**MINISTRY OF EDUCATION AND TRAINING**

**FPT UNIVERSITY**

Capstone Project Document

**School Feedback Management System**

|  |  |
| --- | --- |
| **Group 11** | |
| **Group members** | SE61897 - Nguyễn Việt Tú (Leader)  SE61882 - Trần Hồ Minh Thuấn  SE61892 - Thái Tiến Hoàng  SE61745 - Hoàng Quốc Việt |
| **Supervisor** | Nguyễn Huy Hùng |
| **Ext. Supervisor** | N/A |
| **Capstone Project**  **code** | SFMS |

-Ho Chi Minh City, ***8th January, 2018***-

*This page is intentionally left blank*

## Table of Contents

[Table of Contents 3](#_bookmark0)

[List of Tables 4](#_bookmark1)

[Definitions, Acronyms, and Abbreviations 6](#_bookmark2)

1. [Report No. 1 Introduction 7](#_bookmark3)
   1. [Project Information 7](#_bookmark4)
   2. [Introduction 7](#_bookmark5)
   3. [Current Situation 7](#_bookmark6)
   4. [Problem Definition 7](#_bookmark7)
   5. [Proposed Solution 8](#_bookmark8)
      1. [Feature functions 8](#_bookmark9)
      2. [Advantages and disadvantages 8](#_bookmark10)
   6. [Functional Requirements 9](#_bookmark11)

6.1 Name Card Management .................................................**Error! Bookmark not defined.**

6.2 Event ................................................................................**Error! Bookmark not defined.**

6.3 Searching..........................................................................**Error! Bookmark not defined.**

6.4 Suggestion ........................................................................**Error! Bookmark not defined.**

6.5 User Management ...........................................................**Error! Bookmark not defined.**

* 1. [Role and Responsibility 9](#_bookmark12)

## List of Tables

[Table 1: Roles and Responsibilities 9](#_bookmark13)

## List of Figures

Figure 1: Modified Waterfall Development Model..............................**Error! Bookmark not defined.**

# Definitions, Acronyms, and Abbreviations

Miêu tả từ viết tắt hay các term dùng trong tài liệu thuyết minh bên dưới

|  |  |
| --- | --- |
| **Name** | **Definition** |
| SFMS | School Feedback Management System |

PAGE \\* MERGEFORMAT 1

* 1. **Report No. 1 Introduction**

## Project Information

* + - * Project name: School Feedback Management System
      * Project Code: SFMS
      * Product Type: **Web app, Mobile App**
      * Start Date: **08/01/2018**
      * End Date:

## Introduction

In this document, we introduce a solution for School Feedback System. Current feedback systems have some problems like impossible to design and customize feedback form that suitable for each major, lecture, course, or inconvenient in analyzing results. Base on our reaserches and analysis, we proposed a solution for university in Vietnam and other developed countries.

We build a system, which helps the universities solve current problems. In the process of analysis, we believe our system is capable to resolve the problem by let users create, design, and customize feedback form through dragging and dropping items. Besides that, we also analyze feedback results and provide reports and suggested improvements, based on the results.

This document also describes our working process in 4 months includes our perspective in the system, component designs and detailed core workflows. We hope the system and our solution will help resolve the problems from universities in Vietnam and other developed countries.

## Current Situation

Currently, in university, we use the same feedback form for every courses and major. So we can just gain feedback of some general information like: if lecturer is on time, if the students can understand the lecture, how skillful the lecturer is... When feedback period’s over, the results will send to lecturer without analysing or providing suggested improvement.

## Problem Definition

* **Customize Feedback form:**

- Use same questions set for all lectures, courses, majors

- Can’t question unique aspect of each course (For example: Can’t question about accent of English lecturer)

- Lack of usefulness and practicality (Can only ask same general questions that used for all courses)

- Hardly use for personal improvement

- Can’t choose level of feedback survey (Feedback for lecture, course, major, or for department)

* **Analyze results and Report**

- Can’t generate graphical charts of statistical results from feedback answers

- Can’t compare results in the past to current

- Don’t provide suggested improvement to lecturers

- Can’t choose target person to send feedback reports to (Send to Lecturer only, or Head of Academic, or both)

## 5. Proposed Solution

Our proposed solution is to build a system named School Feedback Manage System can drag-n-drop feedback items to create new feedback form to resolve those problems of current situations.

SFMS includes a web application to manage and create feedback, and mobile app to conduct feedback, with following functions:

##### **Feature functions**

<Nêu ra các tính năng cốt lỗi, các vai trò cốt lõi trong giải pháp mà nhóm đề xuất, chỉ nên nêu các tính năng chủ chốt giải quyết bài toán, không phải liệt kê toàn bộ tính năng>

* Web Application:

- Create new Feedback form: choose existed template or customize by dragging and dropping items to the form

- Save created feedback template

- Choose scope of Feedback: Feedback for a lecture, a major, a course or a department

- Set suggested improvement based on avaerage points of each feedback

- Compare results in the past to current

- Choose target role to send result report to

- Set interval time for each Feedback

- Remind users to conduct Feedback

- Search and Filter reports

* Mobile Application:

- Conduct Feedback

* 1. **Advantages and disadvantages**
* **Advantages**:

**-** Make realistic improvement from feedback

- Help Head of Academic follows real performance of lecturers

- Feedback can use in different subjects, courses, majors, lecturer, departments

- Represent results as graphical charts

- Save created feedback form to

* **Disadvantages**:

- Staffs have to spend time to design feedback form

- Input suggested improvement manually

## 6. Functional Requirements

**• Base component**

o Login – Logout

**• Student component**

o View list of feedbacks

- Filter based on datetime/semester, status (done or undone), course, major, lecturer, department

o Do feedback

o Remind of undone Feedback

**• Staff component**

o Create new feedback form:

- Choose existed template

- Choose scope (lecture, major, course, department)

- Drag-n-drop feedback items

- Set suggested improvement

- Set interval time for feedback

o View list of current feedbacks

- Filter based on datetime/semester, status (done or undone), course, major, lecturer, department

o Set target role to send report

- Choose between lecturer, head of Academic, or both

o Save feedback template

o View Report

o Filter Reports by scope, name (of lecturer, course, major, department), datetime/semester

o Do feedback

**• Head-of-Academic component**

o Manage users / accounts (CRUD)

o Manage departments (CRUD)

o Manage majors (CRUD)

o Manage courses (CRUD

o Create new feedback form

o Save feedback template

o View list of feedbacks

o View Report

o Filter Reports by scope, name (of lecturer, course, major, department), datetime/semester

**• Lecturer component**

o View Report for targeted feedback templates

o Filter Reports by datetime/semester

o View list of feedbacks

o Do feedback

o Remind of undone Feedback

## 7. Role and Responsibility

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *No* | *Full Name* | *Role* | *Position* | *Contact* |
| *1* | *Nguyễn Huy Hùng* | *Project Manager* | *Supervisor* | [*hungng@fpt.edu.vn*](mailto:hungng@fpt.edu.vn) |
| *2* | *Nguyễn Việt Tú* | *Developer* | *Leader* | [*tunvse61897@fpt.edu.vn*](mailto:tunvse61897@fpt.edu.vn) |
| *3* | *Thái Tiến Hoàng* | *Developer* | *Member* | [*hoangttse61892@fpt.edu.vn*](mailto:hoangttse61892@fpt.edu.vn) |
| *4* | *Hoàng Quốc Việt* | *Developer* | *Member* | [*viethqse61745@fpt.edu.vn*](mailto:viethqse61745@fpt.edu.vn) |
| *5* | *Trần Hồ Minh Thuấn* | *Developer* | *Member* | [*thuanthmse61882@fpt.edu.vn*](mailto:thuanthmse61882@fpt.edu.vn) |

*Table 1: Roles and Responsibilities*

* 1. **Report No.2 Software Project Management Plan**

1. **Problem Definition**

**1.1 Name of this Capstone Project**

* + - Official name: School Feedback Management System
    - Vietnamese name: Hệ Thống Quản Lý Feedback Trường Học
    - Abbreviation: SFMS
  1. **Problem Abstract**

This project is our exertion about School Feedback System in Viet Nam. We called it is School Feedback Management System (SFMS). SFMS will provide users a user-friendly interface to create and conduct feedback.

Organizing a feedback survey in an effectively way is always a problem to improve quality. With the current Feedback system, users use the same feedback for all department, majors and course. Therefore, it’s hard to evaluate the true quality while each department, major and course has its unique aspects.

SFMS will provide a system where users can create several Feedback forms easily by dragging and dropping Feedback items, and save them as templates to reuse later.

To qualify the results, SFMS uses a new critical points-counting system for each question, provides suggested improvement when critical points is low.

