Spectrum Analyzer Analysis Tool

Design Document

Team 6

Ryan O'Connor, Ayorinde Lawani, Richard Luthringshauser, Nathan Reed, Dinesh Sekar

Date: (10/24/2023)
Department: MSSWE
Course: SWE 7903 Capstone
Professor: Dr. Reza Parizi

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Purpose

The primary purpose of this document is to provide the reader with a comprehensive understanding of how and why our application is designed. The intended audience for this document includes software developers, project managers, system engineers, testers and stakeholders.

1.2 Scope

The Spectrum Analyzer Tool is an advanced software application specifically tailored to transform the traditional, labor-intensive process of analyzing RF transmissions from a Spectrum Analyzer into a streamlined, automated one. The software's primary objective is to process video data from the Spectrum Analyzer, identify and interpret RF signal peaks, and then transcribe this information into a numeric format. This process not only reduces the time taken to analyze the data but also minimizes human errors, ensuring the data's accuracy and reliability.

The scope of this software encompasses:

- Reading and processing video data from a Spectrum Analyzer, which displays amplitude on the vertical axis and frequency on the horizontal axis.
- Recognizing and processing the center frequency of a signal and determining the amplitude of the signal peak. The software will track the minimum, maximum, and average amplitudes for each identified peak.
- Providing an intuitive user interface for users to select video files, set processing parameters, and view the results.
- Storing the analyzed data in a structured format, making it easier for further digital manipulation or analysis.

The primary goals of the Spectrum Analyzer Tool are:

1. **Efficiency:** To drastically reduce the time required to analyze RF transmission data.
2. **Accuracy:** To minimize human error by automating the data extraction process.
3. **Usability:** To offer an intuitive interface ensuring ease of use for individuals.

The benefits of this project include:

1. **Cost Savings:** Reducing the hours required for video analysis leads to significant cost savings.
2. **Data Integrity:** Automated processes ensure consistent and error-free data extraction.
3. **Enhanced Decision Making:** With accurate and quickly available data, stakeholders can make more informed decisions.
4. **Scalability:** The software can be extended to handle additional data formats or integrate with other systems if required in the future.

1.3 Overview

This Software Design Document offers a comprehensive blueprint of the Spectrum Analyzer Tool, detailing its architecture, functionality, and design considerations. The document has been meticulously structured to ensure a systematic presentation of information, making it easy for both technical and non-technical stakeholders to grasp the software's intricacies.

Description of each section:

1. **Introduction:** This section establishes the document's context by outlining its purpose, scope, and providing a brief overview. It also includes reference materials, definitions, and acronyms to familiarize the reader with specific terminology and concepts used throughout the document.
2. **System Overview:** A high-level description of the Spectrum Analyzer Tool, shedding light on its primary features, functionalities, and interactions within the system.
3. **System Architecture:** Delves into the architectural design of the software, illustrating how different components interact with each other. The design rationale segment justifies the architectural choices made during the design phase.
4. **Detailed Design:** This section provides an in-depth look into the software's internal workings, elaborating on algorithms, interfaces, and data flow within the application.
5. **Database (Data) Design:** Focuses on the storage aspect of the software, detailing how data is stored, retrieved, and manipulated within the system.
6. **Human Interface Design:** Concentrates on the user's interaction with the software. The UI design sub-section describes the graphical elements, layout, and aesthetics, while the UX design segment emphasizes the user experience, ensuring the software is intuitive and user-friendly.
7. **Appendices:** Contains supplementary information that supports the main content of the document, such as code snippets, additional diagrams, or reference tables.

1.4 Reference Material

List any documents, if any, which were used as sources of information

1.5 Definitions and Acronyms

Terms:

OCR – Optical Character Recognition

2. SYSTEM OVERVIEW

Description:

The Spectrum Analyzer Tool is an advanced software solution tailored specifically for engineers at Robins Air Force Base, located in Warner Robins, GA. Designed to significantly streamline the process of analyzing RF signal intercepts, this tool eradicates the previous labor-intensive manual inspection process.

Functionality:

The Spectrum Analyzer Tool's primary function is the automatic detection of RF signal peaks within videos. Engineers can easily select their desired videos, set a frame skip frequency, and initiate the processing. The software quickly sifts through the video content, identifying and cataloging specific signal peaks within the frames. Moreover, users are kept updated on the processing status via real-time feedback mechanisms, such as a progress bar and processed video frame displays.

Context:

Robins Air Force Base, with its long-standing reputation in innovative aerospace technology and research, often deals with lengthy videos containing vital RF signal intercepts. The manual process of reviewing these videos to detect and analyze signal peaks was not tedious. There was a pressing need for an automated solution that could efficiently handle this task, ensuring accuracy and saving valuable man-hours.

