Software Requirements Specification for Video Merger App

Prepared by: [Bacsain,Jave A.

Parro, Carl Gerald J.

Prestado, Marc Justin N.]

Date: December 9, 2025

Version: 1.0

1. Introduction

The Video Merger App is a desktop tool that will be created to make video editing and uploading as simple and efficient as possible. It will help users combine multiple video clips into a single file and upload the finished video directly to YouTube, no need for complicated editing software or manual uploads. The app will be especially useful for streamers, educators, and content creators who want to save time by automating repetitive steps like merging, naming, and uploading clips. This document outlines what the system will do, how it will work, and what requirements must be met to build it successfully.

1.1 Purpose

This Software Requirements Specification (SRS) describes the goals and requirements for the Video Merger App, version 1.0. The purpose of this document is to guide the development team in creating a simple and reliable video merging and uploading application. It also serves as a reference for all stakeholders from developers to testers to ensure everyone has a clear understanding of what the system is expected to achieve.

1.2 Scope

The Video Merger App is a desktop application designed to provide a streamlined, controlled workflow for assembling, processing, and publishing video content. The system expands beyond a simple video-merging tool to include identity management, administrative oversight, and cloud-synchronized user data

Core Video Pipeline Features

- Clip Selection & Arrangement: Users can select and order multiple video clips to create a customized sequence
- Video Merging: The system merges the arranged clips into a single output file in a user-chosen format and resolution
- Direct YouTube Publishing: Users can upload the merged video directly to their YouTube account. The system supports adding titles, descriptions, tags, and other basic metadata during upload

User Management

- The application maintains **secure user identities**, including authentication and role assignments.
- Users are no longer limited to YouTube login credentials; they also authenticate into the application itself, with permissions controlled by assigned roles (e.g., Creator, Editor, Admin).

Administration Capabilities

- The system includes an Administration Dashboard enabling designated Admin users to:
  - Manage user accounts
  - Assign and modify roles
  - Disable users (ban them from logging in)
  - View activity and usage metrics
  - Monitor system-wide behavior and pipeline performance

Cloud Synchronization

- User profiles, roles, and usage statistics are synchronized via Firebase Firestore, ensuring consistent state across devices and persistent storage of key operational data.
- Cloud-stored configuration allows administrators and users to retain settings securely even when switching devices or reinstalling the application.

Target Users

- General Content Creators: Can quickly combine video segments (gameplay, lectures, vlogs, or other media) without needing full-scale video editing software.
- YouTube Uploaders: Benefit from the integrated YouTube publishing pipeline, eliminating the need to switch between editing tools and browser upload interfaces.
- Administrators: Gain centralized controls for managing user access, security, and usage oversight.

Out of Scope

- The system does not include advanced video-editing capabilities such as trimming, color correction, effects, transitions, multi-track timelines, or audio engineering tools.

1.3 Definitions, Acronyms, and Abbreviations

- SRS: Software Requirements Specification
- UI: User Interface
- API: Application Programming Interface
- YouTube API: Google's interface for uploading and managing videos programmatically
- FFmpeg: A multimedia framework used for processing and merging video files

1.4 References

- ISO/IEC/IEEE 29148:2018 — Systems and software engineering — Life cycle processes — Requirements engineering
- YouTube Data API v3 Documentation (Google Developers)
- Flet Documentation

- FFmpeg Official Documentation

1.5 Overview

This document is divided into three sections:

- Section 1: Introduction to the app and purpose of the document
- Section 2: Overall description including product perspective, user types, and constraints
- Section 3: Specific functional and non-functional requirements

2. Overall Description

This section describes the general factors that affect the product and its requirements

2.1 Product Perspective

The Video Merger App is a standalone desktop application that will be built using Python and FFmpeg for video processing. It will also integrate with the YouTube Data API to authenticate users and upload their videos directly from the app.

The app will feature a graphical user interface (GUI) that will allow drag-and-drop file selection, merging order adjustment, progress tracking, and upload management.

The system follows a Layered Architecture. It integrates Firebase (Firestore & Auth) for user management and real-time state synchronization, alongside the YouTube Data API. It utilizes a local SQLite/Pickle storage mechanism for secure token persistence.

