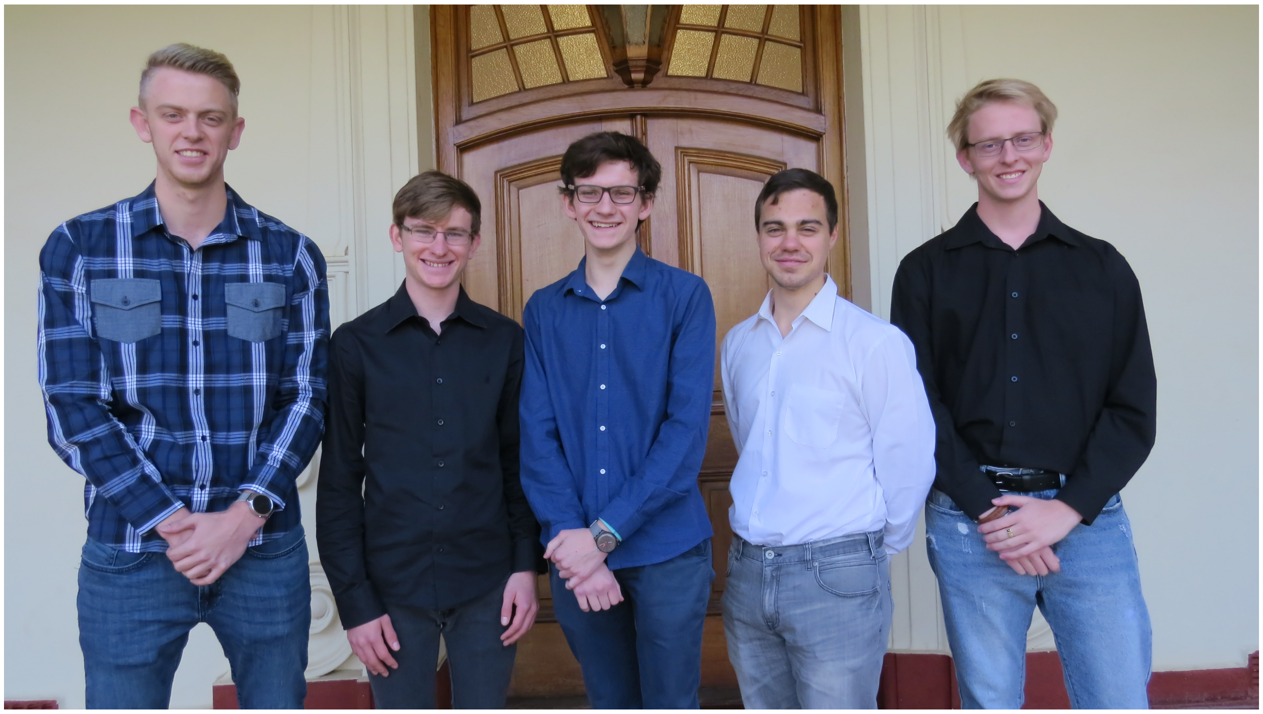
UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

System Requirements Specification

Team: Java the Hutts

2017



Nicolai van Niekerk
`nicvaniek@gmail.com`

Marno Hermann
`marno@barnton-consulting.co.za`

Stephan Nell
`nellstephanj@gmail.com`

Jan-Justin van Tonder
`J.vanTonder@tuks.co.za`

Andreas Nel
`nel.andreas1@gmail.com`

Contents

1 Introduction

This chapter aims to give a description, as well as an overview, of the content of this document. Additionally, it will include any terms, abbreviations, acronyms and references used throughout this document.

1.1 Purpose

The purpose of this document is to present the reader with a detailed description of the Electronic ID Verification system. It will delve into the purpose and features of the system, the various interfaces of the system, the capabilities of the system, as well as the constraints under which the system must operate. The content of this document is intended for both the various stakeholders and the developers of the Electronic ID Verification system.

1.2 Scope

The Electronic ID Verification is a proposed standalone system that provides the core functionality of extracting client details from an image of some form of ID, and comparing existing client information with that of an image of some form of ID.

1.3 Definitions, Acronyms, and Abbreviations

Term	**Definition**
OCR	Optical Character Recognition
ID	Identification Document
API	A set of functions and procedures that allow the creation of applications which access the features or data of an operating system, application, or other service.
Linux	A Unix-like computer operating system assembled under the model of free and open-source software development and distribution
Python	Python is a widely used high-level programming language for general-purpose programming.

