1. Introduction  
 Overview of entire SRS  
 1.1 Purpose

The purpose of this requirements document is to bring clarity to the project by indicating what the hardline requirements are for the project to be considered a success. This will be done with the consideration that the desired end product may not be achievable and as a result will be treated as a form of research project or proof of concept.

Intended target audience for this document is our client and by extension CDK given this is their idea and the success of the project directly assists them. In addition, our instructors and teaching assistant are extensively involved in this project and they will be using this as a grading guideline. The assumption is that the reader is familiar with general concepts within computer science including artificial intelligence, common programming languages, and coding environments.

1.2 Scope

The product being developed will be a variant of open source AI that will be tailored to scanning submitted forms by CDK. With the inclusion of stretch goals this platform may also analyze driver licenses, vehicle imaging, or name to vehicle identification.

This project will be capable of analyzing a submitted form to locate its signature box and check to see if it has been signed. It will also prompt the operator if it is not certain of what it is observing and if a cosigner box is also present. The format of these submissions will be in PDF format. Stretch goals include the capacity to detect expiration dates on driver’s licenses and determine what make and model of a vehicle is based on a submitted image. Additional file formats are also being considered.

Benefits from this product include reduced costs for CDK in error checking forms, rapid validation of a document, and a singular platform that can scan the various form styles that CDK possesses. Another goal of this design is the ability for it to expand upon itself and adapt to new forms that may be introduced by permitting operator confirmation or override on its assessment.

1.3 Definitions, Acronyms, and Abbreviations

Open source AI is a free to access platform in which the code can be used and implemented as seen fit by the operator. OpenCV is a popular image processing example of this as well as TensorFlow and DL4j. Programming languages that may be used include C++, Python, and Java. The use of an IDE is likely, which means Integrated Development Environment which provides a more convenient setup for coding. Examples of IDE’s include Microsoft Visual Studio and Eclipse. Black box design means that the user does not need to understand the internal functions of the program, just what it wants as input and what the user will receive as the output.

1.4 References

[1] IEEE Software Engineering Standards Committee, “IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications”, June 25, 1998.

1.5 Overview

The remainder of the requirements document will contain a greater look into what precisely the project entails. This is going to involve looking at where the project stands for CDK and how it will be utilized as well as the products functions and characteristics for a user. Mention will also be made towards potential constraints placed upon the program.

The following section will be a general overview and description of the project, primary focus being placed on functions, characteristics, constraints, and assumptions/dependencies. After that there will be a section on specific requirements that will bring mention to potential interfacing concerns as well as performance metrics and standards compliance. The document is designed to follow the IEEE Software Engineering Standards [1].

2. Overall Description  
 2.1 Product Perspective  
 a) Indicate if this product is standalone or to be integrated

The core objective as well as the stretch goals all conclude that this product will be an independent platform. For this project, we will explore the potential of A.I. and create the foundation of CDK’s own AI platform.

b) Depict how the software operates within various constraints

The signature detection and driver’s license verification software must support various personal computers such as desktop, laptop, and tablet as an input source. This software will first receive a PDF file of a document as its input. It will then compute the data and give result or feedback to user. With that in mind, the software needs to be robust enough to compute large set of data. Similarly, within our stretch goals, vehicle model detection software must also support various personal computers and be robust enough to evaluate large data. However, it must also support portable devices with camera such as phone and tablet. Software should first take photo from camera then it evaluates the photo and returns model number.

2.2 Product Functions

Main goal

The function of this software is detecting the location of the signature box and determine whether it is signed or unsigned. If document is unsigned: software must provide information of location and quantity of missing components. Also, the program must be capable of making a reasonable guess on whether a signature is sufficiently binding. This entails the AI must be capable of distinguishing between a simple mark in the signature window and an actual signature.

Stretch goal

License validation software: This software must validate whether given photo is driver’s license or not. First, it must validate the format of driver’s license and check legal information such as expiration date.

Vehicle model detection software: Vehicle model detection software must be able to determine the specific model number from appearance of the vehicle. Software will first take photos or video of vehicle taken from different angle. It will determine vehicle model by narrowing down the information. For example, software will search sequentially for make, model, and year.

2.3 User Characteristics

The intended user of the software is vehicle retailers. We can assume a low degree of understanding on artificial intelligence. Therefore, software must be designed as black box model. To accompany this structure the user interface must be agreeable for them. Lastly, the program must not overburden the operator, it will take a single file as input to scan and return a conclusion to the user.

