SE 3XA3: Test Plan
MAC Schedule Importer

Team 12, Team 0C
Cassandra Nicolak, nicolace
Michelle Leung, leungm16
Winnie Liang, liangw15

October 25, 2018

Contents

1 General Information **1**
- 1.1 Purpose . . . 1
- 1.2 Scope . . . 1
- 1.3 Acronyms, Abbreviations, and Symbols . . . 1
- 1.4 Overview of Document . . . 1

2 Plan **2**
- 2.1 Software Description . . . 2
- 2.2 Test Team . . . 2
- 2.3 Automated Testing Approach . . . 2
- 2.4 Testing Tools . . . 2
- 2.5 Testing Schedule . . . 2

3 System Test Description **2**
- 3.1 Tests for Functional Requirements . . . 2
  - 3.1.1 Testing for User Input . . . 2
  - 3.1.2 Testing for Exporting from Mosaic . . . 3
  - 3.1.3 Testing for Importing to Google Calendar . . . 4
  - 3.1.4 Testing for Google API Handling . . . 4
- 3.2 Tests for Nonfunctional Requirements . . . 4
  - 3.2.1 Area of Testing1 . . . 4
  - 3.2.2 Area of Testing2 . . . 5
- 3.3 Traceability Between Test Cases and Requirements . . . 5

4 Tests for Proof of Concept **5**
- 4.1 Parser . . . 5
- 4.2 Link to Google Calendar API . . . 6

5 Comparison to Existing Implementation **6**

6 Unit Testing Plan **6**
- 6.1 Unit testing of internal functions . . . 6
- 6.2 Unit testing of output files . . . 7

7 Appendix **8**
- 7.1 Symbolic Parameters . . . 8
- 7.2 Usability Survey Questions? . . . 8

List of Tables

1 Revision History ii
2 Table of Abbreviations 1
3 Table of Definitions 1

List of Figures

Table 1: **Revision History**

Date	Version	Notes
2018-10-25	1.0	Rough Draft
2018-10-26	1.1	Final Draft

1 General Information

1.1 Purpose

The purpose of the test plan is to build confidence in the implementation correctness for the project.

1.2 Scope

The scope of the test plan is to provide a basic testing platform for the correctness and functionality of the software. The test plan includes a description of the testing procedures, tools and test cases that will be implemented to verify and validate the software for this project.

1.3 Acronyms, Abbreviations, and Symbols

Table 2: **Table of Abbreviations**

Abbreviation	**Definition**
Abbreviation1	Definition1
Abbreviation2	Definition2

Table 3: **Table of Definitions**

Term	**Definition**
Mosaic	McMaster University's online administrative information system.
MacID	A McMaster University student's login account.

1.4 Overview of Document

This application will be a reimplementation of the open source Chrome extension UMD Google Calendar Schedule Importer. The reimplementation will be modified to allow students from McMaster University to export their class schedule from Mosaic and import it into their Google Calendar.

2 Plan

2.1 Software Description

The software will parse Mosaic for a user's schedule and then export it to an array. This array will then be used to import their schedule into their Google calendar. The implementation will be completed in Python 3.6.

2.2 Test Team

The members of Team 0C are Cassandra Nicolak, Michelle Leung and Winnie Liang and are the test team for this project.

2.3 Automated Testing Approach

2.4 Testing Tools

The testing tools that will be used are PyTest and Pycharm. PyTest will be used to automate unit testing and Pycharm will be used for coverage checking and debugging.

2.5 Testing Schedule

See Gantt Chart at the following url: Group12-Gantt03

3 System Test Description

3.1 Tests for Functional Requirements

3.1.1 Testing for User Input

File Explorer

1. Test: FUI-01

   Type: Functional, Dynamic, Manual

   Initial State: File explorer that is used to select the html file of the schedule.

Input: Valid URL or absolute file path.

Output: The specified location of the html file will be saved at the URL required for the parsing component of the software.

How test will be performed: The function that acquires the URL input from the user will check if the XPATHs are present.

2. testid2

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

3.1.2 Testing for Exporting from Mosaic

Parsing

1. testid1

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

2. testid2

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

3.1.3 Testing for Importing to Google Calendar

Input Configuration

1. testid1

   Type: Functional, Dynamic, Manual, Static etc.
   Initial State:
   Input:
   Output:
   How test will be performed:

3.1.4 Testing for Google API Handling

Title of Test

1. testid1

   Type: Functional, Dynamic, Manual, Static etc.
   Initial State:
   Input:
   Output:
   How test will be performed:

3.2 Tests for Nonfunctional Requirements

3.2.1 Area of Testing1

Title for Test

1. testid1

   Type:
   Initial State:
   Input/Condition:
   Output/Result:
   How test will be performed:

2. testid2

   Type: Functional, Dynamic, Manual, Static etc.
   Initial State:
   Input:
   Output:
   How test will be performed:

3.2.2 Area of Testing2

...

3.3 Traceability Between Test Cases and Requirements

4 Tests for Proof of Concept

4.1 Parser

Title for Test

1. testid1

   Type: Functional, Dynamic, Manual, Static etc.
   Initial State:
   Input:
   Output:
   How test will be performed:

2. testid2

   Type: Functional, Dynamic, Manual, Static etc.
   Initial State:
   Input:
   Output:
   How test will be performed:

4.2 Link to Google Calendar API

Authentication and Authorization Test

1. Test: PAAT-01

   Type: Functional, Manual

   Initial State: The client is not connected to their Google account.

   Input: Run function.

   Output: Access to a user's account.

   How test will be performed: Run function for authentication and authorization. User follows instructions. User will receive text confirming authentication and authorization.

Information Retrieval and Upload Test

1. Test: PIRUT-01

   Type: Functional

   Initial State:

   Input: Output:

   How test will be performed:

5 Comparison to Existing Implementation

6 Unit Testing Plan

Test cases will run on different operating systems, including Windows, and Linux.

6.1 Unit testing of internal functions

Every internal function will be tested with the following cases, where applicable:

1. A case where the function takes input it is expected to handle.
2. All edge cases.
3. Cases for all run-time errors and exceptions. (Whitebox)
4. If the function receives input from a user, a case where it does not receive the expected input.

6.2 Unit testing of output files

7 Appendix

This is where you can place additional information.

7.1 Symbolic Parameters

The definition of the test cases will call for SYMBOLIC_CONSTANTS. Their values are defined in this section for easy maintenance.

7.2 Usability Survey Questions?

1. How easy were the instructions to follow?
2. Were you confused at how to use the tool? If so, where and why?
3. Are there any other feature that you would like to see?

This is a section that would be appropriate for some teams.