Software Requirements Specification

For

Digital Archival Tracker

Prepared by Digital Archival Team

School of Computer Science, Northwest Missouri State University

October 15, 2016

Table of Contents

Table of Contents ii

1. Introduction 1

1.1 Purpose 1

1.2 Definitions of AV Preserve Tools 1

1.3 Overview of the System 1

1.4 References 2

2. Overall Description 2

2.1 Product Description 2

2.2 Design Constraints 2

2.3 Product Functionalities 3

2.4 Target User 3

3. Specific Requirements 3

3.1 External Interface Requirements 3

3.2 Functional Requirements 3

3.3 Performance Requirements 3

3.4 Database Requirements 4

4. System Attributes 4

4.1 Reliability 4

4.2 Availability 4

4.3 Security 4

4.4 Maintainability 4

4.5 Portability 4

5. Use Cases 5

5.1 User Interface Use Case 5

5.2 Database Use Case 6

5.3 Service Integration Use Case 9

# Introduction

## Purpose

This software requirement specification document covers the primary goal of the project Digital Archival Tracker and sets clear objective as to what the system does. Digital Archival Tracker is a desktop application that streamlines the process for processing digital files in a historical archive. The system presents a user interface that tracks the progress of data files for the user to view hence replacing the current use of spreadsheet by the client.

## Definitions of AV Preserve Tools

AVPreserve is a data management consulting and software development firm focused on leveraging a deep understanding of technology, information, business, and people to advance the ways in which data is used for the benefit of individuals, organizations, and causes. The various Av Preserve Tools used are:

**Bagger**: BagIt File Packaging Format (an Internet Engineering Task-Force standard, developed by the Library of Congress and the California Digital Library, with current support from George Washington University and the University of Maryland), supports FTP transfer, as well as standard network transfers, and integrates into desktop-based file sharing workflows such as Dropbox or Google Drive.

**Fixity**: Fixity is a utility for the documentation and regular review of stored files. Fixity scans a folder or directory, creating a manifest of the files including their file paths and their checksums, against which a regular comparative analysis can be run. Fixity monitors file integrity through generation and validation of checksums, and file attendance through monitoring and reporting on new, missing, moved and renamed files. Fixity emails a report to the user documenting flagged items along with the reason for a flag, such as that a file has been moved to a new location in the directory, has been edited, or has failed a checksum comparison for other reasons. Supplementing tools like BagIt that review files at points of exchange, when run regularly Fixity becomes a powerful tool for monitoring digital files in repositories, servers, and other long-term storage locations.

## Overview of the System

A standalone windows desktop application that tracks various files on the system that are meant to be archived. The files would be grouped into folders using BagIt. The folders are put through a Fixity check, to examine if there is any change to the files. Fixity would use CDC techniques with checksum or MD5 to be able to discern changes to files in the folder. With so many folders to be processed the user had to manually track the various processes carried out on the folder in a spreadsheet. Our system would simplify this task by reading in reports generated by Fixity and populating the report data in a table like format in the UI. The user also has the ability to add, update, edit and delete information pertaining to the reports visible in the UI.

## References

In this subsection:

* **Accession Guideline Procedures** – Digital – details the procedures involved in transferring files from their original sources to the archives for long-term preservation.
* **Digital Preservation in Action –** details the procedures involved in collecting the files from various systems to be processed at a central system in the future.
* **AIPRecordsMgmt.xlsx** – excel sheet with sample file tracking data filled in by client.
* **AVPreserve** – website of data management consulting and software development firm focused on understanding technology and information.

# Overall Description

## Product Description

The user runs BagIt on archived files, these packaged files are run through Fixity to perform Fixity check so that it generates a report for the user to view. This report contains information about the files (whether the file is moved, renamed, changed or deleted). With this report, the user can easily know about the changes that occurred to the existing folder. The generated report is stored in database and can be displayed on the User Interface upon the user’s request.

