

Software Design Document

for

Box.com and eDefender Integration

Version 2.0.0 approved

Prepared by Branden Zamora, Jacqueline Molina, Luiz Perez Campos, Florian Haule, Leo Gallardo, Donovan Hatfield, Coby Alvarez, Jose Alvarado, Dat Nguyen, Brian McNaughton

Santa Barbara Public Defender's Office

December 8th, 2023

Table of Contents

Table of Contents.....	2
Revision History.....	5
1. Introduction.....	6
1.1 Purpose.....	6
1.2 Document Conventions.....	6
1.3 Intended Audience and Reading Suggestions.....	6
1.4 System Overview.....	7
2. Design Considerations.....	8
2.1 Assumptions and Dependencies.....	8
2.2 General Constraints.....	8
2.3 Goals and Guidelines.....	9
2.4 Development Methods.....	10
3. Architectural Strategies.....	11
4. System Architecture.....	12
5. Policies and Tactics.....	13
5.1 Choice of which specific products used.....	13
5.2 Plans for ensuring requirements traceability.....	13
5.3 Plans for testing the software.....	14
5.3.1 Testing Features.....	14
5.3.2 Protocol of one or more modules/subsystems.....	14
5.3.3. Plans for maintaining software.....	15
5.3.4 Interfaces for end-users, software, hardware, and communication.....	15
5.3.5 Hierarchical organization of the source code into its physical components (files and directories).....	15
6. Detailed System Design.....	16
6.1 Box.com.....	16
6.1.1 Responsibilities.....	16
6.1.2 Constraints.....	16
6.1.3 Composition.....	16
6.1.4 Uses/Interactions.....	16
6.1.5 Resources.....	16
6.2 Box Skills.....	16
6.2.1 Responsibilities.....	16
6.2.2 Constraints.....	16
6.2.3 Composition.....	16
6.2.4 Uses/Interactions.....	17

6.2.5 Resources.....	17
6.3 Microsoft Azure Video Indexer.....	17
6.3.1 Responsibilities.....	17
6.3.2 Constraints.....	17
6.3.3 Composition.....	17
6.3.4 Uses/Interactions.....	17
6.3.5 Resources.....	17
6.4 AWS.....	18
6.4.1 Responsibilities.....	18
6.4.2 Constraints.....	18
6.4.3 Composition.....	18
6.4.4 Uses/Interactions.....	18
6.4.5 Resources.....	18
7. Detailed Lower level Component Design.....	19
7.1 Box.....	19
7.1.1 Classification.....	19
7.1.2 Interface Description.....	19
7.1.3 Processing Detail.....	19
7.1.3.1 Restrictions/Limitations.....	19
7.1.3.2 Performance Issues.....	19
7.1.3.3 Processing Detail for Each Operation.....	19
7.2 AWS.....	20
7.2.1 Classification.....	20
7.2.2 Interface Description.....	20
7.2.3 Processing Detail.....	20
7.2.3.1 Restrictions/Limitations.....	20
7.2.3.2 Performance Issues.....	20
7.2.3.3 Processing Detail for Each Operation.....	20
7.3 Video Indexer.....	21
7.3.2 Interface Description.....	21
7.3.3 Processing Detail.....	21
7.3.3.1 Restrictions/Limitations.....	21
7.3.3.2 Performance Issues.....	21
7.3.3.3 Processing Detail for Each Operation.....	21
8. Database Design.....	22
9. User Interface.....	23
9.1 Overview of User Interface.....	23
9.2 Screen Frameworks or Images.....	23

9.3 User Interface Flow Model.....	26
10. Requirements Validation and Verification.....	27
10.1 Box.com.....	27
10.2 BoxSDK (Developer).....	27
10.3 Box Skills (Developer).....	28
10.4 Microsoft Azure Video Indexer.....	29
10.5 Video Indexer Custom Model.....	29
10.6 Amazon Web Services.....	30
10.7 AWS Secrets Manager.....	30
10.8 AWS S3.....	31
10.9 Transcription Function.....	31
11. Glossary.....	32
12. References.....	33

Revision History

Name	Date	Reason For Changes	Version
Initial Setup	12/25/23		1.0.0
Final Draft	12/08/23		2.0.0

1. Introduction

1.1 Purpose

The purpose of this document is meant to outline the technical aspects of the system for developers and administrators. This document goes in depth into each feature implemented and elaborates on the following topics: high level view of the system, design considerations, architectural strategies, system architecture, detailed system design, and graphical user interface (GUI) design.

1.2 Document Conventions

Not Applicable.

