# Face Recognition System

## Requirements Elicitation

*Authors:* Walid Balegh, Jakob Heyder, Sarpreet Singh Buttar, Henry Pap and Oscar Maris

*Semester:* VT 2017

# Contents

# 1 Introduction

## 1.1 Purpose

The purpose of this document is to describe the requirements for a face recognition system which matches a photo and a Swedish personal number by exposing a well defined API. Additionally it will describe the requirements for a sample client user system which uses the API.
The intended audience includes the software developers and the respective clients assessment person.

## 1.2 Scope

The software system to be produced is a Face Recognition System, which will be referred to as ”FRS” through this document.
The Face Recognition System will allow authenticated services to request a personal number with a given photo. FRS could be used in bank-, state systems or other services where identification of a person is critical. The administrative entity could be a state or other high security organization which provides the data used from the FRS. FRS administrators will be able to register, delete, update and list user data. Therefore the FRS is split into a user component and an administrative component, which each needs different authentication and are logically separate from each other.

## 1.3 Definitions, acronyms and abbreviations

- **URM : User Repository Module**
- **ARM : Admin Repository Module**
- **UAM : User Application Module**
- **AAM : Admin Application Module**
- **EFR : External Face Recognition API**
- **FRS : Face Recognition System to be developed**
- **PN : Swedish Personal Number**
- **Repository: Module where the data can be accessed and processed**
- **User-Entity: A User entity consists of a personal number and a picture of the person**

## 1.4 References

N/A

## 1.5 Stakeholders

- **Customer:** LNUs Teaching Team which represents a fictive company which wants the FRS.
- **User :** Verified and registered partner which is authenticated to use the the User API. E.g. Skatteverket or Swedbank.
- **Admin :** Verified and registered partner which provides the trusted information and has full control over the application and data.
- **Developer :** Fully privileged people which develop the system and have access to the Admin-,User interfaces and the source code during development.

### 1.6 Overview

The rest of this document contains an overall description of the Face Recognition System and the constraints (section 2), the specific functional requirements for the system (section 3) and the Use case and scenario modeling for functional requirements (section 4).

# 2 Overall description

## 2.1 Product perspective

In some services such as financial or official matters there is a need to identify a person fast and securely. The FRS can provide this functionality and make the process easier and more reliable.

### 2.1.1 System Interfaces

The FRS depends on an external face recognition API to compare the photos. This service can be developed in house or made use of existing systems. Beside the external dependency it consists of four different modules: the User Application Module (UCM), the User Repository Module (URM), the Admin Repository Module (ARM) and a database. (see figure 1)