Finally, SFMS analyzed results and presents as charts, help users to review and compare the results through semesters.

### Project Overview

###### Current Situation

Below are the problems encountered in this project:

* Lack of the amount of necessary data: students, lecturers, staffs… data
* Limit in human resources and time: Team has only 4 members and time for all project is about 13 weeks for writing document, implementing the products and testing
* Lack of UI, UX (user experience) design skill: Our team members all study IS major and no one has studied UI, UX design.
* Lack of knowledge about back-office business: How point is counted in survey, how feedback results are treated…
* New technique: Some team members are new to the techniques used in the project. The team need an amount of time to get familiar with those techniques.

###### 1.3.2 The Proposed System

The system will have three sub-systems:

* An API application to serve API for mobile application and web application.
* A web application for staffs, lecturers, head of academic and students. Head of Academic can manage users, majors, courses, departments. Staffs and head of academic can create new feedback or choose existed template. Staff, Head of academic and lecturers can view reports when feedback period finishes. All users can conduct feedback.
* A mobile application for students conduct feedback

Task will be assigned vertically to team members, so that if one member quits, the team will be not lack of resources.

**1.3.2.1 Web Site**

**• Student component**

o View list of feedbacks

- Filter based on datetime/semester, status (done or undone), course, major, lecturer, department

o Do feedback

o Remind of undone Feedback

**• Staff component**

o Create new feedback form:

- Choose existed template

- Choose scope (lecture, major, course, department)

- Drag-n-drop feedback items

- Set suggested improvement

- Set interval time for feedback

o Save feedback template

o View list of current feedbacks

- Filter based on datetime/semester, status (done or undone), course, major, lecturer, department

o Set target role to send report

- Choose between lecturer, head of Academic, or both

o View Report

o Filter Reports by scope, name (of lecturer, course, major, department), datetime/semester

o Do feedback

**• Head-of-Academic component**

o Manage users / accounts (CRUD)

o Manage departments (CRUD)

o Manage majors (CRUD)

o Manage courses (CRUD

o Create new feedback form

o Save feedback template

o View list of feedbacks

- Filter based on datetime/semester, status (done or undone), course, major, lecturer, department

o View Report

o Filter Reports by scope, name (of lecturer, course, major, department), datetime/semester

o Do feedback

**• Lecturer component**

o View Report for targeted feedback templates

o View list of feedbacks

o Do feedback

o Filter Reports by datetime/semester

1.3.2.2 Mobile Application

* View list of feedbacks

- Filter based on datetime/semester, status (done or undone), course, major, lecturer, department

* Do feedback
* Remind of undone Feedback

##### **1.3.2.3 API Application**

The server system takes responsibility to respond all the requests and also manages and processes data.

* Provide APIs for Mobile Application, Web Application

###### 1.3.3 Boundaries of the System

The system can:

* Allow Head of academic to manage users, majors, courses, departments.
* Allow create new feedbacks
* Allow save feedbacks as templates to reuse
* Allow conduct feedback
* Notify users when there is undone feedback
* Count feedback points based on result
* Suggest improvement based on Feedback result
* View feedback reports

###### 1.3.4 Future Plans

Current system is concentrated on core business flow. Therefore, some supporting features are restricted for the development team. These features may be expanded in the future:

* Analyze if users really spend time to do feedback
* Analyze “Other comment” paragraph to see if it’s positive or negative comment and count point

###### 1.3.5 Development Environment

1.3.5.1 Hardware requirements <Yêu cầu phần cứng>

***For server***

|  |  |  |
| --- | --- | --- |
| **Windows** | **Minimum Requirements** | **Recommended** |
| **Internet Connection** | Cable, Wi-Fi (4 Mbps) | Cable, Wi-Fi (8 Mbps) |
| **Operating System** | Window Server 2008 | Window Server 2008 |
| **Computer Processor** | Intel® Xeon ® 1.4GHz | Intel® Xeon ® Quad Core  (12M Cache, 2.50 GHz) |
| **Computer Memory** | 1GB RAM | 2GB or more |

**Table 2: Hardware Requirement for Server**

**For PC**

|  |  |  |
| --- | --- | --- |
| **Windows** | **Minimum Requirements** | **Recommended** |
| **Internet Connection** | Cable, Wi-Fi (2 Mbps) | Cable, Wi-Fi (4 Mbps) |
| **Operating System** | Windows 7 or above | Window 7 or above |
| **Computer Processor** | Intel® Pentinum 4 1.60Ghz | Intel® Pentinum 4 2.00Ghz |
| **Computer Memory** | 1GB RAM | 2GB or more |

**Table 3: Hardware Requirement for PC**

**For mobile**

|  |  |  |
| --- | --- | --- |
| **Windows** | **Minimum Requirements** | **Recommended** |
| **Internet Connection** | Wi-Fi (2 Mbps) | Wi-Fi (4 Mbps) |
| **Operating System** | Android 4.4.2 or above | Android 6.0 or above |
| **Memory** | 1GB RAM | 2GB or more |

**Table 4: Hardware Requirement for Mobile**

1.3.5.2 Software requirements <Yêu cầu phần mềm>

|  |  |  |
| --- | --- | --- |
| Software | Name / Version | Description |
| Operating system | Window 7 or above | Operating system and platform for development |
| Environment | Java EE 8 | Specification for developing web application |
| Modeling tool | StarUML 2.8.0 | Use to draw model models and diagrams |
| IDE | NetBeans 8.2, IntelliJ IDEA  17.2, Android Studio 3.0 | Programming tools |
| DBMS | MySQL 5.7, SQL Server 2005 | Used to create & manage the database for system |
| Source control | Github | Used for source control |
| Web browser | Chrome 42 or above | Testing browser |

1. **Project organization**
   1. **Software Process Model**

This project is developed using Scrum model – part of an agile framework for Software development project. Our team choose Scrum model because of the following reasons:

* Our team only has 4 members, and tasks are assigned vertically, do all steps from design, coding, testing and implementation. Scrum is the most suitable model for small and medium project.
* In the project there are many new technologies that need to be learned. With the Scrum model, the team can learn and develop in parallel to meet deadline.
* There is no leader, no hierarchy in team, so team members work cheerfully, stimulating the initiative and creativity of each member.
* Product owner can change requirement or extend scope. The team will adapt to change better.



Figure 1: Scrum model

<https://www.scrum.org/resources/what-is-scrum>

### Roles and responsibilities <Bảng phân chia vai trò>

|  |  |  |  |
| --- | --- | --- | --- |
| ***No*** | ***Full name*** | ***Role in Group*** | ***Responsibilities*** |
| **1** | Nguyễn Huy Hùng | Project manager | * Specify scope and user requirement * Control the development process * Give out technique and business analysis support |
| **2** | Nguyễn Việt Tú | Scrum master | * Create Sprint Backlog and Product Backlog. * Make sure the Scrum teams understand and follow the process. * Always be present to answer questions and give advice when product owner or scrum member needs. * Help the team master scrum artifacts such as: Sprint Backlog, Product Backlog, ... * Writing report |
| ***3*** | Nguyễn Việt Tú  Trần Hồ Minh Thuấn  Thái Tiến Hoàng  Hoàng Quốc Việt | Scrum team members | * Designing database * Clarifying requirements * Prepare documents * GUI Design * Coding * Testing |

*Table 3: Roles and Responsibilities Details*

**2.3 Tools and Techniques**

|  |  |
| --- | --- |
| Tool / Technique | Name / version |
| Frontend | HTML, CSS, JavaScript, Bootstrap |
| Backend | SpringBoot framework, Java |
| IDE | NetBeans 8.2, IntelliJ IDEA 17.2, Android Studio 3.0 |
| Database | MySQL 5.7, SQL Server 2005 |
| Modelling tool | Star UML 2.8.0 |

**3. Project Management Plan**

### 3.1 Product Backlog

All product backlog could be found here

### 3.2 Sprint Backlog

All sprint backlog could be found [here](https://docs.google.com/a/fpt.edu.vn/spreadsheets/d/1Tj9tsbnGwQrQiTGnWU0NW55C2mwtRrxtwqqfhS-2teA/edit?usp=drive_web)

### 3.3 Deliverables

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Deliverable** | **Deliverable date** | **Deliverable location** | **Note** |
| 1 | Introduction, Project Management Plan, Concept Diagram, Class Diagram, Entity Relationship Diagram, Use Case Overview, Mock UI |  |  | Sprint 1 |
| 2 | Study Spring Boot Framework, Design User Interface for Web Application |  |  | Sprint 2 |
| 3 | Design User Interface for Web Application (continue), Web Service Architecture, Code core flow RESTful API Web services |  |  | Sprint 3 |
| 4 | Code core flow Web application and Mobile application; User Requirement Specification |  |  | Sprint 4 -> 8 |
| 5 | Code low priority functions, Conceptual Diagram, Design Overview, System Architectural Design, Component Diagram. |  |  | Sprint 9-10 |
| 6 | Testing, Entity Relationship Diagram, Database Diagram, Algorithms |  |  | Sprint 11-12 |
|  | | | | |
| Table 6: Deliverables | | | | |

* For each Sprint, deliverables are potentially shippable products, which can be a part of document or prototype implemented based on the project’s core flow.
* Each Sprint has a fixed duration of one weeks.

