Alumni Video Donations
Software Requirements Specification

Simone Davison, Andrew Lanum, Austin Morin

November 22, 2021

Version: 1.0

Ver.	Date	Who	Change

1 Introduction

This document serves to show a more detail-oriented description of how the Alumni-Money project will work. It will show off features, as well as go into how the program is intended to be run.

1.1 Purpose

The purpose of this document is to show how the Alumni-Money project runs on a more low-level basis. It's not quite as in-depth as pseudocode, but not as abstract as a vision and scope document.

1.2 Document Conventions

Throughout this document, a 'snippet' refers to a smaller video that has been created out of a larger video. Throughout this document, a 'stitched video' refers to a video that is composed of snippets played in succession. Throughout this document, hashing and encrypting may be used interchangably, but the plan is to use hashing rather than full-scale encryption.

1.3 Intended Audience and Reading Suggestions

The intended audience of this document is anybody who plans to use the Alumbi-Money project regularly, or needs help understanding how the project works. This will likely be developers, quality control, or possibly a customer.

When reading this document, it is suggested that the reader has or quickly develops a basic understanding of the basic principles of computer science and software creation.

1.4 Project Scope

The goal of this project is to provide an easy-to-use system that allows the WWU CS department to mass-email personalized videos to alumni of the CS deparment soliciting donations. The videos will be created out of snippets of alumni, students, and staff saying specific words and phrases, and possibly giving some anecdotal accounts of their time at WWU.

1.5 References

Documentation for python modules used in this project: moviepy: https://pypi.org/project/moviepy/ pydub: https://pydub.com/

2 Overall Description

2.1 Product Perspective

The Alumni Donation Video Stitcher is projected to increase donations to the computer science department of Western Washington University. Currently, donations are gathered by sending out emails to alumni with a link to the donation site. Our project is going to personalize that process by adding a link to a unique video for each alumnus. A week or so before donation emails are to be sent out, our client will pass into our video stitching program a CSV file with each alumnus's information, name and graduation year, and the program will append to that file the links to URLs for each alumnus. The stitching will take at most 5 days to process, at which point the office who is responsible for sending out donation emails will gain access to the CSV file and add the video URL to the email.

2.2 Product Features

There are two major components that we are designing: Splitting videos and Stitching videos.

Splitting of videos: Will take a directory of unedited videos and will split each one on the spaces between words. The split up snippets will be titled by what word is being said within, and if the program understands it to be one of our accepted words, it will be passed into our edited snippets directory to be accessed by the video stitcher. If it is not deemed to be one of our accepted words, it will be passed into a pending directory to be viewed and either accepted and renamed, or is deleted due to unacceptable content.
Stitching of videos: Will parse through the CSV file of graduate students' names and graduation years, and for each student randomly select one of our prepared sentences and find random snippets from our directory of snippets with precedence to snippets taken in the same year as the student graduated. It will then stitch all the snippets together to create a approximately 30-second long video, create a URL to house it, and hash it. It will then stick that URL in the output file along with the alumni's name and graduation year. The output file will then be stored in a directory where the office has access.

2.3 User Classes and Characteristics

- Maintainers: are those in charge of maintaining and upgrading the code as need be.
- Video collectors: are the people who collect recordings of students and faculty, who then pass it into the video splitter to turn sentences into snippets.
- Video stitchers: the person who provides the input CSV file and prompts the program to stitch the videos together.
- Video senders: the office who will take the output CSV file and access the video URLs to send to students.

2.4 Operating Environment

This program is written in python 3.9 with the latest versions of moviepy, 1.0.3, and pydub, 0.25.1. It is designed to be run on the university CS lab network to deposit and collect videos from in one of our client's Jagodzinski's, directories on the network. It is then to be retrieved by the office to email out links.

2.5 Design and Implementation Constraints

1. We may only use snippets of people who have submitted a release form and had it accepted.

2. We or anyone assigned by the client is to record all videos and snippets to ensure quality and security.

3. To avoid mispronunciations, each student is to record their own name. Meaning, if two people share the same spelling for their names, the snippets will not be used interchangeably. If we do not have a recording of them themselves saying their name, we will instead insert a screen with their name plastered across it for a few seconds.

