|  |
| --- |
|  |
| Capstone Project Document |

**WingS**

Report #2 – Architecture Design

|  |  |  |
| --- | --- | --- |
| **WingS** | | |
| **Group Members** | Đào Trọng Nghĩa | SE03220 |
| Lê Hồng Nhiên | SE03377 |
| Tạ Ngọc Duy | SE03267 |
| Đỗ Văn Tuấn | SE03297 |
| Hoàng Anh Tuấn | SE03108 |
| **Supervisor** | Mr. Nguyễn Văn Sang | |
| **Event code** | WS | |

**- Hanoi, 10/2016 -**

# SIGNATURE PAGE

AUTHOR: Tạ Ngọc Duy 28/10/2016

Team member

REVIEWERS: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 28/10/2016

Team member

APPROVAL: Nguyễn Văn Sang --/--/--16

Supervisor

Record of change

\*A - Added M - Modified D – Deleted

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Effective Date | Changed Item | A,M,D | Change Description | Reason for Change | Rev. Number |
| 28/Oct/2016 | Create Architecture Design | A | First version | Create Architecture Design | 1.0 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

TABLE CONTENTS

[SIGNATURE PAGE 1](#_Toc468718212)

[1 PROJECT OVERVIEW 4](#_Toc468718213)

[1.1 Purpose 4](#_Toc468718214)

[1.2 Scope 4](#_Toc468718215)

[1.3 Definitions, Acronyms, Abbreviations 4](#_Toc468718216)

[1.4 References 4](#_Toc468718217)

[1.5 Overview 4](#_Toc468718218)

[2 CHOICE OF ARCHITECTURE DESIGN 5](#_Toc468718219)

[2.1 MVC Model 5](#_Toc468718220)

[2.1.1 MVC Model Overview 5](#_Toc468718221)

[2.1.2 Advantages and disadvantages of MVC Model 6](#_Toc468718222)

[2.1.3 The reasons for choosing MVC Model 6](#_Toc468718223)

[2.2 .Net Framework 8](#_Toc468718224)

[2.2.1 .Net Framework Overview 8](#_Toc468718225)

[2.2.2 Advantages and disadvantages of .Net Framework 8](#_Toc468718226)

[2.2.3 The reasons for choosing .Net Framework 8](#_Toc468718227)

[2.3 AngularJS 9](#_Toc468718228)

[2.3.1 AngularJS Overview 9](#_Toc468718229)

[2.3.2 Advantages and disadvantages of AngularJS 11](#_Toc468718230)

[2.3.3 The reason for choosing AngularJS 11](#_Toc468718231)

[3 ARCHITECTURAL REPRESENTATION 12](#_Toc468718232)

[4 ARCHITECTURAL GOALS AND CONSTRAINTS 13](#_Toc468718233)

[5 Use-case View 14](#_Toc468718234)

[5.1 User Group Function 14](#_Toc468718235)

[5.1.1 Guest/ Registered Group Function 14](#_Toc468718236)

[5.1.2 Guest Group Function 15](#_Toc468718237)

[5.1.3 Registered Group Function 16](#_Toc468718238)

[5.2 Administrator Group Function 23](#_Toc468718239)

[5.2.1 Manage user's account 23](#_Toc468718240)

[5.2.2 Common Module 24](#_Toc468718241)

[5.2.3 Manage Event 25](#_Toc468718242)

[5.2.4 Manage Thread 28](#_Toc468718243)

[5.2.5 Manage Message 29](#_Toc468718244)

[5.2.6 Manage Report 30](#_Toc468718245)

[6 Logical View 31](#_Toc468718246)

[6.1 Overview 31](#_Toc468718247)

[6.2 Architecturally Significant Design Packages 32](#_Toc468718248)

[7 Process view 36](#_Toc468718249)

[8 Deployment View 38](#_Toc468718250)

[9 Quality 38](#_Toc468718251)

# PROJECT OVERVIEW

## Purpose

This document provides a comprehensive architectural overview of the system, using a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions that have been made on the system.

## Scope

The scope of this document is to depict the architecture of the WingS website for Vietnamese created by WS capstone project team.

## Definitions, Acronyms, Abbreviations

|  |  |  |
| --- | --- | --- |
| **Acronym** | **Definition** | **Note** |
| WS | WingS |  |
| DB | Database |  |
| MVC | Model view control |  |
| IDE | Integrated development environment |  |
| Q&A | Question and answer |  |
| GUI | Graphic user interface |  |
| Admin | Administrator |  |

Table ‑: Definitions and Acronyms

## References

* WS\_ Software Requirements Specification\_v1.0\_EN.docx
* WS\_Data Design\_v1.0\_EN.docx
* Software Architecture Design Illuminated Book

## Overview

The Software Architecture Document contains the following subsections:

* **Section 1**: Provide an overview of entire Software Architecture Document.
* **Section 2**: Choice of Architecture Design
* **Section 3**: Architectural Representation
* **Section 4**: Architectural Goals and Constraints
* **Section 5**: Use-Case view
* **Section 6**: Logical View
* **Section 7**: Process View
* **Section 8**: Deployment view
* **Section 9**: Quality

# CHOICE OF ARCHITECTURE DESIGN

## MVC Model

The purpose of WS is developing as a charity website, where people can start their ideas and make it real. The system of WS is structured based on MVC combined with layered architecture.

