**Spectrum Analyzer Tool**

**Design Document**

# 

Chris Williams, George Vendeta, Seema Kumaran, Jeffrey Amakihe, Alexandro Pinion

Date: 10/25/2023

Department: Computing and Software Engineering

Course: SWE 7903 - Capstone

Professor: Dr. Parizi

**Table of Contents**

[1. INTRODUCTION 2](#_Toc148874040)

[1.1 Purpose 2](#_Toc148874041)

[1.2 Scope 2](#_Toc148874042)

[1.3 Overview 2](#_Toc148874043)

[1.4 Reference Material 2](#_Toc148874044)

[1.5 Definitions and Acronyms 2](#_Toc148874045)

[2. SYSTEM OVERVIEW 3](#_Toc148874046)

[3. SYSTEM ARCHITECTURE 4](#_Toc148874047)

[3.1 Architectural Design 4](#_Toc148874048)

[3.2 Design Rationale 7](#_Toc148874049)

[4. DETAILED DESIGN 7](#_Toc148874050)

[5. DATA (DATABASE) DESIGN 7](#_Toc148874051)

[6. HUMAN INTERFACE DESIGN 7](#_Toc148874052)

[6.1 UI design 7](#_Toc148874053)

[6.2 UX design 7](#_Toc148874054)

[7. APPENDICES 7](#_Toc148874055)

### INTRODUCTION

## Purpose

The purpose of this System Design Document (SDD) is to provide an extensive blueprint that describes the architecture and design of a software. Its main goal is to give a precise and comprehensive explanation of the software's architecture, design, component interactions, and means of meeting the requirements. The SDD fulfills several crucial functions. It serves as a communication tool to assist developers, testers, project managers, and other team members in comprehending the architecture of the software system. Throughout the software development lifecycle, the development team uses it as a reference document. It facilitates the documentation of design choices and justifications. The intended audience for this SDD is for developers, Robins AFB stakeholders, testers and project managers.

## Scope

The goal of this Software project is to create, test, and implement a user-friendly program that the Robins AFB can use to solve video analysis and spectrum analyzer difficulties. An independent deployment environment for the client's platform, formatted CSV data representing spectrum analyzer signals, and image signal processing of video are some of the key features. The project's goal is to provide a reliable, scalable solution that addresses the client's issue within the allocated spending limit and time frame.

## Overview

This document will go over the system architecture, data flow, user interfaces, algorithms, data structures, security considerations, error handling, and other subjects are covered in detail in this document. It serves as an essential means of communication for development teams, testers, project managers, and stakeholders, guaranteeing that all parties participating in the project comprehend the functioning and design of the product. An integral part of the software development lifecycle, the SDD also acts as a point of reference for scalability and future maintenance.

## Reference Material

*This section is optional.*

List any documents, if any, which were used as sources of information for the test plan.

## Definitions and Acronyms

*This section is optional.*

Provide definitions of all terms, acronyms, and abbreviations that might exist to properly interpret the SDD. These definitions should be items used in the SDD that are most likely not known to the audience.

### SYSTEM OVERVIEW

As a key aviation hub, Robins Air Force Base (AFB) uses outdated Spectrum Analyzers to measure the quality of transmission signals within their ground stations. These Analyzers gather information while in flight, producing hours' worth of crucial data that is necessary to preserve the integrity of communications networks. However, there are operational inefficiencies and a high human resource cost associated with the current manual procedure of examining video recordings of these datasets for anomalies in signal quality. In response to this challenge, KSU's Master of Software Engineering Capstone students have been requested by Robins AFB to develop an automated system for the analysis of flight data. This will guarantee prompt identification of problems with signal quality while optimizing operations and resource use. This group of students behind this project will create an open-source standalone application that will operate on the computers that are intended for use by Robins AFB personnel. The application will offer a guided process for loading relevant data, processing it, and producing output resources that will be most helpful to the stakeholders at Robins AFB.