4. Since we cannot tie a video snippet of someone's name to a unique artifact like their W# or their email, we have to rely on their name and graduation year to name the snippet, which may result in students with the same name and graduation year to receive the wrong video.

5. The user, the person requesting videos to be stitched, will need a machine that runs python 3+ as well as has the latest version of moviepy and pydub

2.6 Assumptions and Dependencies

1. The user, the person requesting videos to be stitched, is doing so on a machine that has access to the video snippets and the directory to where to deposit the output file.

2. The user is able to run the program.

3. The office has access to the directory containing the output file.

4. The alumni are able to open links and view a mp4 of 720 resolution.

3 Features

The system shall take long, pre-recorded videos of alumni saying specific words and phrases, and cut it into smaller snippets of video containing just the words and phrases. The system shall be able to combine snippets into longer videos, and have the longer videos be personalized for a specific alumni based on given paramaters (graduation year for example) The system shall store newly created videos in a secure location that can still be accessed online, and then hash the url of that location to be sent to an alumni.

3.1 Name of feature

Video Splitting

3.1.1 Description

The program takes a video of someone saying some words or phrases as input, then splits it into smaller videos where the silence between words/phrases is removed.

3.1.2 Priority

This feature is important because it means videos can be recorded in bulk without the need to manually splice them up.

3.1.3 Stimulus and Response

A user will run the splitting script on the specified file(s), and give an output name/location.

3.1.4 Functional Requirements

The user inputs a video or series of videos, waits until the program is finished running, then recieves many smaller videos that partially make up the initial input.

3.2 Name of feature

Video Stitching

3.2.1 Description

The program takes a series of videos and stitches them together to make a personalized video message.

3.2.2 Priority

This feature is essential because without it, there would be no videos to email to alumni.

3.2.3 Stimulus and Response

A user will run the stitching script with the required paramaters (or a csv file containing all the required information).

3.2.4 Functional Requirements

The user inputs the required paramaters or properly-formatted csv file, waits, then recieves their requested personalized video(s).

3.3 Name of feature

URL Hashing

3.3.1 Description

The program takes a URL of a specified video and hashes part of it, ensuring that the link still goes to the original video.

3.3.2 Priority

This feature is somewhat important, because it wouldn't be good if alumni were able to easily access each others' videos.

3.3.3 Stimulus and Response

Upon a stitched video being stored to its secure location, the program will then run the hashing script on that location's URL.

3.3.4 Functional Requirements

The user runs the stitching script, and recieves the hashed URL of the stitched video.

4 External Interface Requirements

4.1 User Interfaces

1) Users will be required to operate a video playback system of their choice that supports .mp4 files, and know how to access their URL video link using an email inbox of their choice (Outlook, Windows Mail, etc.)
2) QR codes in the video will be Version 3 QR Codes.
3) Linux command line on a Computer Science Lab Computer is responsible for the executing the video stitching program.
4) Cue cards will be drawn to adhere to accommodate visually-impaired users.
5) Videos will be recorded in 720p resolution at the most, and should be scalable to fit the entirety of the user's screen.
6) Release waivers will have specific instructions for signing before recordings occur.
7) Videos will contain minimal non-verbal messages to accommodate visually-impaired users.
8) Program input CSV file contains first name, last name, and graduation year of each alumnus.
9) Program output CSV file contains first name, last name, graduation year, and the hashed URL to their personalized video.

4.2 Hardware Interfaces

1) Users will be able to access the URL video links regardless of what PC they are on (assuming that they are using a modern one that can support video playback).
2) The stitching program will be run on a Computer Science Department computer that supports execution of Linux commands.
3) Videos will be recorded using a Zoomvid-6 WWU/ATUS rental camera.
4) The development team will need a USB cable to upload videos from the Zoomvid-6 camera.