### MVC Model Overview

The **model-view-controller or MVC** is software architecture commonly used for creating web applications or software. In other words, it's a structure for web applications to follow in order to ensure efficiency and consistency. Many of the most popular frameworks use the MVC architecture, including ASP.NET. At the same time, there are many web developers who don't use a coding framework yet still set up their applications to follow the MVC structure.

The Model-View-Controller (MVC) design pattern assigns objects in an application one of  
three roles: model, view, or controller. The pattern defines not only the roles objects play in  
the application, it defines the way objects communicate with each other. Each of the three  
types of objects is separated from the others by abstract boundaries and communicates with  
objects of the other types across those boundaries. The collection of objects of a certain MVC  
type in an application is sometimes referred to as a layer—for example, model layer.

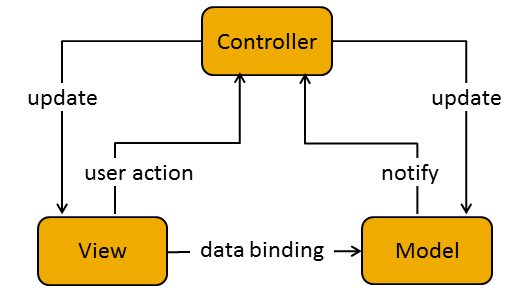


Figure ‑: MVC Model

In addition to dividing the application into three kinds of components, the MVC design defines the interactions between them:

* **A controller:** can send commands to its associated view to change the view's presentation of the model (e.g., by scrolling through a document). It can also send commands to the model to update the model's state (e.g., editing a document).
* **A model:** notifies its associated views and controllers when there has been a change in its state. This notification allows the views to produce updated output, and the controllers to change the available set of commands. A passive implementation of MVC omits these notifications, because the application does not require them or the software platform does not support them.
* **A view:** requests from the model the information that it needs to generate an output representation to the user.

### Advantages and disadvantages of MVC Model

* Advantages:
* The MVC model demonstrates professionalism in programming and design analysis. It is divided into independent components to help develop applications faster, simpler, easier upgrades and maintenance.
* Many MVC vendor framework tool kits are available.
* Multiple views synchronized with same data model.
* Easy to change or plug in new interface views, allowing updating of interface views with new technologies without overhauling the rest of system.
* Develop tools is useful and easy to use.
* Large of documentary sources.
* Disadvantages:
* For small projects that apply MVC model caused cumbersome, time consuming in development process.
* Time consuming to transits data between components.
* Not suitable for agent-oriented applications such as interactive mobile and robotics applications.
* Multiple pairs of controllers and views based on the same data model make data model change expensive.
* The division between the View and the Controller is not clear in some cases.

### The reasons for choosing MVC Model

* Better support for test-driven development.
* It is good support for application built by project team has many developers and designers but still managed application features.
* Better support for test-driven development.
* Tools is useful and documentary source is large makes MVC is easy to develop.
* WS system is not complete system, now. We built the system that towards extensibility and maintainability in the future.

## .Net Framework

### .Net Framework Overview

.NET Framework is a software framework developed by Microsoft that runs primarily on Microsoft Windows. It includes a large class library known as Framework Class Library (FCL) and provides language interoperability (each language can use code written in other languages) across several programming languages.

The .NET Framework is a technology that supports building and running the next generation of applications and XML Web services. The .NET Framework is designed to fulfill the following objectives:

* To provide a consistent object-oriented programming environment whether object code is stored and executed locally, executed locally but Internet-distributed, or executed remotely.
* To provide a code-execution environment that minimizes software deployment and versioning conflicts.
* To provide a code-execution environment that promotes safe execution of code, including code created by an unknown or semi-trusted third party.
* To provide a code-execution environment that eliminates the performance problems of scripted or interpreted environments.
* To make the developer experience consistent across widely varying types of applications, such as Windows-based applications and Web-based applications.
* To build all communication on industry standards to ensure that code based on the .NET Framework can integrate with any other code.

### Advantages and disadvantages of .Net Framework

* Advantages
* Less time to produce product
* Less Complexity.
* Easily to access complex O.S functions
* Easily to build Data Oriented Project, support huge DB functions.
* Managed
* Support Both Windows and Web Application.
* Easy to create Dynamic sites.
* Disadvantages
* Not suitable for High End Application
* Low performance compare to C, C++.
* Unavailability of build in methods.
* .NET framework is free to download but Code Editor is costly.
* Only few O.S supports .NET.

