Software Requirements Specification for ProgName: OAR - Optical Alphabet Recognition

Hunter Ceranic

February 2, 2024

Contents

Revision History

Date	Version	Notes
February 2, 2024	1.0	Initial Revision

[This template is intended for use by CAS 741. For CAS 741 the template should be used exactly as given, except the Reflection Appendix can be deleted. For the capstone course it is a source of ideas, but shouldn't be followed exactly. The exception is the reflection appendix. All capstone SRS documents should have a refelection appendix. —TPLT]

1 Reference Material

This section records information for easy reference.

1.1 Table of Units

Throughout this document SI (Système International d'Unités) is employed as the unit system. In addition to the basic units, several derived units are used as described below. For each unit, the symbol is given followed by a description of the unit and the SI name.

symbol	**unit**	**SI**
px	pixel	picture element

[Only include the units that your SRS actually uses. —TPLT]

[Derived units, like newtons, pascal, etc, should show their derivation (the units they are derived from) if their constituent units are in the table of units (that is, if the units they are derived from are used in the document). For instance, the derivation of pascals as is shown if newtons and m are both in the table. The derivations of newtons would not be shown if kg and s are not both in the table. —TPLT]

[The symbol for units named after people use capital letters, but the name of the unit itself uses lower case. For instance, pascals use the symbol Pa, watts use the symbol W, teslas use the symbol T, newtons use the symbol N, etc. The one exception to this is degree Celsius. Details on writing metric units can be found on the NIST web-page. —TPLT]

1.2 Table of Symbols

The table that follows summarizes the symbols used in this document along with their units. The choice of symbols was made to be consistent with the heat transfer literature and with existing documentation for solar water heating systems. The symbols are listed in alphabetical order.

symbol	**unit**	**description**
x	Pixels	Input image matrix
y	Character	Actual Label of Output

[Use your problems actual symbols. The si package is a good idea to use for units. —TPLT]

1.3 Abbreviations and Acronyms

symbol	description
2D	Two Dimensional
A	Assumption
DD	Data Definition
GD	General Definition
GS	Goal Statement
IM	Instance Model
LC	Likely Change
PS	Physical System Description
R	Requirement
SRS	Software Requirements Specification
TM	Theoretical Model
OAR	Optical Alphabet Recognition

[Add any other abbreviations or acronyms that you add —TPLT]

1.4 Mathematical Notation

[This section is optional, but should be included for projects that make use of notation to convey mathematical information. For instance, if typographic conventions (like bold face font) are used to distinguish matrices, this should be stated here. If symbols are used to show mathematical operations, these should be summarized here. In some cases the easiest way to summarize the notation is to point to a text or other source that explains the notation. —TPLT]

[This section was added to the template because some students use very domain specific notation. This notation will not be readily understandable to people outside of your domain. It should be explained. —TPLT]

[This SRS template is based on Smith and Lai (2005); Smith et al. (2007); Smith and Koothoor (2016). It will get you started. You should not modify the section headings, without first discussing the change with the course instructor. Modification means you are not following the template, which loses some of the advantage of a template, especially standardization. Although the bits shown below do not include type information, you may need to add this information for your problem. If you are unsure, please can ask the instructor. —TPLT]

[Feel free to change the appearance of the report by modifying the LaTeX commands. —TPLT]

[This template document assumes that a single program is being documented. If you are documenting a family of models, you should start with a commonality analysis. A separate template is provided for this. For program families you should look at Smith (2006); Smith et al. (2017). Single family member programs are often programs based on a single physical model. General purpose tools are usually documented as a family. Families of physical models also come up. —TPLT]

[The SRS is not generally written, or read, sequentially. The SRS is a reference document. It is generally read in an ad hoc order, as the need arises. For writing an SRS, and for reading one for the first time, the suggested order of sections is:

- Goal Statement
- Instance Models
- Requirements
- Introduction
- Specific System Description

—TPLT]

[Guiding principles for the SRS document:

- Do not repeat the same information at the same abstraction level. If information is repeated, the repetition should be at a different abstraction level. For instance, there will be overlap between the scope section and the assumptions, but the scope section will not go into as much detail as the assumptions section.