4.3 Software Interfaces

1) Users will be able to access the URL video links regardless of the operating system that they are on (assuming that they are using a modern one that can support video playback).
2) The stitching program will be run on an operating system that supports the Linux command line.
3) The development team will edit code using any IDE that supports Python 3.0 or later.
4) Git will be used to adding, pushing, editing, deleting, and merging files.
5) Gitlab will be used to store the development documents, source code, and video tests.
6) Alumni messages and video URLs are accessible via Microsoft Outlook, Apple Mail, Gmail, Microsoft Mail, and any modern inbox software.
7) Microsoft Excel will be used to store the information from the CSV files.
8) The Python library moviepy version 1.0.3 is used for video editing.
9) The Python library pydub version 0.25.1 is used for audio editing.
10) All user info will be stored remotely both on the CS Dept. Linux Computer and on any cloud storage platform of the development team's choice on their personal devices.
11) The version of Python that the stitching program is written in is 3.9.

4.4 Communications Interfaces

1) Users will be able to access their personalized video on any connection type via WiFi, USB, Ethernet, etc.
2) Users will be able to access their personalized video only under their WWU Computer Science alias (i.e. you must be a graduate or faculty member of the Computer Science Department to receive the video.
3) In order to access Gitlab or Computer Science Lab machines remotely, the development team must use download and use WWU's Cisco AnyConnect Secure Mobility Client VPN.
4) Release waivers that alumni and faculty must sign before being recorded.
5) Users will have access to the URL video link in a message sent via email.
6) Dr. Filip Jagodzinski will have access to all alumni email information and CSV files that he will use to communicate with the faculty responsible for sending out the messages. Means of communication include email and remote transfer of the CSV files between Jagodzinski and the responsible faculty.
7) The final version of the program will be executed on one Linux machine, so there will

be no need for other computers to communicate with each other while running the program concurrently.
8) All video URLs are hashed to ensure security.
9) The cost of data transfer will be kept to a minimum, working with non-HD videos to decrease storage space needed to transfer video files from one location to another.
10) The amount of text in the alumni's emails will comprise of a short greeting, a URL to the final stitched video, and a link to the Computer Science Department's official donation page.

5 Other Nonfunctional Requirements

5.1 Performance Requirements

The process of stitching the videos together using Python's moviepy and pydub libraries shall stitch 10,000 30-second videos in 3 days, where 3,300 are stitched per day. If we choose later to implement threading in our program we shall stitch 10,000 30-second videos in 1.5-2 days, where 4,500 are stitched per day. A single recording session shall take 30 seconds per student, and the transferring of each video file from the Zoomvid-6 camera via USB 3.0 connection to either the Computer Science Lab computer or a developer's personal machine shall take a maximum of 1 hour. Sending out all donation emails and video URL links to alumni shall be near instantaneous, give or take 5 minutes. Users shall be able to access the videos and Computer Science Department Donation Page near instantaneously given that they have a suitable WiFi connection.

5.2 Safety Requirements

The use of recording equipment with flashing lights will be avoided to protect epileptic users.

5.3 Security Requirements

1) Confidentiality: The development team, Computer Science club members, teacher assistants, and alumni does NOT have access to the contents of the input CSV files that contain alumni's first names, last names, years of graduation, nor the output CSV files that contains the first name, last name, graduation year, and the hashed URL to their personalized video. Dr. Filip Jagodzinski will have maintainer access to the CSV files, and the development team will be responsible for creating a video stitching program that he may use to run the CSV files through. The development team has authorized access to all video content and URL links.

2) Integrity: The most important factor contributing towards the integrity of the project is the hashing of the URL video links. This is to prevent possible URL modification with harmful intent such as attackers re-directing alumni to non-trustworthy websites. The implementation of threading in the stitching program can result in race conditions where audio and video data is lost or overwritten. To avoid these situations, breaking critical sections up

where threads are performed is recommended, and ordering threads to execute once their predecessor leaves its critical section. The development team is responsible for controlling all of the video content, including the pop-up URLs and contact information in the videos, to maintain content consistency and integrity. Obsolete and unused CSV and video files will be deleted from Jagodzinski's storage system taking up unnecessary storage space once per two academic years. Instructions in two README files shall be written for the stitching program, one will be used to minimize CSV file processing errors, and the other will be used to document how to collect the video clips. Regular expressions are used to detect errors in CSV file input and output.