### The reasons for choosing .Net Framework

* Consistent with WS system.
* There are many plugins and resources which support creating a website using MVC model.
* Many members can use and have experiences using C# (a language of .Net Framework)

## AngularJS

### AngularJS Overview

AngularJS is a structural framework for dynamic web apps. It lets you use HTML as your template language and lets you extend HTML's syntax to express your application's components clearly and succinctly. Angular's data binding and dependency injection eliminate much of the code you would otherwise have to write. And it all happens within the browser, making it an ideal partner with any server technology.

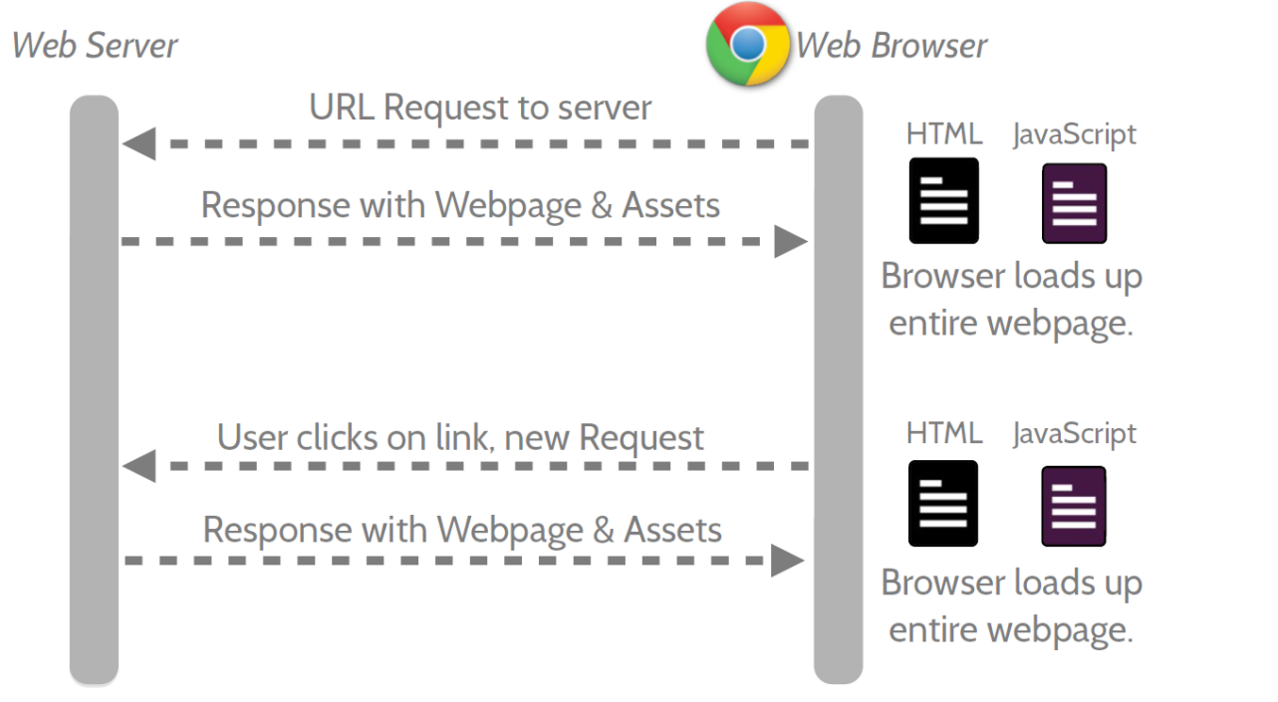


Figure ‑: Multi Page Application



Figure ‑: Single Page Application

* Features:
* AngularJS is a powerful JavaScript based development framework to create RICH Internet Application (RIA).
* AngularJS provides developers options to write client side application (using JavaScript) in a clean MVC (Model View Controller) way.
* Application written in AngularJS is cross-browser compliant. AngularJS automatically handles JavaScript code suitable for each browser.
* AngularJS is open source, completely free, and used by thousands of developers around the world. It is licensed under the Apache License version 2.0.

Overall, AngularJS is a framework to build large scale and high performance web application while keeping them as easy-to-maintain.

* Core futures: following are most important core features of AngularJS:
* **Data-binding:** It is the automatic synchronization of data between model and view components.
* **Scope:** These are objects that refer to the model. They act as a glue between controller and view.
* **Controller:** These are JavaScript functions that are bound to a particular scope.
* **Services:** AngularJS come with several built-in services for example $http to make a XMLHttpRequests. These are singleton objects which are instantiated only once in app.
* **Filters:** These select a subset of items from an array and returns a new array.
* **Directives:** Directives are markers on DOM elements (such as elements, attributes, css, and more). These can be used to create custom HTML tags that serve as new, custom widgets. AngularJS has built-in directives (ngBind, ngModel...)
* **Templates:** These are the rendered view with information from the controller and model. These can be a single file (like index.html) or multiple views in one page using "partials".
* **Routing:** It is concept of switching views.
* **Model View Whatever:** MVC is a design pattern for dividing an application into different parts (called Model, View and Controller), each with distinct responsibilities. AngularJS does not implement MVC in the traditional sense, but rather something closer to MVVM (Model-View-ViewModel). The Angular JS team refers it humorously as Model View Whatever.
* **Deep Linking:** Deep linking allows you to encode the state of application in the URL so that it can be bookmarked. The application can then be restored from the URL to the same state.
* **Dependency Injection:** AngularJS has a built-in dependency injection subsystem that helps the developer by making the application easier to develop, understand, and test.

