**Software Requirements Specification**

**for**

Document Tag Parser

**Version 2.4 approved**

**Prepared by** **Joshua Cabrera, Marco De La Torre, Raul Gallegos, Daniel Guevara-Dominguez, Chuang Huang, Dang Le, Jesica Lopez De Leon, Shaocheng Shi, Sergio Tapia, Luke Williams**.

Advisor: Jungsoo (Soo) Lim

Santa Barbara Public Defender Office

**May 8, 2022**

**Table of Contents**

[**Revision History**](#_gjdgxs) **3**

[**1. Introduction**](#_30j0zll) **4**

[1.1 Purpose](#_1fob9te) 4

[1.2 Intended Audience and Reading Suggestions](#_3znysh7) 4

[1.3 Product Scope](#_2et92p0) 5

[1.4 Definitions, Acronyms, and Abbreviations](#_tyjcwt) 5

[1.5 References](#_3dy6vkm) 5

[**2. Overall Description**](#_1t3h5sf) **5**

[2.1 System Analysis](#_4d34og8) 5

[2.2 Product Perspective](#_2s8eyo1) 5

[2.3 Product Functions](#_17dp8vu) 6

[2.4 User Classes and Characteristics](#_3rdcrjn) 6

[2.2 Operating Environment](#_26in1rg) 6

[2.6 Design and Implementation Constraints](#_lnxbz9) 6

[2.7 User Documentation](#_35nkun2) 6

[2.8 Assumptions and Dependencies](#_1ksv4uv) 6

[2.9 Apportioning of Requirements](#_44sinio) 6

[**3. External Interface Requirements**](#_2jxsxqh) **7**

[3.1 User Interfaces](#_z337ya) 7

[3.2 Hardware Interfaces](#_3j2qqm3) 8

[3.3 Software Interfaces](#_1y810tw) 8

[3.4 Communications Interfaces](#_4i7ojhp) 8

[**4. Requirements Specification**](#_2xcytpi) **9**

[**4.1 Functional Requirements**](#_1ci93xb) **10**

[4.2 External Interface Requirements](#_3whwml4) 11

[4.3 Logical Database Requirements](#_2bn6wsx) 11

[4.4 Design Constraints](#_qsh70q) 11

[**5. Other Nonfunctional Requirements**](#_3as4poj) **11**

[5.1 Performance Requirements](#_1pxezwc) 11

[5.2 Safety Requirements](#_49x2ik5) 11

[5.3 Security Requirements](#_2p2csry) 11

[5.4 Software Quality Attributes](#_147n2zr) 11

[5.5 Business Rules](#_3o7alnk) 12

[**6. Legal and Ethical Considerations**](#_23ckvvd) **13**

[**Appendix A: Glossary**](#_ihv636) **14**

[**Appendix B: Analysis Models**](#_32hioqz) **14**

[**Appendix C: To Be Determined List**](#_1hmsyys) **14**

**Revision History**

| **Name** | **Date** | **Reason For Changes** | **Version** |
| --- | --- | --- | --- |
| Team | 3/8/22 | Update requirements, DTP | 2.1 |
| Team | 5/7/22 | Updated DTP for sections 1,2,3 | 2.2 |
| Team | 5/8/22 | Updated DTP sections 4,5,6 & Appendix A-C | 2.3 |
| Team | 5/9/22 | Revised and Final Draft | 2.4 |

# 

# 1. Introduction

**Document Tag Parser**

The document tag parser app purpose is to extract pdf files tag number and then rename the file with the tag number included. The document tag parser is also able to transcribe text from image files. This application can handle multiple files and will greatly speed up the process compared to doing it manually one by one.

### 1.1 Purpose

The purpose of this document is the following:

1. Identify the requirements for the Document Tag Parser application.
2. Identify the needed software and how it is being used.
3. State how the user is meant to interact and use the product.
4. How the product will interact with any external service.
5. Define and describe concepts that will clarify the functions of the product.

**Document Tag Parser**

The Document Tag Parser Module is coded in Python and implements Pytesseract and a Text Parsing Module.