3) Availability: The program code shall be engineered to restart upon a system crash, power outage, etc., delete a single that was in the process of stitching, and start building the processed video again from scratch. CSV/video files in use during the academic year shall be backed-up on a cloud storage system in case the Computer Science Lab Computer file directories are restructured.

5.4 Software Quality Attributes

The final stitched videos will be 30 seconds long. The pydub library should take at most 0.5 seconds to extract all audio spaces from a 30 video recording. Stitching together one video after adding transitions and editing the audio shall take 25 seconds without threading, and 10-15 seconds with threading. Reading an input CSV file containing 30 rows shall take less than 1 second, and reading an input CSV file containing thousands of rows shall take at most 30 seconds. For the output CSV files, it shall take at most 2 seconds to generate and hash 30 video URLs for an input CSV file with 30 rows, and take at most one second to write the hashed video URLs to a CSV file with 30 rows. Sending a single test email with one video URL link and one CS Donation Page link shall take less than one second on a satisfactory WiFi connection, and recieving a single test email shall take approximately 5-10 seconds on a satisfactory WiFi connection.

A Glossary

1) CSV - A text file that uses a comma to separate values. Each line of the file is a data record. Each record consists of one or more fields, separated by commas. Typically represents data to be inserted into a database. Microsoft Excel is an example of software that accepts CSV files. (Source - https://en.wikipedia.org/wiki/Comma-separated$_values$)

2) moviepy - A Python programming language library for video editing that supports cutting, concatenations, title insertions, video compositing, video processing, and creation of custom effects. (Source - https://pypi.org/project/moviepy/)

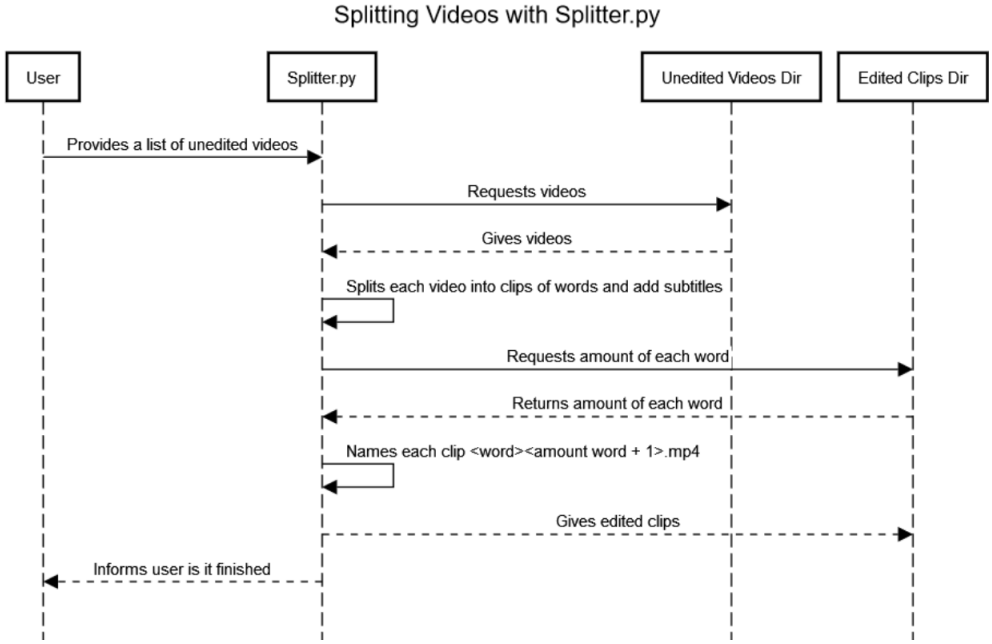
3) pydub - A Python programming language library that allows users to edit audio files.