### SYSTEM ARCHITECTURE

## Architectural Design

This section includes a high-level structure design that outlines the main elements, how they relate to one another, and how the application is organized overall. It acts as a construction blueprint for the software, assisting developers in comprehending how different system components interact to produce the intended functionality. Important non-functional needs like performance, maintainability, and scalability are covered. This section will also provide a high-level road map that directs the development team in creating a system that satisfies the project's functional and quality objectives while taking flexibility and maintenance ease into account is the aim of architectural design. For the Spectrum Analyzer Tool application to remain viable and successful over the long term, a well-designed architectural design is essential. Below are the main components of the systems architecture:

* **User Interface (UI) Layer**:
  + The desktop application's user interface is known as the UI.
  + It offers an easy-to-use interface so people can engage with the program.
  + It has options for image processing, video data display, and file input for choosing video files.
  + Additionally, the UI provides controls for exporting CSV files and starting the data conversion process.
* **Logic Layer**:
  + Video Processor:
    - This part extracts pertinent data by processing the loaded video frames.
    - It could involve filtering, feature extraction, picture analysis, and other image processing methods.
    - The data has been processed and is ready to be converted to CSV format.
  + This layer will perform the business logic for the application.
* **Data Management Layer:**
  + CSV Data Converter:
    - This module creates a CSV file from the processed video data.
    - It arranges the information in a tabular style that can be exported as CSV.
    - The CSV format may be configured by users with options to set headers and delimiters, for example.
  + This layer will be responsible for creating all of the non-volatile data storage for the application.
* **Exceptions and Logging Layer**:
  + Put in place an error-handling system to handle unforeseen problems that arise when loading, processing, or converting videos.
  + Produce logs for reporting errors and debugging.
* **Distribution Layer**:
  + Builds all packages and bundles all preset libraries, dependencies, and components that makes it simpler to deploy the application to the user.
  + Generates a standalone executable that operates independently on a computer system without the requirement for extra resources or dependencies.

The Figure 1 below will showcase an overview diagram of the architectural design for the Spectrum Analyzer Tool and how each of the components in the application provide key functionality. Each of the diagram’s components are described above.

A diagram of a software application

Description automatically generated

Figure - Architectural Diagram of Spectrum Analyzer Tool

## Design Rationale

The primary software architecture pattern chosen was the Model View Controller (MVC) pattern. The key benefit of implementing the MVC architecture in a desktop application is that it encourages a distinct division of responsibilities, which improves the program's maintainability, scalability, and flexibility. It is possible to modify the user interface (View) without also changing the data processing logic (Model) that underlies it, and vice versa. Because developers can work on different components independently, this separation also makes testing and cooperation easier.

**View** - The choice for the user interface is crucial because it acts as the main point of contact between the user and the program, enabling clear and effective communication and will act as the View in our architectural pattern of choice. A well-thought-out user interface improves the user experience by offering a visual framework that leads users through the functionalities of the application, facilitating simpler navigation, comprehension, and usage of the capabilities. It guarantees that users may engage with the program in a way that is both aesthetically pleasing and easy to use, which eventually enhances productivity, usability, and user pleasure.

**Controller** - An image processor was added because it is beneficial to increase the capability and versatility of a desktop program for manipulating video files by adding image processing. Through image processing, the application can extract useful data from video frames, including motion tracking, object recognition, and facial detection. Consequently, this enables users to carry out diverse tasks such as data analytics, video-image manipulation, and surveillance with increased accuracy and automation. Additionally, image processing can enable features like visual effects, noise reduction, and video stabilization, all of which enhance the output's overall quality and aesthetics. It gives the application an extra layer of intelligence, which makes it a potent tool for the specific use-case intended for the Robins customers.

**Model** - For effective data organization, storage, and retrieval, a desktop program processing photos and converting them to CSV data must have a data management layer. This layer gives the software the ability to organize and handle the translated image data in a methodical manner, facilitating user access and searching. Version control, indexing, and metadata tagging are all made possible by it, and these features are very helpful for complex image processing jobs. It also makes it easier to manipulate, transform, and export data into CSV format, which improves the application's ability to handle data and meets the primary requirements of the developed system. The application is a useful tool for a variety of picture data analysis and reporting jobs because a strong data management layer ensures that users can work with structured and relevant data while streamlining the conversion process and enhancing overall data quality – this will afford the Robins customers more leverage with the different data analytics that they will perform on the applications CSV data.