Design:

To ensure that the software met the exacting standards expected at Robins Air Force Base, a multi-threaded architecture was adopted for optimal video processing. The user interface, created using the PyQt5 framework, is intuitive and user-friendly, allowing the engineers to easily navigate and operate the tool. On the back end, the power of OpenCV, a trusted library in computer vision, is harnessed to meticulously analyze video frames.

Background Information:

Traditionally, RF signal intercept analysis at Robins Air Force Base required engineers to manually inspect hours-long videos. This not only consumed vast amounts of time but also risked potential oversights due to human fatigue. Recognizing the inefficiency and inaccuracies of this method, the decision was made to transition to a software-driven approach. The Spectrum Analyzer Tool emerged as the solution, born from the necessity to ensure that the engineers at Robins Air Force Base had the best tools at their disposal. Now, with this tool, they can focus on more critical tasks, confident in the knowledge that their RF signal analysis is accurate and efficient.

3. SYSTEM ARCHITECTURE

3.1 Architectural Design

The Spectrum Analyzer Tool is structured into a modular architecture, where each module or subsystem has a well-defined role. This modular approach ensures flexibility, maintainability, and scalability of the tool.

Subsystem/Service Overview:

User Interface (UI) Module: Built with PyQt5, this module provides an interactive graphical interface for users to upload videos, set parameters, start the analysis, and view results.

Video Processing Module: Harnesses the power of OpenCV to break the video into frames, apply the required filters, and analyze them for RF signal peaks.

Data Storage Module: Responsible for saving the detected peaks, both temporarily during processing and long-term once the analysis is complete by exporting the data in CSV format.

Feedback Mechanism: Provides real-time feedback to the user on the status of the processing, including a progress bar and display of processed frames.

Integration & Communication Layer: Acts as a mediator, ensuring seamless communication between the other modules. It takes requests from the UI, sends them to the Video Processing Module, retrieves results, and then communicates back to the UI.

Subsystem Collaboration:

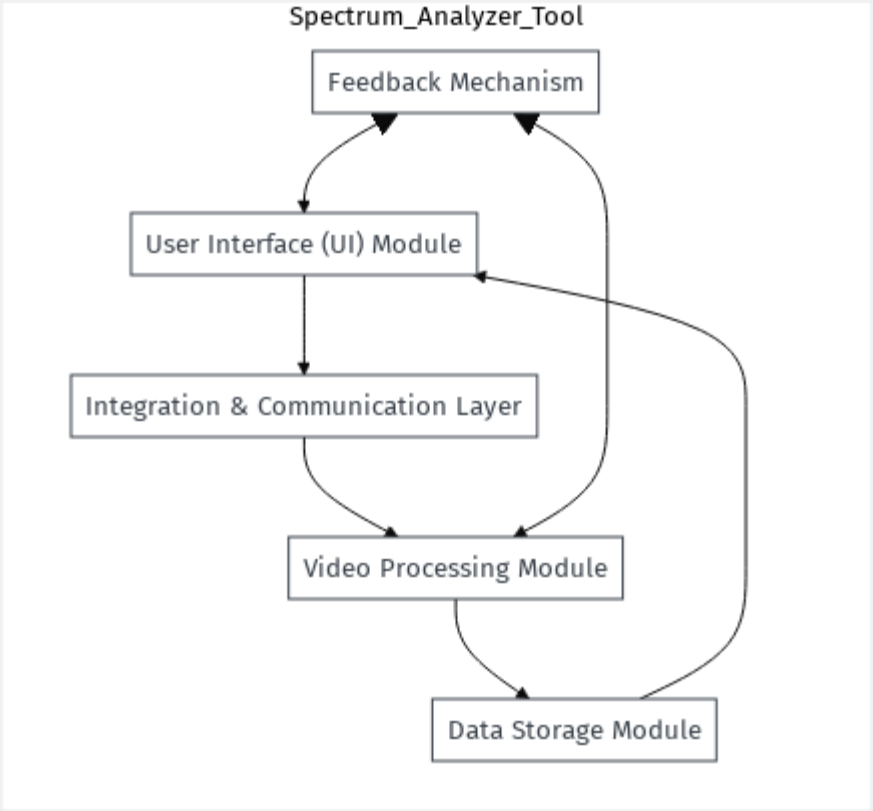
The **User Interface Module** receives input from the user, such as the video file and processing parameters. Once the user initiates the process, it communicates the request to the Integration & Communication Layer.

The **Integration & Communication Layer** directs the video file to the **Video Processing Module**. As the Video Processing Module processes each frame and identifies peaks, it sends this data to the **Data Storage Module** for temporary storage.

The **Feedback Mechanism** actively monitors the progress and communicates updates back to the UI, allowing users to see the status in real-time.

Once the processing is complete, the **Data Storage Module** makes the results available for the user, either for viewing within the tool or as an exported CSV file.

