_statuscode(lst)**

   Gets the status code of the list of urls using threading. It sends a maximum of 70 (requests) threads at a time to maximize speed.

2) **getStatusCode(url)**

   This is a function that checks the status code of each URL in our tranche. Our function accepts URL as a parameter and returns the status code that is associated with that URL. We will be testing whether our code returns the correct status code for each URL passed or not.

URLVerification.py

This is a Python file that reads each URL from the tranche and uses functions from SyntaxChecker.py and StatusCodeChecker.py to analyse the problem.

1) **opener(**base_dir, file_name, b_id_list, url_id_list, url_list**)**

   Opens a csv file that contains all a list of the URLs of the businesses.

2) **write(**base_dir, file_name, lst**)**

   Writes analyzed URL into new file.
   Also contains main method that analyzes client data using functions from other modules.

Generate_Url.py

This is a Python file to try to generate the URL based on the company information from the Csv file.

1) **appends**(company_name, tld, url=[])

   Appends company name to a general URL format

2) **clean_data**(companyName):
   A function that cleans companyName(changes to lowercase, and all remove all white spaces as well as remove any legal designation eg: ',LLC'

3) **validURl**(url):
   A function that checks if found url is valid
4) **getURL(**company_name, Url, rating_sites)
   Return company's URL given company name and state
5) **filter**(url, rating_sites)
   A function that checks if found url are rating sites
6) **status_code**(url)
   A function that checks if found url are rating sites
7) **scrap**(url, companyName, words)
   Gets a single url and returns the status code
8) **correctUrl**(url, company_name, info)
   A function that picks the url with the most count out of the list of potential urls passed
9) **company_URL**(information)
   A function that gets the correct url utilizing all the functions above)

badUrls.py

This is a Python file looking for businesses with broken URLs and businesses with no URLs in the database.

1) **http_error**(base_dir, fileName, bad_bid=[])
   A function that extracts businesses with http error
2) **syntax_error**(base_dir, fileName, bad_bid)

   A function that extracts businesses with syntax error
3) **no_Url**(base_dir, fileName)
   A function that extracts businesses with no urls and writes them into new csv
4) **indentify_business**(base_dir, fileName, bad_bid)
   A function that identifies businesses with bad syntax or http error give their business id and writes them into a file
5) **write**(base_dir, file_name, lst)
   Writes analyzed url into new file

ADDURL.py

This is a Python file that reads each URL information and uses functions from generate_url.py to add urls to the database

1) **reads(**base_dir, file_name**)**

   Reads company information form csv and writes a new file with a potential url included

Also contains main method that analyzes client data using functions from other modules.

4. Testing strategy

We will follow unit testing in this project. We will perform unit and integration testing on all functions.

4.1. Unit testing

- Unit testing is the process in which the smallest testable parts of a project are tested independently and individually to ensure that each unit performs as intended.
- We will utilize unit testing for all the functions to verify the expected outcome.

4.2. Integration testing

- Integration testing involves testing all the functions in the program as a group to see whether they function together seamlessly.
- We will use integration testing as we go on in the project.

5. Testing tools and environment

5.1. Hardware requirements

- For this test we will be using a personal computer with an 11th Gen Intel Core i7 3.30GHz processing power.

5.2. Software requirements

- Install pytest package
- Install pandas package to get the tranche of data

5.3. Workstation

- Pycharm

- SQL management studio
- Microsoft Excel

6. Test Schedule

Test	Start date	End date	Description
Unit test	10/3/2022	10/05/2022	Testing each function separately
Integration test	10/05/2022	10/08/2022	Testing the entire system
Unit test	10/11/2022	10/14/2022	Retesting each function separately after adding partial solutions.
Integration test	10/17/2022	10/21/2022	Retesting the entire system for our whole solution after adding partial solutions.
Unit test	10/24/22	10/28/22	Retesting each function after increasing the tranche into 250
Integration test	10/31/22	11/04/22	Testing the entire system after increasing the tranche
Unit test	11/10/2022	11/15/2022	Retesting each function with the tranche containing 250 rows separately after adding partial solutions.
Integration test	11/15/2022	11/18/2022	Retesting the entire system for our whole solution after adding new partial solutions.

7. Roles and responsibilities

The Testing Lead, Wen Sun, will serve as the primary contact for testing on this project. However, all other team members (Wengel Tsegaselassie, Medhanit Asrat Bekele & Mohammed Ahnaf Khalil) are equally responsible for testing. The following are some responsibilities:

- Understand test requirements
- Develop test conditions, test cases, expected results, and execution scripts
- Perform execution and validation
- Identify, document, and prioritize defects according to the guidance provided by the test lead
- Re-test after source code modification have been made according to the schedule
- Prepare testing metrics and provide regular status

8. Dependencies

The data we are testing is a replica of the data stored in BBB's database (Blue database). Since our team only has read-only access to BBB's database and don't want the modifications we make to go live, we are unable to test it on their database ourselves. The solution for this is to send our documents to their database administrator who would run it themselves, after verifying it is safe.

9. Risk and contingencies plan

Risk	Probability	Influence	Contingency plan
SCHEDULE Tight time limits	High	High	Follow the development schedule. Timely notification of potential problems or shifts in the schedule. Run unit tests before the deadline and run integration tests after it.
RESOURCES Resources onboarding too late	Medium	High	Work with whatever we have until full resource access.
DEFECTS Defects discovered at a late stage	Medium	High	Make plans to quickly fix defects and prioritize test code.

SCOPE Scope not completely defined	Medium	Medium	Have a meeting with client and faculty coach to redefine the scope of the meeting.
LEARNING Gaps in domain knowledge	Medium	High	Use online resources to learn or reach out to technical specialist or faculty coach.
OTHER ISSUES Delayed testing due to other issues	Medium	High	Implement unit testing to make sure minimum requirements are met. Try testing as thorough as possible after that but make sure to have a test plan so that next team can run the test.

10. Testing criteria

10.1. Entry criteria

The entry criteria is a set of conditions or requirements, which are required to be fulfilled or achieved to create a suitable & favorable condition for testing. The following criteria listed could be negotiated and discussed among the team as well as client if they are not met.

- Tranche of a bigger database should be available for easier and faster testing.
- All hardware and software requirements listed above in section 5 should be fulfilled.
- Complete or partially testable code should be available.
- Test cases should be developed and ready to be tested.

10.2. Exit criteria

The exit criteria specify the conditions and requirements that are required to be achieved or fulfilled before the end of software testing process.

- All test cases should be executed.
- All the identified defects should be corrected and closed.
- No critical defects should be left out.

11. Approvals

Approver/Reviewer	Role	Date of Approval/ Reviewal
Ryan Sharp	Main Client	
Eli Johnson	Technical Specialist	
Lin Chase	Faculty Coach	