**3.4 All Meeting Minutes**

All sprint meeting minutes could be found [here](https://drive.google.com/open?id=1uTbg15fL05aFUnV9TXOXZ4Vy-kl4HMBu)

## 4. Coding Convention

* **Naming convention:**

- Variable and method names are in mixed case, with first letter of each internal word capitalized except first word.

* **Method names should be verbs.**

- Class names should be nouns, in mixed case with first letter of each internal word capitalized.

- Constant names should be all uppercase with words separated by underscore.

* **Comment**:

- Using /\* \*/ for block comments.

- Using // for line comments.

Using Java coding convention from:

<http://www.oracle.com/technetwork/java/codeconvtoc-136057.html>

1. **Report No. 3 Software Requirement Specification**
   1. **User Requirement Specification**
      1. **Guest Requirement**

Guest is a person who wants to create or conduct feedback. Guest can do the following functions:

* Login
  + 1. **Student Requirement**

Student is a person who attends in a class. Student conducts most of feedbacks. Student can do the following functions:

* Manage Feedbacks:

+ View list feedbacks

+ Filter feedbacks

* Conduct Feedback
* Be alerted of undone feedback

**1.3 Lecturer Requirement**

Lecturer is a person who teach a class. Lecturer is the target of most of feedbacks. Teacher can do the following functions:

* Manage Feedbacks:

+ View list feedbacks

+ Filter feedbacks

* Conduct Feedback
* Be alerted of undone feedback
* Manage Reports:

+ View list reports that relates to their classes

+ Compare reports by semesters

+ View report details

+ Filter reports

* 1. **Staff Requirement**

Staff is a person who work for the school, but doesn’t have the highest role. Staff can do the following functions:

* Conduct Feedback
* Be alerted of undone feedback

- Create new feedback form:

+ Create blank new feedabck

+ Drag-n-drop feedback items to create content

+ Edit feedback content

+ Set suggested improvement

+ Set name and description for feedback

+ Set interval time for feedback

+ Set semester for feedback

+ Choose scope (lecture, major, course, department)

+ Choose and modify feedback target

+ Choose and modify feedback conductor

* Create new feedback from template
* Save Feedback:

+ Save and publish feedback

+ Save feedback as a new template

+ Update current template

* Manage Feedbacks:

+ View list feedbacks

+ Filter feedbacks

* Manage Reports:

+ View list reports that relates to their departments

+ Compare reports by semesters

+ View report details

+ Filter reports

* Manage Users:

+ View list users

+ Add users

+ Update users

* 1. **Head-of-Academic Requirement**

Head of Academic is a person manage everything and have the highest role. Head of Academic can manage criterias of feedbacks, deactive users and can view all reports. Head of Academic can do the following functions:

* Conduct Feedback
* Be alerted of undone feedback

- Create new feedback form:

+ Create blank new feedabck

+ Drag-n-drop feedback items to create content

+ Edit feedback content

+ Set suggested improvement

+ Set name and description for feedback

+ Set interval time for feedback

+ Set semester for feedback

+ Choose scope (lecture, major, course, department)

+ Choose and modify feedback target

+ Choose and modify feedback conductor

* Create new feedback from template
* Save Feedback:

+ Save and publish feedback

+ Save feedback as a new template

+ Update current template

* Manage Feedbacks:

+ View list feedbacks

+ Filter feedbacks

* Manage Reports:

+ View all reprots

+ Compare reports by semesters

+ View report details

+ Filter reports

* Manage Users:

+ View list users

+ Add users

+ Update users

+ Deactive users

* Manage Criterias:

+ View list criterias

+ Add criterias

+ Update criterias

+ Deactive criterias

### System Requirement

System performs functions that run underground and connects to external system (deposit money, notification system). System does following functions:

* Calculate stistic results for report
* Send notification when users have undone feedbacks
  1. **System Requirement Specification**
     1. **External Interface Requirement**
        1. **User Interface**
* Mobile and Web application use Vietnamese interface, suitable for Vietnamese user.
  + - 1. **Hardware Interface**
* N/A

###### Software Interface

* Web application: work with Firefox, Chromes, Internet Explorer browsers.
* Mobile application: Android operating system.

###### Communication Protocol

* Use HTTP protocol 1.1 for communication between the web browser and the web server, mobile application and the server.

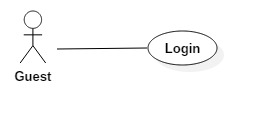
### System Overview Use Case

### 

Figure 2: System Overview Use Case

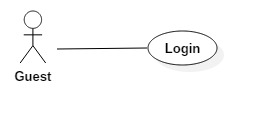
### List of Use Case

* + - 1. ***Guest Overview Use Case***



***Figure 3: Guest Overview Use Case***

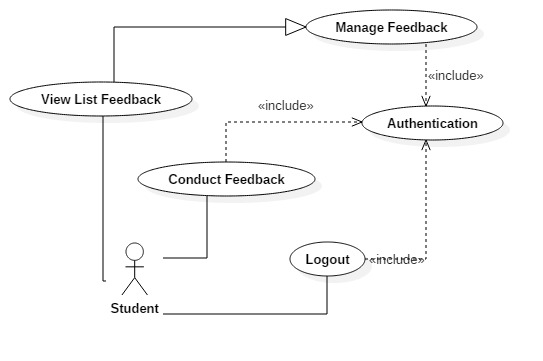
***2.3.1.1 Guest Login***



***Figure 4: <Guest>Login***

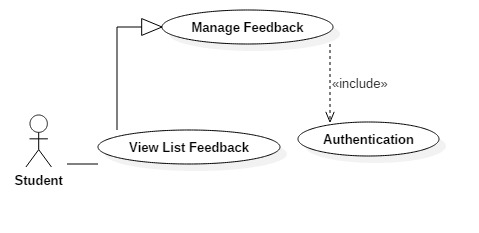
|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC001** | | | |
| **Use Case No.** | 001 | **Use Case Version** | 1.0 |
| **Use Case Name** | Login | | |
| **Author** | TuNV | | |
| **Date** | 27/03/2018 | **Priority** | High |
| **Actor:**   * Guest   **Summary:**   * This use case allows Guest login to the system on website and mobile application.   **Goal:**   * Guest login successfully with the proper role.   **Triggers:**   * Guest sends the login command.   **Preconditions:**   * Guest has an account.   **Post Conditions:**   * **Success:** Guest accesses the system successfully. * **Fail:** System shows error message “Invalid username or password”.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Guest sends command to request login. | System requires identity information form Guest:   * Username: free text input. * Password: free text input. | | 2 | Guest inputs information |  | | 3 | Guest sends command to login to system.  [Alternative 1] | Guests will login system with their specific role. |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor action | System Response | | 1 | Guest input invalid username or password. | System shows error message “Tên đăng nhập hoặc mật khẩu không đúng”. |   **Exceptions :** N/A  **Relationships:** N/A  **Business Rules:**  - After login to system, guest will be redirected to specific view based on their role on the system: student, lecturer, staff or head of academic.   * If role is “Student”, the system will display to Student view. * If role is “Lecturer”, the system will display to Lecturer view. * If role is “Staff”, the system will display to Staff view. * If role is “Head of Academic”, the system will display to Head of Academic view. | | | |

* + - 1. ***Student Overview Use Case***



***Figure 5: <Student> Overview Use Case***

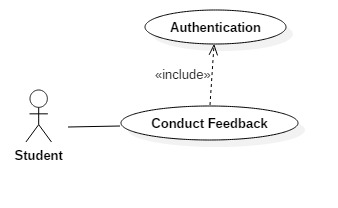
***2.3.2.1 Student View List Feedback***