### 1.2 Intended Audience and Reading Suggestions

This document is intended for developers, testers, project managers, and users to understand the purpose, function, and requirements of the product. Suggested reading sequence for each is the following:

**Document Tag Parser**

* Developers - Understand the purpose and functions of the product to help during development; Recommended Reading: Recommended Reading: pytesseract documentation, pdf2image documentation, python documentation, tkinter documentation.
* Testers - Understand the purpose of the document tag parser and what it does during testing to guarantee intended function; Recommended Reading: pytesseract documentation, pdf2image documentation, python documentation
* Users - Understand the purpose of the product and its functions; Recommended reading: Sections 1-5

### 1.3 Product Scope

**Document Tag Parser**

The document tag parser app will help speed up the process for public defenders by eliminating the process of manually renaming each file one by one, saving a lot of time.

### 1.4 Definitions, Acronyms, and Abbreviations

* See Appendix A: Glossary

### 1.5 References

**Document Tag Parser**

* [**pytesseract documentation**](https://pypi.org/project/pytesseract/)
* [**tkinter documentation**](https://docs.python.org/3/library/tkinter.html)
* [**pdf2image library documentation**](https://pypi.org/project/pdf2image/)
* [**Python Documentation**](https://www.python.org/doc/)

# 2. Overall Description

**Document Tag Parser**

Document Tag Parser retrieves bate stamps from court documents and creates copies of the court documents in pdf. When it creates the copy the new name of the file will contain the bate stamp that was retrieved. Users also have the option to have the program generate extra strings to the file name.

### 2.1 System Analysis

**Document Tag Parser**

* PDF files will contain tag numbers that will be extracted from the pdf and added to the name of the pdf file.
* The goal is to speed up the process of renaming files from court PDF documents. Previously SBPD would manually open up each PDF file and manually retrieve the bate stamps in order to rename the PDF files.

### 2.2 Product Perspective

**Document Tag Parser**Document Tag parser app is created with python. It requires the use of three libraries; pytesseract OCR that extracts printed text from images, pdf2image converts pdf2s to images, and tkinter GUI toolkit to create a GUI.

### 2.3 Product Functions

**Document Tag Parser**

* Read text transcribed from text images in pdfs.
* Retrieve bate stamps from text transcribed text images.
* Transcription of pdf tag and renaming of pdf files based on tag.

### 2.4 User Classes and Characteristics

**Document Tag Parser**

This software is designed with Public Defenders as the main users/recipients.

### 2.2 Operating Environment

**Document Tag Parser**

Currently this software will operate only on windows.

### 2.6 Design and Implementation Constraints

**Document Tag Parser**

Not a common occurrence but bate stamps sometimes come with extra symbols that don't match a description of a bate stamp. The DTP app will not be able to read the bate stamps and will say there are no bate stamps on the pdf form.

### 2.7 User Documentation

**Document Tag Parser**

* DTP user manual.pdf

### 2.8 Assumptions and Dependencies

**Document Tag Parser**

* DTP retrieves bate stamps with two attempts. First attempt is getting the text from the pdf, second attempt is converting the pdf two an image and then transcribing the text from the images with pytesseract. The user interface will use the tkinter python library.
* DTP will need to use pdf2image library, pytesseract, and tkinter python libraries.

