TEST SPECIFICATIONS REPORT

PPTX TO HTML5 CONTENT CONVERTER

Ömer Baykal

Rustem Hashimov

Shamil Farajullayev

Nahid Hamidli

Table of Contents

1. Introduction

This document is the Test Specifications of the “PPTX to HTML5 Content Converter” project [1], created by the project team Limon.

1.1.Goals and Objectives

We will be testing on our project to find out and debug its errors, make them function more properly and successfully. It will not be 100% error-free, but the tests we are going to do will improve the quality of our final product. After that we handle the errors and make the product operate well, there will be some other concerns. Those concerns are the performance of the software, well designed user interfaces and user experience. Fullfilling those concerns also will increase the software quality.

1.2.Statement of Scope

In general, what we will try to be guarantee to the users will be the fact that, there will be almost no data loss during the conversion of the content and logical correctness through the whole execution. Moreover, we will try to make our users to access to what they created in anywhere they desire. So, platform-independence is our other concern. We also would like to make the product user-friendly by easing it use. The tests we are going to apply will help us to meet these requirements.

1.3.Major Constraints

1.3.1. Time

There is almost two months to finalize the project and besides the project, all group members are going on their education with huge workload. So, the time is the biggest constraint for project development and testing.

1.3.2. Software

One of the components of our project (SCORM component) contains the connection and communication with a special server called LMS server. (More details can be found in section 3). To test these communications a test server will be provided to the group by the sponsor, Enocta. Hence, the testing phase will be kindly dependent on the software that will be given by the sponsor.

1.4.Definitions, Acronyms and Abbreviations

Definitions, acronyms and abbreviations that are used throughout this document are listed in the following table.

CVS	Concurrent Version Systems
The Project	PPTX to HTML5 Content Converter Project
SCORM	Sharable Content Object Reference Module

1.5.References

[1] The Project Website,

http://senior.ceng.metu.edu.tr/2012/limon

[2] SCORM Standard,

http://en.wikipedia.org/wiki/Sharable_Content_Object_Reference_Model

2. Test Plan

Overall testing strategy and methods are described below. In addition the special managements, if necessary, are described in this section in order to clarify any further needs to effectively test the software.

2.1.Software to Be Tested

The software to be tested is a Windows Visual Studio 2011 file, more specifically, a Power Point Add-In. It contains C#, JavaScript and HTML codes; therefore needs different aspects and kinds of tests.

2.2.Testing Strategy

We follow our strict steps of testing on every iteration of the project. The steps are as follows:

- ✓ Every member of the team tests their own units (compiles and runs) and checks the results to be as expected
- ✓ Secondly, each member starts testing their units on other computers or (platforms) to check the unit compatibility
- ✓ Coming to the integration phase, all members get together to integrate the whole project after the iteration, and check the integrated parts all together until the desired results are shown.
- ✓ If the integration phase worked (compiled) fine, we try to check the overall results to the results that are expected from the potential costumers.
- ✓ At the end, as high-order testing, Performance tests and Alpha – Beta tests will be applied.

Performance test is applied to ensure if the software works as fast as desired on any platform or operating system. Alpha test of the project will be similar to the final software, however with possible errors and unexpected outputs. Alpha software is tested by many different developers. Beta product is expected to be more similar to the final product having less and only minor (not fatal) errors. Beta product will be available to public freely to be tested by as many people as possible, and gather the error reports from them.

2.2.1. Unit Testing

As stated at section 2.2, every member of the team is expected to test their own parts of the software which they are responsible of. First of all, since all members have different jobs of different sides of the project, it is possible that only that member of the team has the ability, knowledge and experience to successfully test the unit. Therefore, unit testing is among the most crucial parts of the testing process. Once the fish stinks from the head, it will stink all down to the tail. Consequently, unit testing is ought to be done by each member of the group, and continued until the expected result of the specific module is achieved. Changing from unit to unit, the tests may include testing on different browsers, different windows machines with same (or different versions of) operating systems, one to one check to expected outputs and etc. Since our project aims to convert PowerPoint slides to HTML5 files, the desired output is very certain and predetermined. The strategy of unit testing is simple: Do test, edit and test loop until the exact content of the slides is shown on the html pages. Moreover, as our product will be web based, sometimes it may require checking the results of the units to be same as the one company asked the project group to make.