******

***Figure 6: <Student> View List Feedback***

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC002** | | | |
| **Use Case No.** | 002 | **Use Case Version** | 1.0 |
| **Use Case Name** | View List Feedback | | |
| **Author** | TuNV | | |
| **Date** | 27/03/2018 | **Priority** | High |
| **Actor:**   * Student * Staff * Lecturer * Head of Academic   **Summary:**   * This use case allows Actor to get feedback list.   **Goal:**   * System gets feedback list and display it to Actor.   **Triggers:**   * Actor sends get feedback list command.   **Preconditions:**   * Actor has been authorized and accessed the system.   **Post Conditions:**   * **Success:** Feedback list is displayed. * **Fail:** Show error messages.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Actor sends get feedback list command. | [Exception 1]  System displays feedback list to Actor. |   **Exceptions 1:**   |  |  |  | | --- | --- | --- | | Step | Cause | System Response | | 1 | Feedback list is empty | System shows error message “Không có Phản hồi”. |   **Relationships:** N/A  **Business Rules:**   * Actor can view list of feedbacks that they are in conductors list. * Actor can conduct feedbacks that are still in available time from this list. | | | |

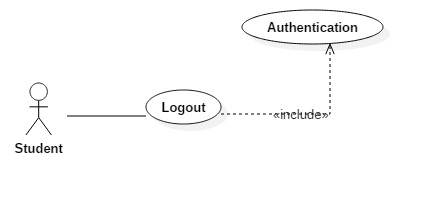
***2.3.2.2 Student Conduct Feedback***

******

***Figure 6: <Student> Conduct Feedback***

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC003** | | | |
| **Use Case No.** | 003 | **Use Case Version** | 1.0 |
| **Use Case Name** | Conduct Feedback | | |
| **Author** | TuNV | | |
| **Date** | 27/03/2018 | **Priority** | High |
| **Actor:**   * Student * Staff * Lecturer * Head of Academic   **Summary:**   * This use case allows Actor to get conduct feedback.   **Goal:**   * Actor submits answers successfully.   **Triggers:**   * Actor clicks button “Hoàn thành ngay” from list feedback. * Actor clicks corresponding feedback from alert button.   **Preconditions:**   * Actor has been authorized and accessed the system. * Actor has feedback needed to conduct. * Feedback is still in available interval time.   **Post Conditions:**   * **Success:** Actor submits answers succesfully. * **Fail:** Show error messages.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Actor sends conduct feedback command. | System displays list of feedback questions to Actor. | | 2 | Actor chooses options |  | | 3 | Actor submits answers | System checks answers.  [Exception 1]  System directs to view list feedback screen and changes button “Hoàn Thành Ngay” to “Đã Hoàn Thành” |   **Exceptions 1:**   |  |  |  | | --- | --- | --- | | Step | Cause | System Response | | 1 | Actor doesn’t answer all required questions | System shows error message “Xin hoàn thành những câu hỏi bắt buộc”. |   **Relationships:** N/A  **Business Rules:**   * Actor answers question of the feedbacks. * Actor has to answer all required questions. | | | |

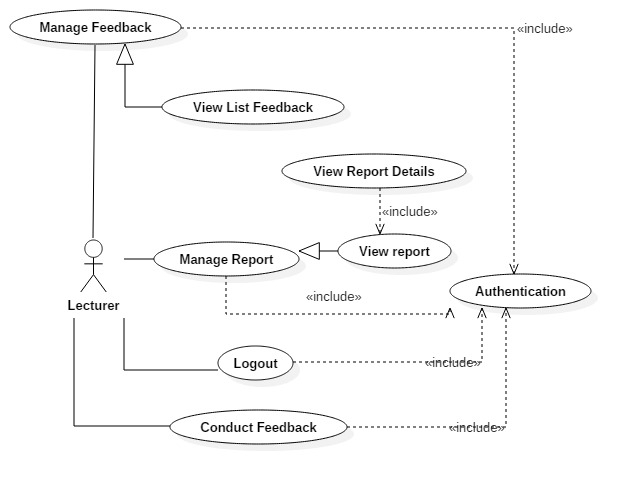
***2.3.2.3 Student Logout***

******

***Figure 7: <Student> Logout***

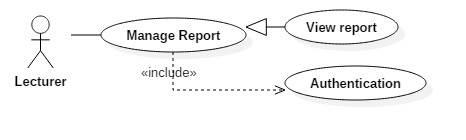
|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC003** | | | |
| **Use Case No.** | 004 | **Use Case Version** | 1.0 |
| **Use Case Name** | Logout | | |
| **Author** | TuNV | | |
| **Date** | 27/03/2018 | **Priority** | High |
| **Actor:**   * Student * Staff * Lecturer * Head of Academic   **Summary:**   * This use case allows Actor to logout.   **Goal:**   * Actor logouts successfully.   **Triggers:**   * Actor clicks button “Logout”.   **Preconditions:**   * Actor has been authorized and accessed the system.   **Post Conditions:**   * **Success:** Actor logouts succesfully. * **Fail:** N/A   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Actor sends command to Logout. | System clears session state if any, takes user out of the system.  System displays sign in view. |   **Exceptions 1:** N/A  **Relationships:** N/A  **Business Rules:**   * After logout, role “Authenticated User” will become “Guest”. | | | |

* + 1. ***Lecturer Overview Use Case***



***Figure 8: <Lecturer> Overview Use Case***

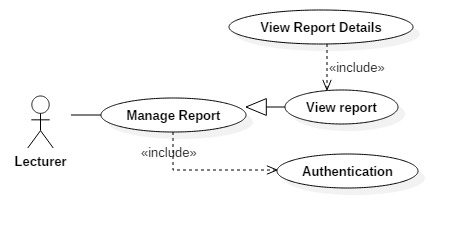
***2.3.3.1 Lecturer View List Report***

******

***Figure 9: <Lecturer> View List Report***

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC005** | | | |
| **Use Case No.** | 005 | **Use Case Version** | 1.0 |
| **Use Case Name** | View List Reprot | | |
| **Author** | TuNV | | |
| **Date** | 27/03/2018 | **Priority** | High |
| **Actor:**   * Staff * Lecturer * Head of Academic   **Summary:**   * This use case allows Actor to get report list.   **Goal:**   * System gets report list and display it to Actor.   **Triggers:**   * Actor sends get report list command.   **Preconditions:**   * Actor has been authorized and accessed the system.   **Post Conditions:**   * **Success:** Report list is displayed. * **Fail:** Show error messages.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Actor sends get report list command. | [Exception 1]  System displays report list to Actor. |   **Exceptions 1:**   |  |  |  | | --- | --- | --- | | Step | Cause | System Response | | 1 | Report list is empty | System shows error message “Không có Báo cáo”. |   **Relationships:** N/A  **Business Rules:**   * Actor can view list of report that corresponds to their roles. * Lecturer can view reports of their classes. * Staff can view reports of their departments. * Head of Academic can view all reports. | | | |

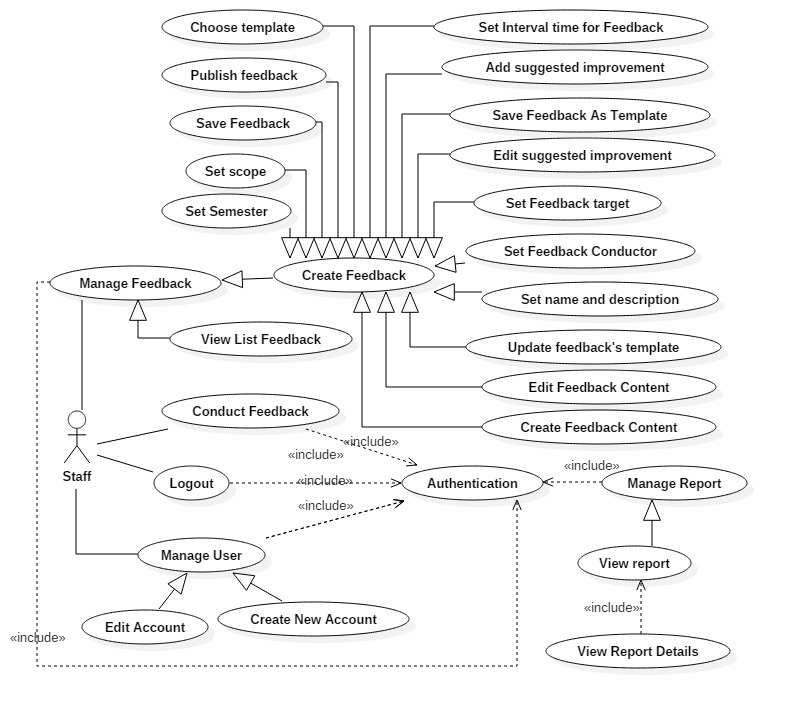
***2.3.3.2 Lecturer View Report Details***

