<UNDECIDABLES>

COSBAS Architectural Requirements Documentation

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

The Client:
Prof Andries Engelbrecht
Head of Department, Computer Science
University Of Pretoria

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

1.1 Introduction

The software architecture requirements for the COSBAS system.

1.2 Architectural Scope and Responsibilities

1.2.1 Persistence

Persisting system valuable information will take place in a database environment. In the scope of this project a non-relational and NoSQL database will be used. MongoDB was chosen as the best solution for the COSBAS project. The reason for this is the schemaless structure of data so there is no need to keep to a strict schema and MongoDB also allows us to store image data for biometrics.

1.2.2 Communication

A client-server architecture will be used for communication. We will use http requests sent from either the web browser or from the rasberry-pi computers.

1.3 Notifications

The appointment system should be able to send notifications to lecturers and visitors. All notifications will be via email, but the system should stay pluggable with the possibility of adding other methods later. In this context, methods refer to sub-systems such as biometric authentication services, third-party calendar services and notification method services.

1.4 Biometric Validation

The system should be able to validate biometric data sent from the client to identify the user it belongs to and grant access to the department for authorized users.

1.5 Quality Requirements

1.5.1 Reliablity

One of the most important quality requirements for the system is that it should be reliable. The reliability quality requirement is of high importance because the system is based on secure actions. 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 who should **not have access**.
- never **refuse access** to someone that **should have access**.
- **never lock someone in**. If someone gained authorized access to the building the system should allow him/her to exit the building as well.

1.5.2 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 and/or 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.

1.5.3 Scalability/Plugability

The system should be scalable, plugable and generic.

The scalability includes,

- **Functional scalability** - Which means to add functionality to the system with minimal effort from developers. The system initially will only have facial recognition and fingerprint scanning as the biometric methods, but it should ideally be expanded to be able use any type of biometric method such as voice recognition for example.
- 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
- To decouple code

2.2 Dependency Injection

We will be using dependency injection to ensure:

- That our code in easily changeable and extendable. By allowing for functionality to be swapped in and out as needed. [Attard, 2013]
- Better testing (Unit Testing and Integration Testing) and the mocking of objects. [Attard, 2013]
- Loose coupling between the different components. [Fowler, 2004]
- Inversion of Control. [Fowler, 2004]

2.3 Adapter Design Pattern

2.3.1 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.

2.3.2 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.

2.4 Strategy Design Pattern

2.4.1 Description

The strategy design pattern abstracts the deep implementations of a concrete class, defining a family of algorithms that can be called during runtime. This makes it easier to call functions based on the way the user wants the program to react dynamically.

2.4.2 Reason for use

- Increased plugability of the system - We use this for the creation of a calendar service. The use of a strategy design pattern makes it easier to create and initialize an instance of a Calendar object by a chosen service such as Google or Microsoft Outlook. The initial implementation of the COSBAS system only focusses on the Google Calendar Service.

2.5 Factory Method Design Pattern

2.5.1 Description

By making use of the Factory Method design pattern we can create an object without exposing the creational logic to the client and refer to newly created objects using a single common interface.

2.5.2 Reason for use

- Optimize code readability and enhance maintainability - The easier code is read and abstracted from the actual implementation, the easier it is to modify or maintain the system to the changing needs of the client and the COSBAS system users.

3 Access and Integration Channels

3.1 Access Channels

3.1.1 Human access channels

This system will be accessible to humans in the following 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.
- 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 and Safari.
  - 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

3.1.2 System access channels

The client (can be a 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.

3.2 Integration Channels

COSBAS will integrate with the following channels:

- The CS LDAP server in order to retrieve login details of the lecturers.

- Any online calendars used, such as Google Calendar or Outlook, in order to gain access to the lecturers' calendar.
- 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.
- Our MongoDB database for access to the registered users' stored biometric data as well as the appointment information of the staff members.

4 Technologies

4.1 Software

4.1.1 Build System

This project will use the Gradle build system.

Our reasons for choosing Gradle:

- Gradle is generally seen as incorporating the best aspects of Maven and Ant - It has Ants power and flexibility with Mavens life-cycle and ease of use. [Farcic, 2014, Timoin, 2013]
- Gradle can download dependencies from Maven and Ivy repositories including the Maven Central repository so all Maven packages can be used with gradle. [Gradle Documentation, 2015]
- Gradle's Groovy-based build script is more concise than Maven's XML pom. [Farcic, 2014]
- Gradle 2.5 allows for multiple build-types and simple custom tasks [Gradle User Manual, 2015]. This is useful for our client app that has one version for access requests and one version for registration.
- Even though Gradle is considered to have a bigger learning curve than Maven, it is extremely powerful [Timoin, 2013] and the DSL is simple to understand once you get the basics. [Farcic, 2014]
- Gradle also has the ability to exclude tasks from running which makes testing certain aspects of our system easier. Maven does not have this ability though.
- Our biggest reason for choosing Gradle is due to the fact that it is a newer build tehnology compared to Maven and Ant.

