Senior Design Final Report

Box.com/eDefender Integration

Version 1.0.0 - 05/03/2014

CSULA Team
Branden Zamora
Jacqueline Molina
Luis Perez Campos
Florian Haule
Leo Gallardo
Donovan Hatfield
Jose Alvarado
Coby Alvarez
Dat Nguyen
Brian McNaughton

Advisor
Dr. Jung Soo Lim

Liaisons
Deepak Dudwani
Luis Ramirez
AJ Voisan
Angie Stokke
Sarah Rothschild

Table of Contents

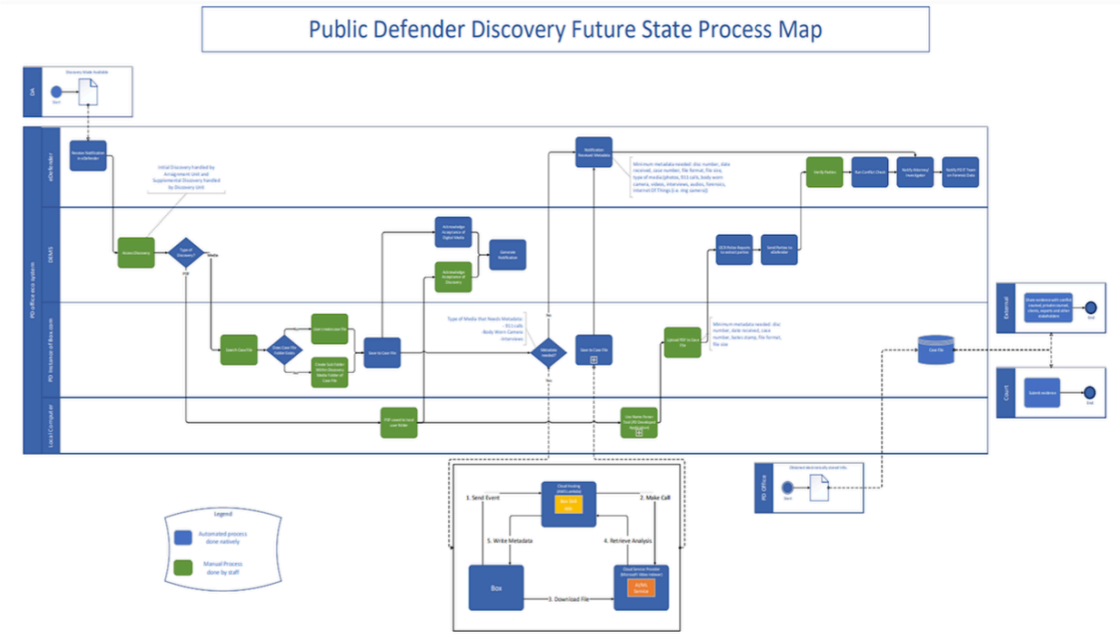
Introduction

1.1 Background

The aim of the project is to use visual analysis provided by Video Indexer to improve the Snata Barbara Public Defender's ability to review digital media evidence involving audio and/or video on their cloud storage website (Box.com) during the initial discovery phase of a case. As of this moment, the amount of data incoming each month is growing exponentially, and if a solution isn't brought forth soon, the sheer amount of discovery media will overwhelm the faculty over at SBPD.

In order to achieve this goal, the team must work with various programs and APIs; including but not limited to: Video Indexer, Box Skills, Azure Custom Models, AWS Lambda, AWS S3, and additional AWS services.