2.4 Constraints

There are few constraints on CDK A.I. data stream project. First, software must be standalone. The future goal of this project is to build CDK’s own platform, so software must be independent. In addition, project must be kept in open source. The software must be made in such a way that, once our team has completed its work, the project be taken over by a new team within CDK. To meet this demand the need for thorough documentation of implementation is vital. This project will also have limits on testing due to the privacy issue surrounding actual consumer data. Since the data that signature detecting and license validation software compute contains personal data, testing data must be done on dummy documents and licenses. The last constraint is that product must function on it’s own without the additional use or maintenance of outside software. Since the users of the product are unlikely to be familiar with the internal workings the interface for them must sufficiently mask the technical workings of the program only provide a clean and simple menu.

2.5 Assumptions and Dependencies

All images provided by the client are sanitized and free of legal conflicts. Cloud based deployment services will be provided by the client. From a developing and testing standpoint, this project will be developped in C++ utilizing the OpenCV picture formatting library. This library assumes that the system is running GCC 4.4.x or later and CMake 2.8.7 or higher.

3. Specific Requirements

The program must take as a input a picture or a collection of pictures and be able to gather from it the following information:

- A percentage based estimation as to whether the document can be signed; that is it contains a signable space lacking a signature.

- A percentage based estimation as to whether the document has been signed.

- A percentage based estimation as to whether the document is lacking a signature.

These are the main priorities outlined by our client. Upon completion, we have been given a set of stretch goals consisting of gathering data pertaining to the following:

- A percentage based estimation on whether the picture is of a car or not.

- A percentage based estimation on whether the car is damaged.

- The identifying and parsing of legal personal information contained within a document.

- License information

- Permit information

- The pin pointing and reading of a license plate within the picture of any vehicle.   
  
3. Specific requirements

3.1 External interface requirements

The main interface design principle of this software are intuitivity and simplicity. Since potential user of the software are people who knows little or non about machine intelligence, software must be easy to use without knowing any of the background information.

3.1.1 User interfaces

The signature detection software interface window is consist with four sections which are top menu bar, pop-up window, detection window, and feedback window. <Figure 1> shows the interface structure of signature detection software.

Following describes the functions of the each section:

1. Top Menu Bar: On top menu bar, there are five button that has different functions. From the left side, load file, save progress, start detection, stop detection, continue detection buttons are located. Load file button loads up the file that needs to be processed. Save progress button allow user to save the progress of the current work and run it later. When user press either load or save button shows pop-up windows for load and save file on <Figure 2>. Detection buttons allow user to start, stop, and continue the detection.
2. Pop-up Window: Pop-up window will be shown whenever error is detected or load or save file button is pressed. This window will show at the middle of the main window to either alarm error message or sub window that gives proper interaction tool.
3. Detection Window: This window displays current working document. Software scan through the given document to find signature box. Orange box on <Figure 1> denotes where the software is currently looking at, so user can get sense of how much work is done. It also highlights the location of missing signature or incorrect with red box <Figure 1>. This box will be shown whenever software locate missing or incorrect signature.
4. Information Window: Information window shows more detail information about each working documents. It displays important information such as the number of missing signature, the number of incorrect signature, presence of co-sign box, the number of pages finished detection, and the number of pages on the queue.



3.1.2 Hardware interfaces

This project does not include building any specific hardware. Therefore, there is no hardware interface for this software.

3.1.3 Software interfaces

The software will be built utilizing a linux installation of the OpenCV image processing framework. In order to ensure the program is portable, the project will utilize the Docker container software. Cloud deployment will be handled by the Amazon Web Services platform.

3.1.4 Communications interfaces

For this project, all softwares are independent. In the future, project will be developed into building a platform, and each program needs to be communicate each other to improve the performance. However, in this project, building individual software is focused.

3.2 System features

3.2.1 Signature detecting capability

3.2.1.1 Introduction/Purpose of feature

The program must be able to detect if a document contains a signature regardless of the format of the form.

3.2.1.2 Stimulus/Response sequence

The program will parse a PDF image of a document, or a document in PDF format, and output a percentage based estimation as to whether it contains a signature.

3.2.1.3 Associated functional requirements

3.2.1.3.1 Functional requirement 1 . . .

3.2.1.3.n Functional requirement n

3.2.2 System feature 2 . . .

3.2.m System feature m . . .

3.3 Performance requirements

3.4 Design constraints

3.5 Software system attributes

3.6 Other requirements

Appendix  
  
Index