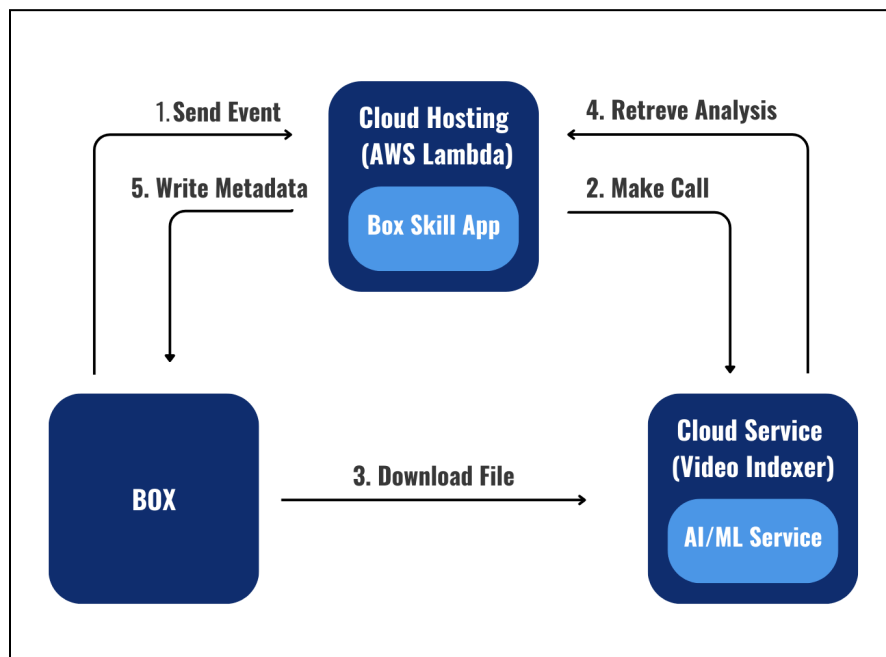
1.3 Intended Audience and Reading Suggestions

The intended audience comprises individuals who will be involved in the ongoing upkeep and technical support of the product. This document specifically targets Developers or Administrators responsible for managing the accounts required for any subsequent maintenance of the product.

1.4 System Overview

The Box.com/eDefender Integration system is a serverless cloud-based program which combines the use of three primary services: Amazon Web Services, Azure Video Indexer, and the Box.com cloud content management system. The goal of the project is to use visual analysis provided by Video Indexer to speed up the Public Defender's ability to review digital media evidence involving audio and/or video on the Box website during the initial discovery phase of a case.

A user is able to initiate the process by uploading an audio or video file into a designated and valid Box folder. A Box skill is then activated, prompting the transfer of data to AWS. Within AWS, a function is triggered to dispatch the data to Microsoft Video Indexer for comprehensive analysis. Once the analysis is complete, the results are sent back to AWS and then redirected to Box. The processed data is presented within the Box interface, appearing as a dedicated Box Skill tab within the file viewer. The outcome of this workflow includes insights that enable users to pinpoint the locations of items (key words, faces, etc.) within the video and/or audio, accompanied by a transcript for comprehensive understanding.



2. Design Considerations

2.1 Assumptions and Dependencies

- Node JS (18.x.x)
- AWS (v3)
- Box
- Box Skill
- Severless
- AWS Lambda
- AWS S3
- AWS Secrets Manager
- AWS API Gateway
- Microsoft Azure Video Indexer

2.2 General Constraints

- Transcription accuracy may differ based on initial media quality.
- Item detection accuracy may differ based on initial media quality.
- Face detection accuracy may differ based on initial media quality.
- Securely handling data for processing is required
- Handling of access tokens and possible expiration of tokens
- Secure Internet connection is required for best results
- Viewing access is limited to only Desktop/Laptop computers

2.3 Goals and Guidelines

2.3.1 AWS Team

- 2.3.1.1 Integrated AWS Secrets Manager to secure credentials for our Lambda Function
- 2.3.1.2 Limited the domains that can access our Lambda Function with API Gateway
- 2.3.1.3 Setup expiration policies for S3 Buckets

2.3.2 Codebase Team

- 2.3.2.1 Manual Removal of S3 Buckets after media has returned from processing.
- 2.3.2.2 Accurate Error Handling and Logging
- 2.3.2.3 Up to date dependency versions (zero vulnerabilities in the code)
- 2.3.2.4 Modularizing; prioritizing efficiency and stability
- 2.3.2.5 Verifying incoming requests

2.3.3 Transcript Team

- 2.3.3.1 Finish confidence marking
- 2.3.3.2 Fix Line numbering with JSON files
- 2.3.3.3 Use more test files to parse documents

2.3.4 Custom Models Team

- 2.3.4.1 Setup the environment needed to train the model
- 2.3.4.2 Created metrics to measure accuracy of the Custom Language Model indexing
- 2.3.4.3 Gathered the necessary data to train the model
- 2.3.4.4 Initial training of language model

2.4 Development Methods

In the execution of our project, we adopted the Agile Development model as our methodology. Our team maintained a weekly schedule for in-person meetings with both our project advisor and each other to ensure regular communication and collaboration. Furthermore, biweekly meetings with our liaison were conducted to deliberate on project milestones and tasks.

