Software Design Document
for
Box.com/eDefender Integration

Version 2.0 approved

Prepared by:
James Yokley, Joseph Comeaux, Ethan Ngo, Alexander Voisan, Jimmy Castillo, Alexis Ponce, Karen Quan, Rosa Robles, Mario Avila, Sherina Mae Marquez

Santa Barbara Public Defender's Office

9 December 2022

Table of Contents.....<pg 2-3>

Revision History.....<pg 4>

1. Introduction.....<pg 5>
   - 1.1 Purpose.....<pg 5>
   - 1.2 Document Conventions…….....<pg 5>
   - 1.3 Intended Audience and Reading Suggestions.....<pg 5>
   - 1.4 System Overview.....<pg 5>

2. Design Considerations.....<pg 6>
   - 2.1 Assumptions and dependencies.....<pg 6>
   - 2.2 General Constraints.....<pg 6>
   - 2.3 Goals and Guidelines.....<pg 6>
   - 2.4 Development Methods.....<pg 6>

3. Architectural Strategies.....<pg 7>

4. System Architecture.....<pg 8>
   - Level 0 DFD.....<pg 8>
   - Level 1 DFD.....<pg 8>

5. Policies and Tactics.....<pg 9>
   - 5.1 Specific Products Used.....<pg 9>
   - 5.2 Requirements traceability.....<pg 9>
   - 5.3 Testing the software.....<pg 9>
   - 5.3.1 Engineering trade-offs.....<pg 9>
   - 5.3.2 Protocols.....<pg 9>
   - 5.3.3 Maintaining the software.....<pg 10>
   - 5.3.4 Interfaces.....<pg 10>
   - 5.3.5 Hierarchical organization.....<pg 10>
   - 5.3.6 How to build.....<pg 10>
   - 5.3.7 Other tactics.....<pg 11>

6. Detailed System Design.....<pg 12>
   - 6.1 Box.com…….....<pg 12>
   - 6.1.1 Responsibilities.....<pg 12>
   - 6.1.2 Constraints.....<pg 12>
   - 6.1.3 Composition.....<pg 12>
   - 6.1.4 Uses/Interactions.....<pg 12>
   - 6.1.5 Resources.....<pg 12>

   - 6.2 Box Skills.....<pg 12>
   - 6.2.1 Responsibilities.....<pg 12>
   - 6.2.2 Constraints.....<pg 12>
   - 6.2.3 Composition.....<pg 12>

6.2.4 Uses/Interactions.....<pg 13>
6.2.5 Resources.....<pg 13>

6.3 Microsoft Azure Video Indexer.....<pg13>
6.3.1 Responsibilities.....<pg 13>
6.3.2 Constraints.....<pg 13>
6.3.3 Composition.....<pg 13>
6.3.4 Uses/Interactions.....<pg 13>
6.3.5 Resources.....<pg 13>

6.4 AWS.....<pg 14>
6.4.1 Responsibilities.....<pg 14>
6.4.2 Constraints.....<pg 14>
6.4.3 Composition.....<pg 14>
6.4.4 Uses/Interactions.....<pg 14>
6.4.5 Resources.....<pg 14>

7. Detailed Lower level Component Design
7.1 Box.com.....<pg 15>
7.2 AWS.....<pg 16>
7.3 Microsoft Azure…….....<pg 16>

8. Database Design.....<pg 17>

9. User Interface.....<pg 18>
9.1 Overview of User Interface.....<pg 18>
9.2 Screen Frameworks or Images.....<pg 18>
9.3 User Interface Flow Model.....<pg 18>

10. Requirements Validation and Verification.....<pg 19-21>

11. Glossary.....<pg 22>

12. References.....<pg 22>

Revision History

Name	Date	Reason For Changes	Version
Team	11/12/2022	First Draft	0.1
Team	12/09/2022	Second Draft	0.3
Team	3/25/2023	Third Draft	1.0
Team	4/17/2023	Final Draft	2.0

1. Introduction

1.1 Purpose
The goal is to use AWS and Microsoft Cognitive skills with a focus on Video Indexer to speed up certain processes within the Public Defender's Office to improve the amount of work that can be done within work hours.

