<UNDECIDABLES>

COSBAS Architectural Requirements Documentation

Git: `https://github.com/undecidables/Requirements-Documentation`

The Team:
Elzahn Botha *13033922*
Jason Richard Evans *13032608*
Renette Ros *13007557*
Szymon Ziolkowski *12007367*
Tienie Pritchard *12056741*
Vivian Venter *13238435*

March 2015

Contents

1 Architectural Requirements **2**
- 1.1 Introduction . . . 2
- 1.2 Architectural Scope . . . 2
- 1.3 Quality Requirements . . . 2
  - 1.3.1 Reliablity . . . 2
  - 1.3.2 Security . . . 2
  - 1.3.3 Scalability And Generic . . . 2

2 Architectural Patterns or Styles **3**
- 2.1 MVC Architectural Pattern . . . 3
  - 2.1.1 Description . . . 3
  - 2.1.2 Reason for use . . . 3
- 2.2 Adapter Design Pattern . . . 3
  - 2.2.1 Description . . . 3
  - 2.2.2 Reason for use . . . 3

3 Use of Reference Architectures and Frameworks **3**

4 Access and Integration Channels **4**
- 4.1 Access Channels . . . 4
  - 4.1.1 Human access channels . . . 4
  - 4.1.2 System access channels . . . 4
- 4.2 Integration Channels . . . 4

5 Technologies **5**
- 5.1 Software . . . 5
  - 5.1.1 Build System . . . 5
  - 5.1.2 Server . . . 5
  - 5.1.3 Database . . . 5
  - 5.1.4 Client . . . 5
  - 5.1.5 Web Access Channel . . . 6
  - 5.1.6 Testing . . . 6
- 5.2 Hardware . . . 6
  - 5.2.1 Server . . . 6
  - 5.2.2 Client . . . 6
  - 5.2.3 Camera . . . 6
  - 5.2.4 Fingerprint Scanner . . . 7
  - 5.2.5 Keypad . . . 7

1 Architectural Requirements

1.1 Introduction

The software architecture requirements for the COSBAS system.

1.2 Architectural Scope

1.3 Quality Requirements

1.3.1 Reliablity

One of the most important quality requirements fot the system is that it should be reliable. The system should therefore produce the correct action/output at any given time when the system is in use.

The system should,

- never **give access** to someone whom should **not have access**.
- never **refuse access** to someone that **have access**.
- **never lock someone in**. If someone gain authorized access to the building the system should allow him/her to exit the building as well.

1.3.2 Security

The system needs to be secure, since personal information is stored on the system. Security as a quality requirement means that the system should prevent malicious actions/attacks.

The system should,

- protect the biometric data and other personal information of all users against unauthorized modification/access.
- protect the biometric data against third party applications.
- prevent loss of information (Personal information, Appointments etc.)

If security as a requirement is not met then the system can be deemed as unimplementable and worthless, since anyone can just gain access to the building.

1.3.3 Scalability And Generic

The system should scalable and generic.

The scalability includes,

- **Functional scalability** - Which means to add functionality to the system with mininal effort from developer(s). Althougth the system initially will only have facial recognition as a biometric method it should ideally be expanded to use any type of biometric method such as fingerprint scanning and voice recognition.

- The system needs to **scale out** (scale horizontally) - Which indicates that the system should be able to add more nodes,that is the system should add more biometric devices, replace existing ones and also add more entry points (entrances/doors), without the loss of performance or any other quality requirement.

2 Architectural Patterns or Styles

2.1 MVC Architectural Pattern

2.1.1 Description

MVC (Model-view-controller) is a software architectural pattern which devides the software application into three interconnected parts, so as to seperate the internal representation from the way the information is represented to the user.

2.1.2 Reason for use

- Client-Server communication
- Reduced code complexity
- Efficient code-reuse
- Decoupled code

2.2 Adapter Design Pattern

2.2.1 Description

The adapter design pattern changes or converts the interface of a class into another interface the client expects. The design pattern makes classes that would normally not be able to work together, interact seamlessly.

2.2.2 Reason for use

- Increased plugability of the system - Because many different biometric access points as well as non-biometric access points will have to interact with the system. This makes it easy for a new type of access point to be added to the system.

3 Use of Reference Architectures and Frameworks

We will be using the Spring MVC framework.