To facilitate efficient learning, we organized ourselves into four distinct teams, each dedicated to mastering a different software aspect. The culmination of our efforts were demonstrated during each collective meeting, where we were allowed to divulge our acquired knowledge for the week. Throughout the development process, we remained flexible, constantly adapting, communicating, and implementing new features to gain a comprehensive understanding of how all components harmonize within the project.

3. Architectural Strategies

Programming Languages

- JavaScript

File Formats

- JSON
- .DOCX
- .MOV
- .MP4
- .MP3

Existing Software and Services

- Amazon Web Services (v3)
 - Lambda
 - S3
 - API Gateway
 - Secrets Manager
- Box
- Microsoft Video Indexer
- Serverless
- NodeJS (18.x.x)
- BoxSDK (v3)

User Interfaces

- AWS
- Box
- Microsoft Video Indexer

Error Detection

While uploading and processing videos with muted or unclear audio, the JavaScript code responsible for parsing transcripts is designed to insert a placeholder or dummy transcript.

External Data Storage Management

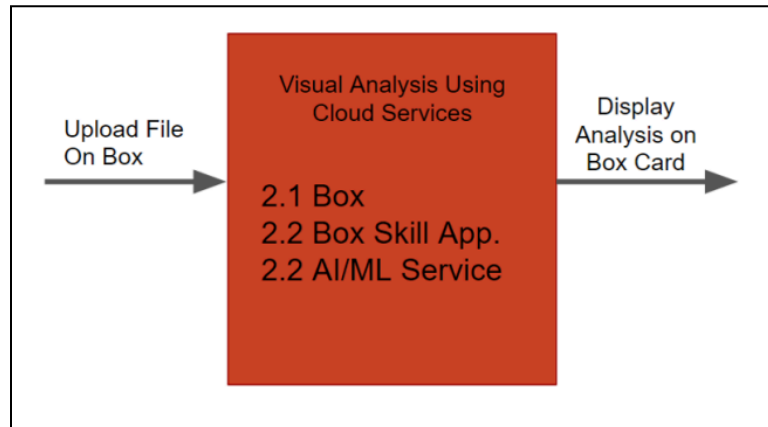
- AWS S3 Buckets
- Box folder and file management
- Video Indexer

Communication Mechanism

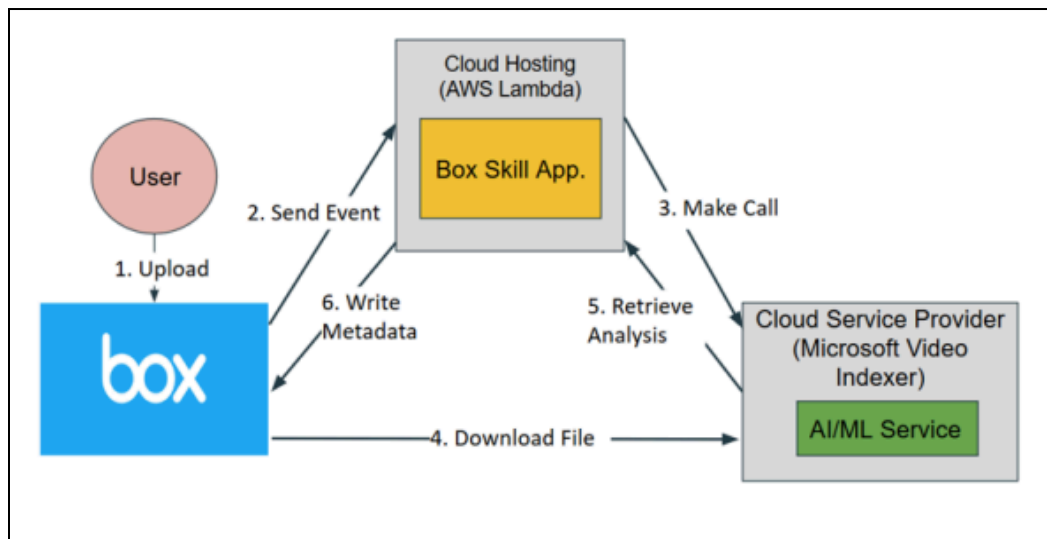
- AWS Lambda

4. System Architecture

Level 0:



Level 1:



5. Policies and Tactics

5.1 Choice of which specific products used

- AWS Services
 - Lambda, S3, Secrets Manager, API Gateway
- Microsoft Services
 - Video Indexer, Custom Model Training
- Box
 - Box Skills
- Severless
- JavaScript (NodeJS)
- Visual Studio Code

5.2 Plans for ensuring requirements traceability

The team engages in consistent collaborative discussions within the group and with the SBPD (presumably Santa Barbara Police Department) to identify and deliberate on requirements and potential features. Following the development phase, the team meticulously reviews the Software Requirements Specification (SRS) to ensure alignment between the implemented features and the documented specifications.

Any instances where requirements from the SBPD are not successfully incorporated into the project are promptly communicated to them for attention and resolution. This process aims to maintain transparency, accuracy, and adherence to the mutually agreed-upon specifications.

5.3 Plans for testing the software

5.3.1 Testing Features

- Evaluating the upload and processing efficiency involves conducting tests with diverse video and media files to assess the time required for both uploading and processing.
 - Muted Audio Files
 - Process data within our unique dataset
 - Videos w/ movement
- Conducting a manual evaluation of video transcription quality involves cross-referencing the output generated by the artificial intelligence with the actual audio content to ensure accuracy and reliability.

5.3.2 Protocol of one or more modules/subsystems

- The foundation of this software is constructed upon pre-existing services, each employing distinct architectural frameworks. As a result, the majority of communication between these services is anticipated to occur through specific modes and programming languages, which are outlined as follows:
 - JSON
 - Javascript
 - AWS Lambda API Calls
 - Severless Endpoints
 - Box Skills
 - Box Security Keys
 - Access Keys
- To utilize the software, it is imperative that the user is logged into a Box.com account with the appropriate authorization to upload files into designated folders.
- Upon uploading a video, a request is initiated through HTTPS and other secure communication channels, necessitating the use of keys. These keys facilitate the seamless transfer of files from Box.com to Azure and back, contingent upon the accurate invocation of API calls within the AWS lambda functions.

5.3.3. Plans for maintaining software

- All services used are maintained by Box, AWS, and Microsoft
 - Unless substantial updates occur that fundamentally revamp and entirely eliminate prior function calls, the current methods incorporated into our project, which interface with these services, should remain unaffected and not require direct maintenance.
- Upon deployment and utilization by the team, the software will employ existing function calls that are expected to remain consistent throughout the entire integration process.

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

- Node.js
 - <https://nodejs.org/en/download>
- Serverless
 - <https://www.npmjs.com/package/serverless>
- AWS
 - <https://docs.aws.amazon.com/AWSJavaScriptSDK/v3/latest/>
- Box.com
 - <https://developer.box.com/guides/applications/custom-skills/>
 - <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 is located within the AWS Lambda Function.

5.3.6 How to build and/or generate the system's deliverables

See Developer and User Guide(s).

6. Detailed System Design

6.1 Box.com

6.1.1 Responsibilities

Box.com will serve as the storage solution for uploaded files, facilitate file sharing among users, and dispatch event triggers to a Box Skills application.

6.1.2 Constraints

To access Box services, it is required to have a Box account. Box offers boundless storage for an unlimited quantity of files. Nevertheless, the permissible file size for uploads is contingent upon the specific account type.

6.1.3 Composition

Box will function as the designated storage platform for uploaded files, enable seamless file-sharing capabilities among users, and initiate the transmission of event triggers to a Box Skills application.

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 a stable internet connection and browser.

6.2 Box Skills

6.2.1 Responsibilities

Box Skills is designed to transmit file information to Microsoft Azure Video Indexer and Amazon Web Services. Following this, Box Skills will transform the processed metadata into Box cards, which will be added to the corresponding file for information integration.

6.2.2 Constraints

To access Box services, it is required to have a Box developer/admin account.

6.2.3 Composition

Box Skills will serve as the application responsible for facilitating the exchange, retrieval, and processing of data between Box.com and Microsoft Azure Video Indexer and Amazon Web Services.

6.2.4 Uses/Interactions

Box Skills initiates the process by fetching an event trigger from Box.com and transmitting the uploaded file's data to the video indexer hosted on Microsoft Azure. Following the completion of the file data processing by the video indexer, Box Skills receives the analysis and proceeds to transform specific metadata into metadata cards. Box Skills applies these metadata cards back to the corresponding file, completing the integration of analyzed information.