1.2 Document Conventions
N/A

1.3 Intended Audience and Reading Suggestions
The expected audience of this document is mainly Software Developers or Administrators for accounts needed to perform any future upkeep for the product.

1.4 System Overview
A user will upload a audio or video file into a valid Box folder, where a box skill will trigger and send the data to AWS. An AWS function will then trigger and send the data to Microsoft Video Indexer for Analysis, which is then sent back to AWS, and then back to Box where the data will then show up as a Box Skill tab within the file viewer in Box along with a DOCX transcription of the video. The results of this should be tags that allow a user to see where items appear in the video as well as the separate transcript file.

2. Design Considerations

2.1 Assumptions and Dependencies

- Node JS of version 16+
- Box.com
- A Box.com skill
- Serverless, for deployment
- AWS Lambda function
- Transcription / Analysis Service
    - Microsoft Video Indexer

2.2 General Constraints

- Quality transcription accuracy
- Quality item detection
- Quality face detection
- Securely handling data for processing
- Handling of access tokens and possible expiration of tokens

2.3 Goals and Guidelines

- Deliver and integrate the final product by the end of the Spring 2023 Semester.
- Focus on a simple and easy to understand process.
- Transcribe videos that are a different language (ex: Spanish) and create a bilingual transcript.
- Transcribe videos that deal with bad audio quality and background noises.

2.4 Development Methods

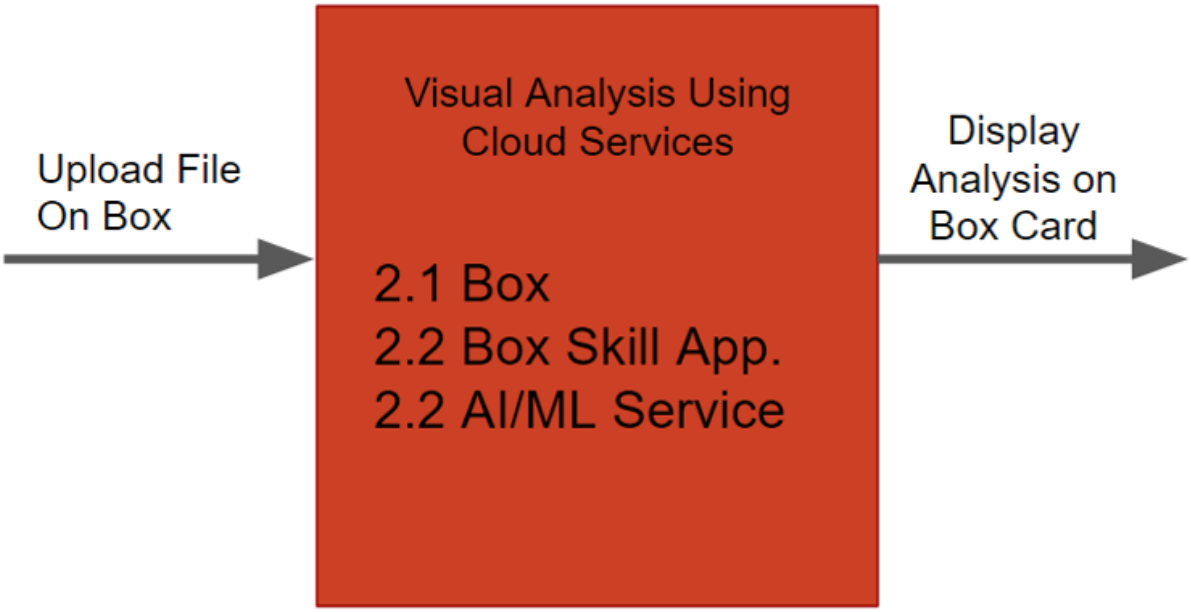
Our project used the Agile Development model as our approach. We met weekly with each other and our project advisor in person. In addition, we met with our liaison biweekly to discuss what should be done. We placed ourselves into three different teams to learn software and combined our knowledge during one meeting. We continuously adapted and implemented new features to discover how everything comes together.