******

***Figure 10: <Lecturer> View Report Details***

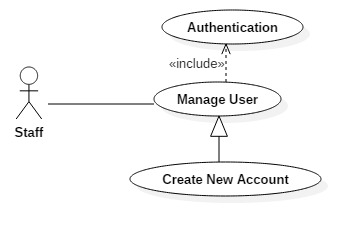
|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC006** | | | |
| **Use Case No.** | 006 | **Use Case Version** | 1.0 |
| **Use Case Name** | View List Reprot Details | | |
| **Author** | TuNV | | |
| **Date** | 27/03/2018 | **Priority** | High |
| **Actor:**   * Staff * Lecturer * Head of Academic   **Summary:**   * This use case allows Actor to view report details.   **Goal:**   * System gets details of a report and displays it to Actor.   **Triggers:**   * Actor sends get report details command.   **Preconditions:**   * Actor has been authorized and accessed the system.   **Post Conditions:**   * **Success:** Details of a report is displayed. * **Fail:** Show error messages.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Actor sends get report details command. | System calculates answer points, statistics and displays report list to Actor. |   **Relationships:** N/A  **Business Rules:**   * Actor can view details of a report. * Details of a report included: Statistic number of answers of each questions, proportions of answers, average point of each criteria, average point of the report. * Average point of each criteria is the average point of all questions belong to that criteria. * Average point of the report is the average point of all criteria belong to that report. | | | |

* + 1. ***Staff Overview Use Case***



***Figure 11: <Staff> Overview Use Case***

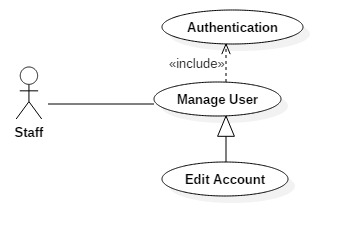
***2.3.4.1 Staff Create New Account***

******

***Figure 12: <Staff> Create new account***

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC007** | | | |
| **Use Case No.** | 007 | **Use Case Version** | 1.0 |
| **Use Case Name** | Create new account | | |
| **Author** | TuNV | | |
| **Date** | 27/03/2018 | **Priority** | High |
| **Actor:**   * Staff * Head of Academic   **Summary:**   * This use case allows Actor to create new account.   **Goal:**   * Account is created successfully and store in database of the system.   **Triggers:**   * Actor clicks on create new account button.   **Preconditions:**   * Actor has been authorized and accessed the system with proper role.   **Post Conditions:**   * **Success:** New account is created. * **Fail:** Account is not created.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Actor sends create new account command. | System requires information:   * Username: free text input, required, length (9-20), unique * Password: free text input, required, length (9-20) * Full name: free text input, required, length (10-50) * Birthdate: date * Email: free text input, email format, unique * Sex: male or female * Role: Student, Lecturer, Staff, or Head of Academic * Major: Only for Lecturer and Student * Department: Only for Staff and Student * Status: Active or Deactive * Code | | 2 | Actor inputs information and sends command to “Save”.  [Alternative 1] | Account created.  [Exception 1]  [Exception 2]  [Exception 3]  [Exception 4] |   **Alternative Scenario 1:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | |  | Actor sends command to reset. | System reset all field to blank. |   **Exceptions :**   |  |  |  | | --- | --- | --- | | No | Cause | System Response | | 1 | Actor inputs username already exist. | System show warning message “Tên đăng nhập này đã có người sử dụng”. | | 2 | Actor inputs email already exist. | System show warning message “Mail này đã được sử dụng”. | | 3 | Actor does not input required field. | System notices that actor need to input all these field:   * “Full name” * “Username” * “Password” * “Code” * “Email” | | 4 | Actor inputs wrong some fields with requirement. | System notices that actor need to re-input all those fields. |   **Relationships:** N/A  **Business Rules:**   * After creating new account and active, user can login with new account. * If choose role “Lecturer” or “Student”, actor can choose field “Major”, field “Department” will be disabled. * If choose role “Staff” or “Head of Academic”, actor can choose field “Department”, field “Major” will be disabled. | | | |

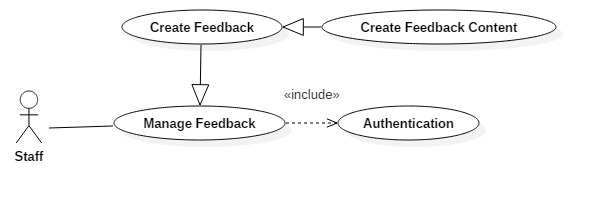
***2.3.4.2 Staff Edit Account***

******

***Figure 13: <Staff> Edit account***

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC008** | | | |
| **Use Case No.** | 008 | **Use Case Version** | 1.0 |
| **Use Case Name** | Edit account | | |
| **Author** | TuNV | | |
| **Date** | 27/03/2018 | **Priority** | High |
| **Actor:**   * Staff * Head of Academic   **Summary:**   * This use case allows Actor to edit account.   **Goal:**   * Profile of account is updated.   **Triggers:**   * Actor clicks on edit account button.   **Preconditions:**   * Actor has been authorized and accessed the system with proper role.   **Post Conditions:**   * **Success:** Information of account is updated and display. * **Fail:** Information of account is not updated.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Actor sends edit account command. | System requires information:   * Username: free text input, required, length (9-20), unique * Password: free text input, required, length (9-20) * Full name: free text input, required, length (10-50) * Birthdate: date * Email: free text input, email format, unique * Sex: male or female * Role: Student, Lecturer, Staff, or Head of Academic * Major: Only for Lecturer and Student * Department: Only for Staff and Student * Status: Active or Deactive * Code | | 2 | Actor inputs information and sends command to “Save”.  [Alternative 1] | Account updated.  [Exception 1]  [Exception 2]  [Exception 3] |   **Alternative Scenario 1:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | |  | Actor sends command to reset. | System reset all field to blank. |   **Exceptions :**   |  |  |  | | --- | --- | --- | | No | Cause | System Response | | 1 | Actor inputs email already exist. | System show warning message “Mail này đã được sử dụng”. | | 2 | Actor does not input required field. | System notices that actor need to input all these field:   * “Full name” * “Password” * “Email” | | 3 | Actor inputs wrong some fields with requirement. | System notices that actor need to re-input all those fields. |   **Relationships:** N/A  **Business Rules:**   * After edited, new information will be save to profile of account. * Actor can’t change Username, Code field. | | | |

***2.3.4.2 Staff Create New Feedback Content***

******

***Figure 14: <Staff> Create New Feedback Content***

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC009** | | | |
| **Use Case No.** | 009 | **Use Case Version** | 1.0 |
| **Use Case Name** | Create Feedback Content | | |
| **Author** | TuNV | | |
| **Date** | 27/03/2018 | **Priority** | High |
| **Actor:**   * Staff * Head of Academic   **Summary:**   * This use case allows Actor to create new feedback content.   **Goal:**   * Actor successfully creates new feedback content.   **Triggers:**   * Actor clicks on create blank feedback button.   **Preconditions:**   * Actor has been authorized and accessed the system with proper role.   **Post Conditions:**   * **Success:** New Feedback with created content is stored in database. * **Fail:** New Feedback content cannot be created.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Actor sends create blank new feedback command. | System displays create blank new feedback screen offering 4 types of feedback elements: radio, multiple choices, text or paragraph. | | 2 | Actor drags and drops suitable feedback elements to create content. | System requires information:   * Radio: Question, Options, “Other” or not, “Required” or not, Point of each Option, Criteria * Multiple Choices: Question, Options, “Other” or not, “Required” or not, Point of each Option, Criteria * Text: Question, Criteria * Paragraph: Question, Criteria | | 2 | Actor sends command to “Save”.  [Alternative 1]  [Alternative 2]  [Alternative 3] | Account updated.  [Exception 1] |   **Alternative Scenario 1:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | |  | Actor sends command to delete a created question. | System deletes chosen question. |   **Alternative Scenario 2:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | |  | Actor sends command to delete a created option. | System deletes chosen option. |   **Alternative Scenario 3:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | |  | Actor drags question to reorder. | System reorders chosen questions. |   **Exceptions :**   |  |  |  | | --- | --- | --- | | No | Cause | System Response | | 1 | Actor does not input required field. | System notices that actor need to input all these field:   * Question * Option * Point |   **Relationships:** N/A  **Business Rules:**   * After clicking save, all questions and options will be stored in database. * Point will be limit from 1 to 5 points. | | | |

1. **Software System Attribute**
   1. **Usability**

* Head of Academic and staff should need less than 0.5 day of training to use the system.
* Mobile and web application use Vietnamese including all dialogs and messages.
* Student and Lecturer can understand and perform functions immediately without the training process.
  1. **Availability**
* System is divided into modules, if a function is down, it will not influence others.
* The system can be adapted for a large number of requests.
  1. **Security**
* All input data should be validated before saving to database.
* Roles permission should be specified clearly and user should be authenticated and authorized when accessing to the system.
  1. **Maintainability**
* System is divided into modules.
* When a module of a function is down, it is easy to take it down to fix without impact other functions.