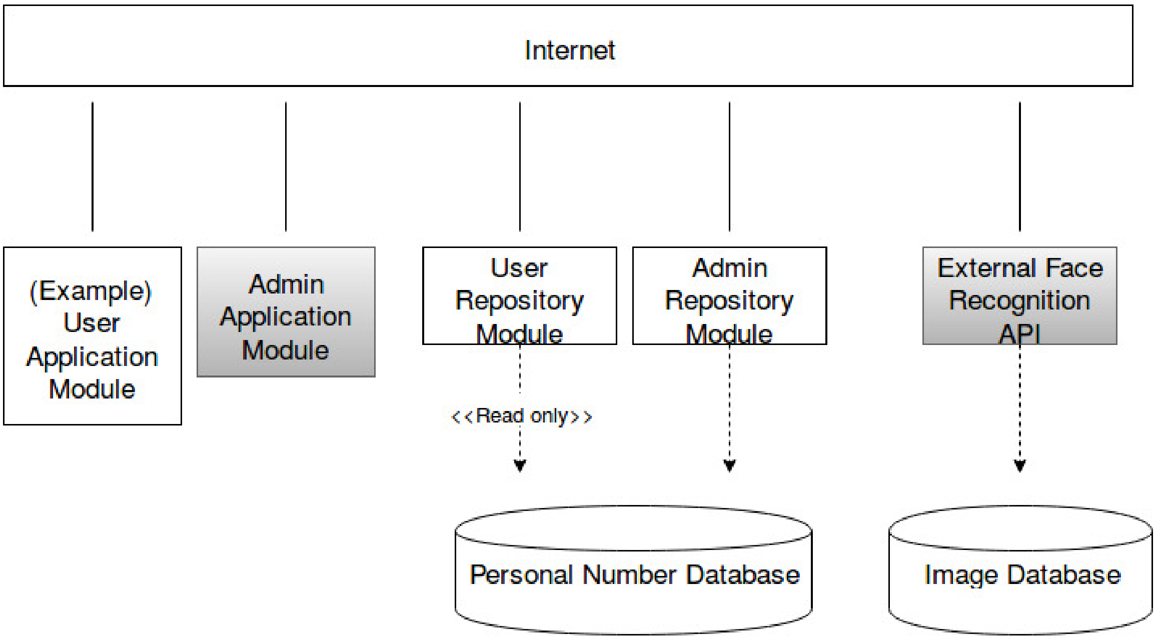


Figure 1: Overview over the system interfaces

The UCM allows User services to log on to the FRS and request a personal number with a given image. The URM is a daemon that accepts connections from the UCM and serves with the requested information. The URM is dependent on the External Face Recogntion API (EFR) to match the given photo to a registered one. The Admin Repository Module serves the API for the ACM to register, delete, update and get user data. It directly interacts with the PN-Database and indirectly uses the EFR to modify the Image-Database.
The User Application Module can be *any* verified and registered Application which depends on the URM API. The given UCM will be an example service. The gray marked components are external components that are not in the scope of this document and will only be mentioned in context of dependencies.

### 2.1.2 User Interfaces

The User Application Module must provide an interface to manage the API. This will be a mobile (e.g. Android/iOS) or web (WWW Browser) interface.

### 2.1.3 Hardware Interfaces

All components must be able to execute on a personal computer.

### 2.1.4 Software Interfaces

The software interfaces of the modules will be further defined in the FRS Design Document.

### 2.1.5 Communication Interfaces

All communication will be over the Internet. The PN-database will be located locally or interacted remotely over an Internet connection, this will be further defined in the FRS Design Document.

### 2.1.6 Security constraints

All sent personal numbers must be encrypted end to end. Services must be authenticated to interact with the API. The external service does not have any information about the personal numbers or other high security informations saved and only the images are shared.

## 2.2 Product functions

The two main functions of the Face Recognition System are to allow user services to retrieve a PN corresponding to a photo of a face, and to allow administrators to manage User-Entities.
For managerial purposes it is useful to have the option to *RETRIEVE* all persons and then have the basic REST functionality to *ADD*,*UPDATE* or *DELETE* User-Entities. A User entity consists of a personal number and a (optimally biometric) picture of the person.

## 2.3 User characteristics

The User as defined in 1.5 is a verified and registered service which provides the User Application Module to interact with the URM.

## 2.4 Constraints

The system should enforce User and Admin authentication security and guarantee encrypted communication of critical data. The system should not take more than 5seconds response time when retrieving a PN with a photo.

## 2.5 Assumptions and Dependencies

The FRS will depend on an external face recognition API to compare photos. This can be developed in house or made use from existing systems. The system is developed under the assumption that the external component is reliable, secure, scalable and functional.

# 3 Specific functional requirements

## 3.1 User Application Module

3.1.1. The UAM shall support the log in of users

3.1.2. If and only if the log in of a user is successful (see 3.1.1) the user shall be able to use the functionality of the application.

