**Software Design**

**Document**

**for**

Document Tag Parser

**Version 2.2 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 9, 2022**

**Table of Contents**

[**Revision History**](#_heading=h.wkva0wft3bz4) **3**

[1.1 Purpose](#_heading=h.s0mvo7sggnqz) 4

[1.2 Document Conventions](#_heading=h.7lbpmg5hr8p4) 4

[1.3 Intended Audience and Reading Suggestions](#_heading=h.2jlip7mynx94) 4

[1.4 System Overview](#_heading=h.z5d9b3yqq30k) 5

[**2. Design Considerations**](#_heading=h.a5uznjextfv2) **5**

[2.1 Assumptions and Dependencies](#_heading=h.bcvp36cghen5) 6

[2.2 General Constraints](#_heading=h.egrdpcadr85y) 6

[2.3 Goals and Guidelines](#_heading=h.l9r28aohga9t) 6

[2.4 Development Methods](#_heading=h.nslqnntahs1g) 7

[**3. Architectural Strategies**](#_heading=h.v3t4bt618ar7) **7**

[**4. System Architecture**](#_heading=h.fmt3pmze7c8) **8**

[**5. Policies and Tactics**](#_heading=h.p16qgilp1cet) **10**

[5.1 Choice of which specific products used](#_heading=h.95br2wx6t4xm) 10

[5.2 Plans for ensuring requirements traceability](#_heading=h.97xw0thw8iz3) 11

[5.3 Plans for testing the software](#_heading=h.2ocmr8ky4cl) 11

[**6. Detailed System Design**](#_heading=h.lfgseeomszbh) **10**

[6.1](#_heading=h.g4k9ny4hqwv3) Text Parsing Module 10

[6.1.1 Responsibilities](#_heading=h.tz2vr13b7md5) 10

[6.1.2 Constraints](#_heading=h.i3bzgtuwj5n) 10

[6.1.3 Composition](#_heading=h.9rd5b7yumy02) 10

[6.1.4 Uses/Interactions](#_heading=h.cjpu8mrmo60e) 10

[6.1.5 Resources](#_heading=h.nea9hghmjnjo) 10

[6.1.6 Interface/Exports](#_heading=h.e05jb66fhbdb) 10

[6.2](#_heading=h.9997zthjzx6i) Pytesseract 10

[6.2.1 Responsibilities](#_heading=h.yve9u1yncfzi) 10

[6.2.2 Constraints](#_heading=h.5gst5gh36dcv) 10

[6.2.3 Composition](#_heading=h.79650923hbss) 11

[6.2.4 Uses/Interactions](#_heading=h.xutu2mnbr4xl) 11

[6.2.5 Resources](#_heading=h.5hqj0qgska5a) 11

[6.2.6 Interface/Exports](#_heading=h.tifqkbyqefet) 11

[6.3](#_heading=h.itq9lkgs7pun) UI Module 11

[6.3.1 Responsibilities](#_heading=h.yro69g1n6w9v) 11

[6.3.2 Constraints](#_heading=h.cj6knulmhqsy) 11

[6.3.3 Composition](#_heading=h.1hz002qie4s) 11

[6.3.4 Uses/Interactions](#_heading=h.b8h3hu9v3ow8) 11

[6.3.5 Resources](#_heading=h.h6dsd9sf3b04) 11

[6.3.6 Interface/Exports](#_heading=h.fitbahfua7pv) 11

[**7. Detailed Lower level Component Design**](#_heading=h.8aidug32iae1) **12**

[**8. Database Design**](#_heading=h.htrw9o8pjxx8) **12**

[**9. User Interface**](#_heading=h.ez4o6c56y7bf) **12**

[9.1 Overview of User Interface](#_heading=h.e2612sz24454) 12

[9.2 Screen Frameworks or Images](#_heading=h.793pqjio35kg) 13

[**10. Requirements Validation and Verification**](#_heading=h.xfssfmpnwqa) **15**

[10.1 Functional Requirements](#_heading=h.kxuv9vrhbme) 15

[**11. Glossary**](#_heading=h.o6puoconkr6y) **21**

[**12. References**](#_heading=h.idea7edo9mgd) **21**

# Revision History

| Name | Date | Reason For Changes | Version |
| --- | --- | --- | --- |
| Team | 3/8/22 | Implement parser app documentation | 2.1 |
| Team | 3/16/22 | Fixed outline to separate parser from Box | 2.1 |
| Team | 5/9/22 | Revised and Final Draft | 2.2 |
|  |  |  |  |

# 1. Introduction

### 1.1 Purpose

The Public Defender’s office of Santa Barbara receives a large volume of legal documents everyday. These documents contain Bates Stamps, which are numbers present on each page of evidence, and are used to organize their case exhibits. Employees at the public defender’s office spend many hours a week on this task.Document Tag Parser is an application intended to automate the process, by reading the Bates stamps on a document and renaming large batches of files based on our sponsor’s naming convention.