### Portability

* Web application can run on Chrome, Firefox, IE browser.
* User can use the mobile application on devices running Android 6 or later.

### Performance

* System converts and returns results in 1 seconds or less depend on information of driver inputted.

1. **Conceptual Diagram**

<Xác định các **thực thể - không cần có thuộc tính** - và **mối quan hệ** giữa chúng với nhau **thông qua các business rule**, **actor**, các **thành phần có mối quan hệ** để hình thành nên các thực thể thông qua các **mô tả trong usecase diagram và usecase specification** đã nêu ra ở trên>

**Chú ý**

* Chỉ sử dụng một tập kí hiệu và cần reference đến địa chỉ mô tả tập kí hiệu để sử dụng cho chính xác
* Các Diagram cần lớn rõ ràng, phải dàn trang cho phù hợp và nên dùng trang A3 để in
* Các thành phần trong diagram phải được thể hiện thông qua dictionary

**Data Dictionary <Đặc tả các thực thể có trong hình>**

|  |  |
| --- | --- |
| **Entity Data dictionary: describe content of all entities** | |
| **Entity Name** | **Description** |
|  |  |

Ví dụ

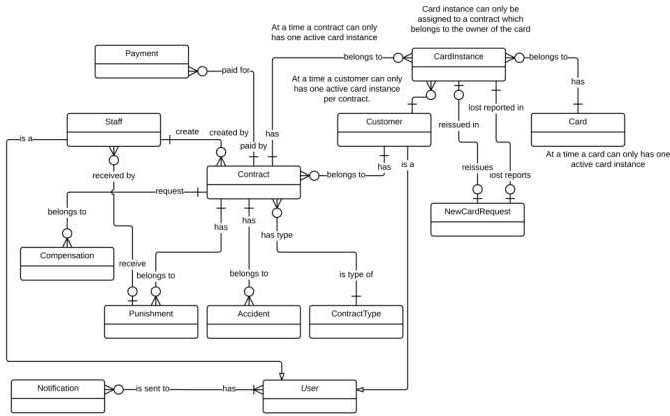


Figure 8 Conceptual diagram

***Data Dictionary***

|  |  |
| --- | --- |
| *Entity Data dictionary: describe all content of all entities* | |
| *Entity Name* | ***Description*** |
| *User* | *Abstract entity describes a user in system* |
| *Customer* | *Contain the customer information.* |
| *Contract* | *Contain the contract information.* |
| *Card* | *Contain the card information* |
| *CardInstance* | *Represent a card assigned to a contract* |
| *Payment* | *Contain the payment information.* |
| *Staff* | *Contain the staff information.* |
| *Compensation* | *Contain the compensation information.* |
| *Punishment* | *Contain the punishment information.* |
| *Accident* | *Contain the accident information.* |
| *ContractType* | *Contain the contract type information.* |
| *NewCardRequest* | *Contain the new card request information.* |
| *Notification* | *Contain the notification information* |

***Table 9 Conceptual Diagram Data Dictionary***

Ví dụ

### Report No. 4 Software Design Description

## Design Overview

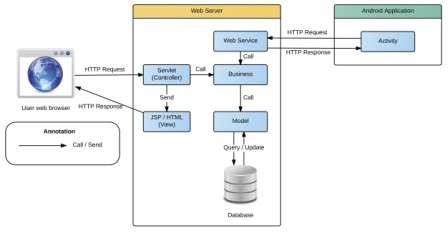
<Nội dung này tham khảo và có thể giữ nguyên và chỉ thay thế các phần phù hợp với đồ án của nhóm. Nhóm có thể viết lại cho hay hơn>

* *This document describes the technical and user interface design of* ***MSSC System****. It includes the architectural design, the detailed design of common functions and business functions and the design of database model.*
* *The architectural design describes the overall architecture of the system and the architecture of each main component and subsystem.*
* *The detailed design describes static and dynamic structure for each component and functions. It includes class diagrams, class explanations and sequence diagrams for each use cases.*
* *The database design describes the relationships between entities and details of each entity.*
* *Document overview:*
  + *Section 2: gives an overall description of the system architecture design.*
  + *Section 3: gives component diagrams that describe the connection and integration of the system.*
  + *Section 4: gives the detail design description which includes class diagram, class explanation, and sequence diagram to details the application functions.*
  + *Section 5: describe screens design.*
  + *Section 6: describe a fully attributed ERD.*
  + *Section 7: describe algorithms****.***
    1. **System Architectural Design**

<Kiến trúc hệ thống mà nhóm xây dựng: sử dụng các pattern và reference đến nội dung và xem xét lựa chọn các diagram mang đầy đủ nội dung như concept, không sao chép, vay mượn và chế kí hiệu. Nếu dùng kí hiệu ngoài UML thì ghi chú giải kí hiệu ngay cạnh hình vẽ.>

<Mô tả kiến trúc của từng thành phần trong ứng dụng nếu có.>

Ví dụ



***Figure 9 System architecture design***

This diagram is referenced and modified from an original concept from: Chapter 6 Architecture Design, SOFTWARE ENGINEERING 9th Edition, by Ian Sommerville.

**2.1 Web application architecture description**

<Giải thích lý do tại sao lựa chọn mô hình này dựa trên SRS, Introduction, và project plan đã nêu ra ở các phần trên>

<Mô tả các thành phần của kiến trúc theo dạng bảng, và sự tương tác giữa các thành phần theo kiến trúc.>

Ví dụ

In Web Application, the system is developed under J2EE MVC architecture style. We choose this architecture for Web application because of following advantages:

* *Web app contains a Web service (public API for mobile app), with MVC architecture, we can separate business code with Controller and View, so we can use the business code in web service without repeat the code.*
* *...*

This project follows MVC architecture with following components:

* *Servlet (Controller) is the parts of the application that acts like event handler to handles user interaction. Typically, controller read data from a request and calls appropriate Business’s method then selects view to return to user.*
* *...*

### 2.2 ...

* + 1. **Component Diagram**

<Thể hiện việc chia hệ thống thành các component. Nội dung này dựa trên kiến trúc đã đề ra ở phần trên để chia cho phù hợp và đúng mô hình>

**Ghi chú:** Xem lại bộ quy ước kí hiệu của UML 2.0 trước khi vẽ các mối quan hệ cũng như hiểu rõ thiết kế để vẽ chính xác. Nếu tool không phù hợp thì nhóm nên dùng Paint để vẽ

<Mô tả từng thành phần trong hình vẽ theo bảng biểu bên dưới.>

|  |  |
| --- | --- |
| **Component dictionary: describe component** | |
| **Component Name** | **Description** |
|  |  |

Ví dụ



**Figure 10 Component Diagram**

|  |  |
| --- | --- |
| *Component Dictionary: Describes components* | |
| *Web Application* | *Web application package: View, Controller* |
| *Mobile Application* | *Mobile application package* |
| *PayPal* | *Handle payment process with PayPal API* |
| *Payment Component* | *Component to handle payment process* |
| *Web Service* | *Provide API for mobile applications to interact with the system.* |
| *Staff Component* | *Component to handle staff activities in the system* |
| *Customer Component* | *Component to handle customer activities in the system* |
| *Public Component* | *Component to handle guest activities in the system* |
| *Admin Component* | *Component to handle admin activities in the system* |
| *Schedule Component* | *Component to handle scheduler in the system* |
| *Business Objects* | *Common objects to handle domain business operations for*  *each components* |
| *Data Access Objects* | *Component to handle interaction between the system and*  *database* |

***Table 10 Component Dictionary***

## Detailed Description

### Class Diagram

<Hình thiết kế class diagram: tham khảo các mối quan hệ giữa các lớp trong đặc tả UML, nắm rõ về dependency, association, composition, aggregation, inheritance. Bên cạnh đó, cần xác định rõ cardinality giữa các quan hệ với nhau. Đây là dạng conceptual class diagram, do vậy, cần căn cứ trên conceptual diagram và nội dung xây dựng object cần thiết khi lập trình và xây dựng ứng dụng trong lúc viết chương trình>

<Mô tả từng thành phần class theo bảng biểu bên dưới.>

|  |  |
| --- | --- |
| **Class dictionary: describe Class** | |
| **Class Name** | **Description** |
|  |  |