### User Interfaces

The user interface follows the basic windows style and functionality conventions. The interface has an option to upload the user’s desired files to Fixity and the generated report from the fixity will be displayed to the user. Functionality to add comments and annotations is provided within the report.

### System Interface

This application is interlinked with Fixity to run Fixity check and produce the report. The system is configured to a database to store, retrieve, edit and view the report for the user.

### Hardware Interface

The application can run on any machine which supports:

* Windows 7/8x plus OS
* Minimum 1 GB ram
* Minimum 10 GB Hard disk
* Hardware: Keyboard, mouse/trackpad and monitor
* Intel core i3 or equivalent

## Design Constraints

* Windows System – The user must have a Windows 7/8.x plus OS installed on their machine.
* Desktop application – The system is specific to a machine and does not run in a browser (not a web application), instead can be run on Windows as a desktop application.
* Specific Database – When the system is running on a machine, a database must be pre-installed.

## Product Functionalities

* **Automating Fixity upload and run**

The system automates the process of uploading the folders from BagIt to Fixity by providing the user with the option of entering the source folder. The files from this source folder are uploaded and Fixity check is performed and the report generated is obtained and saved in the database.

* **Clear presentation of Fixity Reports**

The reports generated by Fixity, which are saved in the database can be viewed in the user interface in a very clear fashion.

* **Commenting and Annotating Fixity Report**

The user interface displaying the reports provides the functionality of adding, editing and deleting comments and annotations.

## Target User

Any user that has access to files uploaded to DuraCloud, backup hard drives, Fixity desktop application and BagIt will be our target user.

# Specific Requirements

## External Interface Requirements

* This application interfaces with Fixity in order to generate reports.
* The system is configured to a database to store, retrieve, edit and view the report for the user.
* Windows File Explorer to navigate the machine for retrieving the reports generated by Fixity.

## Functional Requirements

* The user will have an option of selecting when to run Fixity by the means of a simple “Run Fixity” button.
* The user will have access to the existing reports in the “View Reports” list.
* The user will have the ability to add comments and annotations by “Add Annotation” button.

## Performance Requirements

* The data must be retrievable from the database when the user requests.
* Fixity must generate reports when user runs the fixity check.

## Database Requirements

The system uses a relational database to store the contents of the report generated by Fixity and the comments and annotations generated by the user.

# System Attributes

After receiving the report from Fixity, the system reads the report, saves it to the database and makes the report available in the User Interface for the User to view and edit the data.

## Reliability

The system is able to read the report correctly and display the appropriate data in the User Interface.

## Availability

As this is not a web application, the system is always available to the user as a desktop and it makes the data available upon user request.

## Security

When a user requests for viewing/saving the data, the process of communication with the database is hidden. Input validation is also performed, hence, securing the data from unauthorized tampering.

## Maintainability

Once the system is running on the client machine it requires no essential updates except for changes in Fixity’s report generation process and report template.

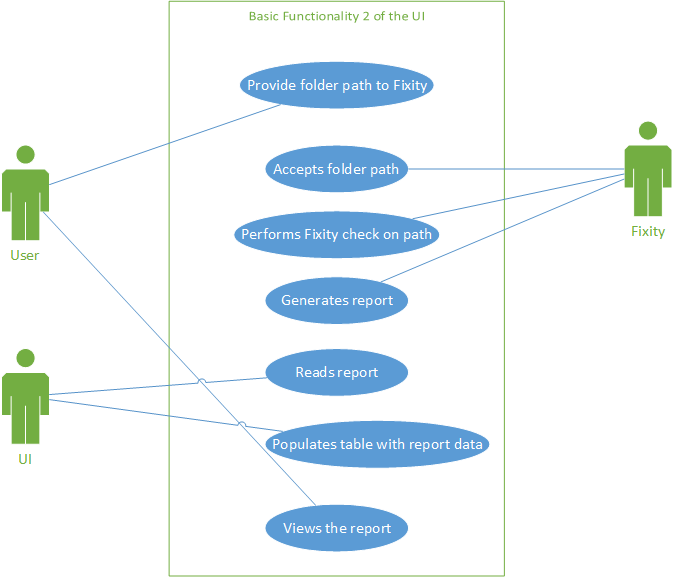
## Portability

The system can be installed on any system that has the specified database preconfigured and the machine must be running Fixity.