—TPLT]

[The template description comments should be disabled before submitting this document for grading. —TPLT]

[You can borrow any wording from the text given in the template. It is part of the template, and not considered an instance of academic integrity. Of course, you need to cite the source of the template. —TPLT]

[When the documentation is done, it should be possible to trace back to the source of every piece of information. Some information will come from external sources, like terminology. Other information will be derived, like General Definitions. —TPLT]

[An SRS document should have the following qualities: unambiguous, consistent, complete, validatable, abstract and traceable. —TPLT]

[The overall goal of the SRS is that someone that meets the Characteristics of the Intended Reader (Section 2.3) can learn, understand and verify the captured domain knowledge. They should not have to trust the authors of the SRS on any statements. They should be able to independently verify/derive every statement made. —TPLT]

2 Introduction

Images often contain important information to be used in many different modern applications. One such type of information that can be contained within images are readable characters, and the motivation for the Optical Alphabet Recognition project is to create a system from scratch, that can classify upper-case English letter characters characters accurately. More information about the problem itself can be found within the Problem Statement document. Furthermore, a foundational goal of this project is to be useful for educating people about the fundamentals of image classification.

The following section is an overview of the Software Requirement Specifications for the OAR project, specifically outlining the purpose of the document, the scope of the requirements, characteristics of the intended reader, and the organization of the document.

2.1 Purpose of Document

The purpose of this SRS document is to establish and clearly communicate the requirements, limitations, definitions, and models, in regards to the OAR project. This document will also serve as basis for reference for the rest of the documents supporting the project.

2.2 Scope of Requirements

The scope for this project is constrained to the recognition of images of handwritten and printed capital-letter English alphabet characters, in their correct orientations. This does not include hand-written cursive letters. See the assumptions section (Section 4.2.1) for further details.

2.3 Characteristics of Intended Reader

Readers of this document are intended to have a basic level understanding of 2D image processing techniques (equivalent to a standard undergraduate course on machine learning). However, readers with a prerequisite of at least level one university mathematics knowledge (specifically including matrix linear algebra) will be able to understand and follow most of the concepts in the document, as the project is also intended to be a useful tools for learning about image processing.

2.4 Organization of Document

This SRS document contains an introduction to the problem being addressed by the software, and the overarching goals of the project. It is to be used as reference for readers based on the SRS template by Smith and Lai(2005); Smith et al.(2007). The scope, terminology, definitions and models used in the project are outlined, and more details about the specifics of the problem and the explored solution are presented. Requirements, potential future changes and traceability information are also defined.

3 General System Description

This section provides general information about the system. It identifies the interfaces between the system and its environment, describes the user characteristics and lists the system constraints.

3.1 System Context

[Your system context will include a figure that shows the abstract view of the software. Often in a scientific context, the program can be viewed abstractly following the design pattern of Inputs $\rightarrow$ Calculations $\rightarrow$ Outputs. The system context will therefore often follow this pattern. The user provides inputs, the system does the calculations, and then provides the outputs to the user. The figure should not show all of the inputs, just an abstract view of the main categories of inputs (like material properties, geometry, etc.). Likewise, the outputs should be presented from an abstract point of view. In some cases the diagram will show other external entities, besides the user. For instance, when the software product is a library, the user will be another software program, not an actual end user. If there are system constraints that the software must work with external libraries, these libraries can also be shown on the System Context diagram. They should only be named with a specific library name if this is required by the system constraint. —TPLT]

- User Responsibilities:
  - Provide an image of a character to the program in the proper orientation.
  - Be able to interact with the software program using a keyboard, computer mouse, and monitor.
  - Run the software on a system with the capabilities to sufficently process computer graphics.
- OAR Responsibilities:
  - Provide the ability to load or change image files to be classified.
  - Detect and warn about invalid or corrupt image files, upon input.
  - Display the confidence in the result of classified characters.