3.1.3. The UAM should be able to access the camera of a users device.

3.1.4. The UAM should be able to send a picture to the URM.

3.1.5. The UAM should be able to process and show PNs.

3.1.6. The UAM shall encrypt all outgoing requests and be able to decrypt all incoming responses.

## 3.2 User Repository Module

3.2.1. Upon successful login request from the UAM the URM shall grant access. 3.1.1.

3.2.2. The URM shall be able to decrypt all incoming requests and encrypt all outgoing responses.

3.2.3. The URM should only process requests from authenticated users.

3.2.4. The URM shall be able to retrieve a PN from a Photo.

3.2.5. The URM shall be able to send a well formated response to the UAM.

## 3.3 Admin Repository Module

3.3.1. The ARM shall be able to authenticate admins.

3.3.2. The ARM shall be able to decrypt all incoming requests and encrypt all outgoing responses.

3.3.3. The ARM should only process requests from authenticated admins.

3.3.4. The ARM should be able to READ, DELETE, CREATE, UPDATE User-Entities.

# 4 Domain Model

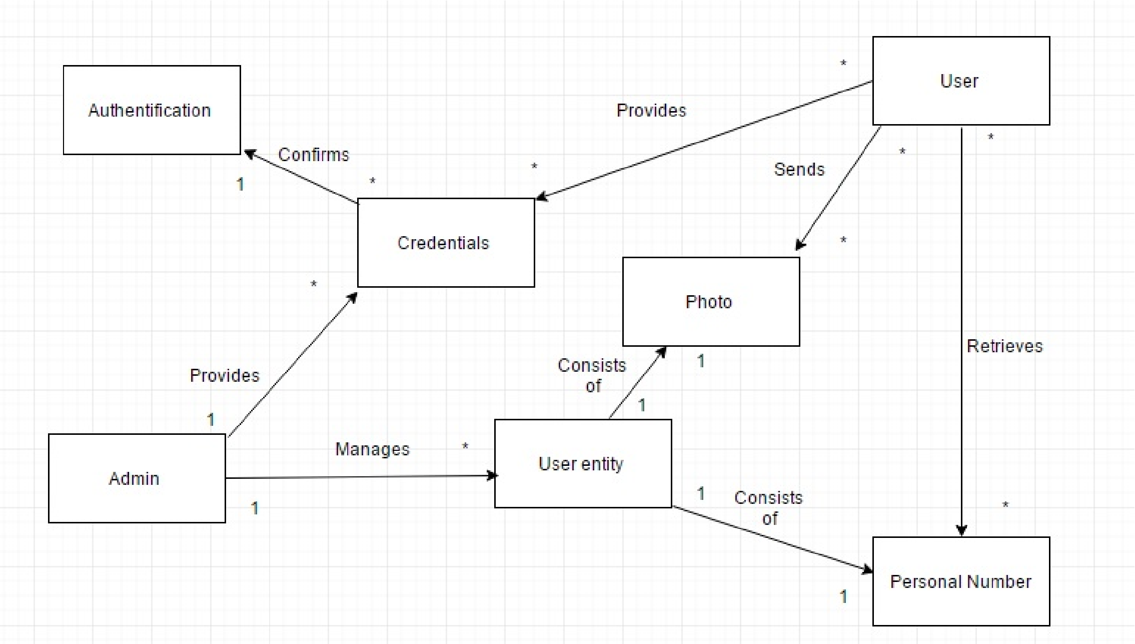
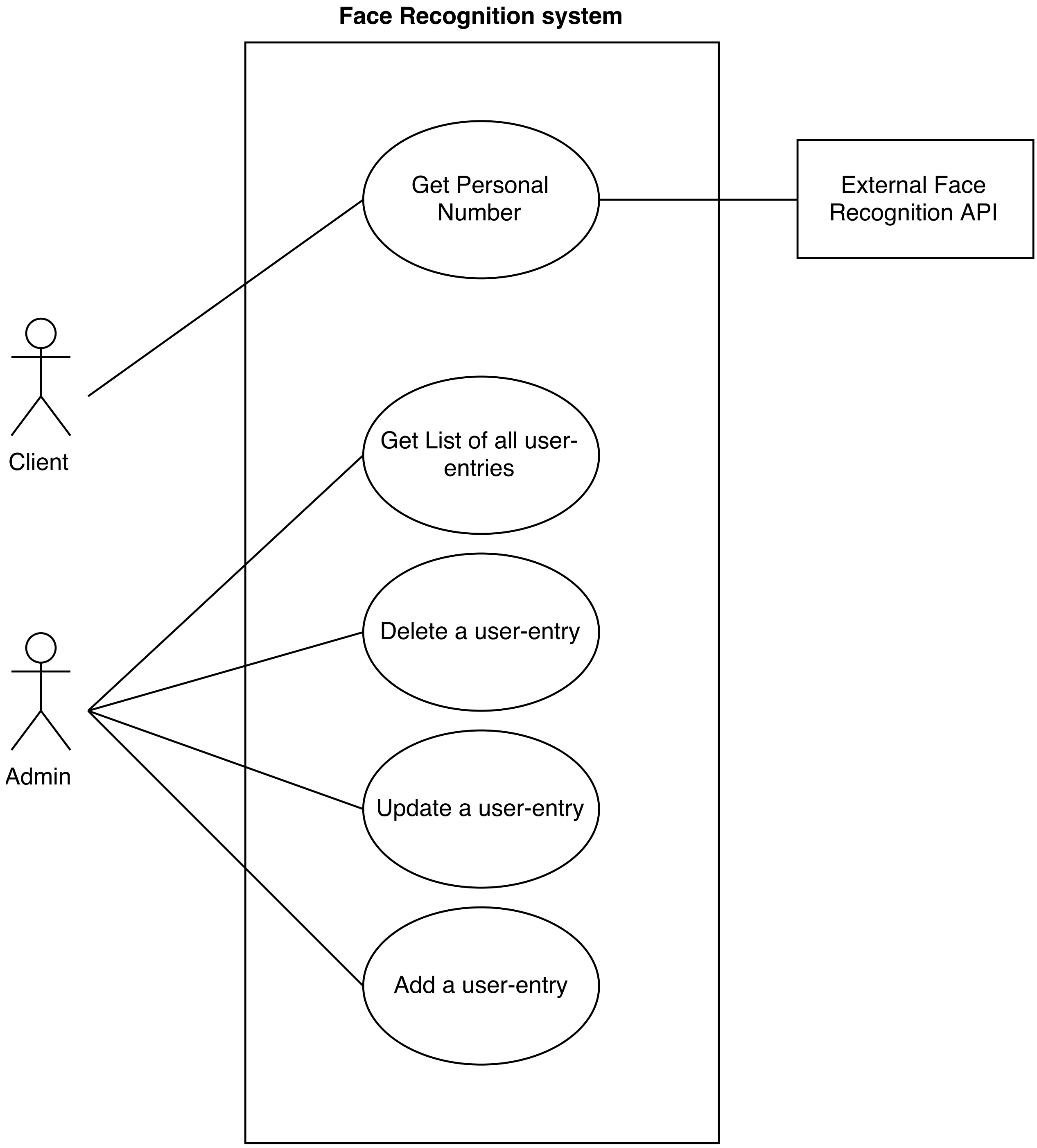