**2.9 Apportioning of Requirements**

**Document Tag Parser**

N/A

# 

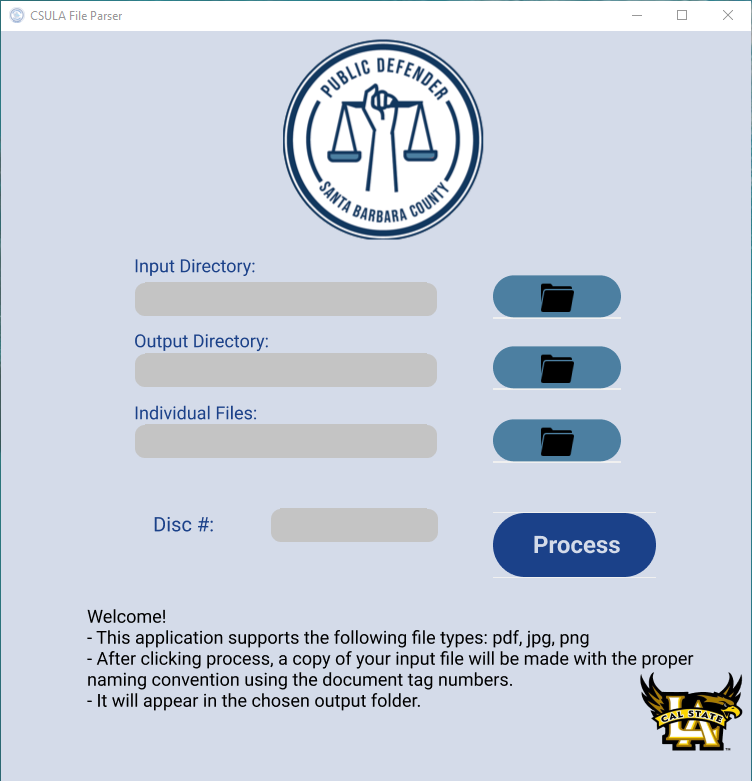
# 3. External Interface Requirements

**Document Tag Parser**

The document tag parser app will be a desktop program installed on the machine of the user and the GUI will be built with the tkinter library.

### 3.1 User Interfaces

**Document Tag Parser App**

The user will interact with a desktop program that is installed on the user's machine. The program lets the user choose the file directory they would like to read from, and the directory to place the text file into. 

### 3.2 Hardware Interfaces

**Document Tag Parser App**

N/A

### 3.3 Software Interfaces

**Document Tag Parser App**

DTP uses a Graphical User Interface that is built with the tkinter python library.