### 1.2 Document Conventions

See glossary

### 1.3 Intended Audience and Reading Suggestions

This document is intended for software developers, project managers, testers, and users. This document covers the initial specs and research done on the Box.com/ eDefender integration. It also covers the specifics of the Document Tag Parser application, its frontend and backend design, as well as how it fulfilled the requirements set by SBPD Suggested reading sequence for each is the following:

* Developers - Understand the purpose and functions of the product to help during development; Recommended Reading: Section(s) 2, 4, 6-8
* Project Managers - Understand the purpose of the product and its functions; Recommended Reading: Section(s) 2-5
* Testers - Understand the purpose and requirements of the product to help during testing to guarantee intended function; Recommended Reading: Section(s) 2, 4, 5, 9
* Users - Understand the purpose of the product and its functions; Recommended Reading: Section(s) 1, 9

### 1.4 DTP System Overview

Document Tag Parser is a Python-based desktop GUI application which utilizes several popular libraries to assist in the processing of pdf documents as well as images. In addition to standard pdf text documents, it uses the machine learning based pytesseract library to convert pdf images, png, and jpg files to text for the parsing process. The libraries are grouped together in three separate modules: Text Parsing, Pytesseract, and the UI Module. Please refer to the data flow diagram for a more concise overview.

# 2. Design Considerations

### 2.1 Assumptions and Dependencies

* Pytesseract OCR Library
* Pdf2image Library
* PyMuPDF
* Tkinter

### 

### 2.2 General Constraints

* Accuracy of file renaming depends on image to text conversion of the tesseract machine learning algorithm.
* Processing large numbers of image based documents can increase the time to completion of a batch of files significantly.
* Documents vary based on file type and position of the tag number on each document. Multiple rounds of testing will be needed to ensure accurate file renaming.

### 2.3 Goals and Guidelines

* Deliver final project in April 2022
* Ensure accuracy of document renaming by handling different document file types appropriately
* In the case of user error or unanticipated edge cases, log which files have failed to process and give a summary of the reason for each
* Create a user guide for installing the application
* Create a video demonstration on the features of the application
* Create a developer guide for future modifications and improvements

### 2.4 Development Methods

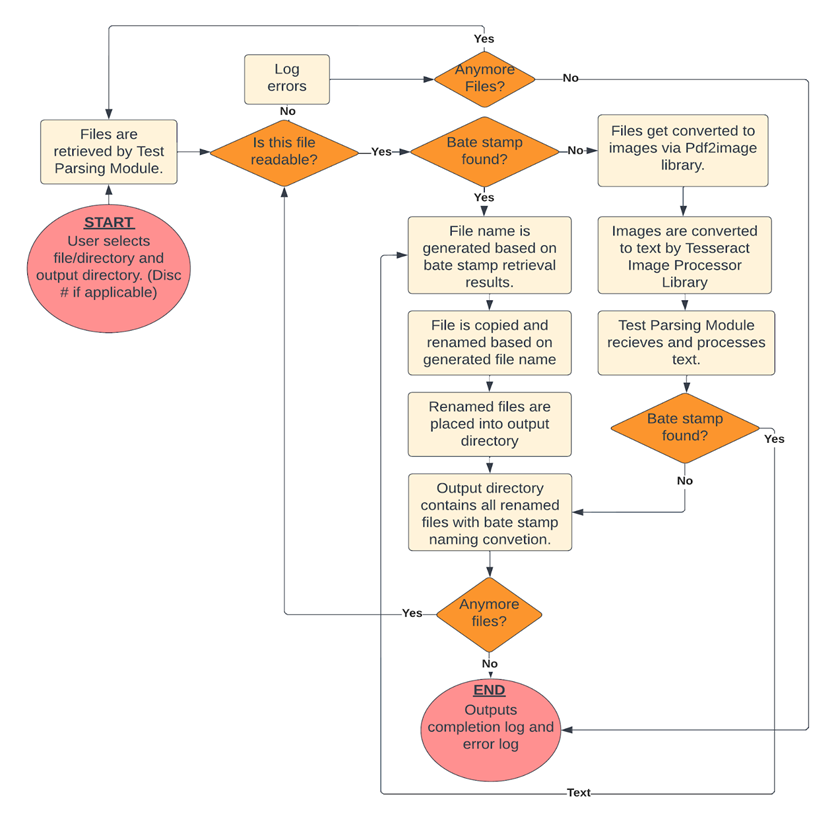
* We are using the Agile Development Methods
* A github repository stores all the code and history. Team members are assigned features and have branches associated with their update. These branches are then merged into the main branch.
* InnoSetup is used to create an installer which can be ran by any user regardless of technical experience
* The pyinstaller library is an alternative used to compile the program into an exe which can be run on any Windows system
* Testing the Document Tag Parser software utilizes example files provided by the Public Defender's office of Santa Barbara. Testing so far has been conducted manually using the GUI by each team member. So far, over 180 separate files have been tested, and the most recent round achieved a 94% accuracy rate. Improvements have been made since then, and it is likely that this rate will increase.