Architectural Diagram:



3.2 Design Rationale

General Rationale:

The chosen architecture for the Spectrum Analyzer Tool was rooted in the immediate requirements and constraints presented by the Robins Air Force Base engineers and the context of the project being a one-time graduate software engineering capstone.

Modular Architecture:

Rationale: A modular design was adopted to ensure that each distinct function of the system could be developed, tested, and maintained independently. Even if there are no future enhancements, this approach ensures that the system is robust, and each module works within its purview.

Trade-offs: While modular design can sometimes add initial overhead in terms of integration, its inherent benefits in terms of clear segregation of functions and potential ease of troubleshooting if issues arise justified the approach.

Use of OpenCV in Video Processing:

Rationale: OpenCV is a proven library for computer vision tasks. To process videos and detect RF signals effectively, OpenCV provided the necessary functionalities.

Trade-offs: While OpenCV may be more comprehensive than required, its reliability and performance made it a suitable choice.

EasyOCR in Video Processing:

Rationale: EasyOCR is an open-source library for performing optical character recognition allowing for automatic detection of spectrometer settings.

Trade-offs: Implementation of OCR in the video processing pipeline will increase start up time as OCR must complete before the process can complete.

PyQt5 for UI Development:

Rationale: Given the need for an intuitive interface for the engineers, PyQt5 was chosen for its capabilities to deliver a responsive and user-friendly interface.

Trade-offs: While there are other frameworks available, PyQt5's native performance and seamless integration with the chosen tech stack made it the preferred choice.

Data Storage Module:

Rationale: With the intent to allow RF signal data to be revisited or analyzed by the engineers, a dedicated data storage module was deemed necessary. The capability to export data in CSV format ensures that even in our absence, the users can utilize the data with other tools.

Trade-offs: The addition of a storage solution might add a layer of complexity, but the need for persistent storage and user-friendly data export capabilities justified the decision.

Feedback Mechanism:

Rationale: Given the duration of the videos, real-time feedback on the processing status was incorporated to enhance the user experience and ensure transparency in operations.

Trade-offs: While real-time feedback requires a bit more computational resources, the value it adds to the user experience made it a beneficial inclusion.

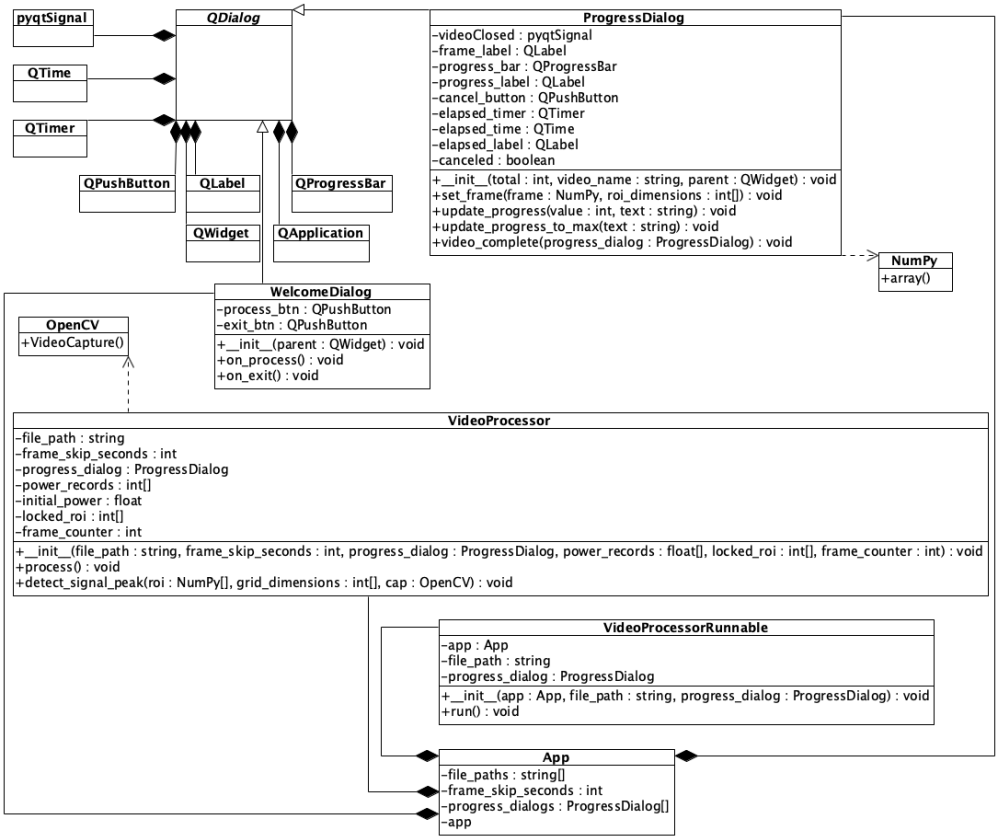
Alternatives Considered:

Cloud-Based Processing: While cloud solutions could offer more computational power and ease of deployment, the lab's secure PCs lack internet access. Thus, an offline processing solution was not just a preference but a necessity.