6.2.5 Resources

This module necessitates a reliable internet connection, the implementation of AWS Lambda functions, and the configuration of a serverless setup.

6.3 Microsoft Azure Video Indexer

6.3.1 Responsibilities

The Microsoft Azure Video Indexer acquires file data and extracts valuable insights such as topics, facial recognition details, transcripts, and timeline information.

6.3.2 Constraints

To utilize Video Indexer on Microsoft Azure, it is required to have a Microsoft Azure account. A free, trial account provides limited video indexing capabilities, allowing up to 40 hours. For extended usage beyond this limit, a paid subscription is required.

6.3.3 Composition

Microsoft Azure Video Indexer is designed to retain a duplicate copy of the processed data for every uploaded file.

6.3.4 Uses/Interactions

The Microsoft Azure Video Indexer receives file data from a Box Skills application through AWS Lambda functions and performs extraction of insights including keywords, facial recognition details, transcripts, and timeline information. Following the extraction process, the Video Indexer then sends the analysis back through Lambda function to the Box Skills application for further processing.

6.3.5 Resources

This module requires a stable internet connection and browser.

6.4 AWS

6.4.1 Responsibilities

AWS Lambda facilitates the connection between the Box Skills application and Microsoft Azure Video Indexer. The uploaded file data is stored in an AWS S3 bucket as part of the processing workflow. The AWS team also is responsible for increasing security within an application.

6.4.2 Constraints

An AWS account is required to use AWS services. Permission or key is also needed to run aws secrets manager.

6.4.3 Composition

This module will encompass both the Lambda function component and the S3 bucket component. It will also include API Gateway and Secrets Manager to help the project become more secure.

6.4.4 Uses/Interactions

AWS hosts the Box Skills application on a Lambda function. By utilizing the invocation URL, AWS establishes a connection between the Box Skills application and the Video Indexer, enabling seamless communication between the two.

6.4.5 Resources

This module requires a stable internet connection and a browser.

7. Detailed Lower level Component Design

7.1 Box

7.1.1 Classification

A secure cloud content management system.

7.1.2 Interface Description

Box is used in place of an interface for standard users. Here they are able to send and receive data (in the form of media).

7.1.3 Processing Detail

7.1.3.1 Restrictions/Limitations

There is no conceivable way to relay the status (percentage) of the processed video other than a 'processing' or 'error' card.

7.1.3.2 Performance Issues

When uploading multiple or large quantities of video, a substantial delay is expected.

7.1.3.3 Processing Detail for Each Operation

Immediately after an upload has occurred to a Box folder, the AWS invocation function will send out the data to Video Indexer. Which then processes the media and returns metadata insights in the form of cards to Box.

7.2 AWS

7.2.1 Classification

A data storage and management provider who additionally provides a suite of other services and software for developers to use.

7.2.2 Interface Description

Developers are able to use the AWS website to manage their AWS services. Developers are also able to track the status of their processes.

7.2.3 Processing Detail

7.2.3.1 Restrictions/Limitations

Communication with Box and Video Indexer may result in slower processing times. Cost for AWS service usage is to be kept in mind.

7.2.3.2 Performance Issues

Delays will inevitably occur due to cross system communication. Size of uploaded media may alter processing times.

7.2.3.3 Processing Detail for Each Operation

AWS will act as a middle-man for Box and Video Indexer. The uploaded data is uploaded and saved onto an AWS service and then sent out for processing to Video Indexer. The program receives the newly modified media, handles some additional processing, and then returns the new metadata insights to Box.

7.3 Video Indexer

7.3.1 Classification

Program classifies itself as a video recognition software that utilizes Artificial Intelligence and Machine Learning to process various types of media.

7.3.2 Interface Description

Developers will utilize Video Indexer to observe the progress of uploaded videos sent by AWS from Box. Developers will additionally create, manage, and train custom language and speech models.

7.3.3 Processing Detail

7.3.3.1 Restrictions/Limitations

When using a trial account, a limit is imposed of ~40 hours of media.

7.3.3.2 Performance Issues

When working with larger or multiple videos, performance is expected to be slower and/or delayed.

7.3.3.3 Processing Detail for Each Operation

Video Indexer obtains a piece of media from AWS, processes it, and returns metadata to AWS in a JSON format. This newly returned data includes transcription, face recognition, and keyword detection.

8. Database Design

Not Applicable.

All data will be managed by the Box Cloud Storage System.