# 3. Architectural Strategies

* Standalone desktop application which is installed and made available on a user’s machine as a folder containing all assets and an exe file.
* Combines the functionality of libraries mentioned in Section 2.1, represented by three modules
* Application uses 4 threads, which greatly increases the processing time, particularly for image-based documents.
* An additional thread is used to maintain the responsiveness of the UI during processing
* Refer to DFD Level 1 below for detailed architecture

# 

# 4. System Architecture

**Document Tag Parser DFD Level 1**

# 

# 5. Policies and Tactics

### 5.1 Choice of which specific products used

The Pytesseract OCR Library is used to convert images into lines of text which can be analyzed. The Pdf2image library allows conversions of pdf images into standard image file types such as jpg and png which can be converted by Pytesseract into text. PyMuPDF/Poppler provides a convenient framework for manipulating pdf files, and provides features such as retrieving page count, opening files, and writing files. Tkinter is a popular UI framework which was used to create the desktop graphical user interface.

### 5.2 Plans for ensuring requirements traceability

N/A

### 5.3 Plans for testing the software

Testing the Document Tag Parser software utilizes example files provided by the Public Defender's office of Santa Barbara. Our software processes pdfs that are in both text and image format. Test cases will include both formats, as well as invalid file formats such as images and videos. See 2.4 for further details on testing results.

# 6. Detailed System Design

### 6.1 Test Parsing Module

#### 6.1.1 Responsibilities

Handles parsing of text and file manipulation.

#### 6.1.2 Constraints

The poppler library that PyMuPDF depends on is not able to be compiled into the exe using the framework we used, so it must be included as a separate folder within the application folder.

#### 6.1.3 Composition

Uses the PyMuPDF library which depends on the Poppler pdf rendering library to manipulate pdf files.

#### 6.1.4 Uses/Interactions

Interacts with the GUI to show the current status of the selected files and works with Pytesseract as needed to process image based files.

#### 6.1.5 Resources

Relies on multiple threads to split a batch of files into separate groups for greater efficiency when parsing. This is especially important when dealing with large batches of files.

#### 6.1.6 Interface/Exports

N/A

### 6.2 Pytesseract

#### 6.2.1 Responsibilities

#### Processes images and converts them into text files for use by the Text Parsing Module.

#### 6.2.2 Constraints

Cannot directly convert pdf images into text files. A separate library, pdf2image, is required to handle this task.

#### 6.2.3 Composition

#### It uses the pytesseract image processing library for conversion of images to text. It also utilizes the separate pdf2image library which converts pdf images into png files.

#### 6.2.4 Uses/Interactions

Interacts with the Text Parsing Module and GUI module.

#### 6.2.5 Resources

Relies on multiple threads to divide the computationally complex process of image conversion. This is especially important when dealing with large batches of files.

#### 6.2.6 Interface/Exports

N/A

### 6.3 UI Module

#### 6.3.1 Responsibilities

Renders the UI and works with both the Text Parsing and Pytesseract modules.

#### 6.3.2 Constraints

Users must have authorization provided by SBPD. Currently a lot of manual work is being done by the SBPD support team. The UI must be suitable for any user regardless of technical acumen.

#### 6.3.3 Composition

Uses the popular python GUI framework Tkinter to render the user interface. It also depends on a folder of assets consisting of icons, buttons, and graphics.

#### 6.3.4 Uses/Interactions

Provides a user interface for choosing files, and interacts with the text parsing module to manipulate and rename files.

#### 6.3.5 Resources

Relies on multiple threads to show the progress of the processing of image files. Without a multi-threaded approach, the program does not respond to the user or provide any feedback until processing is complete.

#### 6.3.6 Interface/Exports

N/A

# 7. Detailed Lower level Component Design

N/A

# 8. Database Design

There is no database for this application, states are stored entirely within the app itself.