Single Integrated Design: A holistic, non-modular approach was initially pondered upon to reduce complexities. However, for clarity of development and potential troubleshooting, a modular design was more suitable.

4. DETAILED DESIGN

UML Class Diagram



Class Descriptions

WelcomeDialog:

- This class inherits from QDialog.
- Attributes:
  - process_btn: Represents the "Process Video(s)" button.
  - exit_btn: Represents the "Exit" button.
- Methods:
  - __init__(self, parent=None): Constructor method to initialize the dialog.
  - on_process(self): Handles the click event of the "Process Video(s)" button.
  - on_exit(): Handles the click event of the "Exit" button.

VideoProcessorRunnable:

- This class inherits from QRunnable.
- Attributes:
  - app: Reference to the main application.
  - file_path: Path to the video file to be processed.
  - progress_dialog: Dialog to show processing progress.

- Methods:
  - __init__(self, app, file_path, progress_dialog): Constructor method to initialize the runnable.
  - run(self): Main processing method for the runnable.

VideoProcessor:

- Attributes:
  - file_path: Path to the video file.
  - frame_skip_seconds: Number of seconds to skip between processed frames.
  - progress_dialog: Dialog to show processing progress.
  - power_records: List to store detected power records.
  - initial_power: Initial detected power.
  - locked_roi: Bounding box of the Region of Interest.
  - frame_counter: Counts processed frames.

- Methods:
  - __init__(self, file_path, frame_skip_seconds, progress_dialog): Constructor method to initialize the video processor.
  - process(self): Main method to process the video.
  - _write_power_records_to_csv(self): Save detected power records to a CSV file.
  - detect_signal_peak(self, roi, grid_dimensions, cap): Detect the peak of a signal within a Region of Interest (ROI).

ProgressDialog:

- This class inherits from QDialog.
- Attributes:
  - videoClosed: Signal emitted when the dialog is closed.
- Methods:
  - __init__(self, total, video_name, parent=None): Constructor method to initialize the dialog.
  - And other methods to handle the progress dialog's behavior, such as updating progress, handling button clicks, and displaying elapsed time.

App Class:

- Attributes:
  - file_paths: Paths of the selected video files.
  - frame_skip_seconds: Number of seconds to skip between frames during processing.
  - progress_dialogs: List of ProgressDialog instances showing the progress of video processing.

- Methods:
  - __init__(self): Constructor method to initialize the app.
  - show_welcome_screen(self): Display the welcome dialog to the user.
  - select_and_process_videos(self): Launch a dialog for video file selection and initiate processing.
  - video_completed(self, progress_dialog): Handle the completion of video processing.
  - on_video_closed(self): Handle the event when a video processing dialog is closed.
  - process_videos(self): Start the video processing tasks.

Pseudocode:

Video Processing Module Pseudocode:

```
function process(video_path):
   load video from video_path
   while video is not ended:
      frame = read_next_frame()
      signals = detectRFSignals(frame)
      save signals to data storage
   end while
end function

function detectRFSignals(frame):
   process frame using OpenCV methods
   detect patterns indicative of RF signals
   return list of detected signals
end function
```

UI Module Pseudocode:

```
function startProcessing():
   video_path = get_input_video_path_from_user()
   processor.process(video_path)
   displayResults()
end function

function displayResults():
   data = processor.get_detected_signals()
   display data in a user-friendly format on the UI
end function
```

```
function exportData():
    data = processor.get_detected_signals()
    save data as CSV
    notify user of successful export
end function
```

Data Storage Pseudocode:

```
function save(data):
    open storage_path for writing
    write data to storage_path
    close storage_path
end function

function exportAsCSV():
    open storage_path for reading
    convert data to CSV format
    save as a CSV file
    return CSV file
end function
```

Feedback Mechanism Pseudocode:

```
function updateStatus(status):
    current_status = status
    notify user of current_status
end function
```

5. DATA (DATABASE) DESIGN

The system employs a direct and simplified approach for storing the detected RF signals' data through the utilization of CSV files. This section will detail the data design specific to this CSV-based storage.

Data Transformation:

The primary informational entities:

- **Video Streams:** Raw video data containing potential RF signals.
- **Detected Signals:** Information related to signals detected from the video stream, which includes attributes like timestamp, signal frequency, power, minimum, maximum, and average values.

The transformation process involves the VideoProcessor module analyzing the video streams. Once signals are detected, their data is directly written into the CSV

file.