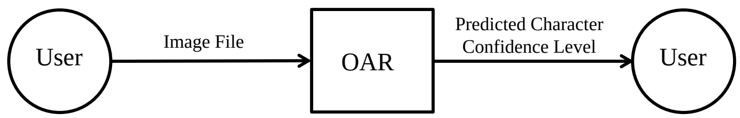


Figure 1: System Context

– Provide a user interface which is able to take user input at anytime, with minimal down time where the program is unresponsive.

3.2 User Characteristics

The end user of the OAR software is anyone who can use character recognition software or wants to learn about it. This may include the intended readers of this document (as described in Section 2.3), however none of those reader requirements are neccessary to use the software.

3.3 System Constraints

The only real world design constraint for this project is that all input images must have the character in their proper legible orientation.

4 Specific System Description

This section first presents the problem description, which gives a high-level view of the problem to be solved. This is followed by the solution characteristics specification, which presents the assumptions, theories, definitions and finally the instance models.

4.1 Problem Description

OAR is intended to classify the characters in images according to their corresponding English language alphabet characters. It should be able to do this with a high level of confidence that the classifcation is correct.

4.1.1 Terminology and Definitions

[This section is expressed in words, not with equations. It provide the meaning of the different words and phrases used in the domain of the problem. The terminology is used to introduce concepts from the world outside of the mathematical model The terminology provides a real world connection to give the mathematical model meaning. —TPLT]

This subsection provides a list of terms that are used in the subsequent sections and their meaning, with the purpose of reducing ambiguity and making it easier to correctly understand the requirements:

•

4.1.2 Physical System Description

[The purpose of this section is to clearly and unambiguously state the physical system that is to be modelled. Effective problem solving requires a logical and organized approach. The statements on the physical system to be studied should cover enough information to solve the problem. The physical description involves element identification, where elements are defined as independent and separable items of the physical system. Some example elements include acceleration due to gravity, the mass of an object, and the size and shape of an object. Each element should be identified and labelled, with their interesting properties specified clearly. The physical description can also include interactions of the elements, such as the following: i) the interactions between the elements and their physical environment; ii) the interactions between elements; and, iii) the initial or boundary conditions. —TPLT]

The physical system of ProgName, as shown in Figure ?, includes the following elements:

PS1:

PS2: ...

[A figure here makes sense for most SRS documents —TPLT]

4.1.3 Goal Statements

[The goal statements refine the "Problem Description" (Section 4.1). A goal is a functional objective the system under consideration should achieve. Goals provide criteria for sufficient completeness of a requirements specification and for requirements pertinence. Goals will be refined in Section "Instanced Models" (Section 4.2.6). Large and complex goals should be

decomposed into smaller sub-goals. The goals are written abstractly, with a minimal amount of technical language. They should be understandable by non-domain experts. —TPLT]
Given an image, the goal statements are:

GS1: Reprocess the image such that classifying calculations can be performed on the image.

GS2: Calculate and display the predicted corresponding character label for the unknown character in the image.

GS3: Calculate and display the confidence level that the predicted label is correct.

