Lab III - Blackboard Archive Extractor Prototype Test Plan/Procedure Outline

1 Objectives

The purpose of this Test Plan and Procedure is to establish the overall approach, testing sequence, and specific tests to be conducted to demonstrate the successful operation of the Blackboard Archive Extractor (BAE) prototype. The Blackboard Archive Extractor provides a user with simple and easily accessible interfaces to process a Blackboard Archive quickly. The plan and procedures described in this document are designed to verify stable and operational performance of the Blackboard Archive Extractor (BAE) prototype.

2 References

- Blackboard Archive Extractor Lab 1 - Prototype Description
- Blackboard Archive Extractor Lab 2 - Prototype Specification

3 Test Plan

3.1 Testing Approach

The functionality of the Blackboard Archive Extractor prototype will be validated by major functional component. System and Integration testing will validate performance of the entire application.

Functionality of the Blackboard Archive Extractor prototype will be verified by the following types of tests:

1. User Interface tests to verify that the controls for the GUI and commands for the CLI function properly.

2. Systems tests to verify that the application successfully extracts content from a Blackboard Course Archive and produces a webpage with the correct formatting and correct content.
3. Core Functionality tests to verify that the major components of the application function correctly.
    a. Internal interface tests to verify proper communication between projects within the solution.
    b. Integration tests to verify proper functionality between interacting components.
4. Extraneous Functionality to verify optional functionality for the application is working correctly.

Tests will be performed to verify functionality of the above areas in a an environment with controlled parameters. After completion of controlled tests, non-controlled tests will be conducted to discover potentially undiscovered edge cases.

All tests will be conducted in Windows 7. User Interface and System tests will be completed through observation. Internal interface and integration tests will be completed through the use of the application's test harness.

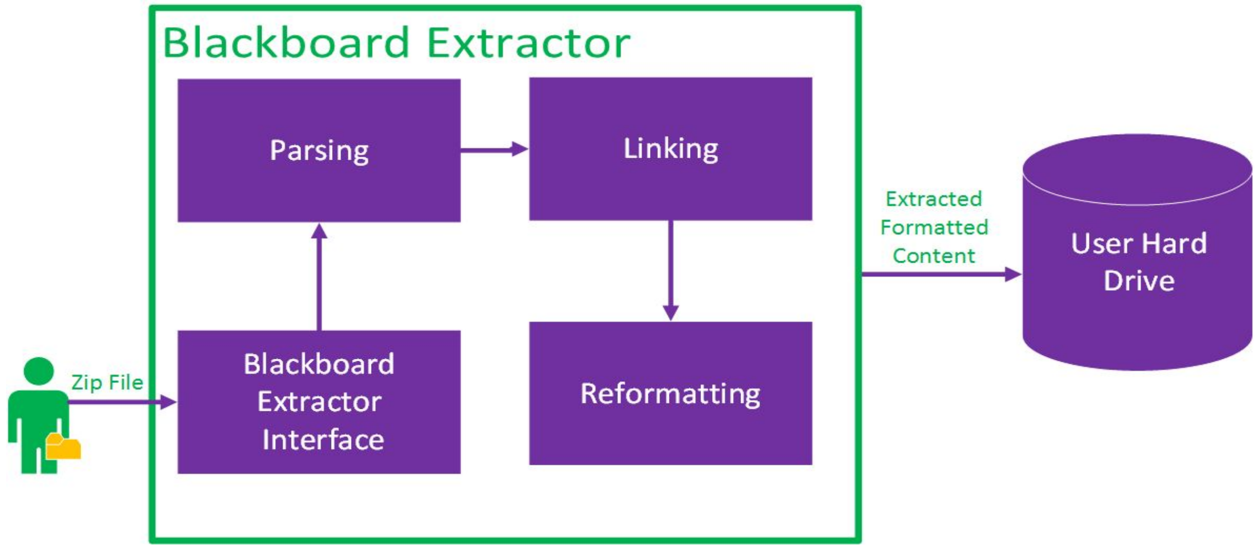

Figure 1 – BAE Prototype Functional Diagram

3.2 **Identification of Tests**

The following tests will be performed:

Category ID	Description	Test Case	Description	Objective
1	User Interface	1.1	GUI Functionality	Verify that all functions of BAE can be accessed via the Graphical User Interface
		1.2	CLI Functionality	Verify that all functions of BAE can be accessed via the command line interface
2	Core Functionality	2.1	Extraction of Blackboard archive	Verify an input archive can be extracted for processing
		2.2	Conversion of resource files to HTML elements	Verify the ability of BAE to convert resource files to HTML elements
		2.3	Generation and population of output directory	Confirm BAE's ability to create an output directory and populate with resources found during parsing
		2.4	Generation of index file	Verify BAE's ability to

				create a structured index file linking to various resources
3	Extraneous functionality	3.1	Verification of expired links	Verify BAE’s ability to detect hyperlinks to expired/dead content