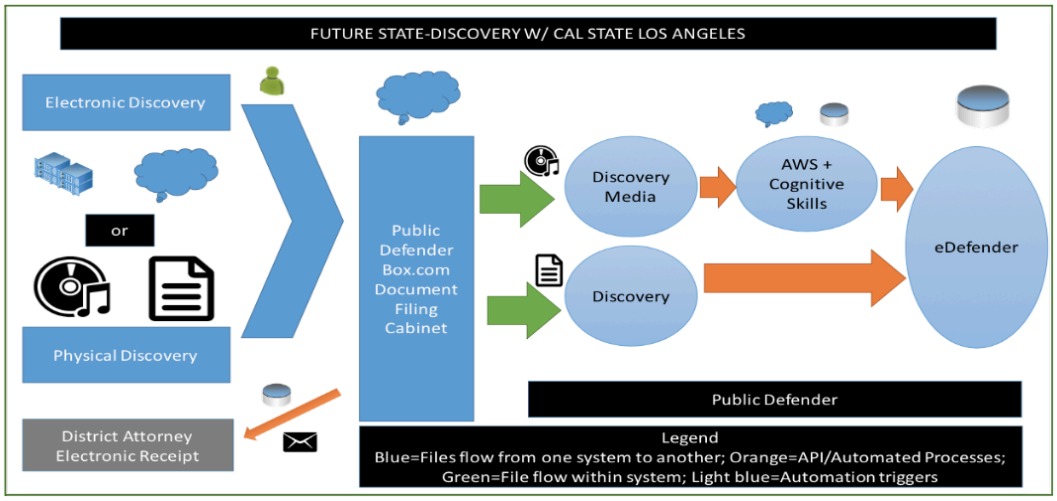
Overall, the team promises to deliver a user-friendly and easy to navigate platform to the Santa Barbara Public Defender's Office.



1.2 Design Objectives

Continuing the work from the previous year; the main objective of the program is to allow all discovery media uploaded onto Box.com to enter a fully automated process where each piece of media is processed and returned with insights (faces, transcripts, keywords) along with a transcribed document.

As soon as a piece of media is uploaded onto the cloud storage website, the file contacts a serverless endpoint link which in turn invokes an AWS lambda function which sends out each video to be processed by Video Indexer. As soon as the video has finished processing, the insights and data are returned to be turned into skill cards that will be displayed to the user. In addition, upon the return on a processed video a .docx document is returned with a generated transcript that adheres to courtroom standards.



1.3 Design Benefits

The program we plan to develiver has no outright effect on a typical ‘everyday’ user. Since all of the code runs in the background, the user will not need to know any specifics to how the program is implemented or how the program runs. In turn, making the design extremely simple to use, with no additional steps required. These ‘everyday’ users will be able to upload discovery media onto Box.com like usual, and all that is required of them is to wait for their videos to be processed.

The code, additionally, aids public defenders in reviewing media in a more efficient manner. The user may choose to focus on a singular individual in a video (returned skill cards allow users to ‘select’ individuals to listen to), or they may choose to find keywords (returned skill cards allow users to jump ahead in a video to where a specific word is mentioned). The transcript returned in addition to these skill cards allows users to search for relevant information without watching the video or listening to the audio. To close, the returned transcript is in line with courtroom standards and is available to be used during a case.

1.4 Achievements

Throughout the course of the year, our team has been able to complete all given tasks and add on to the project that was handed to us. To ensure maximum efficiency, we organized ourselves into four subgroups; each dedicated to a specific coding/software aspect (AWS Team, CodeBase Team, Transcript Team, Custom Models Team). We adopted the Agile Development model as our methodology by maintaining weekly meetings with both our project liaisons and our project advisor in order to ensure consistent communication and updates.

As the year progressed, we were able to remain flexible and constantly adapt to any challenge we encountered. Below are the features/tasks we were able to accomplish this semester:

AWS Team

- Integrated AWS Secrets Manager to secure credentials for our Lambda Function
- Limited the domains that can access our Lambda Function with API Gateway
- Set up expiration policies for S3 Buckets (remove a bucket in 30 days)

Codebase Team

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

Transcript Team

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

Custom Models Team

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

2. Related Programs

2.1 Existing Solutions

This product is similar to an already existing product employed by the Los Angeles Public Defender's Office which is used to automatically transcribe and tag discovery media uploaded to Box.com. While SBPD is looking for a product like the one mentioned, they would like to create a separate product in order to curate the software to the specific needs of their team. In our instance, they would like the digital transcription of discovery media (.docx file), along with returned insights from Video Indexer (key words, face recognition, digital transcript).

Additionally, last year, the sponsors made the team aware that there exists no solutions catered for government sectors (legal implementations for automating the generation of case metadata and insights).

2.2 Reused Solutions

The program has been developed with Javascript Node.js, Serverless, Box Skills SDK, AWS (S3, Lambda, CloudFormation, CloudWatch), and Microsoft Video Indexer.