3. Architectural Strategies

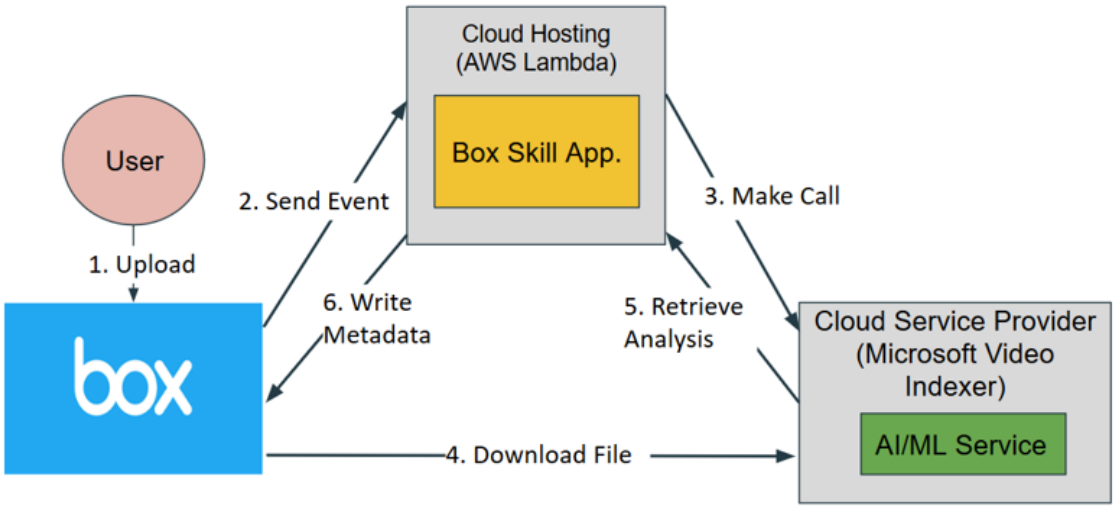
- Programming Languages and file formats
    - JavaScript
    - JSON
    - .DOCX
    - .MOV
    - .MP4
- Existing Software and Services
    - AWS
        - Lambda
        - S3 Buckets
    - Box.com
    - Microsoft Video Indexer
- User Interfaces
    - AWS
    - Box.com
    - Microsoft video Indexer
- Error detection
    - During uploaded and processing of deafened/muted videos the Javascript code which parses transcripts shall input a base/dummy transcript during this type of error
- External data storage management
    - AWS S3 Buckets
    - Box.com folder and file management
    - Video Indexer uploads
- Communication mechanism
    - AWS Lambda function

4. System Architecture

LEVEL 0 DFD



LEVEL 1 DFD



5. Policies and Tactics

5.1 Choice of which specific products used

- AWS Services
    - AWS Lambda
- Microsoft Services
    - Video Indexer
- Serverless
- Box.com
    - Use of Box Skills

5.2 Plans for ensuring requirements traceability

- Requirements and possible features are brought up and discussed among the group as well as SBPD. After development of features the team will check the SRS and confirm that the features match what has been done. Any requirements that we fail to implement from SBPD will be brought to their attention.