4.2 Solution Characteristics Specification

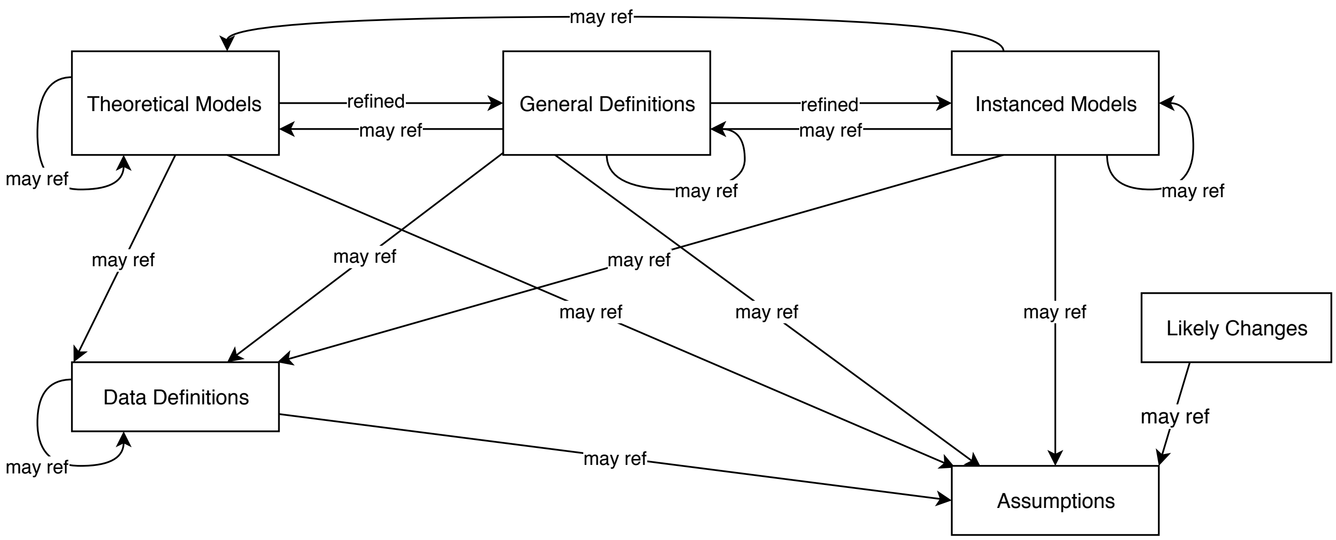
This section provides the assumptions, theoretical models, instance models, general definitions, and data constraints. The information in this section is intended to reduce ambiguity about the project and to present the problem in clear mathethematical or logical terms.

[This section presents the solution characteristics by successively refining models. It starts with the abstract/general Theoretical Models (TMs) and refines them to the concrete/specific Instance Models (IMs). If necessary there are intermediate refinements to General Definitions (GDs). All of these refinements can potentially use Assumptions (A) and Data Definitions (DD). TMs are refined to create new models, that are called GMs or IMs. DDs are not refined; they are just used. GDs and IMs are derived, or refined, from other models. DDs are not derived; they are just given. TMs are also just given, but they are refined, not used. If a potential DD includes a derivation, then that means it is refining other models, which would make it a GD or an IM. —TPLT]

[The above makes a distinction between "refined" and "used." A model is refined to another model if it is changed by the refinement. When we change a general 3D equation to a 2D equation, we are making a refinement, by applying the assumption that the third dimension does not matter. If we use a definition, like the definition of density, we aren't refining, or changing that definition, we are just using it. —TPLT]

[The same information can be a TM in one problem and a DD in another. It is about how the information is used. In one problem the definition of acceleration can be a TM, in another it would be a DD. —TPLT]

[There is repetition between the information given in the different chunks (TM, GDs etc) with other information in the document. For instance, the meaning of the symbols, the units etc are repeated. This is so that the chunks can stand on their own when being read by a reviewer/user. It also facilitates reuse of the models in a different context. —TPLT]
[The relationships between the parts of the document are show in the following figure. In this diagram "may ref" has the same role as "uses" above. The figure adds "Likely Changes," which are able to reference (use) Assumptions. —TPLT]



The instance models that govern ProgName are presented in Subsection 4.2.6. The information to understand the meaning of the instance models and their derivation is also presented, so that the instance models can be verified.

4.2.1 Assumptions

[The assumptions are a refinement of the scope. The scope is general, where the assumptions are specific. All assumptions should be listed, even those that domain experts know so well that they are rarely (if ever) written down. —TPLT] [The document should not take for granted that the reader knows which assumptions have been made. In the case of unusual assumptions, it is recommended that the documentation either include, or point to, an explanation and justification for the assumption. —TPLT] [If it helps with the organization and understandability, the assumptions can be presented as sub sections. The following subsections are options: background theory assumptions, helper theory assumptions, generic theory assumptions, problem specific assumptions, and rationale assumptions —TPLT]

This section simplifies the original problem and helps in developing the theoretical model by filling in the missing information for the physical system. The numbers given in the square brackets refer to the theoretical model [TM], general definition [GD], data definition [DD], instance model [IM], or likely change [LC], in which the respective assumption is used.