### 3.4 Communications Interfaces

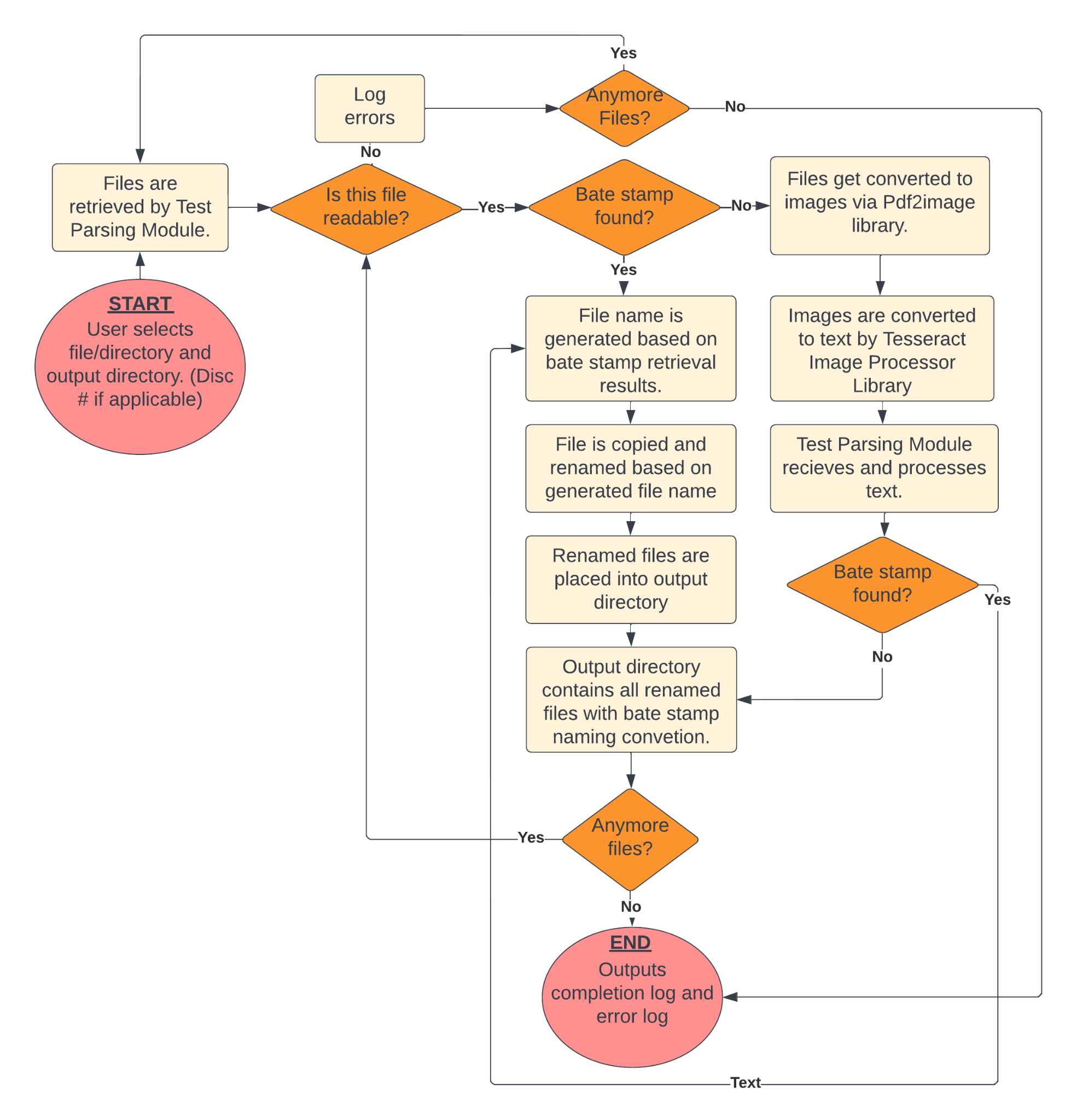
**Document Tag Parser App**

N/A

# 

# 4. Requirements Specification

**Document Tag Parser App**

****

This is an installer program which allows users to parse any pdf and image files with specific requirements set by SBPD. Furthermore, the program will be able to process multiple files at once and give users the flexibility of processing by folder or by files. The program will accept a disc number from the user and locate the starting and ending bate stamps within each file and combine all information to rename and make a copy of the files using SBPD’s naming convention. For better user experience, the program will stay responsive while processing files and a loading bar lets the user know it is processing files. To speed up the process, we utilize multi threads to process files in parallel sessions with four threads running concurrently. At the end, the program will provide a completion log indicating the files that potentially need user attention, and all the original files will remain unchanged.

# 4.1 Functional Requirements

**Document Tag Parser App**

| **1.1:Document Tag Parser(DTP)** | |
| --- | --- |
| 4.2-1.1-1 | DTP shall convert pdfs and images into text files for parsing |
| 4.2-1.1-2 | DTP shall read the tag numbers on the first and last page of each document |
| 4.2-1.1-3 | DTP shall rename the file based on the SBPD’S naming convention |
| 4.2-1.1-4 | DTP shall add the disc number based on the user’s choice within the application |
| 4.2-1.1-5 | DTP shall not modify the original file |
| 4.2-1.1-6 | DTP shall create new copies with the appropriate file naming convention |
| 4.2-1.1-7 | DTP shall allow input by files and/or by folder |
| 4.2-1.1-8 | DTP shall display processing messages |
| 4.2-1.1-9 | DTP shall give a completion log indicating files need user attention |
| 4.2-1.1-10 | DTP shall be responsive while processing the files |
| 4.2-1.1-11 | DPT shall optimize processing speed with multithreading |
| 4.2-1.1-12 | DPT shall display loading bar while processing |

### 4.2 External Interface Requirements

**Document Tag Parser**

* Processing files must be pdf, png, or jpg
* Currently only compatible with windows system computers.

### 4.3 Logical Database Requirements

N/A

### 4.4 Design Constraints

**Document Tag Parser App**

* The program can only pick up 6 digit bate stamp numbers, edge cases with extra symbols next to bate stamp numbers will not work.
* The program has a considerably long processing time with big image files
* When input by folder, sub folders will not be processed.

# 

# 5. Other Nonfunctional Requirements

### 5.1 Performance Requirements

**Document Tag Parser**

* Currently only compatible with windows system computers.
* Large image files can cause longer processing/execution time
* Supports multithreading

### 5.2 Safety Requirements

N/A

### 5.3 Security Requirements

N/A

### 5.4 Software Quality Attributes

**Document Tag Parser App**

Document Tag Parser is a standalone application that runs after installation. It can be installed as necessary and requires no external resources other than the Python modules listed above. See appendix for more information.