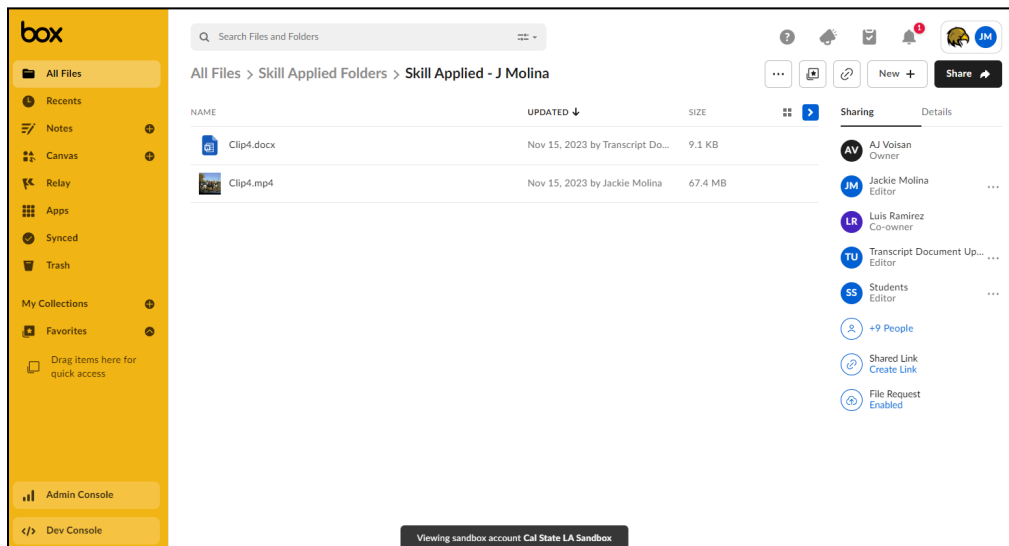
9. User Interface

9.1 Overview of User Interface

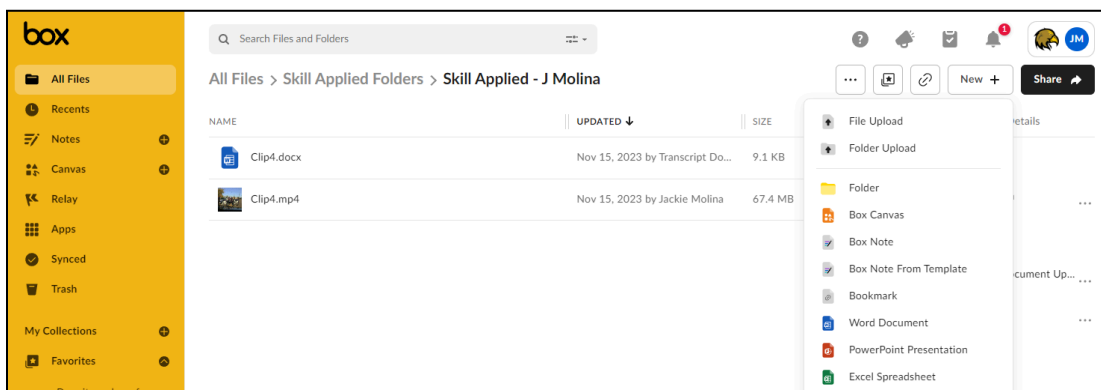
A standard user will only be interacting through the Box.com provided interface. The video indexing is performed on cloud services that do not require the input of a standard user. To upload media, select a skill enabled folder, press the new button on the top right of the screen. Once upload is complete, wait for the processing to finish.

9.2 Screen Frameworks or Images

Box.com Folder Interface



Box.com Upload Media Interface



Box.com Processing Card

Status ▲

We're working on processing your file - please hold!

Box.com Error Card


Error ▲

Something went wrong with running this skill or fetching its data.

Box.com Skill Insights

v1 Clip4.mp4
Skill Applied - J Molina · Updated Nov 15, 2023 by Jackie Molina

1 ... Open Download Share X



Skills

Topics

blood sample blood test alcohol driving fucking required submit vehicle court legal truck bullshitting fuck side

Transcript

0:00 I was working, driving.
0:02 I'm not going to fucking.
0:03 OK, just commit to something that I did not.
0:06 I guess you're not going to cooperate, right?
0:08 OK, You are required by state law to submit to a chemical test to determine the alcohol and or drug content.
0:13 Yeah, but I wasn't driving that.
0:14 I believe you're under the influence of alcohol or or combination of

Viewing sandbox account Cal State LA Sandbox

Box.com Transcript

v1Clip4.docxSkill Applied - J Molina - Updated Nov 15, 2023 by Transcript Document Uploader

OpenDownloadShare

1

Clip4.docx

1 Speaker 1 : I was working, driving.

2 Speaker 1 : I'm not going to fucking.

3 Speaker 1 : OK, just commit to something that I did not.

4 Speaker 2 : I guess you're not going to cooperate, right?

5 Speaker 2 : OK, You are required by state law to submit to a chemical test to determine the alcohol and or drug content.