A1: [Short description of each assumption. Each assumption should have a meaningful label. Use cross-references to identify the appropriate traceability to TM, GD, DD etc., using commands like dref, ddref etc. Each assumption should be atomic - that is, there should not be an explicit (or implicit) "and" in the text of an assumption. —TPLT]

4.2.2 Theoretical Models

[Theoretical models are sets of abstract mathematical equations or axioms for solving the problem described in Section "Physical System Description" (Section 4.1.2). Examples of theoretical models are physical laws, constitutive equations, relevant conversion factors, etc. —TPLT]

[Optionally the theory section could be divided into subsections to provide more structure and improve understandability and reusability. Potential subsections include the following: Context theories, background theories, helper theories, generic theories, problem specific theories, final theories and rationale theories. —TPLT]

This section focuses on the general equations and laws that ProgName is based on. [Modify the examples below for your problem, and add additional models as appropriate. —TPLT]

RefName: **TM:COE**

Label: Conservation of thermal energy

Equation: $-\nabla \cdot \mathbf{q} + g = \rho C \frac{\partial T}{\partial t}$

Description: and ∇ is the gradient operator. For this equation to apply, other forms of energy, such as mechanical energy, are assumed to be negligible in the system (A**??**). In general, the material properties (ρ and C) depend on temperature.

Notes: None.

Source: http://www.efunda.com/formulae/heat_transfer/conduction/overview_cond.cfm

Ref. By: GD**??**

Preconditions for TM:COE: None

Derivation for TM:COE: Not Applicable

[“Ref. By” is used repeatedly with the different types of information. This stands for Referenced By. It means that the models, definitions and assumptions listed reference the current model, definition or assumption. This information is given for traceability. Ref. By provides a pointer in the opposite direction to what we commonly do. You still need to have a reference in the other direction pointing to the current model, definition or assumption. As an example, if TM1 is referenced by GD2, that means that GD2 will explicitly include a reference to TM1. —TPLT]

4.2.3 General Definitions

[General Definitions (GDs) are a refinement of one or more TMs, and/or of other GDs. The GDs are less abstract than the TMs. Generally the reduction in abstraction is possible through invoking (using/referencing) Assumptions. For instance, the TM could be Newton’s Law of Cooling stated abstracting. The GD could take the general law and apply it to get a 1D equation. —TPLT]

This section collects the laws and equations that will be used in building the instance models.

[Some projects may not have any content for this section, but the section heading should be kept. —TPLT] [Modify the examples below for your problem, and add additional definitions as appropriate. —TPLT]

Number	GD1
Label	**Newton's law of cooling**
SI Units	Something
Equation	$q(t) = h\Delta T(t)$
Description	Newton's law of cooling describes convective cooling from a surface. The law is stated as: the rate of heat loss from a body is proportional to the difference in temperatures between the body and its surroundings. $q(t)$ is the thermal flux. h is the heat transfer coefficient, assumed independent of T (A**??**) . $\Delta T(t)$= $T(t) - T_{\text{env}}(t)$ is the time-dependent thermal gradient between the environment and the object.
Source	Citation here
Ref. By	DD1, DD**??**

Detailed derivation of simplified rate of change of temperature

[This may be necessary when the necessary information does not fit in the description field. —TPLT] [Derivations are important for justifying a given GD. You want it to be clear where the equation came from. —TPLT]

4.2.4 Data Definitions

[The Data Definitions are definitions of symbols and equations that are given for the problem. They are not derived; they are simply used by other models. For instance, if a problem depends on density, there may be a data definition for the equation defining density. The DDs are given information that you can use in your other modules. —TPLT]

[All Data Definitions should be used (referenced) by at least one other model. —TPLT]

This section collects and defines all the data needed to build the instance models. The dimension of each quantity is also given. [Modify the examples below for your problem, and add additional definitions as appropriate. —TPLT]