2.2.2. Integration Testing

Without integration, our project does not have anything to demonstrate. Thus, integration should be done carefully to continue the development progress and to be able to demonstrate the state of the project at the end of each iteration.
The steps of the integration are as followed:

- ✓ 2 members of the project who have the related parts of the project get together to form 2 groups of 2 people, and integrate their units to each other's product.
- ✓ Now that there are two separate parts of the project, which are slightly unrelated to one another by means of developing methods, are combined together more easily to check if they work together as a whole, and output correctly.
- ✓ After all the integration is done, all members altogether do the testing process once again inspecting if everything works fine.

2.2.3. Validation Testing

Every software product needs to be fully tested before being demonstrated to the costumers. Since we do not show the product of each iteration to the costumer or company, but to the teaching assistants, we still need to make sure all the progress is going well and nothing wrong is missed by us. Therefore, one more general test with more complicated testing inputs is done, if there is no problem, it is demonstrated to the assistants.

2.2.4. High-Order Testing

High order testing includes two main tests:

- ✓ Performance testing
- ✓ Alpha – Beta testing, which is also consisted of two parts.

Those methods will be discussed with more details at the section 3.4. More generally, those tests are done to check if the product works as fast as desired, without causing any discomforts due to slow speed. Secondly, Alpha and Beta phases of the product are generated to be able to gather information about the errors from the tests of different people. Alpha product is tested by developers or the people who understand enough about the implementation process of the product, so that he/she can give us crucial feedback once a fatal error is occurred. Beta product is tested by anyone for free, to gather information about less fatal or less important errors to be able to release a perfect final product.
The strategy of performance testing is to test the product on several computers from fastest to slowest and keep the detailed data of the conversion process.
The strategy of the alpha-beta testing phase is to generate a test product as similar as possible to the final expected product, with the ability to gather and send feedbacks back to developers.

3. Test Procedure

In this section, test procedure including test tactics and test cases for the software is described in detail.

3.1.Unit Testing

3.1.1. Hierarchy Editor Component Testing Procedure

Hierarchy Editor Component has properties that are listed in following subsections, to be tested.

3.1.1.1. Hierarchy Tree Edit Functionality

Check whether the Hierarchy Tree Edit functionality works properly or not. There are two cases:

1. Drag-Drop Functionality: makes one of the nodes children of another.
2. Move Up-Down Functionality: changes the order of sibling children.

These two cases are tested separately.

3.1.1.2. Save Button Functionality

Check whether the Save Button saves Hierarchy Tree in XML format as a file named “PresentationName.xml”. Besides, check for the correctness of XML attribute names and whether all the Hierarchy Tree content is saved correctly. There are two cases for testing:

1. If there is already a saved XML file named as “PresentationName.xml", the file must be overwritten.
2. If there is no such XML file, new file is created and saved.

These two cases are tested separately.

3.1.1.3. Load Button Functionality

Check whether the Load Button loads the Hierarchy Tree that is saved as a file named “PresentationName.xml" successfully. There are two cases for testing:

1. If there is such an XML file, load function must fetch that tree.
2. Otherwise, it loads tree in initial format that does not contain any hierarchy, all are in sequence.

These two cases are tested separately.

3.1.1.4. Reset Button Functionality

Check whether the Reset Button resets tree to its initial format that is with no hierarchy, when it is clicked. Reset Button has different functionality than Load Button. Load Button initially checks memory to see if there is already an existing hierarchy tree and if there is not such a tree, it loads initial form. However, Reset Button directly initiates the initial form.

3.1.2. Publish Component Test Procedure

Publish Component has two functionality to be tested. Firstly, it publishes IMS Manifesto, Bookmark Panel and Hierarchy Tree. However, because this functionality is related with other components, it will be evaluated in Integration Test. Secondly, Publish Component publishes presentation content in HTML5 format and this property is evaluated in this part.

3.1.2.1. HTML Content

- ✓ **Unzipping presentation file (.pptx file):**
  This process is done for accessing presentation objects such as images and themes. If this function executes successfully, there must be a folder with the name "unzipped" which contains extraction of the pptx file. Since this action affects image and theme conversion directly, it must be tested before those conversions.
- ✓ **Placing image content within <canvas>:**
  The test that is applied checks if the image from presentation document is placed in correct coordinates with correct dimensions. Besides, all of the image formats (like .jpg, .png, etc.) must be tested.
- ✓ **Placing text content within <canvas>:**
  The test that is applied checks if the text from presentation document is placed in correct coordinate with correct dimensions. Besides, properties of the texts like underlined, bold, font size, font color, italic must be represented.