6 Speaker 1 : Yeah, but I wasn't driving that.

7 Speaker 1 : I believe you're under the influence of alcohol or or combination of alcohol drugs.

8 Speaker 1 : I wouldn't in that.

9 Speaker 1 : Vehicle or blood test I wouldn't in that vehicle if.

10 Speaker 2 : You are.

11 Speaker 2 : On DUI probation, you're required to submit to a preliminary alcohol screening test.

12 Speaker 2 : OK, either way, we're going to take your picture.

13 Speaker 2 : So just.

14 Speaker 2 : Yeah.

15 Speaker 1 : Yeah, right.

16 Speaker 1 : For 100.

17 Speaker 1 : How how are you going to

1/2103%

Viewing sandbox account Cal State LA Sandbox

Activity

Add Task

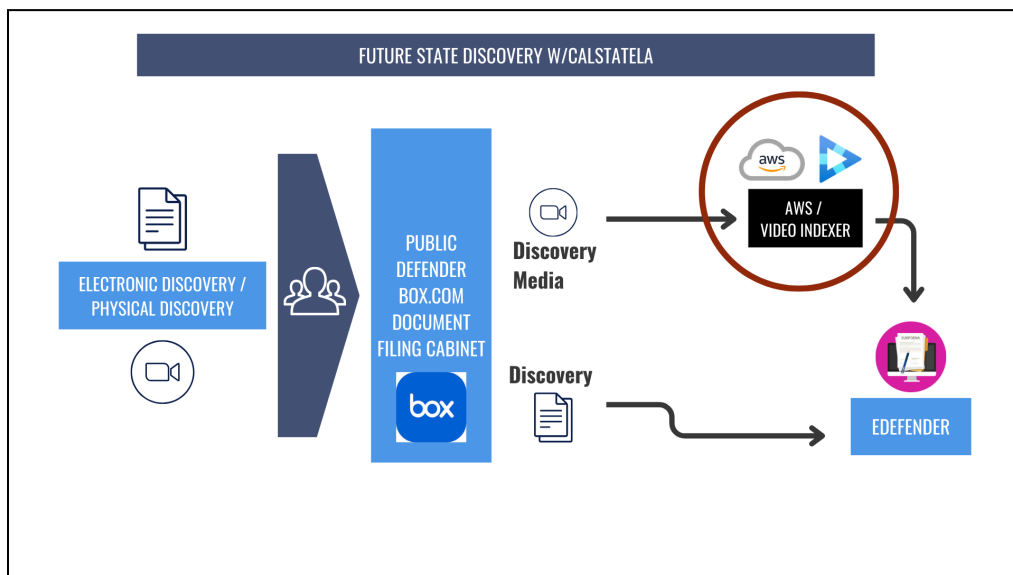
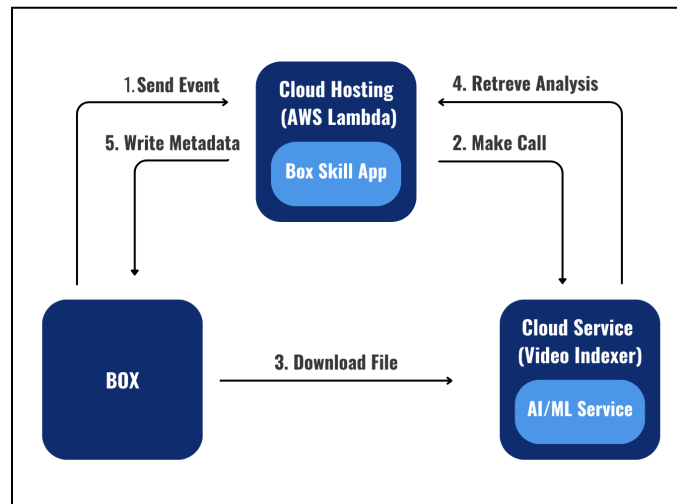
No activity to show

Hover over the preview and use the controls at the bottom to annotate the file.

JMWrite a comment

@mention users to notify them.