### Advantages and disadvantages of AngularJS

* Advantages:
* AngularJS provides capability to create Single Page Application in a very clean and maintainable way.
* AngularJS provides data binding capability to HTML thus giving user a rich and responsive experience
* AngularJS code is unit testable.
* AngularJS uses dependency injection and make use of separation of concerns.
* AngularJS provides reusable components.
* With AngularJS, developer write less code and get more functionality.
* In AngularJS, views are pure html pages, and controllers written in JavaScript do the business processing.
* Disadvantages:
* **Not Secure:** Being JavaScript only framework, application written in AngularJS are not safe. Server side authentication and authorization is must to keep an application secure.
* **Not degradable:** If your application user disables JavaScript then user will just see the basic page and nothing more.

### The reason for choosing AngularJS

* AngularJS supports to create a website faster via Rest Api and MVC Model.
* Provide a more fluid user experience akin to a desktop application.
* All team members want to study a new language.

# ARCHITECTURAL REPRESENTATION

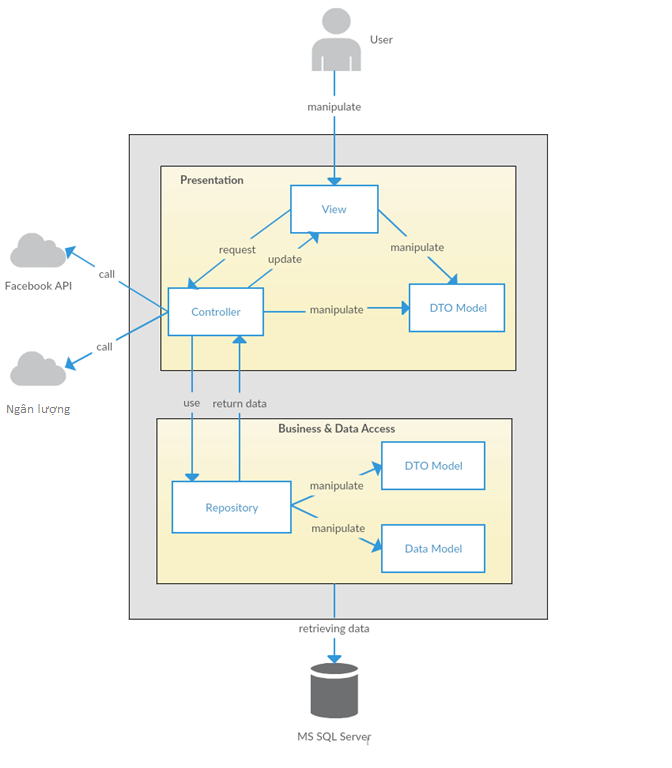


Figure ‑: System Overview

We follow MVC architecture to implement the WS Project. MVC offers architectural benefits over standard Jquery and AngularJS — it helps us write better-organized and therefore more maintainable code.

* **DTO Model (data transfer object model)** is an object that defines how the data will be sent over the network to remove circular references from data model, Hide particular properties that clients are not supposed to view, omit some properties in order to reduce payload size, flatten object graphs that contain nested objects, to make them more convenient for clients, avoid “over-posting” vulnerabilities and decouple your service layer from your database layer.
* **View** is what is presented to the users and how users interact with the system. The view is expected to render the model in a meaningful way to the user. In WS, the view is made with .cshtml file including css, AngularJS and jQuery, it sends user gestures to controller and allows controller to select view.
* **Controller** is the decision maker and the glue between the model and view; it handles user actions and gestures, and responds to user events. For example, in CMS, when a user clicks the “Create” button to create a new contract, the controller for that action is invoked. The controller will then make changes to the contract model. The view will then render the modified contract model to the display so that user can view the new contract he added in the contract list.
* **Data Model** is where the application’s data objects are stored. A model object is in charge of encapsulating application state and one object could be related to other objects establishing a one-to-one or one-to-many relationship.
* **Repository** is intermediate layer which used to separate the controller and the data access layer (database context). It queries the data source for data, maps it to DTO models, processes data and returns data to controller.