### 5.5 Business Rules

N/A

# 

# 

# 6. Legal and Ethical Considerations

N/A

# Appendix A: Glossary

* SRS: Software Requirements Specification
* AI/ML: Artificial Intelligence/Machine Learning
* DTP: Document Tag Parser
* SBPD: Santa Barbara Public Defender

# Appendix B: Analysis Models

* Refer to flow diagram in Section 4

# Appendix C: To Be Determined List

* Document parser app to extract tags from documents and append it to file names

**Software Requirements Specification**

**for**

Box.com/eDefender Integration

**Version 2.1 approved**

**Prepared by** **Joshua Cabrera, Marco De La Torre, Raul Gallegos, Daniel Guevara-Dominguez, Chuang Huang, Dang Le, Jesica Lopez De Leon, Shaocheng Shi, Sergio Tapia, Luke Williams**.

Advisor: Jungsoo (Soo) Lim

Santa Barbara Public Defender Office

**May 10, 2022**

**Table of Contents**

[**Revision History**](#_gjdgxs) **18**

[**1. Introduction**](#_30j0zll) **19**

[1.1 Purpose](#_1fob9te) 19

[1.2 Intended Audience and Reading Suggestions](#_3znysh7) 19

[1.3 Product Scope](#_2et92p0) 20

[1.4 Definitions, Acronyms, and Abbreviations](#_tyjcwt) 20

[1.5 References](#_3dy6vkm) 20

[**2. Overall Description**](#_1t3h5sf) **20**

[2.1 System Analysis](#_4d34og8) 20

[2.2 Product Perspective](#_2s8eyo1) 20

[2.3 Product Functions](#_17dp8vu) 21

[2.4 User Classes and Characteristics](#_3rdcrjn) 21

[2.2 Operating Environment](#_26in1rg) 21

[2.6 Design and Implementation Constraints](#_lnxbz9) 21

[2.7 User Documentation](#_35nkun2) 21

[2.8 Assumptions and Dependencies](#_1ksv4uv) 21

[2.9 Apportioning of Requirements](#_44sinio) 22

[**3. External Interface Requirements**](#_2jxsxqh) **22**

[3.1 User Interfaces](#_z337ya) 22

[3.2 Hardware Interfaces](#_3j2qqm3) 22

[3.3 Software Interfaces](#_1y810tw) 23

[3.4 Communications Interfaces](#_4i7ojhp) 23

[**4. Requirements Specification**](#_2xcytpi) **24**

[**4.1 Functional Requirements**](#_1ci93xb) **25**

[4.2 External Interface Requirements](#_3whwml4) 26

[4.3 Logical Database Requirements](#_2bn6wsx) 26

[4.4 Design Constraints](#_qsh70q) 26

[**5. Other Nonfunctional Requirements**](#_3as4poj) **27**

[5.1 Performance Requirements](#_1pxezwc) 27

[5.2 Safety Requirements](#_49x2ik5) 27

[5.3 Security Requirements](#_2p2csry) 27

[5.4 Software Quality Attributes](#_147n2zr) 27

[5.5 Business Rules](#_3o7alnk) 27

[**6. Legal and Ethical Considerations**](#_23ckvvd) **28**

[**Appendix A: Glossary**](#_ihv636) **29**

[**Appendix B: Analysis Models**](#_32hioqz) **29**

[**Appendix C: To Be Determined List**](#_1hmsyys) **29**

**Revision History**

| **Name** | **Date** | **Reason For Changes** | **Version** |
| --- | --- | --- | --- |
| Team | 10/7/21 | First Draft | 1.0 |
| Team | 12/7/21 | Update Requirements, Project Scope | 1.1 |
| Team | 5/10/22 | Revised and Final Draft | 2.1 |
|  |  |  |  |

# 1. Introduction

**Box.com/eDefender Integration**

The purpose of this Box.com and eDefender integration software is to serve as an intermediary between Box.com and E-Defender in order to provide automatic transcription services and other advanced machine learning functionality.

### 1.1 Purpose

The purpose of this document is the following:

1. Identify the requirements for Box.com and eDefender integration (Version
2. Identify the needed software and how it is being used.
3. State how the user is meant to interact and use the product.
4. How the product will interact with any external service.
5. Define and describe concepts that will clarify the functions of the product.

**Box.com/eDefender Integration**

This Box.com and eDefender integration borrows heavily from previous implementations that provided similar functionality for the City of Los Angeles. This document covers the features that were implemented on behalf of the City of Santa Barbara by California State University, Los Angeles which includes facial recognition, voice transcription, pdf text transcription and tagging, and translation for supported languages.