Ví dụ

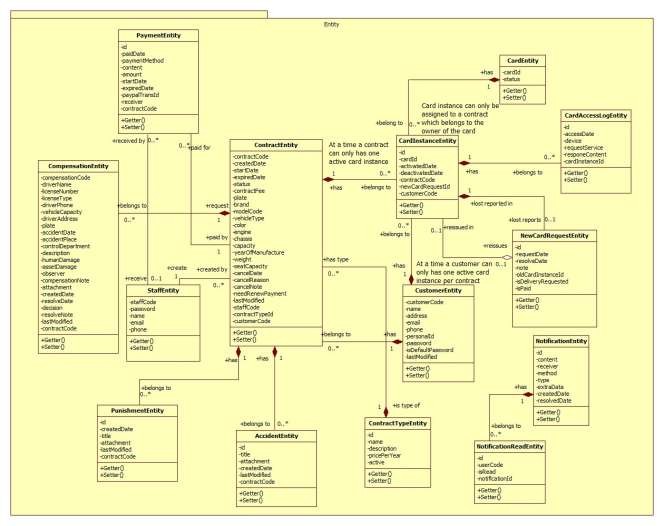


Figure 11 Class Diagram

|  |  |  |
| --- | --- | --- |
| *Class dictionary: describe Class* | | |
| *Class Name* | ***Mapping column***  ***with Conceptual diagram*** | ***Description*** |
| *PaymentEntity* | *Payment* | *Contain the payment information.* |
| *CardEntity* | *Card* | *Contain the card information.* |
| *CardInstanceEntity* | *CardInstance* | *Contain the card instance information* |
| *CustomerEntity* | *Customer* | *Contain the customer information.* |
| *ContractEntity* | *Contract* | *Contain the contract information.* |
| *StaffEntity* | *Staff* | *Contain the staff information.* |
| *CompensationEntity* | *Compensation* | *Contain the compensation information.* |
| *PunishmentEntity* | *Punishment* | *Contain the punishment information.* |
| *AccidentEntity* | *Accident* | *Contain the accident information.* |
| *ContractTypeEntity* | *ContractType* | *Contain the contract type information.* |
| *NewCardRequestEntity* | *NewCardRequest* | *Contain the new card request information.* |
| *CardAccessLogEntity* | *N/A* | *Not exist in conceptual diagram. But needed*  *in class diagram to contain the card access log information.* |
| *NotificationEntity* | *N/A* | *Not exist in conceptual diagram. But needed in class diagram to contain the notification*  *information.* |
| *NotificationReadEntity* | *N/A* | *Not exist in conceptual diagram. But needed*  *in class diagram to know what notifications is read.* |

Ví dụ

***Table 11 Class dictionary***

### Class Diagram Explanation

<Mô tả các thành phần cụ thể cho các lớp đã được vẽ ra ở phần trên>

* + - * 1. ***Role***

*Attribute*

|  |  |  |  |
| --- | --- | --- | --- |
| ***Attribute*** | ***Type*** | ***Visibility*** | ***Description*** |
| *RoleID* | *int* | *Private* | *Unique identifier of a role* |
| *Name* | *string* | *Private* | *Role name* |

*Method*

|  |  |  |  |
| --- | --- | --- | --- |
| ***Method*** | ***Return type*** | ***Visibility*** | ***Description*** |
| *Getter* | *Attribute type* | *Public* | *Get attribute value* |
| *Setter* | *Void* | *Public* | *Set value of attribute* |

***4.2.2 ...***

**4.3 Interaction Diagram**

**4.3.x Tên Interaction Diagram**

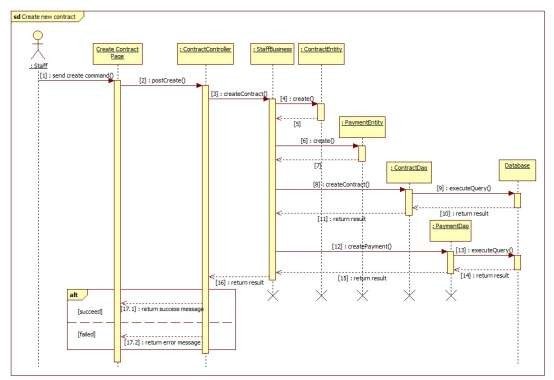
<Sử dụng **sequence diagram là chủ yếu để trình bày nội này**. Sequence diagram cần kết hợp giữa các class đã trình bày ở trên kết hợp với các kiến trúc đã được thuyết minh để có mô hình phù hợp. Đối với ứng **dụng điện thoại di động thì nên sử dụng activity diagram**>

**Summary:** <Nên có phần tóm tắt trước diagram để trình bày về mục đích của diagram trước khi thể hiện hình vẽ>.

Ví dụ

* + - 1. ***Create new contract***

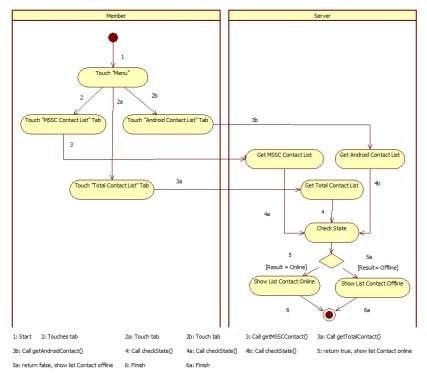
Summary: this diagram show process of staff creates new contract



***Figure 12 Sequence diagram - <Staff> Create new contract***

* + - 1. <Member> View Friend List

***Summary:*** *This diagram shows how member views all contacts that include MSSC contacts and android cell phone contacts.*



* + 1. **Interface**

***Figure 13: <Member> View Friend List***

* + - 1. **Component interface**

<Mô tả các interface như của web service hay các signature của core flow được sử dụng trong hệ thống>

Nội dung được đặc tả theo dạng bảng như sau

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Signature | Description | Input | Output | Output Format | Exception |
| Tên hàm | Mô tả mục đích | Tham số truyền | Kết xuất khi hàm xử lý xong | Kiểu dữ liệu | Xử lý lỗi |

Ví dụ

***Web Service Interface***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Signature* | *Description* | *Input* | *Output* | *Output Format* | *Exception* |
| *public ResponseObject getCheckConnection(R r)* | *Check server status* | *Request object r* | *Json Boolean the status of server* | *Boolean* | *JsonProcessi ngException* |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *...* |  |  |  |  |  |

Ví dụ

### User Interface Design

<Chụp và mô tả màn hình>.

**Lưu ý phải đánh số đặc tả các control trên giao diện cùng với các thành phần trong ràng buộc**

* + - 1. ***Guest Interface Design***
         1. ***Login***



***Fields***

***Figure 14: Login***

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ***No*** | ***Field Name*** | ***Description*** | ***Read only*** | ***Mandatory*** | ***Control Type*** | ***Data Type*** | ***Length*** |
| *1* | *Username* | *Fill user*  *name* | *No* | *Yes* | *Textbox* | *String* | *N/A* |
| *2* | *Password* | *Fill*  *password* | *No* | *Yes* | *Password* | *String* | *N/A* |

***Buttons/Hyperlinks***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***No*** | ***Function*** | ***Description*** | ***Validation*** | ***Outcome*** |
| *3* | *Signin* | *Log-in into the system* | *N/A* | *Transfer to home page* |

* + 1. **Database Design**
       1. **Entity relationship diagram (ERD)**

<Thiết kế ERD. Được suy ra và hình thành từ conceptual diagram, class diagram và quá trình hình thành architectural>

* + - 1. **Data Dictionary**

<Mô tả về các thực thể>

|  |  |
| --- | --- |
| **Entity Data dictionary: describe content of all entities** | |
| **Entity Name** | **Description** |
|  |  |

<Mô tả các thành phần bên trong thực thể>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entity name** | **Attributes** | **Description** | **Domain** | **Null** |
| Tên | Thuộc tính 1 {PK} | Mô tả | Kiểu dữ liệu | Y/N |
| ... | ... | ... | ... |

**Table 12: Detail Data Dictionary**

\* Business integrity constraint:

<Mô tả các ràng buộc về toàn vẹn dữ liệu để đảm bảo nghiệp vụ>

* + 1. **Algorithms**

<Các thành phần thuật toán - các giải pháp để giải quyết phần core flow mà nhóm đã áp dụng>

**Chú ý**

* Không nhất thiết phải là thuật toán nổi tiếng mà có thể là cách tổ chức dữ liệu cũng như giải thuật do nhóm đang thực hiện ở bên trong hệ thống: ghi rõ bản chất, phân tích về độ phức tạp, nếu tham khảo phải ghi rõ nguồn
* Cách giải quyết hay cách áp dụng các qui trình nghiệp vụ hay cách chuyển đổi bài toán khi làm bằng tay - chưa áp dụng máy tính và chương trình để cho thấy việc áp dụng giải bài toán hay giải quyết vấn đề rồi chuyển đổi cách đó sang thành chương trình máy tính

