**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**System Architectural**

**Analysis**

**for**

**The Automatic**

**Attendance Checking System**

**Version 1.0**

**Prepared by Huynh Vinh Nam**

**Le Huy Duc**

**Cao Phuong Linh**

**OOAD Group 2**

**9-Dec-2018**

**Table of Contents**

**Revision History**

**1. Introduction**

**1.1. Purpose**

This is a report on the subject Object-oriented Analysis and Design of group two, class ICT-BI7 about System Architectural Analysis of the project AACS.

**1.2. Intended Audience and Reading Suggestions**

*The different types of reader that the document is intended for are:*

**● Project managers:** who manage and take respond for the quality of the system. Project managers should read the whole document for planning and assigning work.

**● Developers:** Dev is the person who implement the system from the design and documents into a runnable version. Dev have to read the whole document to implement the right system.

**● Documentation writers:** who will write the future document (report, minutes).

Documentation writers should read to understand the Use Case Main Diagram part.

*The content of report includes Analysis Mechanisms, Key Abstraction, ten architectural main parts:*

**● Part 1 Architectural Representation:** This section describes what software architecture is for the current system, and how it is represented. Of the Use-Case, Logical, Process, Deployment, and Implementation Views, it enumerates the views that are necessary, and for each view, explains what types of model elements it contains.

**● Part 2 Architectural Goals and Constraints:** This section describes the software requirements and objectives that have some significant impact on the architecture, for example, safety, security, privacy,... It also captures the special constraints that may apply: design and implementation strategy, development tools, team structure, schedule, legacy code, and so on.

**● Part 3 Use Case View:** This section lists Use Cases or scenarios from the use-case model if they represent some significant, central functionality of the final system.

**● Part 4 Logical View:** This section describes

**● Part 5 Process View:** This section describes

**● Part 6 Deployment View:** This section describes

**● Part 7 Implementation View:** This section describes

**● Part 8 Data View:** A description of the persistent data storage perspective of the system.

**● Part 9 Size and Performance:** A description of the major dimensioning characteristics of the software that impact the architecture, as well as the target performance constraints.

**● Part 10 Quality:** A description of how the software architecture contributes to all capabilities (other than functionality) of the system: extensibility, reliability, portability, and so on.

**1.3. Product Scope**

The software’s main users are students and lecturers. Software will create an environment where user (student) can check for the attendance and user (lecturer) can view and/or manage the attendance list in the course(s).

**1.4. References**

[1] Form of presentation IEEE. IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications. IEEE Computer Society, 1998.

**2. Analysis Mechanisms**

**2.1. Persistence mechanisms**

For all classes with instances that may become persistent, you need to identify:

**● Granularity:** What is the range of size of the objects to keep persistent?

**● Volume:** How many objects (number) do you need to keep persistent?

**● Duration:** How long does the object typically need to be kept?

**● Retrieval mechanism:** How is a given object uniquely identified and retrieved?

**● Update frequency:** Are the objects more or less constant? Are they permanently updated?

**● Reliability:** Do the objects need to survive a crash of the process, the processor, or the whole system?

**2.2 Communication mechanisms**

For all model elements that need to communicate with components or services that are running in other processes or threads, you need to identify:

**● Latency:** How fast must processes communicate with another?

**● Synchronicity:** Asynchronous communication.

**● Size of message:** A spectrum might be more appropriate than a single number.

**● Protocol:** Flow control, buffering, and so on.

**2.3 Security mechanisms**

For all classes, packages, subsystem of AACS, you need to identify:

**● Data granularity:** .

**● User granularity:** How many user roles does system have?

**● Security rules:** Security Rule establishes national standards to protect individuals’ user data.

**● Privilege types:** What can a user role can do on the system?

**2.4 The other mechanisms**

All other analysis mechanisms:

**● Redundancy:** is the duplication of critical components or functions of a system with the intention of increasing reliability of the system, usually in the form of a backup or fail-safe.

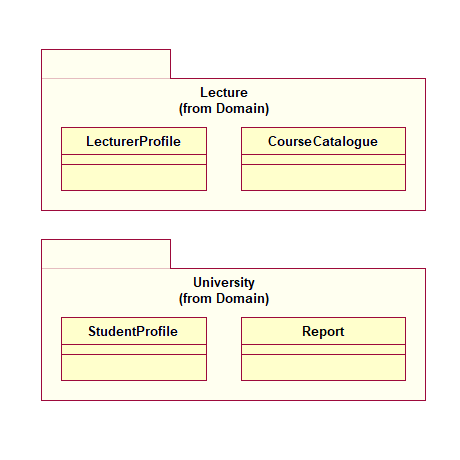
**● Error detection / handling / reporting:** How the errors should be detected / handled /

reported?

**● Distribution:** Which server should data be saved in?

**3. Key Abstractions**

**3.1. Diagram**

****

*Figure 0: Key Abstractions*

**3.3. Key Abstraction definitions**

**4. Architectural Representation**

This document presents the architectural as a series of views; use case view, process view, deployment view, and implementation view. These views are presented as Rational Rose Models and use the Unified Modeling Language (UML).