### 1.2 Intended Audience and Reading Suggestions

This document is intended for developers, testers, project managers, and users to understand the purpose, function, and requirements of the product. Suggested reading sequence for each is the following:

**Box.com/eDefender Integration**

* Developers - Understand the purpose and functions of the product to help during development; Recommended Reading: REST + API Documentation.
* Testers - Understand the purpose and requirements of the product to help during testing to guarantee intended function; Recommended Reading: REST + API Documentation
* Users - Understand the purpose of the product and its functions; Recommended Reading: Sections 1 - 5

### 1.3 Product Scope

**Box.com/eDefender Integration**

Box.com and E-Defender integration will allow for reduced workloads on lawyers and public defenders by providing transcription services for digital audio and/or video evidence. The transcription will include timestamps for the dialogue as well as faces of parties involved.

### 1.4 Definitions, Acronyms, and Abbreviations

* See Appendix A: Glossary

### 1.5 References

**Box.com/Edefender integration**

* REST+API.doc
* user\_manual\_box\_indexer.pdf
* https://journaltech.com/
* https://box.com
* <https://aws.amazon.com/lambda/>

# 

# 2. Overall Description

**Box.com/eDefender Integration**

Box Skills makes use of third-party AI/ML to get metadata within media files, it is not a product on its own and requires the use of other APIs. Box skills node is a middleware and needs to be hosted on a serverless function.

### 2.1 System Analysis

**Box.com/eDefender Integration**

* The goal is to provide automatic transcription services for audio and/or video files through Box.
* File extension support may vary, and errors or debugging is not covered.
* Our design right now is to have AWS Lambda as our serverless function and Azure Video Analyzer to analyze .

### 2.2 Product Perspective

**Box.com/eDefender Integration**

Box Skills acts as middleware between third-party AI/ML services and other API technologies. As a result, Box Skills needs to be hosted on a serverless function. Many other solutions exist, including but not limited to, Google and Microsoft transcription services. Prices will differ and Box was chosen by sponsor for their skills technology.

### 2.3 Product Functions

### **Box.com/eDefender Integration**

* Transcription of audio and/or video files into English, Spanish.
* Facial recognition and indexing, providing timestamps
* Transcription of pdf tag and renaming of pdf files based on tag.

### 2.4 User Classes and Characteristics

**Box.com/eDefender Integration**

This software is designed with Public Defenders as the main users/recipients.

### 2.2 Operating Environment

**Box.com/eDefender Integration**

This software will operate on serverless technologies and cloud-based networks.

### 2.6 Design and Implementation Constraints

**Box.com/eDefender Integration**

There are associated costs for each video being transcribed of $0.0000166667 per GB-s as of December 2021. Additionally, not all languages are as readily transcribable. For languages that are supported, accuracy of transcription services is not guaranteed to be 100% accurate. Access to an internet connection is required for accessing Box and E-Defender.

### 2.7 User Documentation

**Box.com/eDefender Integration**

* REST+API.doc
* user\_manual\_box\_indexer.pdf

### 2.8 Assumptions and Dependencies

**Box.com/eDefender Integration**

* AWS Lambda retrieves a file from Box and analyzes it before the metadata is sent back to Box from AWS.
* The Box Skill Application will need to be hosted on the Serverless Function AWS Lambda.

### 2.9 Apportioning of Requirements

**Box.com/eDefender Integration**

* Transcribing multiple languages
* Sending a notification of completion to desired parties involved after transcription