4.1.2 Server

Programming Language: Java
Platform: The server-program needs to be deployed on a Linux server.
Application Server: Jetty.
Reasoning: We chose a Jetty server over other application servers such as TomCat because the Jetty application server is much more third-party API friendly. Jetty is also much more focused on scalability and better performance with regards to serving static content than TomCat. Jetty's focus on multi-connection HTTP and features such as SPDY can significantly reduce page load latencies. The ultimate focus of Jetty is on speed, scalability and reliability, which is very important for our system.

4.1.3 Frameworks and libraries:

- Spring
  - Spring Framework (MVC, IoC, AOP)
    - We chose the Spring framework over other technologies for the following reasons:
      - Dependencies are explicit and evident in JavaBean properties.
      - Spring enhances modularity.
      - Spring provides more readable code.
      - Spring allows us to have loose coupling between different modules. This means that we can inject dependencies at runtime.
      - Spring provides a more flexible way to do dependency injection - dependencies can be configured by XML based schemas or annotations.
      - Spring makes it easier to implement Inversion of Control because we leave the work of dependency injection to the underlying framework.
      - Spring AOP (aspect oriented programming) is implemented purely in Java, and thus we don't need a separate compilation process. Also, because Spring AOP integrates with the underlying Spring framework, we get an advantage in terms of declarative programming for our security and logging purposes.
      - Spring isn't application server dependent.
      - Spring doesn't require any special deployment steps.
      - Spring simplifies Unit Testing because of its loose coupling, thus it makes it very easy to test a class independently (with or without mock objects).
      - Spring is innovative - for example, Spring was the first to bring out CDI, while other technologies like JavaEE took nearly 3 years to do the same thing. Thus, the possibility exists that Spring will keep innovating their technology and bring out new features. For example, Spring implements Spring Data, Spring Social and Spring Mobile.
      - Spring enables POJO programming which enables continuous integration and testability.
      - Spring is open source and has no vendor lock in.
      - Spring has a layered architecture, which means we only have to use what we need and we can leave what we don't.
      - The main reason why we are using it, however, is because of it's outstanding MVC framework. It is highly configurable with strategy interfaces, which is one of the requirements of our project (because we need to be able to use different types of Biometric Access Systems).
  - Spring LDAP
  - Spring Data MongoDB
    - Enabled easy integration with MongoDB in Java.
    - Provides build in operations/functions for the CRUD operations of the database.
    - Has CDI support for the Mongo Repositories that enables the system to have custom query functions.
  - Spring Security
    - Handles the login to LDAP
    - Does session handling for our system automatically
    - Easy web access control

- Thymeleaf template language for HTML and XML
- OpenCV graphics processing library
- DynamicReports
  - Allows for dynamic generation of reports meaning the system won't be stuck with a particular template as when using JasperReport (which makes use of static reports).
  - Allows reports to be exported to different formats such as pdf, docx, csv, html, ods, odt, xml, xlsx and may more.
- Log4J2 which is a logging framework.
  - Enables asynchronous logging which is good for logging within our server.
  - This provides us with the ability to configure the logger without restarting the server.
- JavaFX for the creation of the registration application GUI

4.1.4 Database

COSBAS will have a MongoDB database for the persistence of the biometric data of the staff members, the appointment information of each staff member, the authentication keys that is generated for each appointment and the credential objects needed by Google Calendar.

The reasons for using MongoDB:

- It is a NoSQL database which enables us to store our objects such as the appointment objects and credentials object easily.
- MongoDB also works well with Spring through SpringData.
- The flexible data model allows us to persist our images (Faces and Fingerprints) that will be used as the biometric data to authenticate a staff member.
- The scalability that MongoDB provides enable us to reach our scalability quality requirement (Horizontal Scaling) by the means of splitting the database up to run over two servers if we need to accomodate more staff members or visitors in the future.

4.1.5 Client

The client application will be developed in Java. 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).

4.1.6 Web Access Channel

- HTML5
- Javascript
- JQuery
- Ajax
- Bootstrap

4.1.7 Testing

We will use JUnit for unit testing.

4.2 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.

4.2.1 Server

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

4.2.2 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.

COSBAS will be using a **Raspberry PI 2 Model B** as the **client**. This device has a low power consumption, small form factor and is inexpensive. It also meets the requirements listed above.

4.2.3 Camera

The camera will be used to capture the facial features of a user, 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 Raspberry PI 2 Model B.
- Is supported by the operating system on the client.