5.3 Plans for testing the software

5.3.1 Testing features

- Testing the upload and processing time by uploading various types of videos and media
    - Muted Audio Videos
    - Stable environment videos
    - Videos with movement
- Testing the quality of video transcription manually by cross referencing the AI with heard audio.
- Ability to add videos to Box folders via mobile phone

5.3.2 Protocol of one or more modules

- This software is built upon existing services with different types of architecture, therefore most communication between them will take place in the following ways and languages:
    - JSON
    - JavaScript
    - AWS Lambda API Calls
    - Serverless Endpoints
    - Box.com Skills
- The software requires the user is logged onto a Box.com account with the proper authority to upload files into specific folders. When a video is uploaded a request

is made through HTTPS and other secure channels which require keys, that will transfer the files to Azure and back to Box.com given the correct API calls from the AWS lambda functions.

5.3.3 Plans for maintaining software

- All services used are maintained by box.com, AWS, and Microsoft
    - Unless major updates that overhaul and completely remove previous function calls, existing methods implemented into our project which interact these services should not need direct maintaining
- Software when pushed into use by the team shall use existing calls which should remain the same throughout the integration process

5.3.4 Interfaces for end-users, software, hardware, and communication

- Interfaces used during development:
    - **Node.js**
        - **https://nodejs.org/en/download/**
    - **Serverless**
        - **https://www.npmjs.com/package/serverless**
    - **AWS Lambda**
        - **https://aws.amazon.com/premiumsupport/knowledge-center/create-access-key/**
    - **Box.com**
        - **http://developer.box.com/guides/applications/custom-skills/setup/**
        - **https://developer.box.com/guides/authorization/custom-skill-approval/**
    - **Microsoft Video Indexer**
        - **https://api-portal.videoindexer.ai/**

5.3.5 Hierarchical organization of the source code into its physical components (files and directories)

- Source code located within AWS Lambda function
    - See source code for more details

5.3.6 How to build and/or generate the system's deliverables (how to compile, link, load, etc.)

- See User Guide

5.3.7 Other tactics

- Sticking to communication between Box, AWS Lambda, Indexer/Azure
    - Allows for a minimal number of possible changes that would disrupt service
    - Allows for a clean way of communication between few services
    - Using existing example code and functions/models the companies promote and high use numbers will allow for longevity

6. Detailed System Design

6.1 Box.com

6.1.1 Responsibilities

Box.com shall provide storage for uploaded files, share files with other users, and send event triggers to a box skills application.

6.1.2 Constraints

A Box account is required to use box services. Box.com provides unlimited storage for an unlimited number of files. However, the file size to be uploaded will be limited depending on the account type.

6.1.3 Composition

Box.com is used as a primary place to put our files in. The folder that the file is placed in will then contact the Box Skills that the folder is attached to.

6.1.4 Uses/ Interactions

When a file is uploaded, Box.com shall send an event trigger to a box skills application.

6.1.5 Resources

This module requires stable internet connection and a browser.

6.2 Box Skills

6.2.1 Responsibilities

Box skills shall send file information to Microsoft Azure video indexer. Box skills shall retrieve the analysis from Microsoft Azure video indexer. Then, it shall convert the analysis to box metadata cards and attach them back to the corresponding file.

6.2.2 Constraints

A Box account is required to use Box services. The uploaded file must be a video and the input video must have dialogue.

6.2.3 Composition

Box skills shall hold the application that will send, retrieve, and process data between box.com and Microsoft Azure video indexer.

6.2.4 Uses/ Interactions

Box skills retrieves an event trigger from Box.com and sends the uploaded file's data to the video indexer. Once the video indexer has finished processing the file data, box skills receive the analysis and convert certain metadata into metadata cards. Box skills shall then apply the metadata cards back onto the corresponding file.

6.2.5 Resources

This module requires stable internet connection, aws lambda function, and serverless configuration.

6.3 Microsoft Azure Video Indexer

6.3.1 Responsibilities

Microsoft Azure video indexer receives file data and extracts insights like topics, facial recognition, transcript, and timeline.

6.3.2 Constraints

Microsoft Azure account required to use video indexer. Limited video indexing of 40 hours on a free account, will need paid subscription for more.