Data Structures and Entities in CSV format:

- **Signal CSV:** Contains records of detected RF signals from the videos.
    - Columns:
        - timestamp: Time at which the signal was detected in the video.
        - frequency: The frequency of the detected signal.
        - power: Power level of the detected signal.
        - min: Minimum value observed for the signal during its occurrence.
        - max: Maximum value observed for the signal during its occurrence.
        - average: Average value computed for the signal during its presence.

Data Storage Mechanism:

Given the offline nature of the Robins Air Force Base environment, the data is stored locally in a CSV file named power_records_{file_path}.csv. This naming convention ensures a unique and identifiable filename for each processed video:

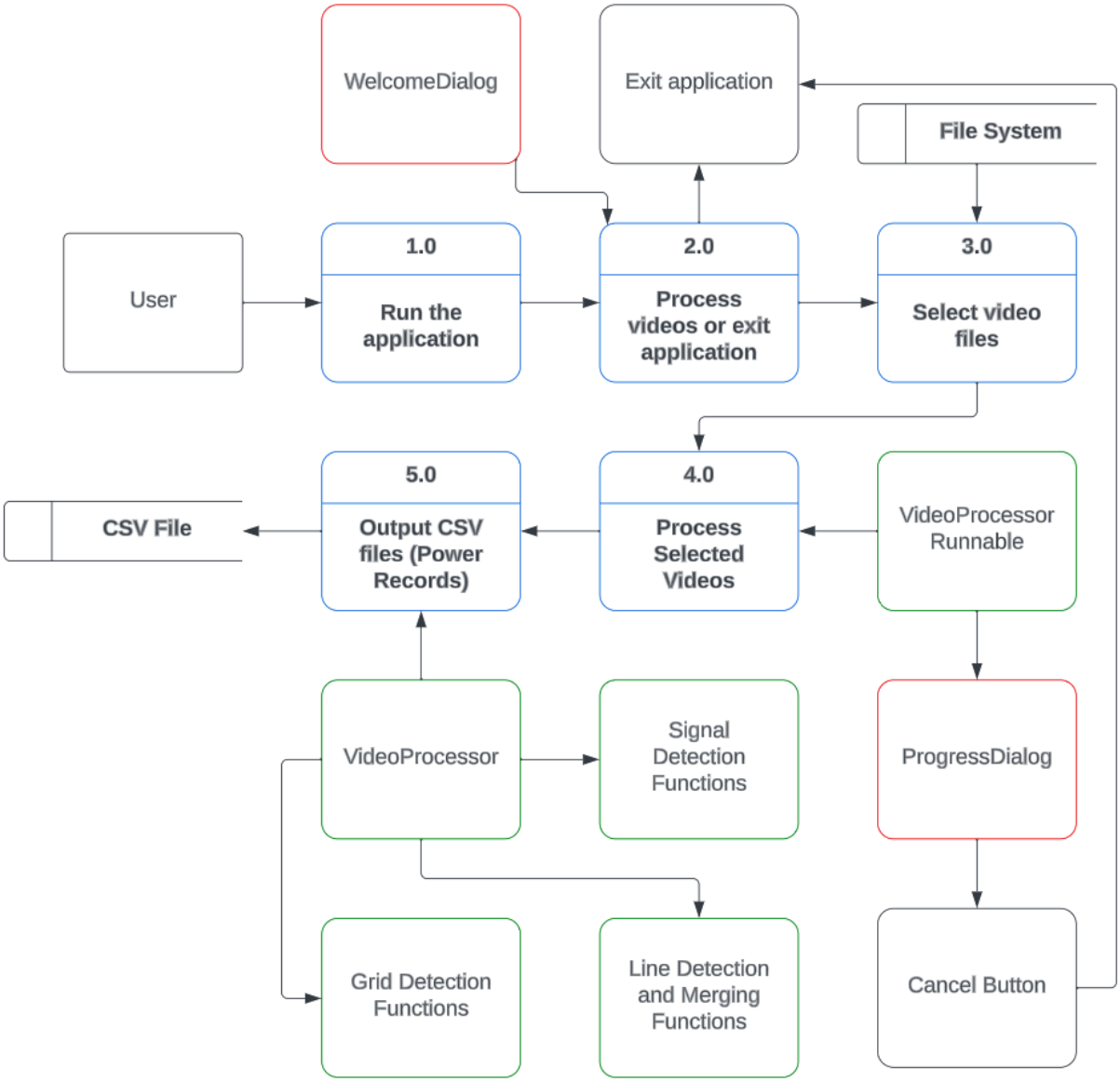
- Once the VideoProcessor detects signals in a video stream, it writes the detected signal data into the CSV file.
- The CSV file is saved to a predefined directory on the local drive.

Benefits of this approach:

- **Simplicity:** Eliminates the need for a complex database setup, management, or maintenance.
- **Portability:** The CSV file can be easily transferred, backed up, or shared, given the appropriate permissions.
- **Compatibility:** Being a universally accepted format, CSV ensures potential future adaptability or integration with other systems.

