Test Plan

Michigan Treasury Local Government Data Parser

[Adham Abdalla](mailto:adhama@umich.edu),

Hasan Alameh,

[Tanis Daniels-Wanamaker](mailto:tanisd@umich.edu),

[Ahmed Mawari](mailto:afmawari@umich.edu)

**1.0 Introduction**

This section provides an overview of the entire test document. This document describes both the test plan and the test procedure.

**1.1 Goals and objectives**

The objective of testing the software is to see if the software meets the requirements assigned when it is put to work. The goal of the software is to extract the required data values from given PDF files containing fiscal treasury values within the counties of Michigan.

**1.2 Statement of scope**

The scope of the software testing is to test the functionality and features of the software. The functionalities and features being tested are:

* The software will take in a given PDF file from the user
* The software scan through the file and extract the required information within the PDF
* The software will skip any unnecessary information that isn’t needed
* The software will generate all the extracted values within a csv file
* The software will generate a csv file that can be saved on the computer

**1.3 Major constraints**

A business constraint that may impact the way the software is to be tested is that the pdf must be readable and cannot be a scanned image, there is a small-time frame to complete the testing, and the program doesn’t use a database server, so the program is not generating its maximum efficiency.

**2.0 Test Plan**

This section describes the overall testing strategy and the project management issues that are required to properly execute effective tests.

**2.1 Software (SCIís) to be tested**

The software to be tested is identified by name. Exclusions are noted explicitly.

* Interface components:
  + File select
  + Open output file
* System components:
  + PDF data extract
  + Data cleansing
  + Excel export

**2.2 Testing strategy**

**2.2.1 Unit testing**

The main testing strategy to be used is brainstorming the possible main test cases with their expected outputs, then running the test through the program to see whether the results are as expected.

The program outputs a CSV file based on the PDF the user inputs, so the components tested will involve the PDF, program, and output file.

**2.2.2 Integration testing**

We will input a PDF, submit it, and have the CSV file open for viewing the results.

**2.2.3 Validation testing**

For validation, we will use unit testing and performance validation. This will help us evaluate the program’s reliability.

**2.2.4 High-order testing**

High-order testing is the next step of module testing. Unit testing, integration testing, validation, and performance testing all ensure the fulfillment of client requirements and criteria.

**2.3 Testing resources and staffing**

Testing staffing: Hasan, Adham, Tanis, and Ahmed

Resources: Windows PC, WIP version of the program, sample PDF files, and Excel to view the CSV.

**2.4 Test work products**

The test work product will be a comma-separated CSV file, expected to be ordered and organized.

**2.5 Test record keeping**

Test results will be added to the test table, with a sample of the output added.

**2.6 Test metrics**

For performance testing, a timer will be used to calculate the approximate time the program needs to output the results.

For the output results, the user’s personal evaluation of the CSV output will determine its success.

**2.7 Testing tools and environment**

The program will run on a Windows 10 PC with Python 3.7 installed. Visual Studio Code will be used to compile and run the code. The CSV will be viewed on Excel and Notepad.

**2.8 Test schedule**

The flow will be as follows:

Unit Testing -> Integration Testing -> Validation Testing -> High-order Test

**3.0 Test Procedure**

This section describes a detailed test procedure including test tactics and test cases for the software.

**3.1 Software (SCIís) to be tested**

The software to be tested is identified by name. Exclusions are noted explicitly.

* Interface components:
  + File select
  + Open output file
* System components:
  + PDF data extract
  + Data cleansing
  + Excel export

**3.2 Testing procedure**

The overall procedure for software testing is described.

**3.2.1 Unit test cases**

The procedure for unit testing is described for each software component (that will be unit tested) is presented. This section is repeated for all components i.

**3.2.1.1 File Select**

The file select functionality will be tested in a white box manner. Pre-selected audit PDFs will be loaded into the program.

**Stubs and/or drivers for component**

* None

**Test cases component**

* Pre-selected audit PDFs will be loaded into the program and readied for processing
* Selecting a different file before processing will use the most recently-selected file

**Purpose of tests for component**

* The purpose of these tests is to ensure the correct file is used when selected by the user

**Expected results for component**

* The interface is expected to perform in accordance with the design specifications.

**3.2.1.2 Open Output File**

The open output file functionality will be tested in a white box manner. Pre-selected audit PDFs will be processed through the program and opened through the interface.