# ARCHITECTURAL GOALS AND CONSTRAINTS

* **Availability:**
* The application must be available 95% of time. Users can access to it everywhere from there .Web browser with internet connection.
* **Maintainability:**
  + Coding standards and naming conventions:
    - Output of the project must include coding standards and naming conventions documentations. Implementation code must be easy to maintain.
    - All code must be clearly commented, including class, method documentations.
    - If some components are reused, the documentations of those components must also be included.
* Design:
  + - The design of the system must be loosely coupled that chances on some module will not affect others.
* Logging:
  + - All the errors should be logged, supporting for bug fixing and maintenance.
    - All strange or sensitive situations should also be logged.
* **Usability:**
  + Intuitiveness: all help/error messages are simple to understand; user can know exactly how to do each feature after one time using it.
* **Capacity and scalability:**
  + Throughput, storage and growth requirements.

# Use-case View

* **This application includes two parts:**
* The first part is User module. User module includes registered and guest.
* Next part is Administrator module.

## User Group Function

### Guest/ Registered Group Function

#### Common Module

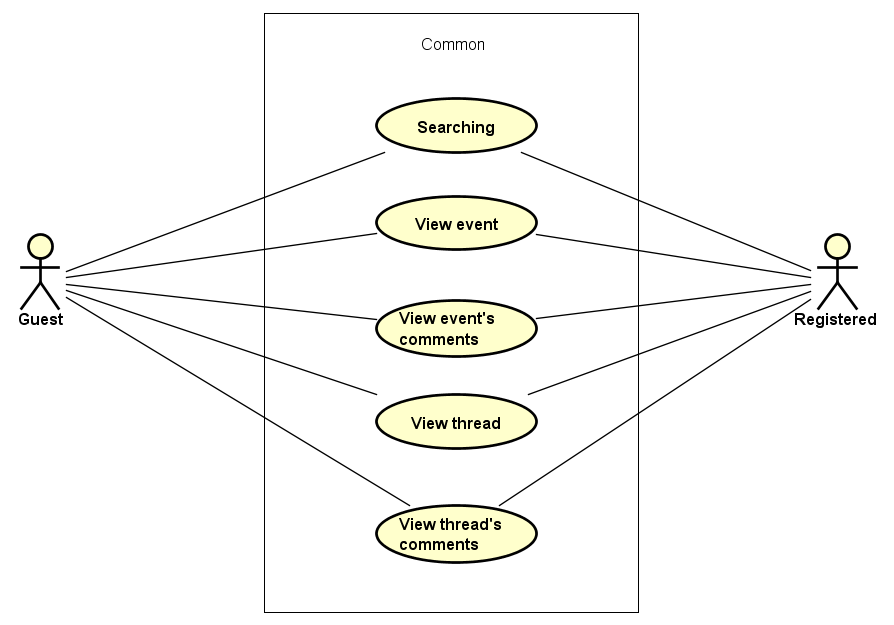


Figure ‑: Common Module

|  |  |  |  |
| --- | --- | --- | --- |
| No | Use-case name | Actor | Description |
| 1 | Searching | Guest & Registered user | Search existing information about User/Account/Event/Thread/  Organization |
| 2 | View Event | Guest & Registered user | Show all existing events stored on WS |
| 3 | View comment in event | Guest & Registered user | This function allows user to view all comments in event |
| 4 | View Thread | Guest & Registered user | Show all existing threads stored on WS |
| 5 | View comment in thread | Guest & Registered user | This function allows user to view all comments in thread |

Table ‑: Common Module UC

### Guest Group Function

#### Common ModuleGuestOnly

Figure ‑2: Guest’s Common Module

|  |  |  |  |
| --- | --- | --- | --- |
| No | Use-case name | Actor | Description |
| 1 | Register | Guest | Create new account to use more function as a register user |