4 Access and Integration Channels

4.1 Access Channels

4.1.1 Human access channels

This system will be accessible to humans in the followings ways:

- From a thin client (this can be any computer with the client program but the preference is a Raspberry Pi) which will be installed at each entrance/exit of the building through non-intrusive bio-metrics or keypad.
- Humans can access the web client in the following ways:
  - Web Browser (client)
    - A web browser is needed to display the web pages of the website
    - Specifically Firefox, Chrome, Opera, Safari and IE.
  - Physical Devices
    - Since the website will be responsive, any device which has a web browser will be able to connect to the website
    - PCs, Laptops, Tablets and Smartphones

4.1.2 System access channels

The client(can be computer with the client program but in this case it will be a Raspberry Pi) should be able to access the services provided by the system to authenticate a user who would like to enter or exit the building. This will be done through SOAP based web services.

In order to access the web client, the following hardware will be needed:

- Internet Connection (ADSL) (wired or Wi-Fi)
  - A decent internet connection (at least ADSL level speed 384kbs) will be needed to connect to the website (since its being run on a server)
- Ethernet Connection
  - This is necessary if an Internet Connection is not available specifically if the web server is run locally, local computers can connect to it via Ethernet and thus have no need to use an Internet Connection

4.2 Integration Channels

COSBAS will integrate with the following channels:

- The CS LDAP server in order to retrieve login details of the lecturers.
- The postgrad meeting system in order to help with making appointments.
- Any online calendars used, such as Google calendar or Outlook, in order to gain access to the lecturers' calendars.

- The COSBAS-server to process the data and grant or deny access.
- The spring MVC framework, to help with dependency injection and connecting all the componenets together.

5 Technologies

5.1 Software

5.1.1 Build System

This project will use the Gradle build system.

5.1.2 Server

Programming Language: Java
Platform: The server-program needs to be deployed on a Linux server.

Frameworks and libraries:

- Spring
  - Spring Framework (MVC, IoC, AOP)
  - Spring LDAP
  - Spring Data MongoDB
  - Spring Security
- Thymeleaf template language for HTML and XML
- OpenCV graphics processing library
- JasperReports

5.1.3 Database

We will use a MongoDB database for persistence. The main reason for this choice is that biometric data (pictures, sounds, etc.) needs to be stored.

5.1.4 Client

The client application will be developed in either Python or Java depending on the biometric devices' APIs. It will communicate with the server using http requests. Each http request's data will contain the client's door id, whether the user is entering or exiting and the types of biometric devices and the captured data.

The client application should run on Linux (Raspbian)

5.1.5 Web Access Channel

- HTML5
- Javascript
- JQuery
- Ajax
- Bootstrap

5.1.6 Testing

We will use JUnit for unit testing.

5.2 Hardware

Our client gave us the task of choosing the equipment we would need in order to build a fully working system that will be implemented at a later stage. Below is a list of equipment we would need for this project and their requirements.

5.2.1 Server

The system will be deployed on an existing server owned by the Department of Computer Science.

5.2.2 Client

The client will be a device installed at the entrance and exit of the buildings where we would want to control access. The following requirements, listed below, must be met by the client in order to be compatible with the system.

- Capable of running a Linux based operating system.
- Can connect to a network.

We will be using a Raspberry PI as the client. This device has a low power consumption, small form factor and is quite cheap. It also meets the requirements listed above.

5.2.3 Camera

The camera will be used to capture the facial features of user's, which will be stored temporarily on the client. The client will then send this data to the server for authentication. The camera must satisfy the following requirements:

- Has a minimum horizontal resolution of 1920.
- Can connect to the client.
- Is supported by the operating system on the client.

5.2.4 Fingerprint Scanner

Fingerprint scanners will allow us to capture a user's finger print which will then be authenticated on the server against a database. For the fingerprint scanner to be compatible with the system it is must meet the following requirements:

- Has a minimum resolution of 500 pixels per inch.
- Can connect to the client.
- Is supported by the operating system on the client.

We have 2 fingerprint scanners that meet the above requirements. They are the Futronic FS80 and FS88. The main difference between the two scanners is that the FS88 can detect fake fingers.

5.2.5 Keypad

Visitors that do not have biometric access will use the keypad with the temporary access code they received when booking their appointments.

Requirements:

- Can connect to the client.
- Is supported by the operating system on the client.