**Stubs and/or drivers for component**

* None

**Test cases component**

* The output of the most recently-processed audit data will be opened through the program interface

**Purpose of tests for component**

* The purpose of these tests is to ensure the correct file output is opened when selected by the user

**Expected results for component**

* The interface is expected to perform in accordance with the design specifications.

**3.2.1.3 PDF Data Extract**

The PDF data extraction will be tested in a white box manner. Pre-selected audit PDFs will be loaded into the program and be extracted accordingly.

**Stubs and/or drivers for component**

* None

**Test cases component**

* The program will extract all relevant data from the table-like formats of the pre-selected audit PDFs
* The program will properly store the raw extracted data to be cleansed
* The program will extract the correct data regardless of formatting differences between different audits

**Purpose of tests for component**

* The purpose of these tests is to ensure the relevant audit data is correctly extracted from the raw PDFs

**Expected results for component**

* The component is expected to perform in accordance with the design specifications.

**3.2.1.4 Data Cleansing**

The data cleansing functionality will be tested in a white box manner. Pre-selected audit PDFs will be loaded into the program and their extracted data will be cleansed for output.

**Stubs and/or drivers for component**

* None

**Test cases component**

* Extra and unnecessary characters will be removed from the final output data
* The cleansed data figures and attribute names will be properly stored into individual cells in the corresponding rows and columns

**Purpose of tests for component**

* The purpose of these tests is to ensure the output is consistent and ready to be analyzed in Excel

**Expected results for component**

* The component is expected to perform in accordance with the design specifications.

**3.2.1.5 Excel Export**

The Excel export will be tested in a white box manner. Pre-selected audit PDFs will be loaded into the program and processed, with the relevant data exported to an Excel file.

**Stubs and/or drivers for component**

* None

**Test cases component**

* The relevant cleansed data attributes are exported as a .csv Excel file

**Purpose of tests for component**

* The purpose of these tests is to ensure the data is properly exported by the program

**Expected results for component**

* The component is expected to perform in accordance with the design specifications.

**3.2.2 Integration testing**

**3.2.2.1 Testing procedure for integration**

For Integration testing, a top-down approach will be used. We will test our cases on the high level interface first. For failed tests, we will break down into lower level components to find the issues.

**3.2.2.2 Stubs and drivers required**

Submit button: submits PDF into next stage of parsing

Page lists: keeps track of pages to be parsed

**3.2.2.3 Test cases and their purpose**

Running program: making sure there are no compile errors

Browsing/selecting PDF: test whether the file path is correctly stored and navigated to

Submitting: testing for errors with opening the PDF

Checking CSV file: making sure the CSV is correctly written to

**3.2.2.4 Expected results**

Running program: the code is compiled with no error messages

Browsing/selecting PDF: The GUI shows the file path correctly

Submitting: The code runs and closes with no error messages/exceptions thrown

Checking CSV file: CSV file is formatted correctly

**3.2.3 Validation testing**

**3.2.3.1 Testing procedure for validation**

The team will run the software with predetermined input data. Testing

every UI element and component of the software. The team will utilize

a test log to keep track of every test scenario.

**3.2.3.3 Expected results**

The team has sample input with highlighted desired output by the client. The team will use those samples for expected results.

**3.2.3.4 Pass/fail criterion for all validation tests**

Every test must produce the predetermined expected result in order to pass.

**3.2.4 High-order testing (a.k.a. System Testing)**

**3.2.4.1 Recovery testing**

The software does not save data, it only processes data into an output file, this is not applicable.

**3.2.4.2 Security testing**

No authentication is required to use the software as it was not required by the client nor is it necessary for the use case of the software.

**3.2.4.3 Stress testing**

The software will be tested with large size pdf files to stress test it.

**3.2.4.4 Performance testing**

The team will test for the efficiency of the software with multiple different formats of pdf files.

**3.2.4.5 Alpha/beta testing**

Alpha will test runs on developer hardware.

beta will be test runs on client hardware.

**3.2.4.6 Pass/fail criterion for all validation tests**

stress test: no time out or crashes

performance: no memory leak or excessive use of resources. Also, reasonable wait times.

**3.3 Testing resources and staffing**

Testing staffing: Hasan, Adham, Tanis, and Ahmed

Resources: Windows PC, WIP version of the program, sample PDF files, and Excel to view the CSV.

**3.4 Test work products**

The test work product will be a comma-separated CSV file, expected to be ordered and organized.

**3.5 Test record keeping and test log**

Test results will be mainly stored in a word file containing the test table.