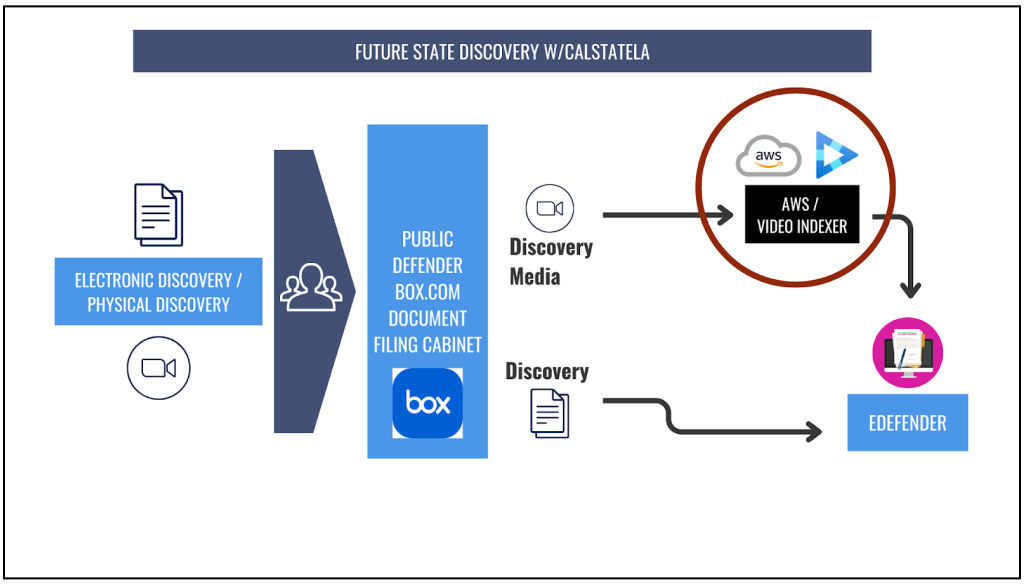
The Box SDK allows us to trigger an event through an upload on Box.com folders, as well as the ability to upload files and box skill cards (insights) back to Box after processing. The Video Indexer API allows us to create processing requests, upload files, and process returned insights/metadata. AWS services and Serverless allow us to create an automated program which is triggered through set endpoints.

3. System Architecture

3.1 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 overarching 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.