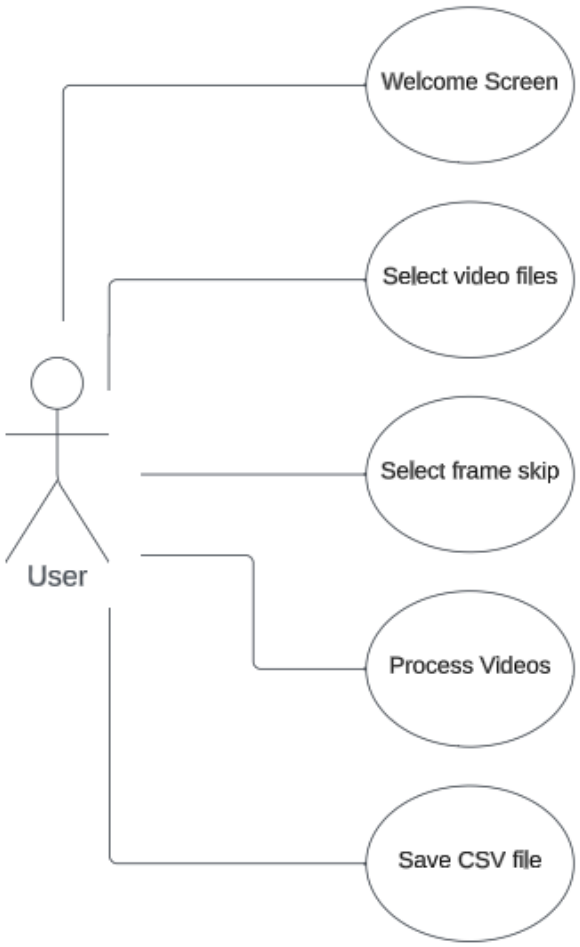
This data design, centered around a single CSV file, ensures that all detected RF signals are securely and efficiently stored on the local drive, making it an apt solution for the requirements at Robins Air Force Base.