# 

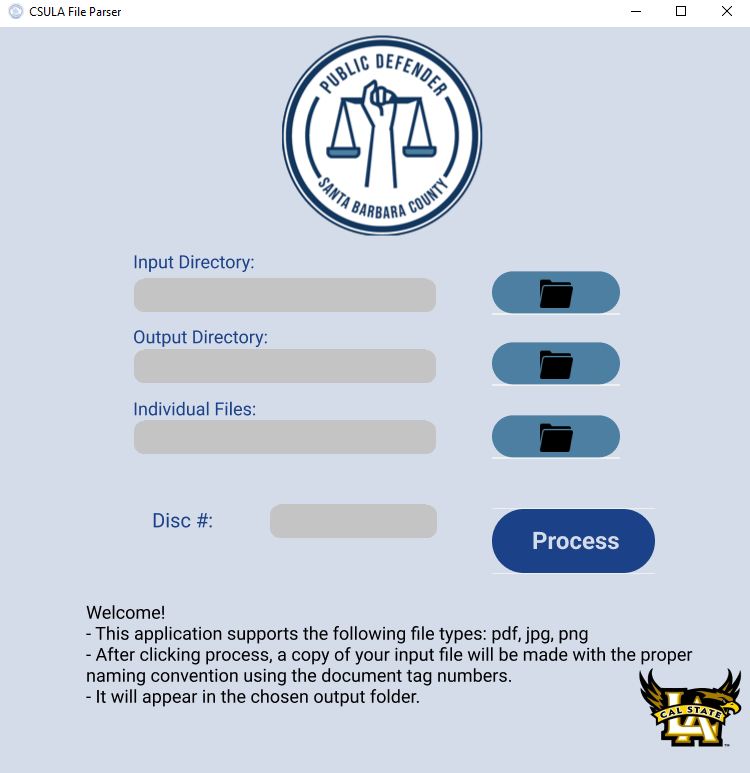
# 9. User Interface

### 9.1 Overview of User Interface

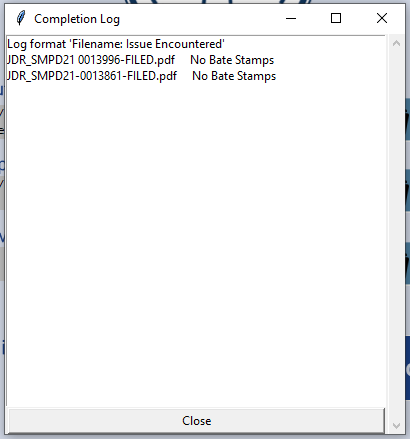
Upon startup, the app displays the interface shown in the image below. The user has a choice of choosing a folder where all documents inside will be processed, choosing specific files, or choosing a combination of both. A disc number can be entered if the source comes from a disc, as specified by SBPD’s requirements. If none exists, the field can be left blank. Upon completion, if all documents are processed without issue, a success message is shown, otherwise, an error log pop up is displayed (see next page for example).

### 9.2 Screen Frameworks or Images

Document Tag Parser Current UI



Example of DTP Error Log



**9.3 User Interface Flow Model**

N/A

# 10. Requirements Validation and Verification

### 10.1 Functional Requirements

| **Functional Requirements** | | **Met by** |
| --- | --- | --- |
| 10.1-1 | DTP shall convert pdfs and images into text files for parsing | 6.2 Pytesseract |
| 10.1-2 | DTP shall read the tag numbers on the first and last page of each document | 6.1 Text Parsing Module |
| 10.1-3 | DTP shall rename the file based on the SBPD’S naming convention | 6.1 Text Parsing Module |
| 10.1-4 | DTP shall add the disc number based on the user’s choice within the application | 6.1 Text Parsing Module |
| 10.1-5 | DTP shall not modify the original file | 6.1 Text Parsing Module |
| 10.1-6 | DTP shall create new copies with the appropriate file naming convention | 6.1 Text Parsing Module |
| 10.1-7 | DTP shall allow input by files and/or by folder | 6.3 UI Module |
| 10.1-8 | DTP shall display processing messages | 6.3 UI Module |
| 10.1-9 | DTP shall give a completion log indicating files need user attention | 6.3 UI Module |
| 10.1-10 | DTP shall be responsive while processing the files | 6.3 UI Module |
| 10.1-11 | DPT shall optimize processing speed with multithreading | 6.1 Text Parsing Module |
| 10.1-12 | DPT shall display loading bar while processing | 6.3 UI Module |

**11. Glossary**

* SBPD: Santa Barbara Public Defender Office
* SRS: Software Requirements Specification
* SDD: Software Design Document
* AI/ML: Artificial Intelligence/Machine Learning
* DTP: Document Tag Parser
* CJIS/HIPPA: Criminal Justice Information Services and Health Insurance Portability and Accountability Act
* Pytesseract: a library which allows developers to utilize Google’s Tesseract optical character recognition engine
* OCR: the conversion of handwritten or printed text into machine readable form.

# 12. References

* Documentation for libraries used:
  + <https://docs.python.org/3/library/tkinter.html>
  + <https://pymupdf.readthedocs.io/en/latest/>
  + <https://pypi.org/project/pdf2image/>
  + <https://pypi.org/project/pytesseract/>
  + <https://pypi.org/project/auto-py-to-exe/>
  + https://jrsoftware.org/isinfo.php