1.4 Overview

The remainder of this document will consist of two remaining chapters.

The second chapter will address the overall description of the Electronic ID Verification system. In addition to this, chapter two will describe the context of the Electronic ID Verification system, its relations and the potential interfaces with other systems. This chapter will also provide a summary of the functions of the Electronic ID Verification system as well as consider the numerous user characteristics, constraints, assumptions and dependencies relevant to the system.

The third chapter serves the purpose of describing the software requirements of the Electronic ID Verification system. This chapter will address the External Interface Requirements, Functional Requirements, Performance Requirements, Design Constraints, Software System Attributes and any other requirements not previously explored.

2 Overall Description

This chapter aims to give an overview of the entire Electronic ID Verification system. The system will be contextualised in order to demonstrate the basic functionality of the system as well as demonstrate how the system interacts with other systems. It will also describe the levels, or types, of users that will utilise the system and describe the functionality that is available to said user. At the end of this chapter, the constraints and assumptions for the system will be addressed.

2.1 Product Perspective

The system will be designed in the form of a Web API which will be hosted on an independent server.

The application will focus on two main criteria. The first is that the application should be able to extract a name, surname, ID number and face, from a photo of either an ID book, ID card, driver's license or passport. The application should then take the information that was extracted and compare it to existing profile and then a percentage match score for each corresponding value should be returned.

When comparing the face for a similarity percentage, the application will deal with problems like an old photo or changed facial features like a beard. By manipulating the photo with different methods to ensure that highest accuracy is ensured when matching with a photo.

The second feature extracts the information from the identification photo and returns the collection of the information that the application extracted.

2.2 User Characteristics

The intent of the Electronic ID Verification system is to function as an Web API, thus the users will require some knowledge in restful programming. Electronic ID Verification is platform independent, therefore user requires no platform specific knowledge.

2.3 Constraints

Only a valid South African ID book, South African ID card, South African Driver license or South African Passport can be used with the application.

People's facial features could change or the photo provided could be very old, or the individual in the photo could be standing skew and not providing a full frontal facial image.

Low DPI, lighting and resolution of images can drastically affect performance.

2.4 Assumptions and Dependencies

- It is assumed that enough resources will be provided to test multiple ID cards, ID books, driver's licenses or passports.
- It is assumed that a clear, well-lit and sufficiently high DPI will be provided photo will always be provided.
- It is assumed that all identification documentation follows the format of documentation issued in South Africa.

3 Specific Requirements

This chapter addresses all the functional requirements of the Electronic ID Verification system. It gives a detailed description of the system and all its features.

3.1 External Interface Requirements

3.1.1 User-interfaces

The system will be implemented as an API that will be called from within the client's exsisting application. As a result there are no user-interface requirements.

3.1.2 Hardware Interfaces

No extra hardware interface is needed, since the application is designed as a API and is completely detached from any hardware requirements.

3.1.3 Software Interfaces

Since the system will be implemented as an API, it will interface with the existing systems of the client.

3.1.4 Communication Interfaces

Communication is an important aspect of an API to function correctly. The application must be able to pass the correct information to any system that makes use of the application functionality. It is also important that descriptive responses are provided in case of an error, or if the need arises to provide extra information.

3.2 Functional Requirements

This section describes the functional requirements of the system. The requirements are derived from the specific use cases that are modelled using use case diagrams. Non-trivial use cases are also further elaborated using Actor-System interaction diagrams.

1. **FR-01:** The system must be able to accept the relevant input data and a(n) ID/Passport/License photo in a specified format.
2. **FR-02:** The system must be able to extract text from the provided image.
3. **FR-03:** The system must be able to extract the photo from the image.
4. **FR-04:** The system must be able to detect a face from the extracted image.
5. **FR-05:** The system must be able to age the detected face to a specified date.
6. **FR-06:** The system must be able to align a face for better matching.
7. **FR-07:** The system must be able to compare the extracted text with provided data.
8. **FR-08:** The system must be able to compare two faces after image processing.
9. **FR-09:** The system must be able to give a percentage match on the validated data.
10. **FR-10:** The system must be able to visually show the differences in images.
11. **FR-11:** The system must be able to perform extraction on a South African ID book.
12. **FR-12:** The system must be able to perform extraction on a South African ID card.
13. **FR-13:** The system must be able to perform extraction on a South African driver's license.