Data Flow Diagram



6. HUMAN INTERFACE DESIGN

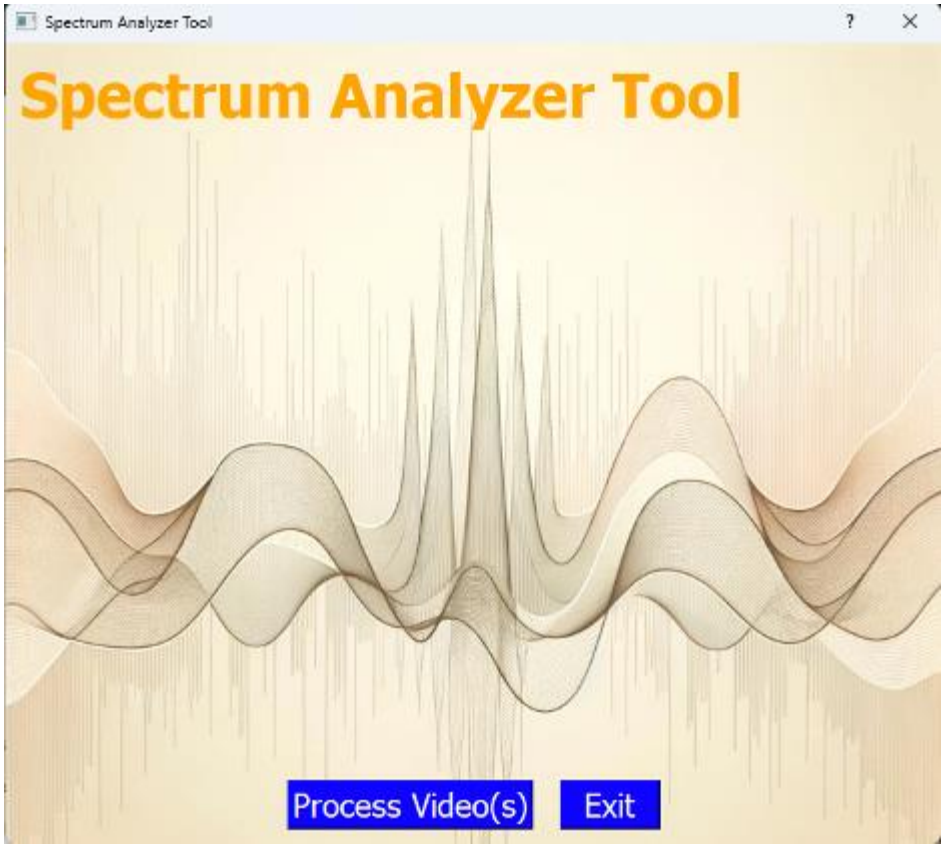
6.1 UI design



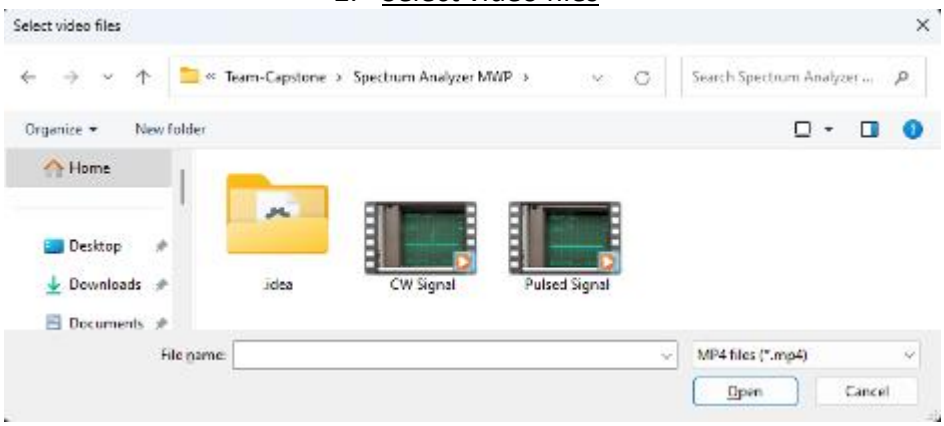
The UI design diagram features the following UI elements:

1. Welcome Screen: The welcome screen is shown to the user when the application starts. The screen offers the options, through clickable buttons, to process videos or exit the application.
2. Select video files: The select video files file picker enables the user to select multiple video files from connected filesystems.
3. Select frame skip: The select frame skip screen prompts the user to choose a value from a dropdown list. The options are 1, 5, 10, 15, 30 and 60 seconds. This selection determines how many video frames are excluded from processing in order to decrease processing time.
4. Process videos: The process videos screen displays four items. It shows a video of the file being processed, a progress bar, a Cancel button, and elapsed processing time.
5. Save CSV file: Processing results are saved to a CSV file on the users filesystem.