In addition to the Model-View-Controller (MVC) pattern, there are two other architectural patterns that were taken into account: the Single Responsibility Principle (SRP) and the Model-View-ViewModel (MVVM). Because MVVM uses a ViewModel to segregate the View from the underlying data, it makes data binding simpler and allows for dynamic UI updates. This makes it ideal for applications with sophisticated user interfaces, such image processing apps. However, we did not need the added sophistication to our user interface, and preferred the MVC to allow us the benefits of having a designated controller – the controller was more beneficial for our requirements of image processing opposed to the MVVM architecture overall. Based on architectural design principles, SRP promotes modular and maintainable code by highlighting the Single Responsibility Principle. It may not give as clear a boundary between the View and Model as MVC or MVVM, which could be a disadvantage in our desktop application being developed, even though it can result in simpler code and higher testability. On that notion, choosing the MVC seemed better suitable for our system since our customers may build upon our code base and add features to the application after our delivery.

### DETAILED DESIGN

In this section, provide details (with respect to 3.1) at what each component does in a more systematic way. For instances, provide UML Class diagrams with all methods, attributes and relationships for all involving objects, if you are using OO approaches, provide a summary of algorithms in procedural description language (PDL) or pseudocode.

As referenced in section 3.1 and depicted in Figure 1, our system design is comprised of a few core components. This section will explain the purpose of each component and how they are broken down with an object-oriented approach for developing the design of the system. Our design leverages an object-oriented design in order to help create organized, modular and reusable code. It mirrors real-world structures and relationships, making it easier for developers and our Robins stakeholders to understand and maintain the code during development, and after delivery. We will use class diagrams to represent the objects, attributes, methods and classes used to design the application. The components described will be: (1) UI Component, (2) Processor Component, (3) Data Component, and (4) Exception and Logging Component.

## UI Component

The user interface is a critical component of our chosen design since it serves as the primary point of contact between the user and the program, facilitating efficient and transparent communication. A well-designed user interface enhances the user experience by providing a visual framework that guides users through the application's operations, making it easier for them to navigate, understand, and utilize the capabilities. It ensures that users can interact with the application in a way that is easy to use and aesthetically beautiful, which subsequently improves user satisfaction, productivity, and usability. Below we will show the main objects, attributes and methods of the UI component shown in a class diagram.

A screenshot of a diagram

Description automatically generated

Figure - UI Component Class Diagram

## Processor Component

The addition of an image processor was made since it improves the functionality and adaptability of a desktop application for working with video files. The program can extract relevant information from video frames, such as motion tracking, object recognition, and facial detection, by using image processing. As a result, users can do a variety of tasks with more accuracy and automation, including data analytics, video-image editing, and surveillance. Furthermore, image processing can enable elements that improve the overall quality and aesthetics of the output, such as visual effects, noise reduction, and video stabilization. It adds another level of intelligence to the application, making it a powerful tool for the use-case meant for Robins clients. Below we will show the main objects, attributes and methods of the Image Processor component shown in a class diagram.

A computer code with text

Description automatically generated with medium confidence

Figure - Processor Component Class Diagram

## Data Manager Component

The application will process images and converts them to CSV – this requires that the data has a data management component for efficient data organization, storage, and retrieval. This component allows the software to systematically process and arrange the translated visual data, making it easier for users to access and search. It enables capabilities like version control, indexing, and metadata tagging, all of which are particularly useful for intricate image processing tasks. Additionally, it facilitates data manipulation, transformation, and export into CSV format, which enhances the application's data handling capabilities and satisfies the main needs of the system that was designed. The data manager component is useful for a variety of picture data analysis and reporting jobs because it ensures that users can work with structured and relevant data while streamlining the conversion process and enhancing overall data quality. Below we will show the main objects, attributes and methods of the Data Manager component shown in a class diagram.

### DATA (DATABASE) DESIGN

Explain how the information domain of your system is transformed into data structures. Describe how the major data or system entities are stored, processed and organized. List any databases or data storage items.

### HUMAN INTERFACE DESIGN

## UI design

## UX design

### 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.