# 

# 3. External Interface Requirements

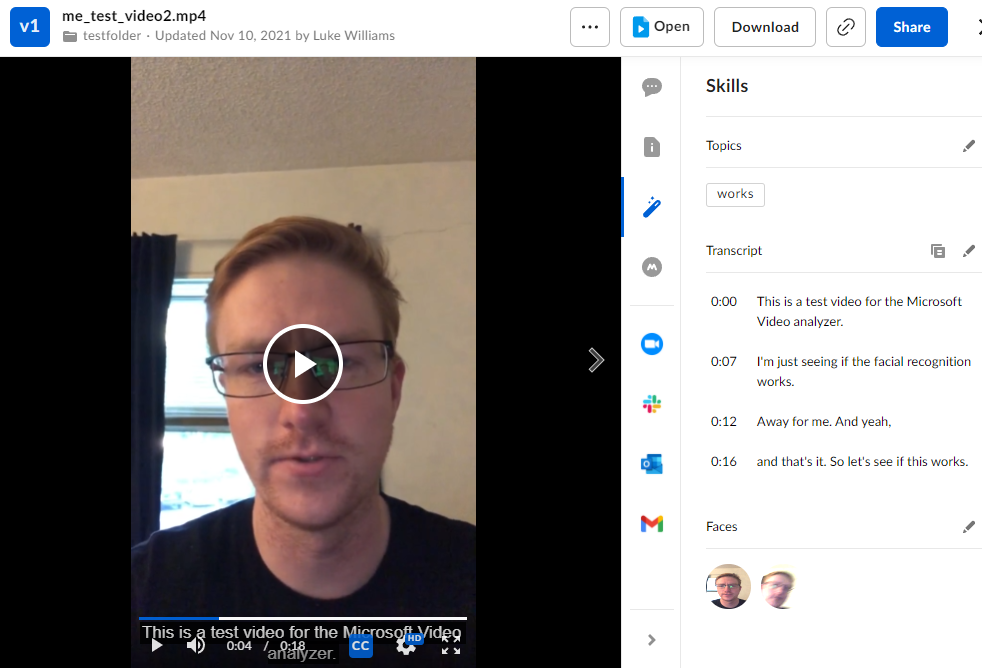
**Box.com/eDefender Integration**

All aspects of the Box skill will be running and controlled through the cloud and webapps. Once deployed, the Box skill will run automatically once a file is uploaded to Box.

### 3.1 User Interfaces

**Box.com/eDefender Integration**

The user must have a Box account given to them by the Santa Barbara County Public Defender Office. With the account, the user may login and upload a video to its respective skill enabled folder. The user will receive an eDefender notification of the action, and now only has to wait for a confirmation of the file being successfully uploaded and transcribed. Once the confirmation is given, the metadata will be displayed on the side as a skill.



### 3.2 Hardware Interfaces

**Box.com/eDefender Integration**

N/A

### 3.3 Software Interfaces

**Box.com/eDefender Integration**

N/A

### 3.4 Communications Interfaces

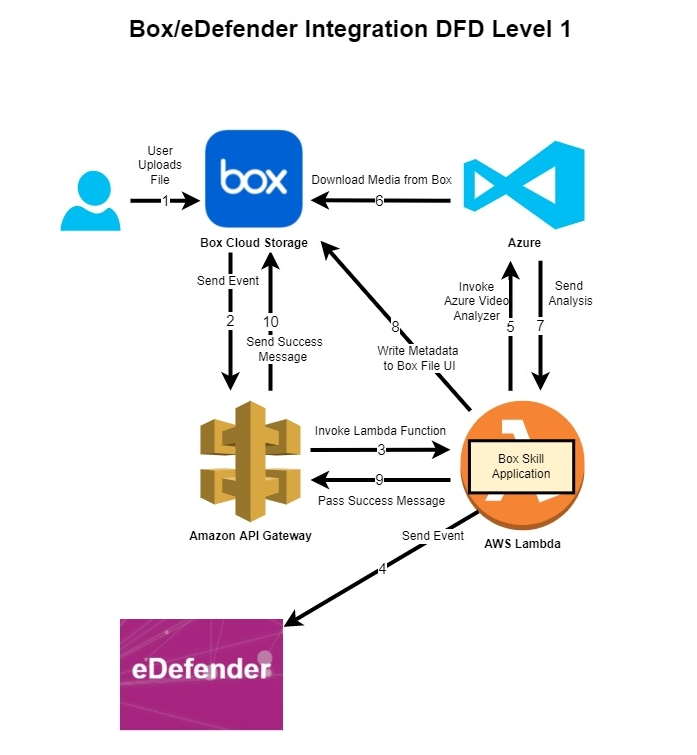
**Box.com/eDefender Integration**

Box invokes AWS, which then invokes the lambda function and then finally the rest of the program. For this to happen, these portions of the program will need to be interconnected via security keys, invocation URLs, and access tokens.

# 

# 4. Requirements Specification

**Box.com/eDefender Integration**



Users should be able to upload video and audio to Box and Box custom skill should trigger upon uploading files. Box skill should send access tokens to serverless hosting sites(AWS), and later pass to ML service(Azure Video Indexer) for file retrieval. ML service should produce metadata including transcription, facial recognition, and topic detection and the hosting site should write that metadata back to the video in Box. A notification should be sent to e-Defender once the video is ready to view with all the metadata embedded to it.