**5. Architectural Goals and Constraints**

There are some key requirements and system constraints that have a significant bearing on the architecture. They are:

**●** All functions must be available for the lecturer so that he/she can easily manage his/her checklist.

**●** The history of change always need to be backed up and stored in a safe place (in case a lecturer want to rollback).

**●** All performance and loading requirements, as the Supplementary Specification, must be taken into consideration as the architecture is being developed.

**6. Use Case View**

A description of the use-case view of the software architecture. The Use Case View is important input to the selection of the set of scenarios and/or Use Cases that are the focus of an iteration. It describes the set of scenarios and/or Use Cases that represent some significant, central functionality.

It also describes the set of scenarios and/or Use Cases that have a substantial architectural coverage (that exercise many architectural elements) or that stress or illustrate a specific, delicate point of the architecture.

The Use Cases in this system are listed below. Use Cases in **bold** are significant to the architecture.

A description of these Use Cases can be found later in this section.

**●** Sign in

**● Generate QR code**

**● Checking attendance**

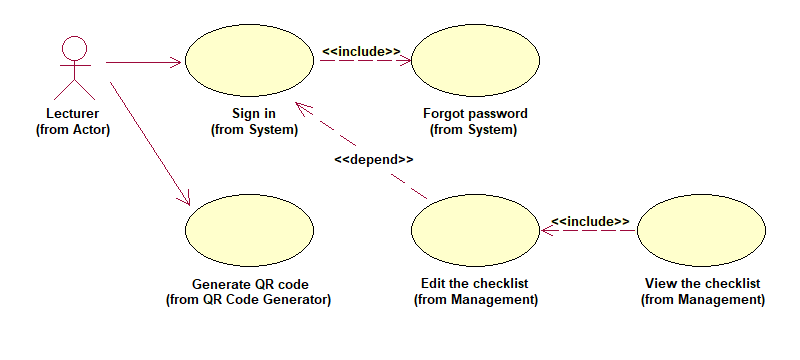
**●** View the checklist

**● Edit the checklist**

**6.1. Use Case Realizations**

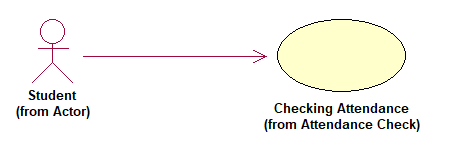
The following diagram depict the Use Cases in the system.

**6.1.1. Lecturer Use Case**

****

*Figure 1: Lecturer Use Case*

**6.1.2. Student Use Case**

****

*Figure 2: Student Use Case*

**6.2. Significant Use Case Descriptions**

**● Generate QR code**

The Use Case allows a logged-in lecturer to generate new QR code by providing required URL. This Use Case starts when a lecturer wants to generate a QR Code for students to check their attendance in his/her current lecture.

**● Checking attendance**

This Use Case starts when a student wishes to check for his/her attendance in the current lecture.

**● Edit the checklist**

This Use Case starts when a lecturer wants to edit the attendance checklist.

**7. Logical View**

**7.1. Overview**

The logical view of the AACS is comprised of 5 main packages:

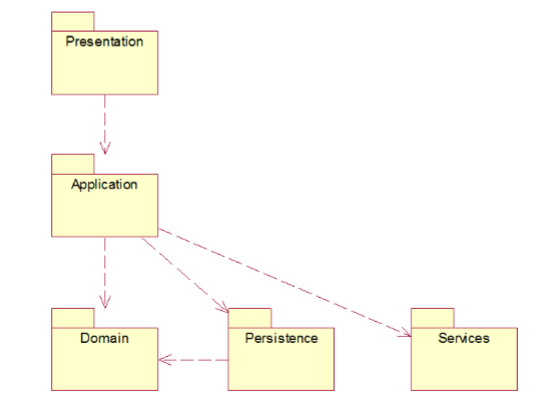
**● Presentation:** contains classes for each of the forms that the actors use to communicate with the System.

**● Application:** contains classes for major processing functionality within the system. Control classes exist to support generate QR code, checklist management.

**● Domain:** contains packages containing classes to support Content, Profile and Attendance.

**● Persistence:** contains classes to persist specific objects within the system. At this point in the design, only Profiles are persisted.

**● Services:** contains classes to provide system-level classes for maintenance purposes.

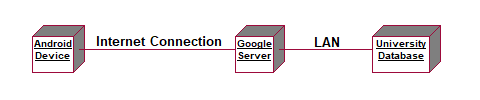
****

*Figure 3: Logical View Package Diagram*

**7.2. Architecturally Significant Design Packages**

**8. Process View**

**9. Deployment View**

****

*Figure 4: Deployment View model*

**10. Implementation View**

**11. Size and Performance**

The software as designed will support 300 users simultaneously.

The network request for checking attendance should have no more than 3 seconds latency.

**12. Quality**