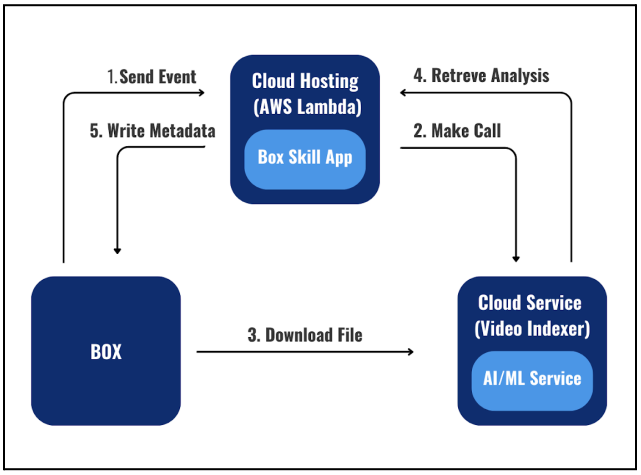
DFD Level 0



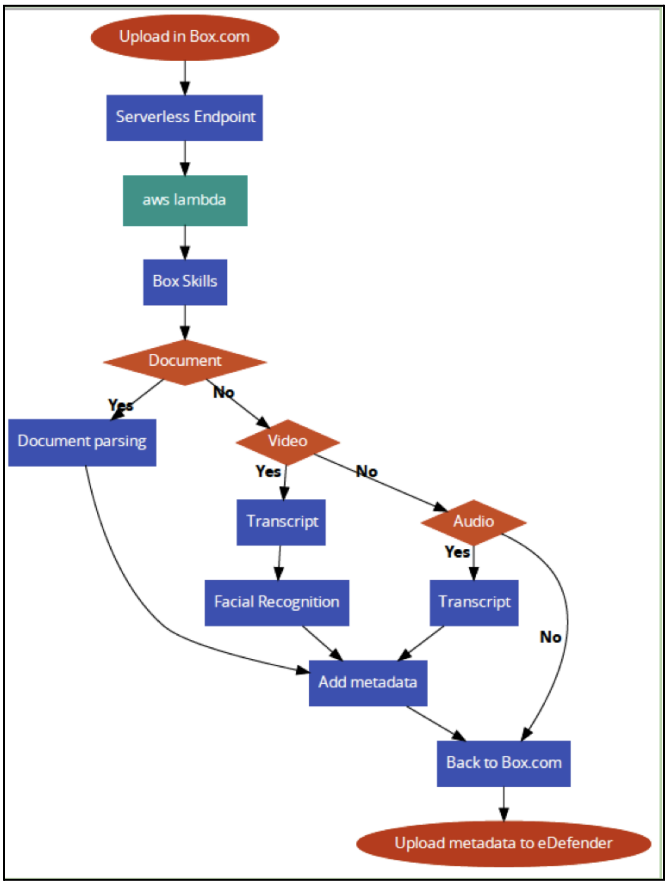
A user initiates the process by uploading an audio or video file into a designated and valid Box folder which has an activated Box Skill. Upon an upload, the AWS Lambda function is triggered. Within AWS, the data is sent to Microsoft Video Indexer for analysis. Once complete, the insights/metadata are 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 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.

3.2 Data Flow

DFD Level 1



DFD Level 2



The data flow in our project is as shown in the figure above, starting with a user uploaded video to an enabled Box.com folder (enabled with a Box Skill). Once the video or document is fully uploaded to Box.com, the site will check its extension type to confirm if the call needs to be triggered and then triggers an event to AWS along with the data of the video.

The data has now been passed on to AWS and is sent to Video Indexer to be analyzed by their AI/ML algorithms. When this is done, the insights/metadata is sent back to AWS and stored in an AWS S3 bucket. AWS will use the transcription data from Video Indexer in a new event call to create a transcription document in the proper format (according to courtroom standards). As this happens, the metadata is also sent back to Box.com to be added to the video through Box Skill Cards (which will be displayed directly to the user through the Box interface).

Data on AWS is deleted after the calls are over, with an additional provision to erase any remaining S3 buckets after 30 days.

3.3 Implementation

The system can be split into three distinct components, each of which are closely interconnected: (Box.com, AWS, and Video Indexer). Each component plays a key role in the handling and processing of discovery media.

Box.com

Box.com serves as the front-facing user interface that ‘everyday’ users will use and interact with. It additionally serves as the cloud based storage solution for uploading files, facilitating file sharing among users, and dispatching event triggers to a Box Skills application. Box Skills is designed to transmit file information to Video Indexer and AWS (Amazon Web Services). Once media has been processed and returned, Box Skills will transform the processed metadata into Box cards. Box Skills then applies these metadata (insight) cards back to the corresponding file, completing the integration of analyzed information.

AWS Services

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. 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 technologies.

Video Indexer

The Microsoft Azure Video Indexer acquires file data and extracts valuable insights (key words, facial recognition, transcripts, and timeline information). Video Indexer receives file data from a Box Skills application through AWS Lambda functions and performs its AI/ML analysis for obtaining insights. Following the extraction process, Video Indexer sends the analysis back through the AWS Lambda function to eventually return Box Skill Cards to the front-facing user.

4. Conclusions

4.1 Struggles

AWS Team

Our team faced several issues to fix the code and reach our goal. We faced issues with the Secrets Manager as the current version of SDK was outdated and needed to be changed for it to run. We made changes to the dependencies such as deleting sdk and adding the necessary properties needed. We also faced issues with the S3 bucket creating two buckets rather than one and not assigning the rule to the correct bucket. To fix this we had to fix the code and rather than assign the rules to a new bucket assign it to the original bucket.

CodeBase Team

Throughout the year, the most prevalent struggles we had to face were completely understanding the code we were given and understanding how to best improve upon it. It took a while before we were able to fully comprehend what each statement of the code contributed to the project on a larger scale, but once we were able to do so, we quickly adapted and were able to add in our own adjustments without fail.

Transcript Team

During the process of fixing and formatting the code to create the transcript documents, we needed to start from the base on how the text was being extracted from the documents. At first, we managed to fix the line numbering since it was one of the main dilemmas before we started working on the code.

Custom Models Team

4.2 Results

AWS Team

Our goals for this year's AWS team was to increase security and optimize our storage system. The most important task assigned to us was to secure a specific file (config.json). This file contains the API keys necessary to access our Box.com content. When thinking about what resources to use in order to keep these credentials secured, we decided to use Amazon Secrets.