Figure 2: Domain Model

# 5 Modeling of functional requirements (Usecases & Scenarios)

## 5.1 Overview Use cases



## 5.2 UAM: GET Personal Number

| | |
|---|---|
| **Title** | User *GET* personal number |
| | |
| **Covered requirements** | 3.1.4 and 3.1.5 |
| | |
| **Description** | The user should be able to provide an image of a person's face within the UAM. Using the EFR the image will be compared to other images within the server. The user will then receive a PN matching the originally provided image. In the event that request fails an error message will be provided to the user further specifying the issue. |
| | |
| **Primary actor** | User |
| | |
| **Pre-conditions** | User must be authenticated (see 3.1.1) |
| | |
| **Post-conditions** | User must receive a notification |
| | |
| **Primary flow** | **1)** User send an image of a person's face<br>**2)** User receives a success notification and matching PN |
| | |
| **Secondary flow** *(Face does not match)* | **2a)** User receives a failure notification |

## 5.3 ARM: Get List of User-Entities

| | |
|---|---|
| **Title** | Admin *RETRIEVE* list of user-entities |
| | |
| **Covered requirements** | 3.3.3 |
| | |
| **Description** | The admin should be able to get a list of all the user-entities from the FRS. |
| | |
| **Primary actor** | Admin |
| | |
| **Pre-conditions** | Admin must be authenticated (see 3.3.1) |
| | |
| **Post-conditions** | Admin must receive a notification |
| | |
| **Primary flow** | **1)** Admin sends a GET request<br>**2)** Admin receives a success notification and list of all the user-entities currently |
| | |
| **Secondary flow** *(Internal server error)* | **2a)** Admin receives a failure notification |