6.3.3 Composition

Microsoft Azure video indexer shall hold a copy of the data processed for each uploaded file.

6.3.4 Uses/ Interactions

Microsoft Azure video indexer receives the file data from a box skills application and extracts insights such as keywords, facial recognition, transcript, and timeline. After extraction, it shall send the analysis back to the box skills application.

6.3.5 Resources

This module requires stable internet connection and a browser.

6.4 AWS

6.4.1 Responsibilities

AWS lambda connects the box skills application, that is hosted on the function, to Microsoft Azure video indexer. AWS S3 bucket shall store the uploaded file data.

6.4.2 Constraints

An AWS account is required to use aws services.

6.4.3 Composition

This module shall hold the lambda function component and s3 bucket component.

6.4.4 Uses/ Interactions

AWS hosts the box skills application on a lambda function. Through the invocation url, AWS shall bridge the box skills application and the video indexer.

6.4.5 Resources

This module requires stable internet connection and a browser.

7. Detailed Lower level Component Design

7.1 Box.com

7.1.1 Classification

Box.com classifies itself as a secure cloud content management platform

7.1.2 Interface Description

Developers use a GUI interface to interact with the Box.com platform, in our case we used Box.com due to design requirements which allows us to send and receive data from AWS.

7.1.3 Processing Detail

7.1.3.1 Design Class Hierarchy

N/A

7.1.3.2 Restrictions/Limitations

Limited when receiving data back as there is no way to include a progress bar

7.1.3.3 Performance Issues

When uploading tons of data, data returns may be slow.

7.1.3.4 Design Constraints

N/A

7.1.3.5 Processing Detail For Each Operation

When uploading a video, Box will send that data to AWS and await its appended data back to give to the user. It will check if the data uploaded is allowed to be sent to AWS firstly.

7.2 AWS

7.2.1 Classification

AWS is a data storage and management solutions provider who also contains a suite of other software included in their environment.

7.2.2 Interface Description

Developers use a GUI interface to interact with AWS, in our case we used AWS due to design requirements which allows us to send and receive data from Box.com and append data using AWS Lambda Functions.

7.2.3 Processing Detail

7.2.3.1 Design Class Hierarchy

N/A

7.2.3.2 Restrictions/Limitations

When sending data to Azure, there will be a delay depending on microsoft return times.

7.2.3.3 Performance Issues

When uploading tons of data, data returns may be slow. Appending data may take longer if a video is long as well.

7.2.3.4 Design Constraints

Constrained to what AWS has built in for lambda functions, like Node.JS and other language frameworks.

7.2.3.5 Processing Detail For Each Operation

When uploading a video, AWS will send that data to Azure and await it to return data. It will check if the data has a transcript then create a custom transcript document for it, returning both the document and the data back asynchronously.

7.3 Microsoft Azure

7.2.1 Classification

Microsoft Azure is a video recognition software.

7.2.2 Interface Description

Developers use a GUI interface to interact with uploaded videos and allow them to change ML models.

7.2.3 Processing Detail

7.2.3.1 Design Class Hierarchy

N/A

7.2.3.2 Restrictions/Limitations

When sending data to Azure free tier, there is a limit of around 4k minutes

7.2.3.3 Performance Issues

When uploading tons of data, data returns may be slow.

7.2.3.4 Design Constraints

N/A

7.2.3.5 Processing Detail For Each Operation

When an uploaded video finishes processing, it will be returned to AWS.