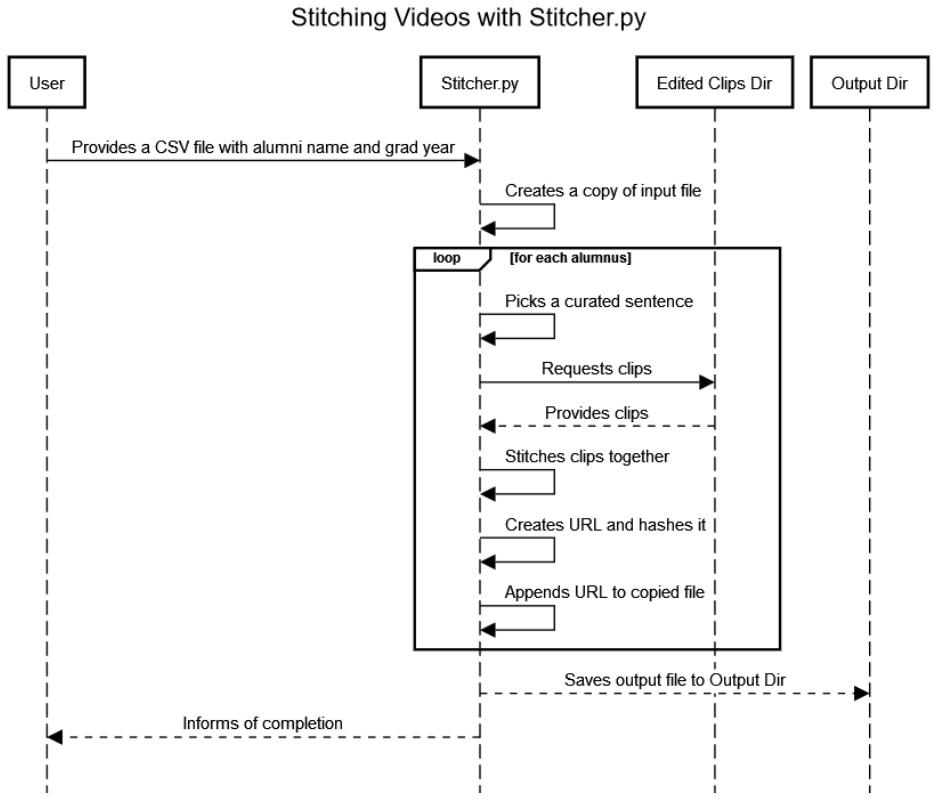
4) QR-Code - A type of barcode that can be read easily by a digital device and which stores information as a series of pixels in a square-shaped grid. QR codes can include website URLs, phone numbers, store up to 4,000 characters of text, link directly to download an app on the Apple App Store or Google Play, authenticate online accounts and verify login details, access Wi-Fi by storing encryption details such as SSID, password, and encryption type, send and receive payment information, and more. (Source - https://usa.kaspersky.com/resource-center/definitions/what-is-a-qr-code-how-to-scan)

5) VPN - A virtual private network (VPN) gives you online privacy and anonymity by creating a private network from a public internet connection. VPNs mask your internet protocol (IP) address so your online actions are virtually untraceable. Most important, VPN services establish secure and encrypted connections to provide greater privacy than even a secured Wi-Fi hotspot. (Source - https://us.norton.com/internetsecurity-privacy-what-is-a-vpn.html)

6) Race condition - A race condition is an undesirable situation that occurs when a device or system attempts to perform two or more operations at the same time, but because of the nature of the device or system, the operations must be done in the proper sequence to be done correctly.

B Analysis Models





C Issue List

1. When splitting our recorded videos into snippets, it relies on audio detection to make sure a student or staff member is saying the anticipated word. It is known that AI

has a diversity problem and is not tested on many non-English accents. Therefore, we are anticipating having many videos sent to pending to be viewed manually due to the program being unable to detect a word based on an accent.

2. There is going to have to be someone to look through the pending videos and name them manually. We anticipate our client as the department chair will not want to do this himself. Finding someone with the time and patience to go through them will be something we will have to address.

3. The possibility of race conditions will need to be handled appropriately to avoid the loss of audio and/or video data. This could occur if two near identical audio, video files, or input CSV files are attempted to be processed at the EXACT same time.