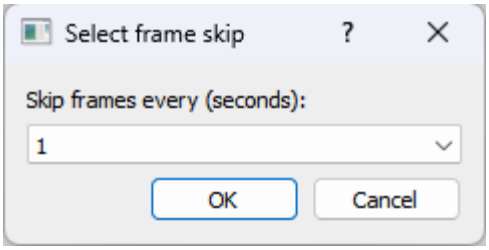
1. Welcome Screen



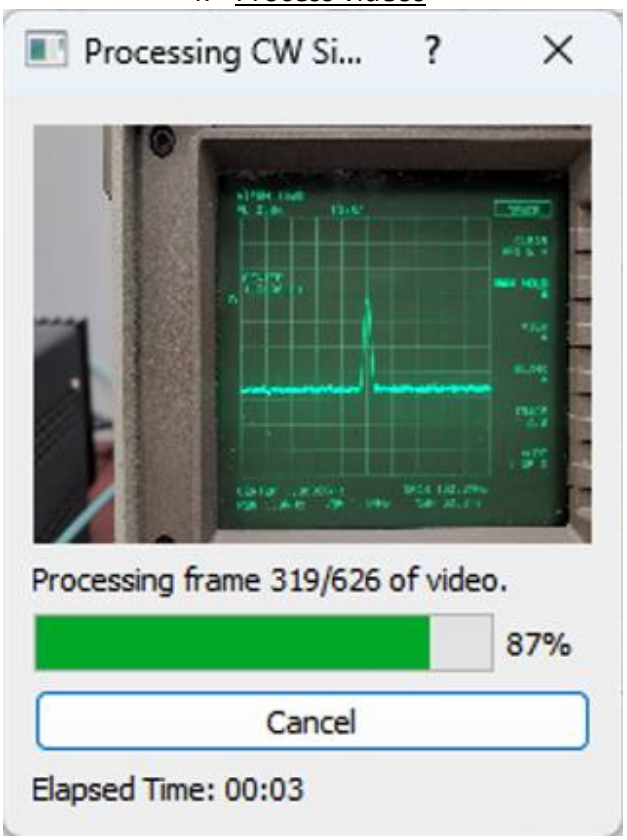
2. Select video files



3. Select frame skip



4. Process videos



5. Save CSV file

main.py | power_records_CW Signal.csv

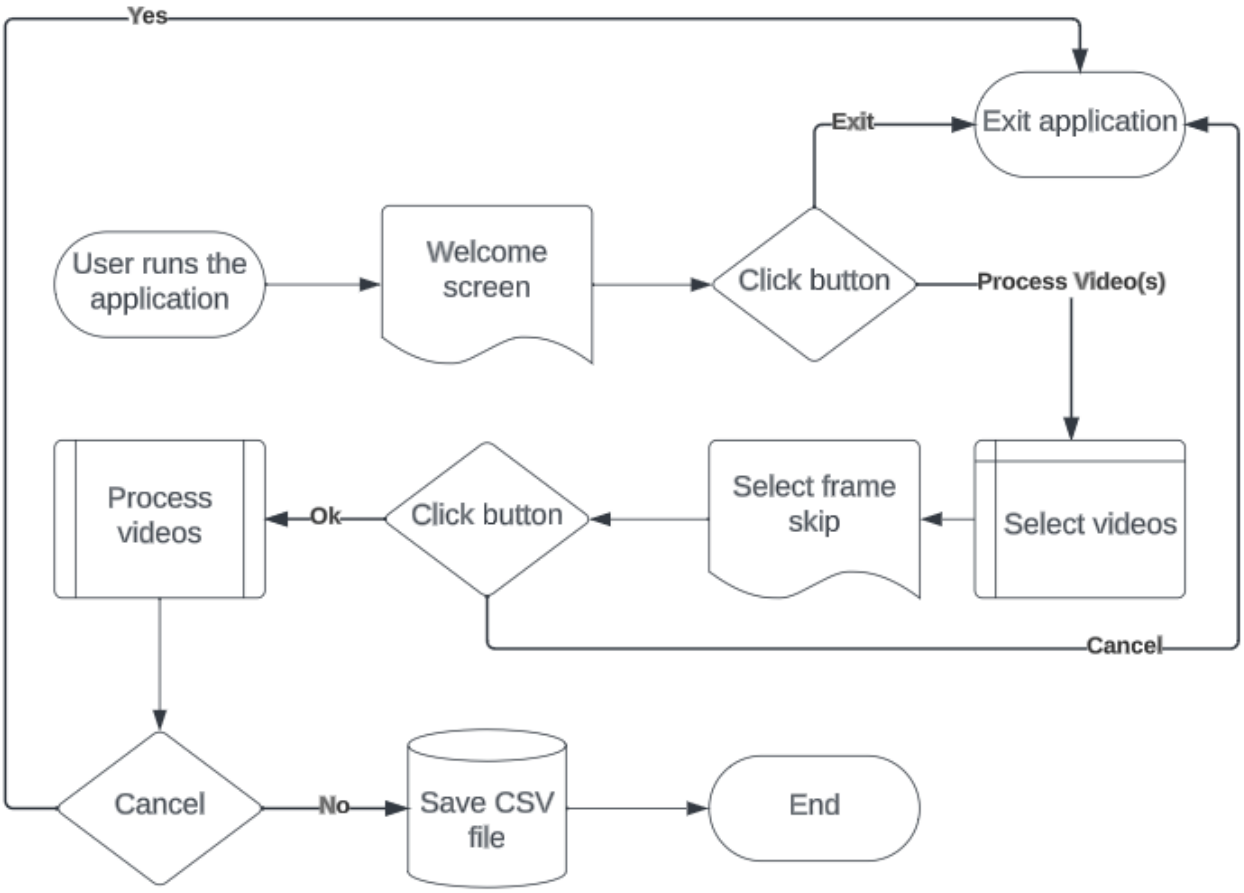
	timestamp (1)	frequency (2)	power (3)	min_power (4)	max_power (...	avg_power (6)
1	timestamp	frequency	power	min_power	max_power	avg_power
2	00:01	1.03	-53.16	-53.16	-53.16	-53.16
3	00:01	1.03	-53.94	-53.94	-53.16	-53.55
4	00:01	1.01	-33.18	-53.94	-33.18	-46.76
5	00:01	1.01	-32.98	-53.94	-32.98	-43.31
6	00:01	1.00	-32.01	-53.94	-32.01	-41.05
7	00:01	1.00	-32.88	-53.94	-32.01	-39.69
8	00:01	1.00	-32.88	-53.94	-32.01	-38.72
9	00:01	1.00	-33.08	-53.94	-32.01	-38.01

6.2 UX design

User Goal:
The user has the goal of processing spectrum analyzer signals contained in video files in less than real time.

Process:

- The user runs the application and is greeted by the welcome screen. The screen contains two buttons – ***Process Video(s)*** and ***Exit***.
- The user clicks on the ***Process Video(s)*** button and is prompted to select video files from their file system. The user can also click ***Exit*** to exit the application.
- Once video files are opened, the user is presented with the ***Select frame skip*** form. The ***Skip frames every (seconds):*** dropdown determines the speed of video processing.
- The user selects and clicks Ok. The user can also click ***Cancel*** to exit the application.
- Processing of the video files commences, and the screen displays both video and a progress bar. The user can ***Cancel*** to exit the application.
- Once video processing is complete, the application saves power record results to the file system in the form of a CSV file.



7. APPENDICES

This section is optional.

Appendices may be included, either directly or by reference, to provide supporting details that could aid in the understanding of the Software Design Document.