8. Database Design

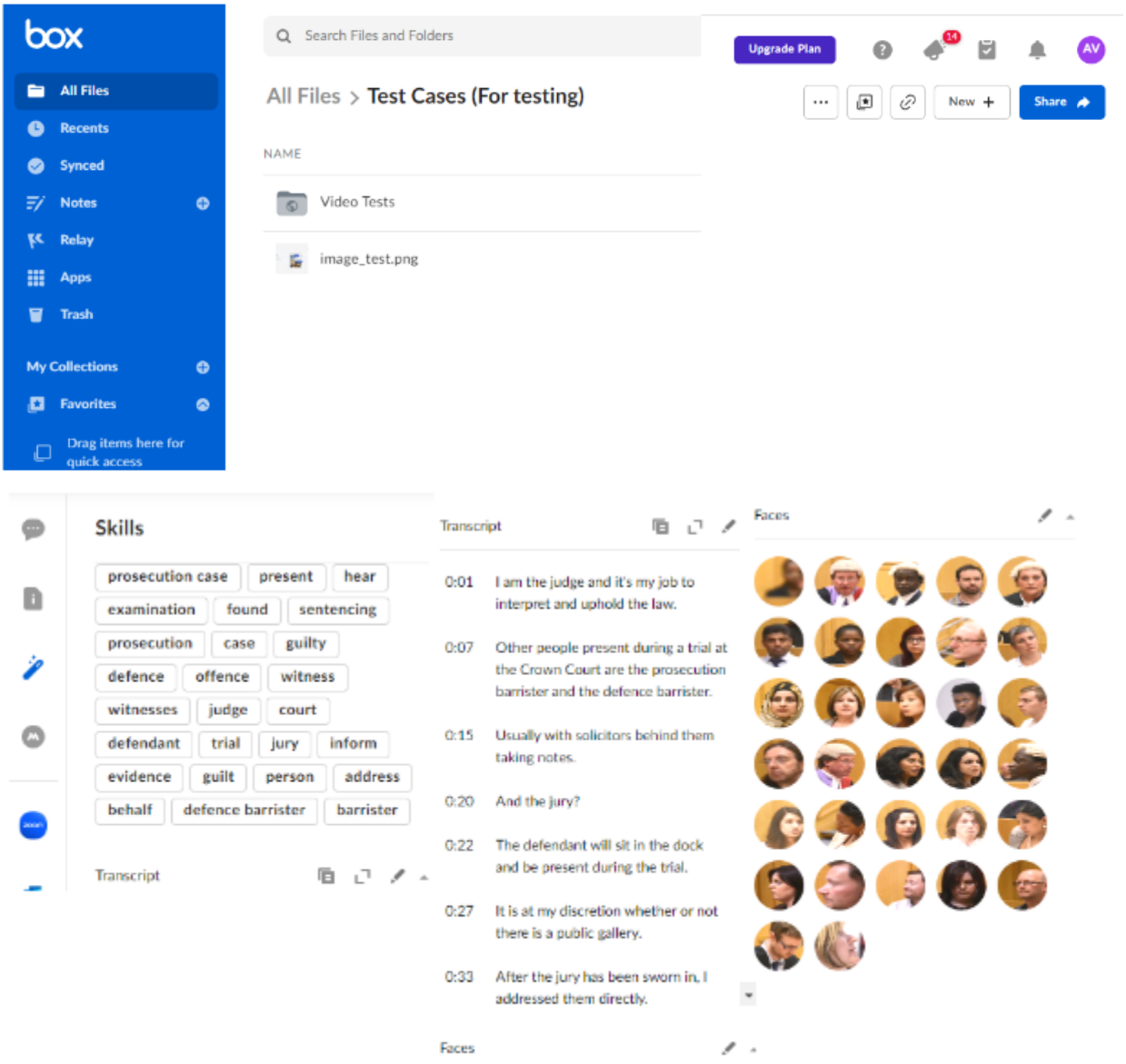
Our project relies heavily on AWS, Box.com and Microsoft Azure as the main places where data is stored and all databases are handled by these services. Box.com is the main cloud storage service used where Microsoft Azure will temporarily hold video files during processing until manual deletion.