Table 1 – Test identification table

3.3 **Test Schedule**

Tests will be completed within a 35 minute period in accordance with the following schedule:

Start Time (hours:min)	Duration (Minutes)	Test Objective	Test Event	Dependencies	Comments
0	3	User Interfaces	1.1	Operating System Compatible	Download software and open GUI
0:05	2	No options - GUI Run	1.1	1.1	Execute program from GUI without options
0:07	5	Check program Output and Directory output	2.1, 2.2, 2.3, 2.4	1.1	Visually look at output content and directory location

		location			
0:13	2	No options - CLI Run	1.2	1.1	Execute program from CLI without options
0:15	5	Check program Output and Directory output location	2.1, 2.2, 2.3, 2.4	1.2	Visually look at output content and directory location
0:20	~10	Check dead hyperlinks option - Both Interfaces	3.1	1.1, 1.2	Check both CLI and GUI if they can detect dead links
0:30	5	Check program Output and Directory output location with bad hyperlinks removed	2.1, 2.2, 2.3, 2.4	3.1	Check content has removed hyperlinks that are dead
Total Time = ~35 minutes					

Table 2 – Test schedule table

3.4 **Fault Reporting and Data Recording**

Test failures will have individual error results depending on the failure. All of the failures and that occur when the program is executed will be viewable from the interface being used. For example, if a user specifies to use the check hyperlinks option from the CLI or GUI interface an error message should be shown if the user is not connected to the

Internet. Data will not be recorded if a failure does occur, and the program will have to be executed again to restart the processing of a Blackboard Archive.

3.5 **Resource Requirements**

The resource requirements required to setup and execute the test may include:

- Hardware Requirements: Standard desktop or Laptop with monitor, keyboard and mouse, Internet Access
- Software Requirements: Windows 7, Visual Studio 2015, Microsoft .Net Framework 4.6.1
- Test Data: Zipped Blackboard course archive

3.6 **Test Environment**

The application will be tested in a Windows 7 environment with the Microsoft .NET 4.5 framework installed.

3.7 **Test Responsibilities**

Team Member	**Role**	**Responsibilities**
Merly Mathis	Test Manager	Main presenter, answers questions
Austin Tillery	CLI User	Subject matter expert on CLI functionality
Dominic Nguyen	GUI User	Subject matter expert on GUI functionality
Chris Soffos	Test Environment Configurator	Configures test environment
Devin Haslam	Test Recorder	Records test results
Tristan Pressley	Archive Manager	Produces Test Archives
Grant Atkins	Guest Website User	Demonstrates guest navigation through website content.

Table 3 – Team responsibilities table

4 **Test Procedures**

4.1 **User Interface Tests**

Test Case 1.1 - GUI Functionality

Specification Reference: 3.1.1.1, 3.1.1.2, 3.1.1.3

Test Description: This test will verify that all functions of BAE can be accessed via the Graphical User Interface

Initialization: BAE software is not installed

Test Inputs: None

Test Procedure:

1. Download software
2. Open the Graphical User Interface
3. Execute program from GUI without options
4. Verify functions of BAE can be accessed via GUI

Expected Test Results:

1. Software installs successfully (*Pass/Fail*)
2. GUI opens successfully (*Pass/Fail*)
3. Program is able to execute via GUI (*Pass/Fail*)
4. All functions accessible from GUI (*list - Pass/Fail each*)

Test Case 1.2 - CLI Functionality

Specification Reference: 3.1.2.1

Test Description: This test will verify that all functions of BAE can be accessed via the command line interface

Initialization: BAE software is installed

Test Inputs: None

Test Procedure:

1. Execute program from command line interface without options
2. Verify functions of BAE can be accessed via CLI

Expected Test Results:

1. Program is able to execute via CLI (*Pass/Fail*)
2. All functions accessible from CLI (*list - Pass/Fail each*)

4.2 **Core Functionality Tests**

Test Case 2.1 - Extraction of Blackboard Archive Functionality

Specification Reference: 3.1.1.1, 3.1.1.2, 3.1.1.3

Test Description: This test will verify that an inputted archive can be extracted for processing

Initialization: BAE Software installed

Test Inputs: Blackboard archive file

Test Procedure:

1. Open the Graphical User Interface
2. Select archive to extract
3. Select location for output
4. Click the "Extract" button

Expected Test Results:

1. GUI opens successfully (*Pass/Fail*)
2. Archive selected (*Pass/Fail*)
3. Location selected (*Pass/Fail)*
4. Extraction completes (*Pass/Fail*)

Test Case 2.2 - Conversion of Resource Files to HTML Functionality

Specification Reference: 3.1.3.1, 3.1.3.2, 3.1.3.3, 3.1.6.1, 3.1.6.2

Test Description: This test will verify that the application correctly produces HTML files from the content in the input archive.