### 5.4 ARM: Add an User-Entity

| | |
|---|---|
| **Title** | Admin *ADD* a new user |
| | |
| **Covered requirements** | 3.3.3 |
| | |
| **Description** | Admin want to add a new user. In order to add a new user, admin send a POST request to FRS which include user's image and PN. In result, admin receives an appropriate notification from the FRS. |
| | |
| **Primary actor** | Admin |
| | |
| **Pre-conditions** | Admin must be authenticated (see 3.3.1) |
| | |
| **Post-conditions** | Admin must receive a notification |
| | |
| **Primary flow** | **1)** Admin sends a POST request<br>**2)** Admin receives a success notification |
| | |
| **Secondary flow** *(User image is corrupted)* | **2a)** Admin receives a failure notification |
| | |
| **Secondary flow** *(User PN format is incorrect)* | **2a)** Admin receives a failure notification |
| | |
| **Secondary flow** *(User PN is invalid)* | **2a)** Admin receives a failure notification |
| | |
| **Secondary flow** *(User image is corrupted and PN is invalid)* | **2a)** Admin receives a failure notification |
| | |
| **Secondary flow** *(User image is corrupted and PN format is incorrect)* | **2a)** Admin receive a failure notification |

## 5.5 ARM: Delete an User-Entity

| | |
|---|---|
| **Title** | Admin *DELETE* a user |
| | |
| **Covered requirements** | 3.3.3 |
| | |
| **Description** | Admin want to delete a user. In order to delete a user, admin send a DELETE request to FRS which include user's unique id(should be integer). In result, admin receives an appropriate notification from the FRS. |
| | |
| **Primary actor** | Admin |
| | |
| **Pre-conditions** | Admin must be authenticated (see 3.3.1) |
| | |
| **Post-conditions** | Admin must receive a notification |
| | |
| **Primary flow** | **1)** Admin sends a DELETE request<br>**2)** Admin receives a success notification |
| | |
| **Secondary flow** *(User id not found)* | **2a)** Admin receives a failure notification |
| | |
| **Secondary flow** *(User id format is incorrect)* | **2a)** Admin receives a failure notification |

## 5.6 ARM: Update User-Entity

| | |
|---|---|
| **Title** | Admin *UPDATE* a user |
| | |
| **Covered requirements** | 3.3.3 |
| | |
| **Description** | Admin want to update a user. In order to update a user, admin sends an update request to FRS which includes the user's new unique id(should be integer) and/or the new photo. In result, admin receives an appropriate notification from the FRS. |
| | |
| **Primary actor** | Admin |
| | |
| **Pre-conditions** | Admin must be authenticated (see 3.3.1) |
| | |
| **Post-conditions** | Admin must receive a notification |
| | |
| **Primary flow** | **1)** The admin sends the new PN and/or picture<br>**2)** The Admin receives a success notification. |
| | |
| **Secondary flow** *(User id not found)* | **2a)** Admin receives a failure notification |
| | |
| **Secondary flow** *(User PN format is incorrect)* | **2a)** Admin receives a failure notification |
| | |
| **Secondary flow** *(User PN is invalid)* | **2a)** Admin receives a failure notification |
| | |
| **Secondary flow** *(User new image is corrupted)* | **2a)** Admin receives a failure notification |

### 5.7 ARM: Authenticate Admin

| | |
|---|---|
| **Title** | Admin *AUTHENTICATES* |
| | |
| **Covered requirements** | 3.3.1 |
| | |
| **Description** | Admin want to authenticate. In order to authenticate, admin sends an authentication request to FRS which includes the correct credentials. In result, admin can access the system. |
| | |
| **Primary actor** | Admin |
| | |
| **Pre-conditions** | Admin accessed to the main page of the system |
| | |
| **Post-conditions** | Admin receives access to the system |
| | |
| **Primary flow** | **1)** Admin sends an authentication request with credentials<br>**2)** Access granted |
| | |
| **Secondary flow** *(Credentials are incorrect)* | **2a)** Admin receives a failure notification<br>**2b)** Access denied |

### 5.8 URM: Authenticate User

| | |
|---|---|
| **Title** | User *AUTHENTICATES* |
| | |
| **Covered requirements** | 3.1.1 |
| | |
| **Description** | User want to authenticate. In order to authenticate, user sends an authentication request to FRS which includes the correct credentials. In result, user can access the system. |
| | |
| **Primary actor** | User |
| | |
| **Pre-conditions** | User accessed to the main page of the system |
| | |
| **Post-conditions** | User receives access to the system |
| | |
| **Primary flow** | **1)** User sends an authentication request with credentials.<br>**2)** Access granted |
| | |
| **Secondary flow** *(Credentials are incorrect)* | **2a)** User receives a failure notification<br>**2b)** Access denied |

¡¡¡¡¡¡¡

Updated upstream