**Software Design**

**Document**

**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 9, 2022**

**Table of Contents**

[**Revision History**](#_heading=h.wkva0wft3bz4) **3**

[1.1 Purpose](#_heading=h.s0mvo7sggnqz) 4

[1.2 Document Conventions](#_heading=h.7lbpmg5hr8p4) 4

[1.3 Intended Audience and Reading Suggestions](#_heading=h.2jlip7mynx94) 4

[1.4 System Overview](#_heading=h.z5d9b3yqq30k) 5

[**2. Design Considerations**](#_heading=h.a5uznjextfv2) **5**

[2.1 Assumptions and Dependencies](#_heading=h.bcvp36cghen5) 6

[2.2 General Constraints](#_heading=h.egrdpcadr85y) 6

[2.3 Goals and Guidelines](#_heading=h.l9r28aohga9t) 6

[2.4 Development Methods](#_heading=h.nslqnntahs1g) 7

[**3. Architectural Strategies**](#_heading=h.v3t4bt618ar7) **7**

[**4. System Architecture**](#_heading=h.fmt3pmze7c8) **8**

[**5. Policies and Tactics**](#_heading=h.p16qgilp1cet) **10**

[5.1 Choice of which specific products used](#_heading=h.95br2wx6t4xm) 10

[5.2 Plans for ensuring requirements traceability](#_heading=h.97xw0thw8iz3) 11

[5.3 Plans for testing the software](#_heading=h.2ocmr8ky4cl) 11

[**6. Detailed System Design**](#_heading=h.lfgseeomszbh) **12**

[6.1 Box](#_heading=h.g4k9ny4hqwv3) 12

[6.1.1 Responsibilities](#_heading=h.tz2vr13b7md5) 12

[6.1.2 Constraints](#_heading=h.i3bzgtuwj5n) 12

[6.1.3 Composition](#_heading=h.9rd5b7yumy02) 12

[6.1.4 Uses/Interactions](#_heading=h.cjpu8mrmo60e) 12

[6.1.5 Resources](#_heading=h.nea9hghmjnjo) 12

[6.1.6 Interface/Exports](#_heading=h.e05jb66fhbdb) 12

[6.2 Box Skills App](#_heading=h.9997zthjzx6i) 12

[6.2.1 Responsibilities](#_heading=h.yve9u1yncfzi) 12

[Box Skills makes use of third-party AI/ML to get metadata within media files. Analyzes audio and video to produce a transcript of the speech detected in English and Spanish. Detects people in the videos using facial recognition.](#_heading=h.n423vweph3cd) 12

[6.2.2 Constraints](#_heading=h.5gst5gh36dcv) 12

[6.2.3 Composition](#_heading=h.79650923hbss) 13

[6.2.4 Uses/Interactions](#_heading=h.xutu2mnbr4xl) 13

[6.2.5 Resources](#_heading=h.5hqj0qgska5a) 13

[6.2.6 Interface/Exports](#_heading=h.tifqkbyqefet) 13

[6.3 eDefender](#_heading=h.itq9lkgs7pun) 13

[6.3.1 Responsibilities](#_heading=h.yro69g1n6w9v) 13

[6.3.2 Constraints](#_heading=h.cj6knulmhqsy) 13

[6.3.3 Composition](#_heading=h.1hz002qie4s) 13

[6.3.4 Uses/Interactions](#_heading=h.b8h3hu9v3ow8) 13

[6.3.5 Resources](#_heading=h.h6dsd9sf3b04) 13

[6.3.6 Interface/Exports](#_heading=h.fitbahfua7pv) 13

[**7. Detailed Lower level Component Design**](#_heading=h.8aidug32iae1) **14**

[**8. Database Design**](#_heading=h.htrw9o8pjxx8) **14**

[**9. User Interface**](#_heading=h.ez4o6c56y7bf) **15**

[9.1 Overview of User Interface](#_heading=h.e2612sz24454) 15

[9.2 Screen Frameworks or Images](#_heading=h.793pqjio35kg) 15

[**10. Requirements Validation and Verification**](#_heading=h.xfssfmpnwqa) **17**

[10.1 Functional Requirements](#_heading=h.kxuv9vrhbme) 17

[**11. Glossary**](#_heading=h.o6puoconkr6y) **21**

[**12. References**](#_heading=h.idea7edo9mgd) **21**

# Revision History

| Name | Date | Reason For Changes | Version |
| --- | --- | --- | --- |
| Team | 10/7/21 | First Draft | 1.0 |
| Team | 12/7/21 | Update Project Design | 1.1 |
| Team | 5/9/22 | Revised and Final Draft | 2.1 |
|  |  |  |  |

**1. Introduction**

### 1.1 Purpose

The purpose of this software is to integrate Box.com with Santa Barbara Public Defender's office case management system, eDefender. Allowing for uploaded documents/files to be transcribed/translated. This is achieved by utilizing Azure Video Analyzer which provides artificial intelligence to transcribe audio and video files. It builds upon a project created by a previous team at CSULA in 2020 titled “Extension for Public Defender's Client Case Management System.” This document is based on the software design document titled “Visual Analysis Using Cloud Services.”