9. User Interface

9.1 Overview of User Interface

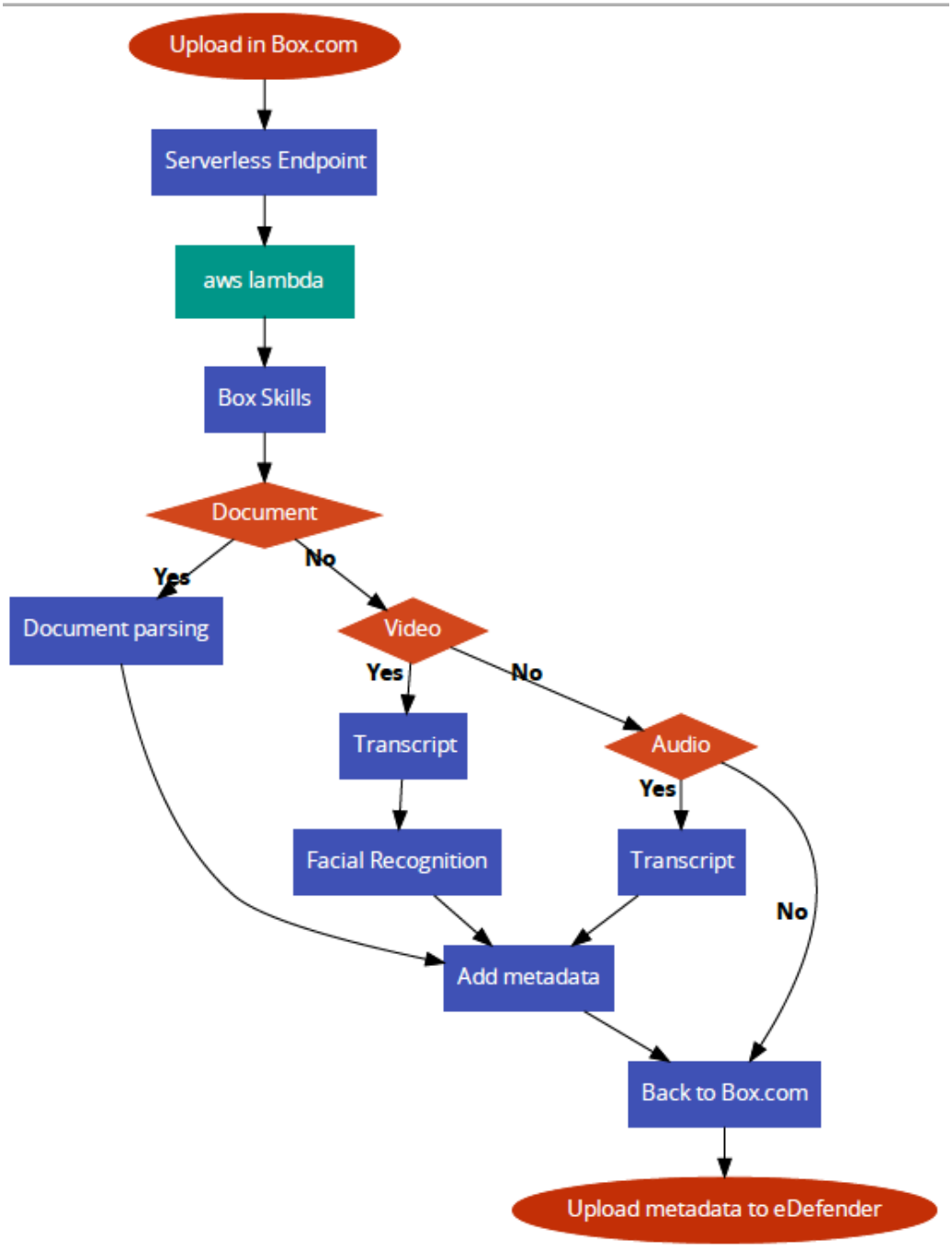
The system will provide an area where the user can upload a file (.txt, .mp4, .mp3, etc.). The file will then process and can take X minutes (depending on file size). After the processing stage is complete, there will be key words, transcription, and facial recognition.