Table ‑2: Guest’s Common Module UC

### Registered Group Function

#### Common Module

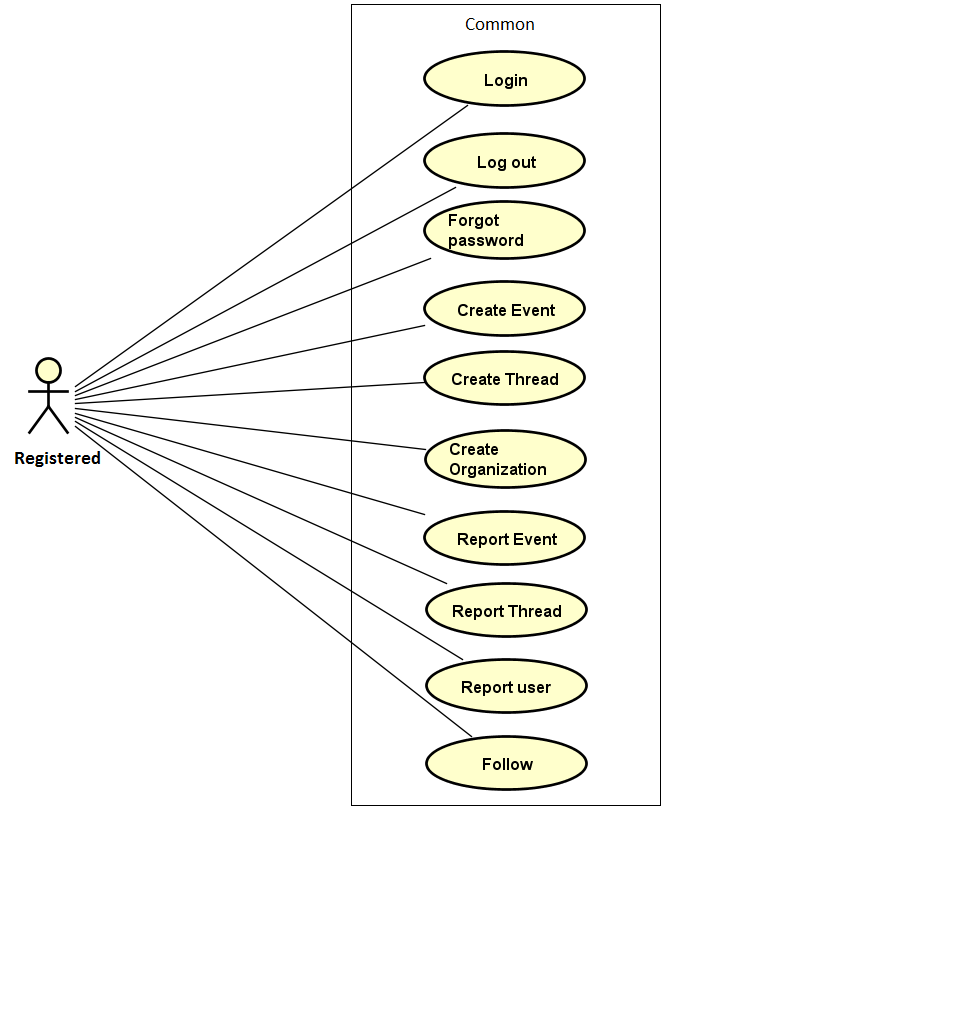


Figure ‑3: Registered Common Module

|  |  |  |  |
| --- | --- | --- | --- |
| No | Use-case name | Actor | Description |
| 1 | Login | Registered User | User login by Facebook/ Register account to use more functions in website |
| 2 | Logout | Registered User | User want to logout their account |
| 3 | Forgot password | Registered User | When User forgot their password to login the system, this function will receiving their password to email |
| 4 | Create event | Registered User | This function allows user to create an event |
| 5 | Create thread | Registered User | This function allows user to create a thread |
| 6 | Create organization | Registered User | This function allows user to create a organization |
| 7 | Report event | Registered User | This function allows user to report an event |
| 8 | Report thread | Registered User | This function allows user to report a thread |
| 9 | Report user | Registered User | This function allows user to report an user |
| 10 | Follow |  | This function allows user to follow an event and receive notification about this event |

Table ‑3: Registered Common Module UC

#### View Personal page

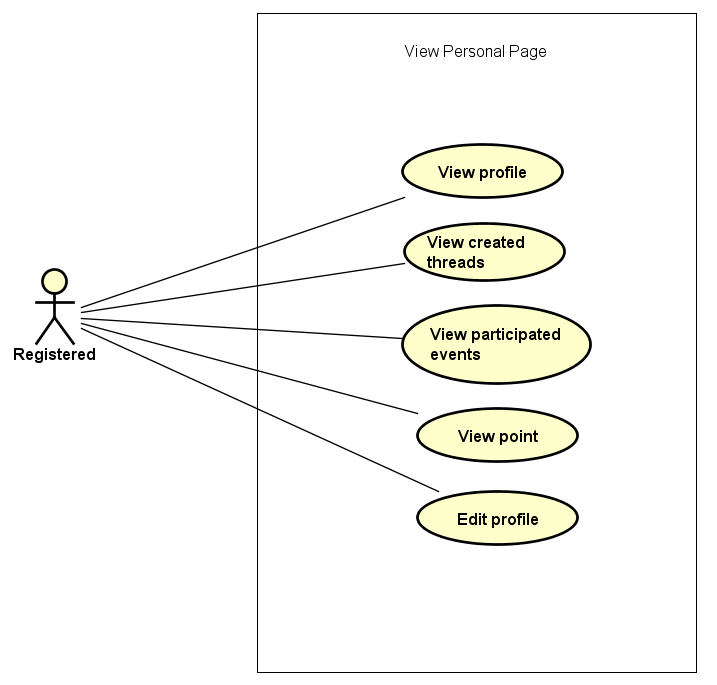


Figure ‑4: View Personal Page

|  |  |  |  |
| --- | --- | --- | --- |
| No | Use-case name | Actor | Description |
| 1 | View profile | Registered User | Display all user profile |
| 2 | Display created threads | Registered User | WS records all thread which user created |
| 3 | Display number donated events | Registered User | WS records all events which user donated |
| 4 | View activities point | Registered User | WS records all user’s activities and display point base on it. |
| 5 | Edit Profile | Registered User | This function allows user to edit profile |
| 6 | Change Password | Registered User |  |