# Use Cases

## User Interface Use Case

**Scenario 1:** In the following use case, the user provides the path of the folders for Fixity to perform the Fixity check. As the path is uploaded to Fixity and Fixity runs the check, a report is generated and is automatically read and important information from the report is populated inside the system UI. The user can now view the report.



**Fig 1: Scenario 1 User Interface Use Case.**

**Scenario 2:** This use case diagram depicts basic functionality of the UI for our software. Fixity performs checksum on files and generates a report. The software’s UI presents the fixity report to the user. The report contains file/folder status like “changed”, “moved” or “deleted” for each file mentioned in the report. After the report is available, the user acknowledges the generated report. Based on the file status for the mentioned files, the user inserts annotations/comments in the report which is updated in the UI. In the end, the report is then saved to the database.

# 

**Fig 2: Scenario 2 User Interface Use Case.**

## Database Use Case

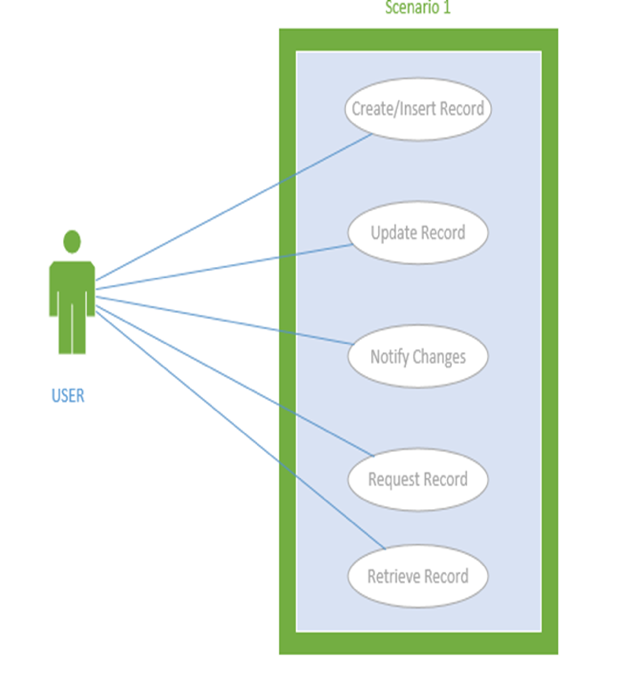
This use case diagram(s) depicts the various interactions of the user with the database. The User is the main actor in all the scenarios since the user will be directly interacting with the UI which is directly interacting with the database. The scenarios are defined based on the exchange of data with the database and the user.

|  |  |  |
| --- | --- | --- |
| **Symbol** | **Goal-Level** | **Description** |
| ++ | Very High Summary | Rachel doing checklist on interface |
| + | Summary | Displaying reports on what actions are performed on files and what are the action to be performed |
| ! | User Goal | Reports generated by fixity |
| - | Subfunction | The interface talks to bagger and fixity |
| -- | Too Low | Adding annotations and comments along with reports to database |