9.2 Screen Frameworks or Images



9.3 User Interface Flow Model

The User Interface Flow Model shows the steps that a file goes through after the user inputs it.



10. Requirements Validation and Verification

	10.1 Box.com	Met by
10.1.1.1	The user shall have valid Box.com credentials with access to SBPD folders	Box.com
10.1.1.2	The user should be able to view a file along with its metadata	Box.com
10.1.1.3	The user shall have permissions to upload files to SBPD Box.com folders	Box.com
10.1.1.4	Must be configured with a Box Skills Application	Box.com

	10.2 Box Skills Application	Met by
10.1.2.1	The user shall have administrator credentials to use Box Skills	Box Skills
10.1.2.2	The user shall have valid folder permissions to apply a Box Skill to a Box.com folder	Box Skills
10.1.2.3	Must create a custom skills application	Box Skills
10.1.2.4	Must be applied manually to each folder needing transcription capabilities	Box Skills
10.1.2.5	Must be configured with a valid AWS Lambda invocation URL	Box Skills
10.1.2.6	Must be configured to trigger event upon file upload	Box Skills

10.1.2.7	Shall provide access tokens to invocation URL	Box Skills

	10.3 Microsoft Azure Video Indexer	Met by
10.1.3.1	The user shall have valid Microsoft Azure credentials	Microsoft Azure Video Indexer
10.1.3.2	Credentials must be configured in AWS Lambda environment variables	Microsoft Azure Video Indexer
10.1.3.3	Must account for uploaded media without audio before indexing	Microsoft Azure Video Indexer
10.1.3.4	Must account for uploaded media without video before indexing	Microsoft Azure Video Indexer
10.1.3.5	Must account for language translation before indexing	Microsoft Azure Video Indexer
10.1.3.6	Must configure Azure skill for video indexing	Microsoft Azure Video Indexer
10.1.3.7	Should periodically clear indexed files from Microsoft Azure	Microsoft Azure Video Indexer

	10.4 Amazon Web Services Lambda	Met by
10.1.4.1	The user shall have valid AWS credentials	Amazon Web Services Lambda
10.1.4.2	Credentials must be configured in Serverless Framework environment	Amazon Web Services Lambda

10.1.4.3	Lambda function must use Microsoft Azure VI credentials to authenticate	Amazon Web Services Lambda
10.1.4.4	Lambda function must utilize Microsoft Azure VI API to access Microsoft machine-learning services	Amazon Web Services Lambda
10.1.4.5	Lambda function must utilize BoxSDK API to access Box.com files and metadata cards	Amazon Web Services Lambda

	10.5 BoxSDK Developer	**Met by**
10.1.5.1	Must use BoxSDK API to manipulate Box.com files	BoxSDK Developer
10.1.5.2	Must use BoxSDK metadata tiles and cards to apply metadata to file	BoxSDK Developer
10.1.5.3	Must have valid Box Skills token to access Box.com files	BoxSDK Developer

11. Glossary

SRS: Software Requirement Specification.

AI/ML: Artificial Intelligence Markup Language.

VAUCS: Visual Analysis Using Cloud Services.

Box (Box.com): A cloud storage.

Box Skills: A framework for bringing Cloud Services to files stored on Box.

AWS: Amazon Web Services.

AWS Lambda: A computing service on AWS.

Azure: Microsoft Azure Cognitive Services

VI: Microsoft Video Indexer

12. References

- Serverless Framework documentation - https://www.serverless.com/framework/docs/providers/aws/cli-reference/deploy
- Box Skills documentation - http://developer.box.com/guides/applications/custom-skills/setup/
- Box.com documentation - http://support.box.com/hc/en-us/articles/360043696394-Create-New-Files-And-Folders
- AWS Lambda documentation - https://docs.aws.amazon.com/lambda/latest/dg/welcome.html
- Microsoft Azure Video Indexer documentation - https://vi.microsoft.com/en-us