3.1.3. SCORM Initialization and Run-time Process Test Procedure

SCORM [2] Process has two steps: Initialization and Run-time. Firstly, Initialization step creates connection to LMS and this is tested by creation of a sample connection. The important file for connection setup is IMS Manifesto, which is created by Publish component. Secondly, Run-time which is compound of get and set functions used to communicate with LMS is handled. Run-time process is tested by running sample education html file and checking the result that is reported to LMS.

3.2.Integration Testing

3.2.1. Hierarchy Editor - Publish Component

Hierarchy Editor serves hierarchy tree for Publish Component, which is used in IMS Manifesto and Bookmark Panel. The testing cases are:

1. Check if Publish Component gets hierarchy tree successfully. This is tested by the hierarchy tree xml file that is located in output folder.
2. Check if Publish Component handles null tree situation, which occurs when the Publish button is pressed in an empty presentation file.

3.2.2. Publish Component - SCORM

Publish Component creates IMS Manifesto with the help of hierarchy tree. Since IMS Manifesto is used in SCORM Connection and Communication, the test is applied in connection by checking if all the slides are included in SCORM.

3.3.Validating Testing

3.3.1. Testing Procedure

The requirements specified in SRS will be examined to see whether software satisfies all of the requirements.

3.3.2. Test Cases

In Microsoft Office PowerPoint side, the test cases are:

- ✓ Is Hierarchy Editor Panel accessible from Add-In Tab?
- ✓ Does Hierarchy Editor edit function works properly?
- ✓ Does Hierarchy Editor save & load functions work properly?
- ✓ Is Publish Component accessible from Add-In Tab?
- ✓ Is "Select Directory" window opened?

In Output side, the cases:

- ✓ Does the output folder contain IMSManifesto.xml, bookmark panel html, hierarchy tree xml and unzipped folder?
- ✓ Does the html file contain bookmark panel and all the slides in panel?
- ✓ Does the touch functionality work in mobile devices with touch capability?
- ✓ Does SCORM functionality works?

3.4.High-Order Testing

3.4.1. Performance Testing

Firstly, system will be tested for conversion of presentations with no data loss. That is main performance measurement. Secondly, conversion of presentations to html must be finish in an adequate time. So, for time testing, presentations that contains large amount of slides will be used and finish time will be evaluated for speed performance.

3.4.2. Alpha Testing and Beta Testing

System will be tested by Limon team and our sponsor company Enocta as alpha testing. Company will provide education presentations that are used by customers to the team. For beta testing, we will release the whole system and users will be able to express both defects of the system and their opinion about the software using a tracking system.

4. Testing Resources and Staffing

- ✓ Each module will be tested firstly by its developer(s). So, unit tests will be conducted by the developers of each module.
- ✓ Integration testing will be conducted by the developers of the modules that are going to be integrated.
- ✓ Validation testing will be conducted by the developer(s) of the related module(s) with a member who is not a developer of that module for getting the benefit of outside perspectives.
- ✓ System testing will be conducted by the whole team.

5. Test Work Products

Test work product which is going to be used to test our system is mainly the TRAC system. Whenever one of the group members finds a bug on the system, he/she will immediately inform the other users using the TRAC system. Since all group members checks the TRAC regularly, they will be informed automatically.

6. Test Record Keeping and Test Log

Any kind of tests being implemented will be reported in weekly progress reports. To keep logs of the tests implemented, SVN and TRAC are used. SVN provides all versions of the

project codes, which are updated according to test results, and TRAC helps to identify the bugs in the project.

7. Organization and Responsibilities

Components of the project mentioned above are divided into the group members. Each group member will be responsible for reporting test results for his own components. After whole group is informed about the unit testing results, integration testing will be done by group members. As the project team was structured with no team leader, all peers will be monitoring the others' works. Validation and high-order testing are done again as group. Additional responsibility will be added to each member in beta testing phase, which is finding adequate number of volunteer beta testers.
Each group member will be responsible for validating testing phases with respect to this document. Rustem is going to be responsible for whether all unit testing steps mentioned in this document are implemented. Ömer will check if all integration testing cases are tested, and Nahid and Shamil are going to validate that the project passes all high-order testing phases.

8. Test Schedule

Schedule for testing is determined as in Table 8.1.

Task Name	Time Interval
Unit Testing	April 30th – May 4th
Integration Testing	May 4th – May 9th
Validation Testing	May 9th – May 10th
Performance Testing	May 10th – May 16th
Alpha Testing	May 16th – May 18th
Beta Testing	May 18th – May 25th
Finalization	May 25th – June 2nd

Table 8.1 – Test Schedule