Table 1: Hierarchy of Goal Levels

**Scenario 1:**

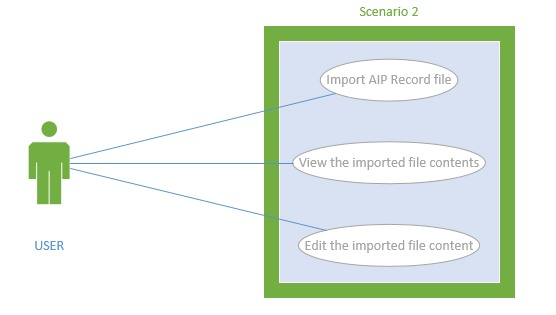
The user enters the AIP record into the system which contains information like whether the files are bagged, uploaded on Duracloud, AIPed, Fixity checks performed, etc. This information is saved in the database as tuples in single or multiple tables. User can request existing record from the database, the system retrieves the records and displays it in the UI. The user can update records, if there are any changes to be made and these changes are saved in database. The user is notified about the record being updated/created in the database.



**Fig 3: Scenario 1 Database Use Case.**

**Scenario 2:**

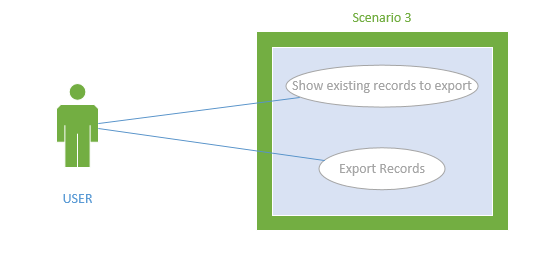
The user has AIP records which is supposed to be inserted into the database. The system reads the file and adds it to the database. After the user has imported the record the user can view/edit the imported file contents in the UI which are populated from the database.



**Fig 4: Scenario 2 Database Use Case.**

**Scenario 3:**

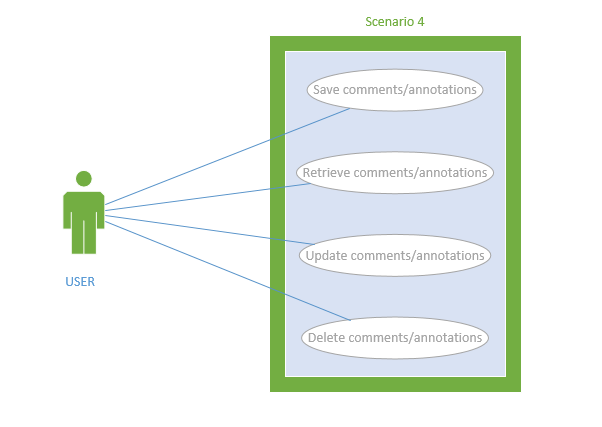
The user can browse existing records and export an AIP record file from the database.



**Fig 5: Scenario 3 Database Use Case.**

**Scenario 4:**

The user adds comments/annotations to the UI which is in turn saved in database, this can be retrieved, updated and deleted based on requirement.

****

**Fig 6: Scenario 4 Database Use Case.**

## Service Integration Use Case

The list of actions which define typical interactions between actors and system are:

* User creates/access a checklist of AIP records
* User performs fixity check which is done using fixity tool
* Reports from fixity are requested by user
* Fixity generates the reports by performing checksums on the files

**Scenario 1: User initiates the fixity for the first time for the file**

The user performs the checklist, wherein he initiates the fixity program for the particular folder. The fixity program would then generate a report based on a checksum or the MD5 function. This report is placed at particular folder on the desktop.



**Fig 7: Scenario 1 Service Integration Use Case.**

**Scenario 2: The CSV report generated from fixity is displayed to the user in appropriate format**

The report is retrieved from the desktop and provided to the user, the user perceives if it is a good report or a bad report



**Fig 8: Scenario 2 Service Integration Use Case.**

**Scenario 3: If the report generated is bad**

If the fixity is provided to the user is bad, then user would add appropriate comments to the report and store it further in the database and would again run the report to find if the aberration would persist.



**Fig 9: Scenario 3 Service Integration Use Case.**

**Scenario 4: If the report generated is good**

If the fixity is provided to the user is good, then user would add appropriate comments to the report and store it further in the database.



**Fig 10: Scenario 4 Service Integration Use Case.**