Number	DD1
Label	**Heat flux out of coil**
Symbol	q_C
SI Units	Something
Equation	$q_C(t) = h_C(T_C - T_W(t))$, over area A_C
Description	T_C is the temperature of the coil. T_W is the temperature of the water. The heat flux out of the coil, q_C, is found by assuming that Newton's Law of Cooling applies (A**??**). This law (GD1) is used on the surface of the coil, which has area A_C and heat transfer coefficient h_C. This equation assumes that the temperature of the coil is constant over time (A**??**) and that it does not vary along the length of the coil (A**??**).
Sources	Citation here
Ref. By	IM1

4.2.5 Data Types

[This section is optional. In many scientific computing programs it isn't necessary, since the inputs and outpus are straightforward types, like reals, integers, and sequences of reals and integers. However, for some problems it is very helpful to capture the type information. —TPLT]

[The data types are not derived; they are simply stated and used by other models. —TPLT]

[All data types must be used by at least one of the models. —TPLT]

[For the mathematical notation for expressing types, the recommendation is to use the notation of Hoffman and Strooper (1995). —TPLT]

This section collects and defines all the data types needed to document the models. [Modify the examples below for your problem, and add additional definitions as appropriate. —TPLT]

Type Name	Name for Type
Type Def	mathematical definition of the type
Description	description here
Sources	Citation here, if the type is borrowed from another source

4.2.6 Instance Models

[The motivation for this section is to reduce the problem defined in "Physical System Description" (Section 4.1.2) to one expressed in mathematical terms. The IMs are built by refining the TMs and/or GDs. This section should remain abstract. The SRS should specify the requirements without considering the implementation. —TPLT]

This section transforms the problem defined in Section 4.1 into one which is expressed in mathematical terms. It uses concrete symbols defined in Section 4.2.4 to replace the abstract symbols in the models identified in Sections 4.2.2 and 4.2.3.

The goals [reference your goals —TPLT] are solved by [reference your instance models —TPLT]. [other details, with cross-references where appropriate. —TPLT] [Modify the examples below for your problem, and add additional models as appropriate. —TPLT]

Number	IM1
Label	**Energy balance on water to find T_W**
Input	m_W, C_W, h_C, A_C, h_P, A_P, t_{final}, T_C, T_{init}, $T_P(t)$ from IM**??** The input is constrained so that $T_{\text{init}} \leq T_C$ (A**??**)
Output	$T_W(t)$, $0 \leq t \leq t_{\text{final}}$, such that $\frac{dT_W}{dt} = \frac{1}{\tau_W}[(T_C - T_W(t)) + \eta(T_P(t) - T_W(t))]$, $T_W(0) = T_P(0) = T_{\text{init}}$ (A**??**) and $T_P(t)$ from IM**??**
Description	T_W is the water temperature. T_P is the PCM temperature. T_C is the coil temperature. $\tau_W = \frac{m_W C_W}{h_C A_C}$ is a constant. $\eta = \frac{h_P A_P}{h_C A_C}$ is a constant (dimensionless). The above equation applies as long as the water is in liquid form, $0 < T_W < 100^o$C, where 0^oC and 100^oC are the melting and boiling points of water, respectively (A**??**, A**??**).
Sources	Citation here
Ref. By	IM**??**

Derivation of ...

[The derivation shows how the IM is derived from the TMs/GDs. In cases where the derivation cannot be described under the Description field, it will be necessary to include this subsection. —TPLT]

4.2.7 Input Data Constraints

Table 1 shows the data constraints on the input output variables. The column for physical constraints gives the physical limitations on the range of values that can be taken by the variable. The column for software constraints restricts the range of inputs to reasonable values. The software constraints will be helpful in the design stage for picking suitable algorithms. The constraints are conservative, to give the user of the model the flexibility to experiment with unusual situations. The column of typical values is intended to provide a feel for a common scenario. The uncertainty column provides an estimate of the confidence with which the physical quantities can be measured. This information would be part of the input if one were performing an uncertainty quantification exercise.

The specification parameters in Table 1 are listed in Table 2.

Table 1: Input Variables

Var	Physical Constraints	Software Constraints	Typical Value	Uncertainty
L	$L > 0$	$L_{\text{min}} \leq L \leq L_{\text{max}}$	1.5 m	10%

(*) [you might need to add some notes or clarifications —TPLT]

Table 2: Specification Parameter Values

Var	Value
L_{min}	0.1 m

4.2.8 Properties of a Correct Solution

A correct solution must come in the form of one of the 26 possible labels, {A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,
as well as an associated confidence level, as a percentage, that the label assigned is correct.