Amazon allows you to store important information such as API keys into a system that is referred to as Secrets. These secrets are encrypted by amazon so that only those with permission can access the content. Because we were already using AWS Lambda to connect Box.com with Video Indexer, we can natively call on the secret that contains the API keys to get access to the APIs. This makes it easy to increase security and maintain efficiency.

Another security measure we took was restricting the domains where our lambda function could be called from. Using API Gateway is part of the serverless framework; we gave it instructions to only allow incoming requests originating from Box.com and Video Indexer domains. Doing this ensures that those with malicious intents do not call our Lambda function.

Our other task was to optimize our storage system. Another amazon service that we use is S3 Buckets. The buckets hold information from the videos uploaded onto Box.com and information from video indexer. As a result, the buckets holding information from multiple videos would increase the cost amount through time. However, through utilizing the S3 lifecycle rule in serverless it will automatically delete a bucket after 30 days. This would allow for lower cost through time and optimizing storage.

CodeBase Team

Our goals for this year consisted of 5 specific tasks to complete in relation to the project. These are modularizing the code, implementing accurate and thorough error handling and logging, updating all dependency versions and mitigating those which resulted in conflicts, verifying incoming event request and making sure they came from Box, and finally manually removing our s3 bucket at the end of each event call to assure that we had no unnecessary memory taking up space in our virtual environment.

We were able to remove all dependency version conflicts through updating our sdks and manually going through the imports to fix any remaining errors. We were also able to migrate our code from AWSv2 implementation to AWSv3 through modularization of the dependency injections. The verification of the incoming initial request was able to be handled through the Box SDK which provided a verification function as well as security keys we were able to add to our environment variables. The S3 bucket manual deletion was a simple statement we had to add to the end of our code provided to us by the AWS documentation.

The error handling and logging was likely the task that took up the most time. We were able to create a system that efficiently handled the most obvious errors. If at any point the system crashed and erred out, then two things always occurred. The S3 bucket in AWS was deleted and an error message was sent to the user. Near the end of the program, if the transcript function

failed to return the docx transcript, then the video insights returned regardless rather than nothing being returned.

Transcript Team

The tasks we were assigned: reworking the parsing format, fixing document line numbering, and implementing confidence marking were all fixed and implemented.

For our first task, we reworked the parsing format by mapping each transcript entry to a TextRun for the Word document. This ensures that the lines are printed out in a more organized manner. Each line will be broken down by speaker and punctuation so the same speaker will occupy as many lines needed so that in turn the line problem can be easily fixed.

Next, the line numbering was easily fixed by adding more exact margins to the page as well as printing out empty text runs, empty lines, so that each page ends at exactly 28 lines. Implementing it dynamically presented problems, so it was done through a series of if statements which were more accurate.

Finally, the confidence marking works by summing up all the confidence levels and getting the average, then from there we get the lowest confidence marking from the JSON file and the calculated average to then get the new average, from there this will be roughly be 25% getting highlighted. We made it this way since confidence measures drastically change per document.

Custom Models Team

4.3 Future

Given the information we now know, there are areas in which the system can improve and/or add on to:

- Send custom notifications/messages to the Box UI interface so the end user is able to see the progress in video processing and indexing
- Migrate code to be in compliance with updated Video Indexer API
    - Any new accounts created will be in compliant with the API following the retirement of Azure Media Service
- Potentially creating a subscription based account on Video Indexer
    - (NOTE) Subscriptions based accounts may need a different type of access token compared to the trial accounts. Might get an ARM api error.
- Modularization of code, perhaps create two files (one for each event call)
- Implementation of working custom language model

5. References

Node.js
https://nodejs.org/en/download

Severless
https://www.npmjs.com/package/serverless

AWS
https://docs.aws.amazon.com/AWSJavaScriptSDK/v3/latest/

Microsoft Video Indexer
https://api-portal.videoindexer.ai/

Serverless Framework Documentation
https://www.serverless.com/framework/docs/providers/aws/cli-reference/deploy

Box.com Documentation
https://developer.box.com/guides/applications/custom-skills/
https://developer.box.com/guides/authorization/custom-skill-approval/

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