2.2 Product Functions

Key features will include:

- Video Selection and Arrangement: Users can import multiple clips and reorder them as needed.
- Video Merging: The app merges all selected clips into a single output file using FFmpeg.
- YouTube Uploading: Users can log in to their YouTube account and upload the final video directly.
- Metadata Editing: Users can input video titles, descriptions, tags, and privacy settings before uploading.
- Progress Tracking: The app displays merging and upload progress to keep the user informed.
- Admin management: Administrators can manage user roles, view user statistics, access audit logs for administrative actions, and control user permissions through a dedicated dashboard with multi-layer security verification.
- Error Handling: Provides alerts for failed merges, invalid files, or upload errors.

2.3 User Characteristics

- General User / Content Creator: A basic to moderately tech-savvy individual familiar with desktop applications and video content creation. The main goal is to allow users to merge multiple video clips and upload them to YouTube with minimal steps (no more than 10 clicks per main action). Users might fall under the Guest and Free category, which restrictions fit

for casual use.

- YouTuber / Streamer: Regularly merges and uploads recordings, often handling multiple short clips per session. Prioritizes efficiency and reliability. Users might fall under the Premium category, which has no restrictions fit for their needs.
- Educator / Lecturer: Creates lecture recordings or tutorial videos and seeks an easy solution to merge and upload them without technical complexity. Users might fall under the Premium category, which has no restrictions fit for their needs.
- Administrator: A new user type responsible for managing the application's user base, viewing audit logs, and promoting users.

2.4 Constraints

- The application must run on Windows 10 or later.
- The app must integrate with FFmpeg for all video merging operations.
- Upload functionality must comply with YouTube Data API v3 authentication and upload policies.
- All user authentication tokens must be stored securely and encrypted locally.
- The system must be developed using Python with the Flet framework for the graphical user interface.

2.5 Assumptions and Dependencies

- Users must have a stable internet connection to upload videos.
- Users must have a YouTube account with permission to upload.
- FFmpeg and YouTube API services must be available and functioning.

3. Specific Requirements

This section contains the detailed requirements necessary to build the system.

3.1 Functional Requirements

This details what the system should do.

FR-001: User Authentication

- The system shall allow users to sign in using their YouTube account through the YouTube Data API (OAuth 2.0).
- After login, the system shall save a secure session token for future uploads.
- The system shall implement Hybrid Authentication. It uses Google OAuth 2.0 for identity verification and Firebase Authentication for session management. It must support distinct user roles (Guest, Free, Premium, Admin).

FR-002: Video Selection and Arrangement

- The user shall be able to import at least 1 video file from local storage without errors.
- The user shall be able to reorder the video clips before merging.

FR-003: Video Merging

- The system shall merge all selected video clips into one continuous video file using FFmpeg.
- The user shall be able to choose the output format (e.g., MP4) and resolution.

FR-004: YouTube Upload

- The system shall upload the merged video directly to YouTube using the user's account.
- The user shall be able to enter a title, description, tags, and privacy setting (Public, Unlisted, or Private).

FR-005: Progress Tracking and Notifications

- The system shall display real-time progress for both merging and uploading.
- The system shall notify the user when merging or uploading is completed or fails.

FR-006: Error Handling and Logging

- The system shall handle errors such as missing files, upload failures, or invalid credentials gracefully.
- The system shall log all errors for troubleshooting. (Note: Currently implemented as console output only; persistent log files not implemented but can be reviewed during active sessions)

3.2 Non-Functional Requirements

This details how the system should perform its functions

NFR-001: Performance

- Merging performance shall depend on system resources, but the app must handle up to 10 clips within a reasonable time (<5 minutes for standard HD).
- Upload feedback must appear within 5 seconds of starting the upload.

NFR-002: Usability

- A user who has never used the app before should be able to merge and upload a video successfully within 3 attempts, without external guidance.

NFR-003: Security

- User authentication data (tokens) must be securely stored locally using serialized storage.
- No passwords will be stored by the app. (Note: OAuth tokens are stored using Python's pickle serialization for local persistence. While not encrypted, tokens are obfuscated and stored with restricted file permissions)

NFR-004: Reliability

- The app should provide error recovery options for upload failures.
- If upload fails due to network loss or other issues, the app shall display detailed error messages and provide a manual retry option through the UI.

- All errors during merging and upload operations are logged to console output for troubleshooting during active sessions.

  (Note: Automatic upload resumption and persistent upload queue were not implemented. Users must manually retry failed uploads using the retry button. Future enhancement could include automatic retry with exponential backoff and persistent queue storage)

NFR-005: Maintainability

- The system should be modular and allow easy updates for new features or API changes.