Table ‑4: View Personal Page UC

#### Message Module

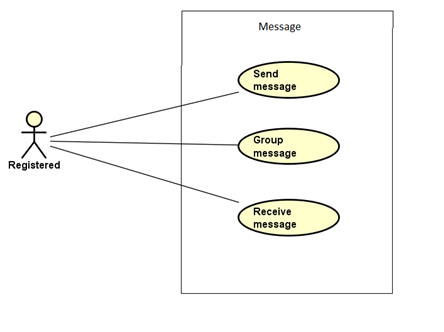


Figure ‑5: Message Module Page

|  |  |  |  |
| --- | --- | --- | --- |
| No | Use-case name | Actor | Description |
| 1 | Send message | Registered User | This function allows user to send a message |
| 2 | Group message | Registered User | This function allows user to group kinds of message to view (All, unread, read) |
| 3 | Receive message | Registered User | This function allows user to receive a message |

Table ‑5: Message UC

#### Chat room Module

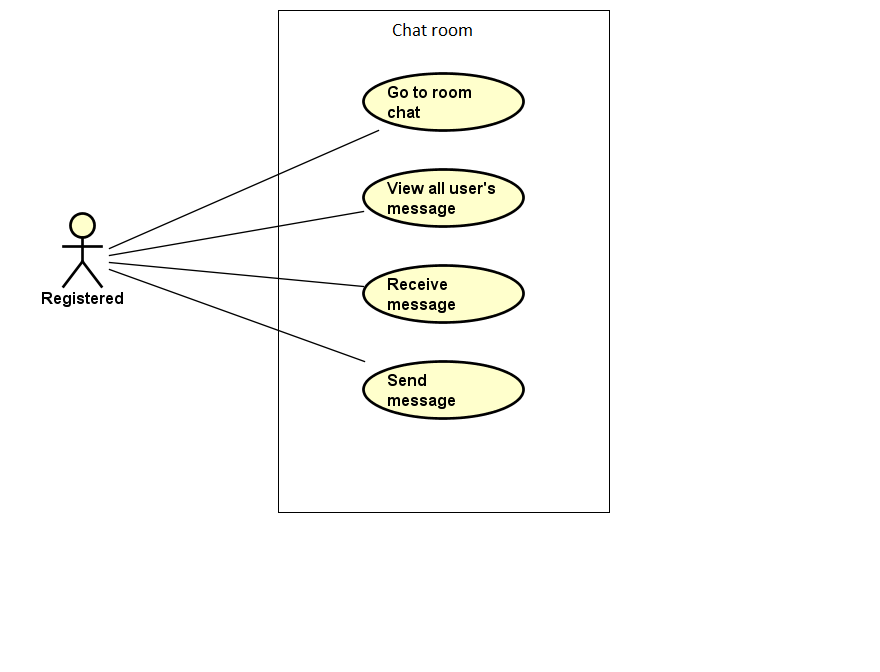


Figure ‑: Chat room

|  |  |  |  |
| --- | --- | --- | --- |
| No | Use-case name | Actor | Description |
| 1 | Go to room chat | Registered User | This function allows user to connect to room chat |
| 2 | View all message | Registered User | This function allows user to view all message in room chat |
| 3 | Receive message | Registered User | This function allows user to receive message of all user who connect to this room chat |
| 4 | Send message | Registered User | This function allows user to send a message |

Table ‑6: Chat room Module

#### Like Module

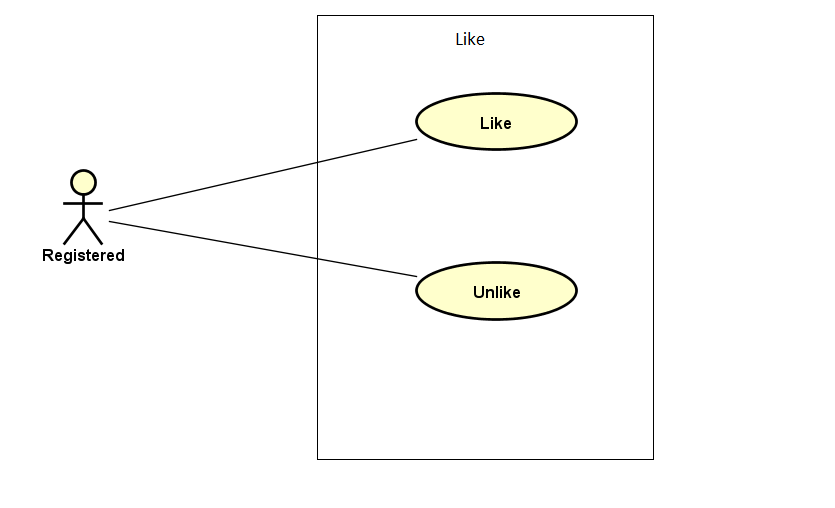


Figure ‑: Like Module

|  |  |  |  |
| --- | --- | --- | --- |
| No | Use-case name | Actor | Description |
| 1 | Like | Registered User | This function allows user to express their feeling by click like an event or thread |
| 2 | Unlike | Registered User | This function allows user to undo their like action before. |