### 1.2 Document Conventions

N/A

### 1.3 Intended Audience and Reading Suggestions

This document is intended for software developers, project managers, testers, and users. This document covers the specifics of the Box.com and eDefender integration along with information from Azure Video Analyzer which is used to transcribe/translate files. Suggested reading sequence for each is the following:

* Developers - Understand the purpose and functions of the product to help during development; Recommended Reading: Section(s) 2, 4, 6-8
* Project Managers - Understand the purpose of the product and its functions; Recommended Reading: Section(s) 2-5
* Testers - Understand the purpose and requirements of the product to help during testing to guarantee intended function; Recommended Reading: Section(s) 2, 4, 5, 9
* Users - Understand the purpose of the product and its functions; Recommended Reading: Section(s) 1, 9

### 1.4 System Overview

Our system will allow for users to upload files to Box.com, which will trigger a custom Box Skill Application that will transcribe and translate files. Once successfully completed users will be able to view metadata from the file that consists of transcription, facial recognition, and timestamps. It will also include a notification system within the eDefender case management system to alert the handling attorney.

The Box Skill application can be deployed to any server that can support a publicly available url such as serverless AWS lambda or Google cloud functions, as well as traditional servers on AWS and Firebase.

# 2. Design Considerations

Note that some of the dependencies and constraints have carried over from the original project, but some have changed.

### 2.1 Assumptions and Dependencies

* For the Box Skill App and Notification Service:
  + Node JS
  + Box SDK
  + AWS Lambda and S3 Bucket
  + Azure Video Analyzer
  + eDefender API

### 2.2 General Constraints

* When multiple people are speaking during a video or audio recording, transcription accuracy is unpredictable
* File security due to sensitive court documents and evidence
* eDefender
  + Notifications will need to be integrated into Santa Barbara Public Defender’s environment, and additional work will be needed outside of the Box Skill App

### 2.3 Goals and Guidelines

* Deliver final project in April 2022
* Box Skill App
  + Ensure accurate transcription and ease of use within Box UI

### 2.4 Development Methods

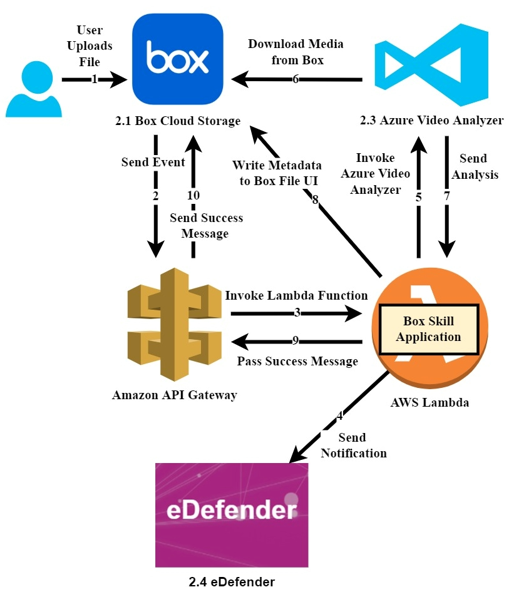
* We are using the Agile Development Methods

# 3. Architectural Strategies

* We are building upon the existing design by the prior CSULA team, document titled “Visual Analysis Using Cloud Services”
* Reusing Original Box Skill App
  + Add eDefender Alert System
  + Azure Video Indexer was used by the prior team, this new version of the application will use the updated version, Azure Video Analyzer
  + Migrate our deployment of the application on our team’s lambda function to Santa Barbara Public Defender’s instance

# 4. System Architecture

**Box/eDefender Integration DFD Level 1**



**.**

# 5. Policies and Tactics

### 5.1 Choice of which specific products used

Box.com which is a cloud content management tool allows the Public Defender to be compliant with the CJIS/HIPAA requirements. This platform also provides the custom Box skills application which enables the ability to apply third-party AI technologies to automatically analyze files as they are uploaded to Box.

Azure Video Analyzer is a cloud application which analyses both visual and audio files by running different AI models. As the video analyzer runs an analysis on the files, the insights are extracted by the AI models that are then provided.

### 5.2 Plans for ensuring requirements traceability

N/A

### 5.3 Plans for testing the software

The Box Skills application will be tested with video file types all of which contain different languages in order to test the accuracy of the Video Indexer. The application will also be tested with videos of varying lengths to test the tokens.

# 6. Detailed System Design

### 6.1 Box

#### 6.1.1 Responsibilities

Allows users to upload and download files.

#### 6.1.2 Constraints

Users must have a Box account provided by SBPD.

#### 6.1.3 Composition

N/A

#### 6.1.4 Uses/Interactions

Provides security by only allowing signed in authorized users. Provides file management features.

#### 6.1.5 Resources

Already existing file management web app Box.com

#### 6.1.6 Interface/Exports

Website Box.com

### 6.2 Box Skills App

#### 6.2.1 Responsibilities

#### Box Skills makes use of third-party AI/ML to get metadata within media files. Analyzes audio and video to produce a transcript of the speech detected in English and Spanish. Detects people in the videos using facial recognition.

#### 6.2.2 Constraints

It is not its own application; it requires use of other APIs. The transcriptions are not 100% reliable.