COSBAS will be using the **Logitech C525** as the **camera** since it met the requirements above.

4.2.4 Fingerprint Scanner

Fingerprint scanners will allow us to capture a user's fingerprint which will then be authenticated on the server against our 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, which is the standard for a fingerprint scan.
- Can connect to the Raspberry PI 2 Model B.
- Is supported by the operating system on the client.

COSBAS will be using the **Futronic FS80H** as the **fingerprint scanner** since it met the requirements above.

4.2.5 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.

References

[Attard, 2013] Attard, A. (2013) *Why should we use dependency injection?* [Online] Available from: http://www.javacreed.com/why-should-we-use-dependency-injection/ [Accessed 31 July 2015]

[Farcic, 2014] Farcic, V. (2014). *Java Build Tools: Ant vs Maven vs Gradle.* [online] Technology. Available at: http://technologyconversations.com/2014/06/18/build-tools/ [Accessed 31 July 2015].

[Fowler, 2004] Fowler, M. (2004) *Inversion of Control Containers and the Dependency Injection pattern*[Online] Available from: http://www.martinfowler.com/articles/injection.html [Accessed 31 July 2015]

[Gradle Documentation, 2015] Gradle Documentation, (2015). *Chapter 8. Dependency Management Basics.* [online] Available at: https://docs.gradle.org/current/userguide/artifact_dependencies_tutorial.html [Accessed 31 July 2015].

[Gradle User Manual, 2015] Gradle User Manual, (2015). Chapter 55. Building native binaries. [online] Available at: http://gradle.monochromeroad.com/docs/userguide/nativeBinaries.html [Accessed 31 Jul. 2015].

[Microsoft 2015] Msdn.microsoft.com, (2015). Chapter 16: Quality Attributes. [online] Available at: https://msdn.microsoft.com/en-us/library/ee658094.aspx [Accessed 29 May 2015].

[MongoDB 2015a] MongoDB. 2015. MongoDB. [ONLINE] Available at: https://www.mongodb.org/. [Accessed 30 July 15].

[MongoDB 2015b] MongoDB. 2015. Purpose of Sharding. [ONLINE] Available at: http://docs.mongodb.org/manual/core/sharding-introduction [Accessed 30 July 15].

[Spring 2015a] Introduction to the Spring Framework. 2015. Introduction to the Spring Framework. [ONLINE] Available at: http://www.theserverside.com/news/1364527/Introduction-to-the-Spring-Framework. [Accessed 31 July 2015].

[Spring 2015b] Five Advantages of Spring Framework — Orange Slate. 2015. Five Advantages of Spring Framework — Orange Slate. [ONLINE] Available at: http://orangeslate.com/2006/11/10/five-advantages-of-spring-framework/. [Accessed 31 July 2015].

[Spring 2015c] Spring Interview Questions FAQs Interview Questions Spring Tutorial JSF Interview Questions Hibernate Interview Questions. 2015. Spring Interview Questions FAQs Interview Questions Spring Tutorial JSF Interview Questions Hibernate Interview Questions. [ONLINE] Available at: http://www.developersbook.com/spring/interview-questions/spring-interview-questions-faqs.php. [Accessed 31 July 2015].

[Spring 2015d] Why using Spring instead of JEE6? - Spring Forum . 2015. Why using Spring instead of JEE6? - Spring Forum . [ONLINE] Available at: http://forum.spring.io/forum/spring-projects/container/124744-why-using-spring-instead-of-jee6. [Accessed 31 July 2015].

[Spring 2015e] Spring advantages over current Java EE - Stack Overflow. 2015. Spring advantages over current Java EE - Stack Overflow. [ONLINE] Available at: http://stackoverflow.com/questions/16234864/spring-advantages-over-current-java-ee. [Accessed 31 July 2015].

[Spring 2015f] What are the benefits of using spring framework.What is Inversion of control (IOC), Aspect Oriented Programming(AOP)—SimpleCodeStuffs. 2015. What are the benefits of using spring framework.What is Inversion of control (IOC), Aspect Oriented Programming(AOP)—SimpleCodeStuffs. [ONLINE] Available at: http://www.simplecodestuffs.com/basic-concepts-of-spring/. [Accessed 31 July 2015].

[Spring 2015g] Spring. 2015. Spring Data MongoDB. [ONLINE] Available at: http://projects.spring.io/spring-data-mongodb/. [Accessed 30 July 15].

[Timoin, 2013] Timoin, J. (2013). *Java Build Tools: Maven, Gradle and Ant plus the DSL vs. XML debate.* [online] Zeroturnaround.com. Available at: http://zeroturnaround.com/rebellabs/java-build-tools-maven-gradle-and-ant-plus-the-dsl-vs-xml-debate/ [Accessed 31 July 2015].