Ví dụ

* 1. ***Document Breakdown***
     1. Definition

*Document breakdown is the way to break the document into many small parts. Each part has it own title and contents of it. And the final data has tree structure.*

* + 1. ***Define Problem***

*All content of document is quite difficute for manage so we must re-construc structure of document for using.*

* + 1. ***Solution***

*To solve this problem, we should follow these steps:*

* + - * *Convert (save) document DOCX file as html type by using Microsoft Word save as Web Filtered.*
      * *Import both html file and directory that incluses all pictures of document.*
      * *Using xpath to get data of html file as we need, include h1, h2, h3,…, image, text content,..*
      * *Save them with structure as below:*

*-TitleA: contentA*

*---TitleA1: contentA1*

*------TitleA1.1: contentA1.1*

*------TitleA1.2: contentA1.2*

*---TitleA2: contentA2*

* + 1. ***Complexity***
       - *In total, the complexity of this algorithm is *
    2. ***Flowchart***

PAGE \\* MERGEFORMAT 1



**Figure 15: Breakdown document flow chart**

#### String Comparison

* + 1. Define Problem

*Given two strings. Calculate their matching percent.*

* + 1. ***Requirement***
* *Robustness to changes of word order: two strings which contain the same words, but in a different order, should be recognised as being similar.*
* *Language independence: the algorithm should work not only in English, but in many different languages.*
  + 1. ***Solution***
* *If a string contains many words, break it into a list of words.*
* *For each word, we find out how many adjacent character pairs are contained in it.*
* *Create a function pairs(s) which returns a list of adjacent character pairs of string s.*
* *Then, we use below formula to calculate matching percent.*
  + 1. ***Example***

*Calculate the matching percent of 2 strings: France and French.*

* + *Upper case 2 strings:*

+ *France FRANCE.*



+ *French FRENCH.*

* + *Break string into list of adjacent character pairs:*

+ *FRANCE*



+ *FRENCH*

* + *Calculate its matching percent.*



1. **System Implementation & Test**
   1. **Introduction**
      1. **Overview**

<Mô tả tống quát mục đích test chủ yếu với thời gian và scope và số lượng nhân lực thì nhóm áp dụng phương pháp gì cho việc test>

Ví dụ

This section provides in detail all necessary information about implementation information and testing procedure of MSSC includes test plans, test cases, test result and risks estimations.

### Test Approach

<Phương pháp kiểm thử của nhóm : black box, white box ...>

* 1. **Database Relationship Diagram**
     1. **Physical Diagram**

<Vẽ database khi cài đặt vật lý trên các RDBMS: chú ý bố cục cũng nhu kích thước cho dễ đọc>

* + 1. **Data Dictionary**

<Mô tả thành phần theo bảng biểu bên dưới>

|  |  |
| --- | --- |
| **Data dictionary: describe content of all tables** | |
| **Table Name** | **Description** |
| Tên | Explanation |

<Mô tả thành phần chi tiết>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entity name** | **Attributes** | **Description** | **Domain** | **Null** |
| Tên | Thuộc tính 1 {PK} | Mô tả | Kiểu dữ liệu | Y/N |
| ... | ... | ... | ... |

**Table 13: Attribute Data Dictionary**

* 1. **Performance Measures**

<Cách nhóm ước lượng việc đo đạc hệ thống>

Ví dụ

* + 1. ***Clustering Performance***
* *Clustering is performed by running K Mean Algorithm which has complexity of : O(n \* k \* I \* d)*
  + *n : number of points*
  + *k : number of cluster*
  + *I : number of iteration*
  + *d : number of attributes (3)*

*Clustering take almost the time of process that we can ignore the time needed to load data from database, digitalize data.*

*The speed of clustering will vary and increase dramatically when n increase. The purpose of this project is not about optimizing K-Mean Algorithm so it is accepted to let the process run till it completes. Moreover, the clustering is designed to run by staff, wait time is acceptable.*

* 1. **Test Plan**

<Đưa ra kế hoạch test>

Ví dụ

The purpose of this section is to verify and ensure that MSSC meets its design specification and other requirements from user. The following part will describe which features to be tested and which will not.

### Features to be tested

<Tính năng sẽ kiểm thử>

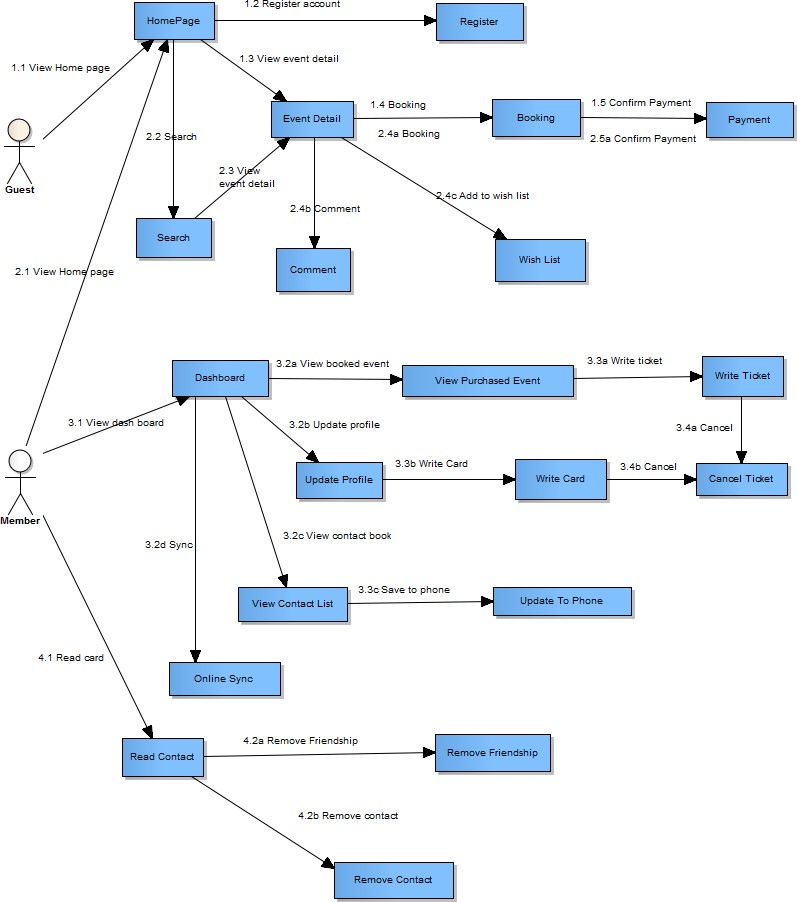
* + 1. **Features not to be tested**

<Tính năng sẽ không kiểm thử>

* 1. **System Testing Test Case**

**<Nên vẽ các workflow tính năng sẽ test để dể hình dung, chú ý dàn trang in ngang, chú ý đánh số, ngày tháng, kết quả, không sao chép>**

Ví dụ



***Figure 16: Guest, Member Core Flow***

MSSC - Introduction

* + 1. ***Guest Test Case***

***5.1.1 Search Event***

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ***ID*** | ***Test Case Description*** | ***Test Case Procedure*** | ***Expected output*** | ***Inter-test Case Dependence*** | ***Result*** | ***Test Date*** | ***Note*** |
|  |  |  |  |  |  |  |  |

MSSC - Introduction

47

1. **Software User’s Manual**
   1. **Installation Guide**
      1. **Setting up environment at server side**

The following software must be installed into the server machine:

* + - 1. **Hardware requirements**

<Yêu cầu phần cứng server, chú ý xem lại các report trước để nhất quán>

* + - 1. **Software requirements**

<Yêu cầu phần mềm server, chú ý xem lại các report trước để nhất quán>

* + 1. **Deployment at server side**

<Mô tả quá trình triển khai lên server thực tế, gợi ý có thể gồm các bước sau, chú ý khi làm phải chụp hình cụ thể để hướng dẫn cũng như so sánh kết quả thành công>

* + - 1. **Prepare deployment package**
      2. **Configure Server before deploy**
      3. **Deploy web application on server**
    1. **Setting up the environment at client side**
       1. **Setting up for computer**

<Ghi rõ phiên bản tối thiểu để sử dụng>

* 1. **User Guide**

<Viết hướng dẫn sử dụng cho người dùng>

**G. Appendix**

<Các thành phần tham khảo của tài liệu chú ý tham khảo thêm cách ghi tại

<http://www.khoahocviet.info/meresci/vi/meresci03d4.html>>