9.3 User Interface Flow Model



10. Requirements Validation and Verification

	10.1 Box.com	Met by
10.1.1	The user shall have valid Box credentials with access to SBPD folders	Box.com
10.1.2	The user shall have permission to view a file along with its returned insights	Box.com
10.1.3	The user shall have permissions to upload files to SBPD Box folders	Box.com
10.1.4	The user shall have permission to remove files from SBPD Box folders	Box.com
10.1.5	The user's given folder must be configured with a Box Skills Application	Box.com

	10.2 BoxSDK (Developer)	Met by
10.2.1	The user must use BoxSDK to manipulate Box files	BoxSDK
10.2.2	The user must use BoxSDK metadata cards to return to the standard user	BoxSDK
10.2.3	The user must have valid access tokens and security keys	BoxSDK

	10.3 Box Skills (Developer)	Met by
10.3.1	The user shall have administrator credentials to configure Box Skills	Box Skills
10.3.2	The user shall have developer credentials to create Box Skills	Box Skills
10.3.3	The user must have created a Box Skills Application to run the program	Box Skills
10.3.4	The user administrator must apply the created skill to selected folders	Box Skills
10.3.5	The user must configure the Box Skills to a valid AWS invocation url	Box Skills
10.3.6	The user must configure the folder to trigger an event upon file upload	Box Skills
10.3.7	The user must provide access tokens to the invocation URL	Box Skills
10.3.8	The user must provide security keys to the lambda function	Box Skills

	10.4 Microsoft Azure Video Indexer	Met by
10.4.1	The user shall have valid Microsoft Azure credentials	Video Indexer
10.4.2	The user shall configure the credentials in the AWS environment	Video Indexer
10.4.3	Video Indexer shall process uploaded media	Video Indexer
10.4.4	Video Indexer shall return media metadata to AWS	Video Indexer
10.4.5	Video Indexer shall communicate with AWS	Video Indexer

	10.5 Video Indexer Custom Model	Met by
10.5.1	The user shall have the necessary privileges and roles for the Custom Model	Video Indexer
10.5.2	The Custom Model shall provide accurate subtitle and closed caption	Video Indexer
10.5.3	The Custom Model should integrate with Azure Video Indexer API for customizing the language model	Video Indexer
10.5.4	The Custom Model should support custom vocabulary and phrases needed by SBPD	Video Indexer
10.5.5	The languages in the video content shall be supported	Video Indexer

	10.6 Amazon Web Services	Met by
10.5.1	The user shall have valid AWS credentials	AWS
10.5.2	The user credentials must be configured within the Serverless Framework	AWS
10.5.3	The Lambda function must use Video Indexer credentials to authenticate	AWS
10.5.4	The Lambda function must user Box credentials to authenticate	AWS
10.5.5	The Lambda function must communicate with VI, Box, and S3	AWS

	10.7 AWS Secrets Manager	Met by
10.7.1	Secrets Manager shall securely store Box.com configurations such as API keys	AWS
10.7.2	Secrets Manager shall have control policies to manage access to secrets	AWS
10.7.3	Secrets Manager will limit access to information to users with permission	AWS
10.7.4	Secrets Manager will deny access to users without permission or credentials	AWS
10.7.5	Secrets Manager will secure the information within it through API Gateway and Secrets Manager	AWS

	10.8 AWS S3	Met by
10.8.1	S3 must store data incoming from Box	AWS
10.8.2	S3 must store data incoming from Video Indexer	AWS
10.8.3	S3 must remove buckets once program has completed it's cycle	AWS
10.8.4	S3 must automatically remove remaining buckets after a 30 day period	AWS

	10.9 Transcription Function	Met by
10.9.1	The user shall have a valid JSON File from Microsoft AI Video Indexer	Codebase
10.9.2	The program is required to format the JSON File	Codebase
10.9.3	The program is required to highlight the confidence marking	Codebase
10.9.4	The program is required to produce a transcribe document	Codebase
10.9.5	The produced document is required to fill in all 28 lines	Codebase

11. Glossary

Term	Definition, Acronym, Meaning
SRS	Software Requirements Specification
AI/ML	Artificial Intelligence/ Machine Learning
Box.com/Box	A secure cloud storage system
Box Skills	Framework for bring cloud services to files on Box
AWS	Amazon Web Services
VI/ Azure/ Video Indexer	Microsoft Azure Video Indexer

12. References

Serverless Framework Documentation

<https://www.serverless.com/framework/docs/providers/aws/cli-reference/deploy>

Box Skill 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 v3 Documentation

<https://docs.aws.amazon.com/s3/>

Video Indexer Documentation

<https://learn.microsoft.com/en-us/azure/azure-video-indexer/>

Ascent Project Homepage

<https://ascent.cysun.org/project/project/view/204>

[2022-2023] Github

https://github.com/James-Yokley/box.com_eDefender_integration

[2023-2024] Github

https://github.com/branden12/Box.com_eDefenderIntegration_23-24