



<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	Reliability	2
1.3.2	Security	2
1.3.3	Scalability	2
1.4	Integration and Access Channel Requirements	2
1.4.1	Human access channels	2
1.4.2	System access channels	2
1.4.3	Integration Channels	3
1.5	Architecture Constraints	3
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	Architectural Tactics or Strategies	4
4	Use of Reference Architectures and Frameworks	4
5	Access and Integration Channels	4
6	Technologies	4
6.1	Software	4
6.1.1	Build System	4
6.1.2	Server	4
6.1.3	Database	4
6.1.4	Client	4
6.1.5	Web Access Channel	5
6.1.6	Testing	5
6.2	Hardware	5
6.2.1	Server	5
6.2.2	Client	5
6.2.3	Camera	5
6.2.4	Fingerprint Scanner	6
6.2.5	Keypad	6

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 Reliability

1.3.2 Security

1.3.3 Scalability

1.4 Integration and Access Channel Requirements

1.4.1 Human access channels

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

- From a thin client(can be computer with the client program but in this case it will be 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

1.4.2 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

1.4.3 Integration Channels

The following integration channels will be made use of for the creation of the COSBAS system.

- 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 components together.

1.5 Architecture Constraints

2 Architectural Patterns or Styles

2.1 MVC Architectural Pattern

2.1.1 Description

MVC (Model-view-controller) is a software architectural pattern which divides the software application into three interconnected parts, so as to separate 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 pluggability 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 Architectural Tactics or Strategies

4 Use of Reference Architectures and Frameworks

5 Access and Integration Channels

6 Technologies

6.1 Software

6.1.1 Build System

This project will use Gradle as a build System

6.1.2 Server

Programming Language: Java

Platform: The server 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

6.1.3 Database

We will use a MongoDB database for persistence.

6.1.4 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)

6.1.5 Web Access Channel

- HTML5
- Javascript
- JQuery
- Ajax
- Bootstrap

6.1.6 Testing

We will use JUnit for unit testing.

6.2 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.

6.2.1 Server

xxxxxx

6.2.2 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.

6.2.3 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.

6.2.4 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 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.

6.2.5 Keypad

Users will make use of the keypad by entering their unique key(password) which the client will send to a server for authentication. The following requirements must be met by the keypad:

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