# 4.1 Functional Requirements

**Box.com/e-Defender Integration**

| **1.1:Box** | |
| --- | --- |
| 4.2-1.1-1 | Box shall allow users upload and download files |
| 4.2-1.1-2 | Box shall trigger BCS upon file upload to pre-configured folder |
| 4.2-1.1-3 | Box shall send access token to hosting site(AWS) |
| 4.2-1.1-4 | Box shall embed metadata to original video/audio |
| 4.2-1.1-5 | Box shall visualize metadata as transcript, face, and topic card |

| **1.2:AWS** | |
| --- | --- |
| 4.2-1.2-1 | AWS shall host the Box skill application |
| 4.2-1.2-2 | AWS shall capture and parse all information of an event send by Box |
| 4.2-1.2-3 | AWS shall send an event to ML service(Azure Video Analyzer) |
| 4.2-1.2-4 | AWS shall send access token to ML service |
| 4.2-1.2-5 | AWS shall receive response from ML service |
| 4.2-1.2-6 | AWS shall retrieve metadata from ML service |
| 4.2-1.2-7 | AWS shall write metadata back to Box |

| **1.3:ML service(AVA)** | |
| --- | --- |
| 4.2-1.3-1 | ML service shall listen for AWS call to process video/audio |
| 4.2-1.3-2 | ML service shall use access token to retrieve file from Box |
| 4.2-1.3-3 | ML service shall produce transcription of a file |
| 4.2-1.3-4 | ML service shall perform language auto detection |
| 4.2-1.3-5 | ML service shall recognize faces and keywords |
| 4.2-1.3-6 | ML service shall provide timestamp for analyzed result |

### 4.2 External Interface Requirements

**Box.com/eDefender Integration**

* Files upload to Box restricted to video or audio only
  + Acceptable video formats: aac, aif, aifc, aiff, amr, au, flac, m4a, mp3, ogg, ra, wav, wma.
  + Acceptable audio format: 3g2, 3gp, avi, m2v, m2ts, m4v, mkv, mov, mp4, mpeg, mpg, ogg, mts, qt, ts, wmv.

### 4.3 Logical Database Requirements

N/A

### 4.4 Design Constraints

**Box.com/eDefender Integration**

* ML services do not provide 100% accurate transcription.
* Inaccuracy increases with multiple languages, voices, and background noises.
* Language auto detection happens when language is not specified, thus increasing inaccuracy and wrong language when generating transcripts.
* Faces recognition not 100% working in picking up faces.
* Topic detection could be irrelevant.

# 5. Other Nonfunctional Requirements

### 5.1 Performance Requirements

* Media metadata extraction time should be at least 1:1 with video length.
* Accuracy should be up to standard with current transcription services.
* Media file will be handled box skills enterprise plan uploaded and handled can be up to 150 gb.

### 5.2 Safety Requirements

N/A

### 5.3 Security Requirements

Files distributed outside of Box should be deleted after metadata is extracted, as they are often important evidence of ongoing cases.

### 5.4 Software Quality Attributes

Box Skill Application is flexible and can be hosted in a number of ways. On a server, on serverless functions, or on a cloud server. AI/ML services can also be replaced. This process is simple as it only needs to be directed by AWS Lambda to make its call to a specific AI/ML.

### 5.5 Business Rules

N/A

# 6. Legal and Ethical Considerations

N/A

# Appendix A: Glossary

* SRS: Software Requirements Specification
* AI/ML: Artificial Intelligence/Machine Learning
* Box: Cloud Storage provider
* Box Skills: A framework used to provide customization for folder/files on Box.
* AWS: Amazon Web Services
* AWS Lambda: A serverless, event-driven compute service
* E-Defender: supplies case management software to courts, prosecutors, public defenders, probation and other justice agencies.
* BCS: Box Custom Skill
* AVA: Azure Video Indexer
* SBPD: Santa Barbara Public Defender

# Appendix B: Analysis Models

* Refer to flow diagram in Section 4

# Appendix C: To Be Determined List

* Document parser app to extract tags from documents and append it to file names