14. **FR-14:** The system must be able to perform extraction on a South African passport.
15. **FR-15:** The system must allow a user to specify a matching accuracy threshold.
16. **FR-16:** The system must be able to remove beards and/or piercings from the detected face.

3.2.1 Use cases

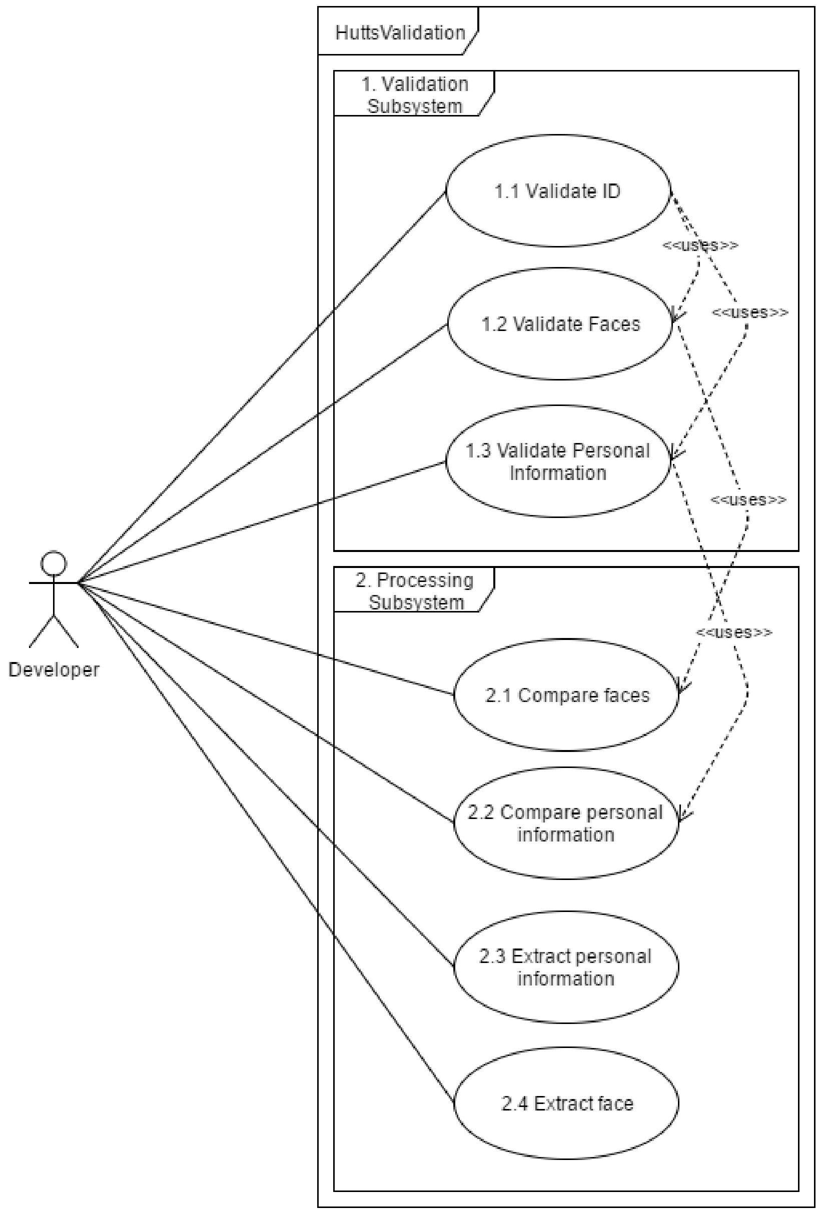


Figure 1: Use Case Diagram for Hutts-Verification

1. Verification Subsystem
   (a) Verify ID
      i. **Description:** The developer must be able to send a picture of an ID and the system should verify the ID against a photo of the individual in question. The personal information of the individual should also be verified.

ii. **Precondition:** Two photos with a high DPI and good proportionate lighting.
iii. **Postcondition:** The system verifies and returns a percentage match.

(b) Verify Faces

i. **Description:** The developer must be able to send a picture of an ID and the system should verify the ID against a photo of the individual in question.
ii. **Precondition:** Two photos with a high DPI and good proportionate lighting.
iii. **Postcondition:** The system verifies and returns a percentage match.

(c) Verify Personal Information

i. **Description:** The developer must be able to send a picture of an ID and the system should verify the ID against provided personal information of the individual in question
ii. **Precondition:** Personal information of the person in question should be provided
iii. **Postcondition:** The system verifies and returns a percentage match.