5 Requirements

[The requirements refine the goal statement. They will make heavy use of references to the instance models. —TPLT]

This section provides the functional requirements, the business tasks that the software is expected to complete, and the nonfunctional requirements, the qualities that the software is expected to exhibit.

5.1 Functional Requirements

R1: Accept an input image in the following formats:

- PNG (Portable Net Graphics)
- JPEG (Joint Photographic Experts Group)
- BMP (Bitmap)
- preloaded training images

R2: Process the image such that the calculations needed to be performed by the program can be executed.

R3: Calculate and display the classified label for the input image.

R4: Calculate and display the confidence level that the assigned label is correct.

[Every IM should map to at least one requirement, but not every requirement has to map to a corresponding IM. —TPLT]

5.2 Nonfunctional Requirements

NFR1: **Accuracy** The accuracy of the computed label confidence should be consistent enough that the results could be considered comparable to other existing solutions for this problem. As such the confidence level output has to be calculated in a similar way to existing solutions so that the output can be verified and compared.

NFR2: **Usability** The user should be able to intuitively use the software and understand the output that is displayed. To accomplish this the software should be user-friendly, simple, and require little setup. A user survey may be conducted to verify the software's percieved usability.

NFR3: **Maintainability** The code should be clear and understandable to make further development by other programmers possible with little hassle, as well as to help educate newer programmers as to how image classification works. Adding further features, or understanding how the software presently works should not be a difficult task.

NFR4: **Portability** The software should be cross-platform (Windows, Linux, MacOS) with little to no setup required. This can be in the the form of an executable python-based application.

5.3 Rationale

[Provide a rationale for the decisions made in the documentation. Rationale should be provided for scope decisions, modelling decisions, assumptions and typical values. —TPLT]

6 Likely Changes

The changes listed below are likely to be implemented as the software is developed further.

LC1: Expand the number of classification labels to include lower-case English letter alphabet, numerical, and punctuation characters.

LC2: Allow the user to specify whether the image is being input from a camera or from a pre-existing file on their system.

7 Unlikely Changes

LC3: Expand the functionality of the program to classify whole English words and sentences instead of just individual characters.

LC4: Provide the choice of different image classifciation algorithms.

8 Traceability Matrices and Graphs

The purpose of the traceability matrices is to provide easy references on what has to be additionally modified if a certain component is changed. Every time a component is changed, the items in the column of that component that are marked with an "X" may have to be modified as well. Table 3 shows the dependencies of theoretical models, general definitions, data definitions, and instance models with each other. Table 4 shows the dependencies of instance models, requirements, and data constraints on each other. Table 5 shows the dependencies of theoretical models, general definitions, data definitions, instance models, and likely changes on the assumptions.

[You will have to modify these tables for your problem. —TPLT]

[The traceability matrix is not generally symmetric. If GD1 uses A1, that means that GD1's derivation or presentation requires invocation of A1. A1 does not use GD1. A1 is "used by" GD1. —TPLT]

[The traceability matrix is challenging to maintain manually. Please do your best. In the future tools (like Drasil) will make this much easier. —TPLT]

	TM??	TM??	TM??	GD1	GD??	DD1	DD??	DD??	DD??	IM1	IM??	IM??
TM??												
TM??			X									
TM??												
GD1												
GD??	X											
DD1				X								
DD??				X								
DD??												
DD??								X				
IM1					X	X	X				X	
IM??					X		X		X	X		
IM??		X										
IM??		X	X				X	X	X		X	

Table 3: Traceability Matrix Showing the Connections Between Items of Different Sections

	IM1	IM??	IM??	IM??	4.2.7	R??	R??
IM1		X				X	X
IM??	X			X		X	X
IM??						X	X
IM??		X				X	X
R??							
R??						X	
R??					X		
R2	X	X				X	X
R??	X						
R??		X					
R??			X				
R??				X			
R4			X	X			
R??		X					
R??		X					