#### 6.2.3 Composition

N/A

#### 6.2.4 Uses/Interactions

N/A

#### 6.2.5 Resources

Uses AWS Lambda

#### 6.2.6 Interface/Exports

Works within Box.com

### 6.3 eDefender

#### 6.3.1 Responsibilities

File management system used by SBPD to organize case files. Alerts attorney when new discovery is received.

#### 6.3.2 Constraints

Users must have authorization provided by SBPD. Currently a lot of manual work is being done by the SBPD support team. Our goal is to automate some of these tasks.

#### 6.3.3 Composition

N/A

#### 6.3.4 Uses/Interactions

Interacts with Box to get transcribed files.

**6.3.5 Resources**

TBD

#### 6.3.6 Interface/Exports

TBD

# 7. Detailed Lower level Component Design

N/A

# 8. Database Design

N/A

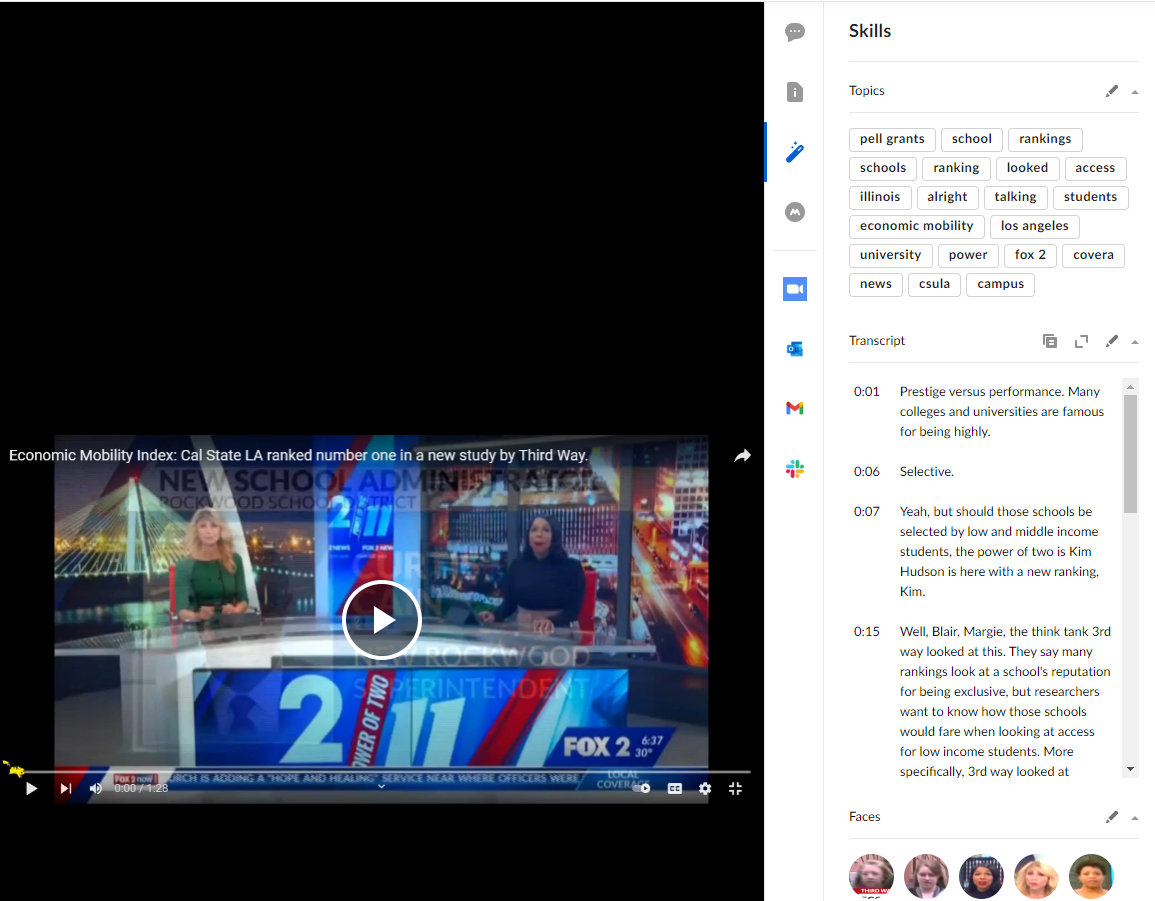
# 9. User Interface

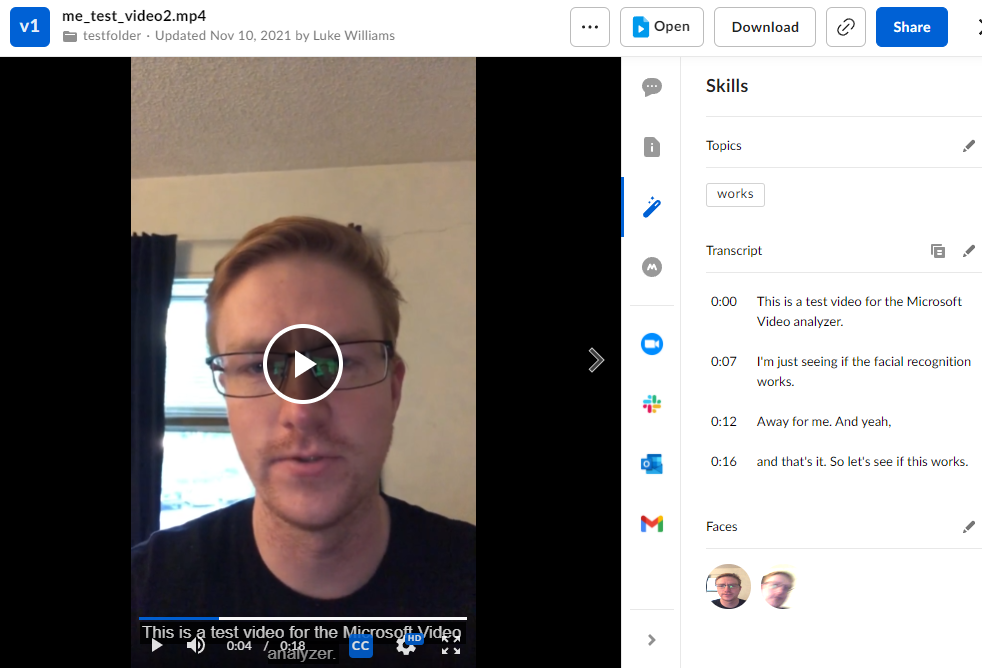
### 9.1 Overview of User Interface

Describe the functionality of the system from the user’s perspective. Explain how the user will be able to use your system to complete all the expected features and the feedback

Information that will be displayed for the user. This is an overview of the UI and its use. The user manual will contain extensive detail about the actual use of the software.

### 9.2 Screen Frameworks or Images

Box.com UI based on Azure Video Analyzer Results



**9.3 User Interface Flow Model**

N/A

# 10. Requirements Validation and Verification

### 10.1 Functional Requirements

**BOX/e-Defender Integration:**

| **1.1:Box** | | **Met by** |
| --- | --- | --- |
| 10.1-1.1-1 | Box shall allow users upload and download files | Box.com |
| 10.1-1.1-2 | Box shall trigger BCS upon file upload to pre-configured folder | Box.com |
| 10.1-1.1-3 | Box shall send access token to hosting site(AWS) | Box.com |
| 10.1-1.1-4 | Box shall embed metadata to original video/audio | Box Skills |
| 10.1-1.1-5 | Box shall visualize metadata as transcript, face, and topic card | Box Skills |

| **1.2:AWS** | | **Met by** |
| --- | --- | --- |