Initialization: BAE Software is installed

Test Inputs: Blackboard archive file

Test Procedure:

1. Open the Graphical User Interface
2. Execute program from GUI without options
3. Select archive to extract
4. Select location to output
5. Click “Extract” button
6. Verify that the application produces the correct html files in the output directory

Expected Test Results:

1. GUI opens successfully (*pass/fail*)
2. Application executes successfully (pass/fail)
3. Archive selected (*pass/fail*)
4. Location selected (*pass/fail*)
5. Extraction completes (*pass/fail*)
6. HTML content matches original archive content (pass/fail)

Test Case 2.3 - Generation and Population of Output Directory

Specification Reference: 3.1.3.1, 3.1.7.1

Test Description: This test will confirm the ability of BAE to populate a specified output directory with resources found during parsing of the Blackboard archive’s manifest file.

Initialization:

- BAE software is installed
- A Blackboard course archive has been provided

Test inputs: Blackboard archive file

Test Procedure:

1. Open a command prompt with the ability to locate the BAE executable
2. Execute BAE, using provided archive and specified output directory as parameters
3. Verify the application has created and populated the location with expected contents

Expected Test Results:

1. Application executes successfully (*Pass/Fail*)
2. Archive selected (*Pass/Fail*)
3. Location selected (*Pass/Fail*)
4. Extraction completes (*Pass/Fail*)
5. Output directory contains all relevant resources found in course archive (*Pass/Fail*)

Test Case 2.4 - Generation and Index File

Specification Reference: 3.1.3.1, 3.1.4.1, 3.1.4.2, 3.1.5.1, 3.1.5.2, 3.1.5.3

Test Description: This test will confirm the ability of BAE to generate an HTML index file based on the XML elements found in the Blackboard manifest file.

Initialization:

- BAE software is installed
- A Blackboard course archive has been provided

Test inputs: Blackboard archive file

Test Procedure:

1. Open a command prompt with the ability to locate the BAE executable
2. Execute BAE, using provided archive and specified output directory as parameters
3. Verify the application has created an HTML index file
4. Verify the HTML index file contains links to all relevant resources in output directory

Expected Test Results:

1. Application executes successfully (*Pass/Fail*)
2. Archive selected (*Pass/Fail*)
3. Location selected (*Pass/Fail*)
4. Extraction completes (*Pass/Fail*)
5. Output directory contains all relevant resources found in course archive (*Pass/Fail*)
6. Index file contains links to all relevant resources found in course archive (*Pass/Fail*)

Test Case 3.1 - Removal of Dead Hyperlinks from HTML Content

Specification Reference: 3.1.8.1, 3.1.8.2, 3.1.8.3

Test Description: This test will verify that the application correctly removes dead hyperlinks from HTML files found in the Blackboard Archive.

Initialization:

- BAE software is installed
- A Blackboard course archive has been provided

Test Inputs: Blackboard archive file

Test Procedure:

1. Open the Graphical User Interface
2. Execute program from GUI with the “remove dead hyperlinks” option
3. Select archive to extract
4. Select location to output
5. Click “Extract” button
6. Verify that the application produces the correct HTML files with no dead links in the output directory

Expected Test Results:

1. Application executes successfully (*Pass/Fail*)
2. Archive selected (*Pass/Fail*)
3. Location selected (*Pass/Fail*)
4. Extraction completes (*Pass/Fail*)
6. Output directory contains all relevant resources found in course archive (*Pass/Fail*)
7. None of the hyperlinks inside the HTML content redirect to dead web pages. (*Pass/Fail*)

[This space intentionally left blank.]

5 Traceability to Requirements

		Test Cases						
		1.1	1.2	2.1	2.2	2.3	2.4	3.1
Functional Requirements	3.1.1.1	x						
	3.1.1.2	x						
	3.1.1.3	x						
	3.1.2.1		x					
	3.1.3.1			x	x	x	x	
	3.1.3.2				x			
	3.1.3.3			x	x			
	3.1.4.1						x	
	3.1.4.2						x	
	3.1.5.1						x	
	3.1.5.2						x	
	3.1.5.3						x	
	3.1.6.1				x			
	3.1.6.2				x			
	3.1.7.1			x		x	x	
	3.1.8.1							x
	3.1.8.2							x
	3.1.8.3							x

Table 4 – Traceability Requirements table