Table 4: Traceability Matrix Showing the Connections Between Requirements and Instance Models

	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??
TM??	X																		
TM??																			
TM??																			
GD1		X																	
GD??			X	X	X	X													
DD1							X	X	X										
DD??			X	X						X									
DD??																			
DD??																			
IM1											X	X		X	X	X			X
IM??												X	X			X	X	X	
IM??														X					X
IM??													X					X	
LC??				X															
LC??								X											
LC??									X										
LC??											X								
LC??												X							
LC??															X				

Table 5: Traceability Matrix Showing the Connections Between Assumptions and Other Items

The purpose of the traceability graphs is also to provide easy references on what has to be additionally modified if a certain component is changed. The arrows in the graphs represent dependencies. The component at the tail of an arrow is depended on by the component at the head of that arrow. Therefore, if a component is changed, the components that it points to should also be changed. Figure **??** shows the dependencies of theoretical models, general definitions, data definitions, instance models, likely changes, and assumptions on each other. Figure **??** shows the dependencies of instance models, requirements, and data constraints on each other.

9 Values of Auxiliary Constants

There are no auxiliary constants defined so this section is not applicable for this project.

References

Daniel M. Hoffman and Paul A. Strooper. *Software Design, Automated Testing, and Maintenance: A Practical Approach*. International Thomson Computer Press, New York, NY, USA, 1995. URL http://citeseer.ist.psu.edu/428727.html.

W. Spencer Smith. Systematic development of requirements documentation for general purpose scientific computing software. In *Proceedings of the 14th IEEE International Requirements Engineering Conference, RE 2006*, pages 209–218, Minneapolis / St. Paul, Minnesota, 2006. URL http://www.ifi.unizh.ch/req/events/RE06/.

W. Spencer Smith and Nirmitha Koothoor. A document-driven method for certifying scientific computing software for use in nuclear safety analysis. *Nuclear Engineering and Technology*, 48(2):404–418, April 2016. ISSN 1738-5733. doi: http://dx.doi.org/10.1016/j.net.2015.11.008. URL http://www.sciencedirect.com/science/article/pii/S1738573315002582.

W. Spencer Smith and Lei Lai. A new requirements template for scientific computing. In J. Ralyté, P. Ȧgerfalk, and N. Kraiem, editors, *Proceedings of the First International Workshop on Situational Requirements Engineering Processes – Methods, Techniques and Tools to Support Situation-Specific Requirements Engineering Processes, SREP'05*, pages 107–121, Paris, France, 2005. In conjunction with 13th IEEE International Requirements Engineering Conference.

W. Spencer Smith, Lei Lai, and Ridha Khedri. Requirements analysis for engineering computation: A systematic approach for improving software reliability. *Reliable Computing, Special Issue on Reliable Engineering Computation*, 13(1):83–107, February 2007.

W. Spencer Smith, John McCutchan, and Jacques Carette. Commonality analysis for a family of material models. Technical Report CAS-17-01-SS, McMaster University, Department of Computing and Software, 2017.

[The following is not part of the template, just some things to consider when filing in the template. —TPLT]
[Grammar, flow and LaTeXadvice:

- For Mac users `*.DS_Store` should be in `.gitignore`
- LaTeX and formatting rules
  - Variables are italic, everything else not, includes subscripts (link to document)
    - Conventions
    - Watch out for implied multiplication
  - Use BibTeX
  - Use cross-referencing
- Grammar and writing rules
  - Acronyms expanded on first usage (not just in table of acronyms)
  - "In order to" should be "to"

—TPLT]
[Advice on using the template:

- Difference between physical and software constraints
- Properties of a correct solution means *additional* properties, not a restating of the requirements (may be "not applicable" for your problem). If you have a table of output constraints, then these are properties of a correct solution.
- Assumptions have to be invoked somewhere
- "Referenced by" implies that there is an explicit reference
- Think of traceability matrix, list of assumption invocations and list of reference by fields as automatically generatable
- If you say the format of the output (plot, table etc), then your requirement could be more abstract

—TPLT]