| 10.1-1.2-1 | AWS shall host the Box skill application | AWS Lambda |
| 10.1-1.2-2 | AWS shall capture and parse all information of an event send by Box | AWS Lambda |
| 10.1-1.2-3 | AWS shall send an event to ML service(Azure Video Analyzer) | AWS Lambda |
| 10.1-1.2-4 | AWS shall send access token to ML service | AWS Lambda |
| 10.1-1.2-5 | AWS shall receive response from ML service | AWS Lambda |
| 10.1-1.2-6 | AWS shall retrieve metadata from ML service | AWS Lambda |
| 10.1-1.2-7 | AWS shall write metadata back to Box | AWS Lambda |

| **1.3:ML service(AVA)** | | **Met by** |
| --- | --- | --- |
| 10.1-1.3-1 | ML service shall listen for AWS call to process video/audio | Azure Video Analyzer |
| 10.1-1.3-2 | ML service shall use access token to retrieve file from Box | Azure Video Analyzer |
| 10.1-1.3-3 | ML service shall produce transcription of a file | Azure Video Analyzer |
| 10.1-1.3-4 | ML service shall perform language auto detection | Azure Video Analyzer |
| 10.1-1.3-5 | ML service shall recognize faces and keywords | Azure Video Analyzer |
| 10.1-1.3-6 | ML service shall provide timestamp for analyzed result | Azure Video Analyzer |

| **1.4:AWS** | | **Met by** |
| --- | --- | --- |
| 10.1-1.2-1 | AWS shall host the Box skill application | AWS Lambda |
| 10.1-1.2-2 | AWS shall capture and parse all information of an event send by Box | AWS Lambda |
| 10.1-1.2-3 | AWS shall send an event to ML service(Azure Video Analyzer) | AWS Lambda |
| 10.1-1.2-4 | AWS shall send access token to ML service | AWS Lambda |
| 10.1-1.2-5 | AWS shall receive response from ML service | AWS Lambda |

# 11. Glossary

* SBPD: Santa Barbara Public Defender Office
* SRS: Software Requirements Specification
* SDD: Software Design Document
* 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

# 12. References

* Extension for Public Defender's Client Case Management System CSULA team 2020-2021 <https://csns.cysun.org/department/cs/project/view?id=7248892>