User Manual

System: Hutts Verification

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 document serves as a user manual for the Hutts Verification system. It contains information about the product description as well as a detailed explanation of how to use the system. The explanation is split into two (2) parts - an explanation of using the included website to test the system (for general usage) and an explanation of how the API can be integrated with an existing system (for developers).

2 Product Description

Hutts Verification is a WebAPI that serves as an electronic ID verification system. The purpose of the system is to extract, process and validate information from a photo of a South African ID book (new) or ID card. By providing the system with personal information (name, surname, ID number, etc.), a current photo of a person's face and a photo of the chosen form of identification documentation, the system will be able to return a percentage match score for each provided element against the information on the identification documentation. Additionally, the system can also be used to extract data from the identification documentation, including a face, an ID number and fields such as the name and surname of the person whose identification documentation is presented.

The Hutts Verification system consists of two (2) separate parts in order to cater for two (2) types of users - general users and developers. The part for general users is a website where the user can upload images, enter data and change system settings in order to see how it affects the extraction and/or verification process. The second and most important part, which is aimed at developers, is the API of the system. The API is set up in such a way that the developer can easily integrate it into an existing system without changing its architecture or design. The API is set up in such a way that the developer can choose to simply start the provided server and make requests to it, or the developer can choose to manually call the provided methods of the API in a manner of their own choosing.

3 Configuration

The settings of the system can be changed on the web interface.

4 Installation

5 General Usage

TODO: insert screenshots, GIFs (it is a very big possibility) and step by step instructions.

6 For Developers

6.1 Using the Server

This section aims to show the developer how to start the server and how to structure the requests that the server is able to handle. All requests are made using the HTTP POST method, in order to add a level of security to the system.

6.2 Extraction

This section aims to explain the format of the requests that can be made to the server in order to extract data from the identification documentation.

6.2.1 Extract Text

1. **Description:** Extracts textual information from the photo of the ID and returns it.
2. **URL:** */extractText*
3. **Method:** POST
4. **URL Parameters:** None
5. **Data Parameters:**

```
{
   "idPhoto" : <stream|URL|path>
}
```

   Example:

```
{
   "idPhoto" : idDocument.jpg
}
```

6. **Success Response:**
   - **Code:** 200
   - **Content:**

```
{
    "country_of_birth": "RSA",
    "date_of_birth": "72-09-17",
    "identity_number": "7209170838080",
    "names": null,
    "nationality": null,
    "sex": "F",
    "status": "Citizen",
    "surname": null
}
```

7. **Error Response:** TODO
8. **Sample Call:**

```
$.ajax({
    type: "POST",
    url: "http://localhost:5000/extractText",
    data: {
            "idPhoto" : idDocument.jpg
          },
    success: function(data){
        console.log(data);
    }
});
```

6.2.2 Extract Face

1. **Description:** Extracts the face from the photo of the ID and returns it.
2. **URL:** */extractFace*
3. **Method:** POST
4. **URL Parameters:** None
5. **Data Parameters:**

```
{
   "idPhoto" : <stream|URL|path>
}
```

   Example:

```
{
   "idPhoto" : idDocument.jpg
}
```

6. **Success Response:**
   - **Code:** 200
   - **Content:**

```
{
   "extracted_face": "..."
}
```

7. **Error Response:** TODO
8. **Sample Call:**

```
$.ajax({
    type: "POST",
    url: "http://localhost:5000/extractFace",
    data: {
            "idPhoto" : idDocument.jpg
          },
    success: function(data){
        console.log(data);
    }
});
```

6.2.3 Extract All

1. **Description:** Extracts textual information and face from the photo of the ID and returns it.
2. **URL:** */extractAll*
3. **Method:** POST
4. **URL Parameters:** None
5. **Data Parameters:**

```
{
   "idPhoto" : <stream|URL|path>
}
```

Example:

```
{
    "idPhoto" : idDocument.jpg
}
```

6. **Success Response:**
   - **Code:** 200
   - **Content:**

```
{
    "extracted_face": "...",
    "text_extract_result": {
            "country_of_birth": "RSA",
            "date_of_birth": "72-09-17",
            "identity_number": "7209170838080",
            "names": null,
            "nationality": null,
            "sex": "F",
            "status": "Citizen",
            "surname": null
        }
}
```

7. **Error Response:** TODO
8. **Sample Call:**

```
$.ajax({
    type: "POST",
    url: "http://localhost:5000/extractAll",
    data: {
            "idPhoto" : idDocument.jpg
        },
    success: function(data){
        console.log(data);
    }
});
```

6.3 Verification

This section aims to explain the format of the requests that can be made to the server in order to verify the provided data and face with the data on the identification document.

6.3.1 Verify ID

1. **Description:** Verifies the similarity between the text and image on the form of identification and the provided personal information with a photo of the individual's face.
2. **URL:** */verifyID*
3. **Method:** POST
4. **URL Parameters:** None
5. **Data Parameters:**

```
{
    "face_img": <stream|URL|path>,
    "id_img": <stream|URL|path>,
    "surname": <string>,
    "names": <string>,
    "gender": <string>,
    "nationality": <string>,
    "idNumber": <string>,
    "dob": <date>,
    "cob": <string>,
    "status": <string>
}
```

Example:

```
{
    "face_img": profilePic.png,
    "id_img": myId.jpg,
    "surname": "Doe",
    "names": "John Joe",
    "gender": "M",
    "nationality": "South African",
    "idNumber": "9877452008082",
    "dob": "1998-07-06",
    "cob": "RSA",
    "status": "Citizen"
}
```

6. **Success Response:**
   - **Code:** 200
   - **Content:**

```
{
    "face_match": 100.0,
    "is_match": true,
    "is_pass": false,
    "text_match": 0.0,
    "total_match": 60.0
}
```

7. **Error Response:** TODO
8. **Sample Call:**

```
$.ajax({
    type: "POST",
    url: "http://localhost:5000/verifyInfo",
    data: {
            "face_img": profilePic.png,
            "id_img": myId.jpg,
            "surname": "Doe",
            "names": "John Joe",
            "gender": "M",
            "nationality": "South African",
            "idNumber": "9877452008082",
```

```
                "dob": "1998-07-06",
                "cob": "RSA",
                "status": "Citizen"
            },
        success: function(data){
            console.log(data);
        }
});
```