2. Processing Subsystem

(a) Compare faces

i. **Description:** The system should be able to receive two images containing faces of the person in question and compare them to return a percentage match.
ii. **Precondition:** The faces should be extracted from the images
iii. **Postcondition:** The system compares the faces and returns a percentage match.

(b) Compare personal information

i. **Description:** The system should be able to receive text extracted from the ID and compare it with provided personal information of the person in question.
ii. **Precondition:** The information should be extracted from the photo of the ID.
iii. **Postcondition:** The system compares the text and returns a percentage match.

(c) Extract personal information

i. **Description:** The system should be able to extract the relevant personal information from a photo of the ID document.
ii. **Precondition:** An ID with a high DPI and good proportionate lighting should be provided.
iii. **Postcondition:** The system returns the extracted text.

(d) Extract profile

i. **Description:** The system should be able to detect and extract the face of the person in question from the photo of the ID document.
ii. **Precondition:** AAn ID with a high DPI and good proportionate lighting. should be provided.
iii. **Postcondition:** The system returns the extracted face.

Table 1: Traceability Matrix

		Functional Requirements															
Use Case	**Priority**	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1.1	7	X															
1.2	5	X														X	
1.3	6	X														X	
2.1	3					X	X		X	X	X						
2.2	4							X		X							
2.3	2	X	X									X	X	X	X		
2.4	1	X		X	X							X	X	X	X		X
Requirement Priority		1	5	4	6	10	11	8	7	3	12	2	14	15	16	9	13

3.3 Performance Requirements

- A single extraction request should return in under 100 milliseconds.
- The system should be able to process 4 concurrent extraction requests in under 200 milliseconds.
- Any facial verification request should be handled in 5 seconds.
- Any internal errors and exceptions should be handle by the system itself.

3.4 Design Constraints

- The system will be implemented using Python 3.5 in order to ensure compatibility between the API and the Quant Solutions system.

3.5 Software System Attributes

This section describes all quality related requirements

3.5.1 Reliability

1. The system should not fail when given a clear, well-lit photo of the identification document.
2. The system should always give a percentage match above the specfied threshold if the faces are the same.
3. Error handling should be implemented and the application should be able to handle all errors in a graceful manner by making use of helpful and descriptive error messages.

3.5.2 Security

1. All incoming and outgoing data will be encrypted using the HTTPS protocol due to the sensitive nature of the data collected.

3.5.3 Availability

3.6 Other Requirements

- Documentation should be of such a standard that anyone can easily implement the application when needed.
- The system should indicate that images match only if the images match with an accuracy of at least 75%.

4 Technology Choices

This section describes the technologies in use, and the arguments behind why we decided on these technologies. It was important to us to try and make use of technologies that are, free to use, efficient, relatively small in terms package size and requires as few as possible dependencies.

4.1 OpenCV-Python Version *3.2.0*

OpenCV (Open Source Computer Vision Library) is free for commercial use. OpenCV is by far the best documented and supported Computer Vision library since it has been available and been developed on for over 15 years now. OpenCV is available in many programming language including Python which was one of the requirements of the client. OpenCV now also support Multi-core hardware acceleration which will increase efficiency drastically.

OpenCV will be used for real-time processing of images in the application which will include image clean-up image extraction and image manipulation.

4.2 Dlib *19.4.0*

Dlib is an Open Source Machine learning toolkit and is also the only machine learning library that support facial landmark optimisation and Face likeness detection out of the box.

Dlib allows us to make use of a High Quality pre-trained classifiers for facial likeness which made use of over 3 million faces during training. This allows for a claimed accuracy of 99.38 % in the *Labeled Faces in the Wild* bench mark, thus allowing us to meet all our performance requirements in terms of the face likeness detection sub-section

Dlib also allows for face detection as well as face landmark detection both of which is required to improve the accuracy and reliability of the face likeness algorithm by supporting the alignment and clean-up of detected faces.

4.3 Pytesseract *0.1.7*

Python-tesseract is an Open Source optical character recognition (OCR) tool for python. This allows for reading and decoding of embedded text in images.

Tesseract allows for easy automated scanning of cards. By making use of Tesseract we no longer need to:

1. Detect Region of text.
2. Process Region of text.
3. Use a trained classifier to read the extracted text.

If it is found later to be a requirement Tesseract does support additional training of fonts which will allow for greater reliability and accuracy when extracting text.