Table ‑: Like Module

#### Comment Module

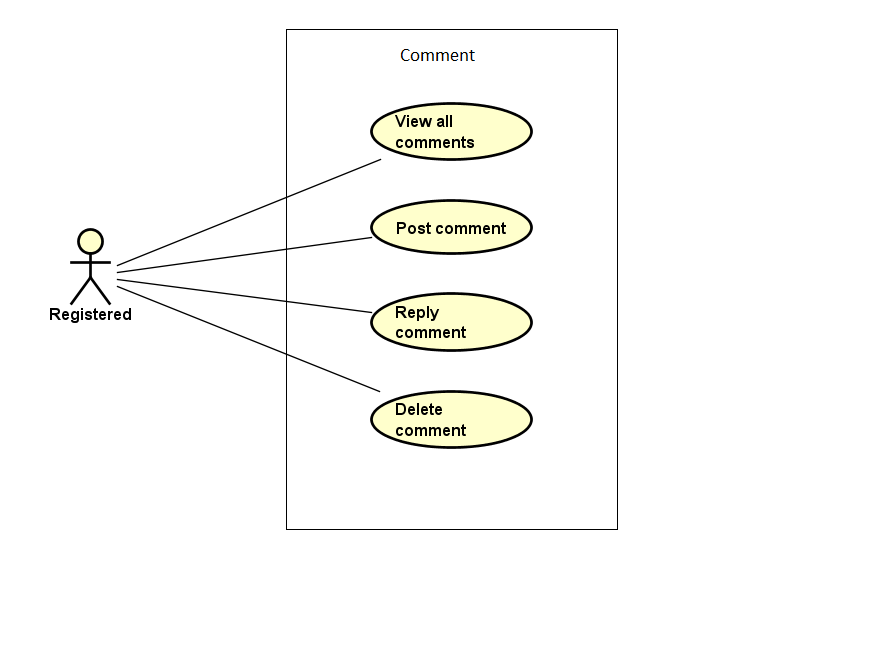


Figure ‑8: Comment Module

|  |  |  |  |
| --- | --- | --- | --- |
| No | Use-case name | Actor | Description |
| 1 | View list comment | Registered User | WS show all comment of a project |
| 2 | Post comment | Registered User | User post a comment |
| 3 | Reply comment | Registered User | User reply a comment |
| 4 | Delete comment | Registered User | User edit comment |

Table ‑8: Comment Module

## Administrator Group Function

### Manage user's account

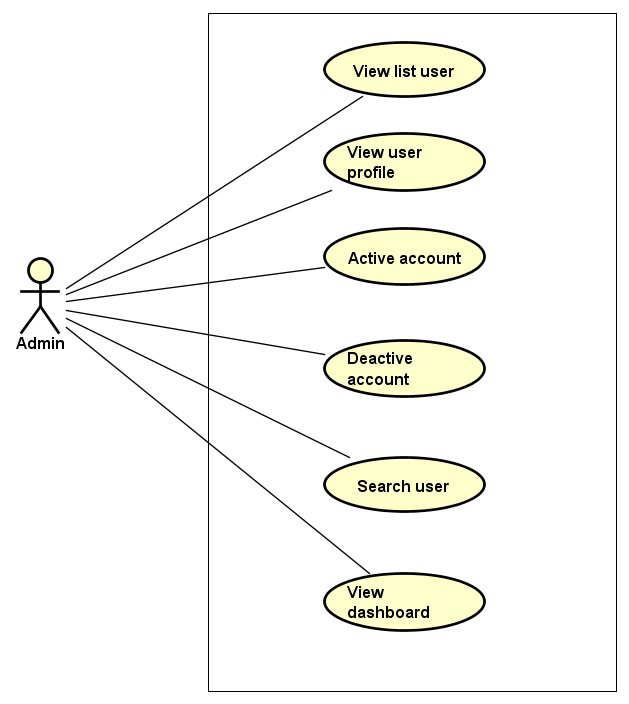


Figure ‑9: Manage user's account

|  |  |  |  |
| --- | --- | --- | --- |
| No | Use-case name | Actor | Description |
| 1 | View user list | Administrator | This function allows Administrator to view User list of the project |
| 2 | View user profile | Administrator | This function allows Administrator to view profile of an user |
| 3 | Activate user's account | Administrator | This function allows Administrator to activate an user’s account |
| 4 | Deactivate user's account | Administrator | This function allows Administrator to deactivate an user’s account |
| 5 | Search user | Administrator | This function allows Administrator to search user on the user list |
| 6 | View user’s dashboard | Administrator | This function allows Administrator to view dashboard about user of the event |

Table ‑9: Manage user's account UC

### Common Module

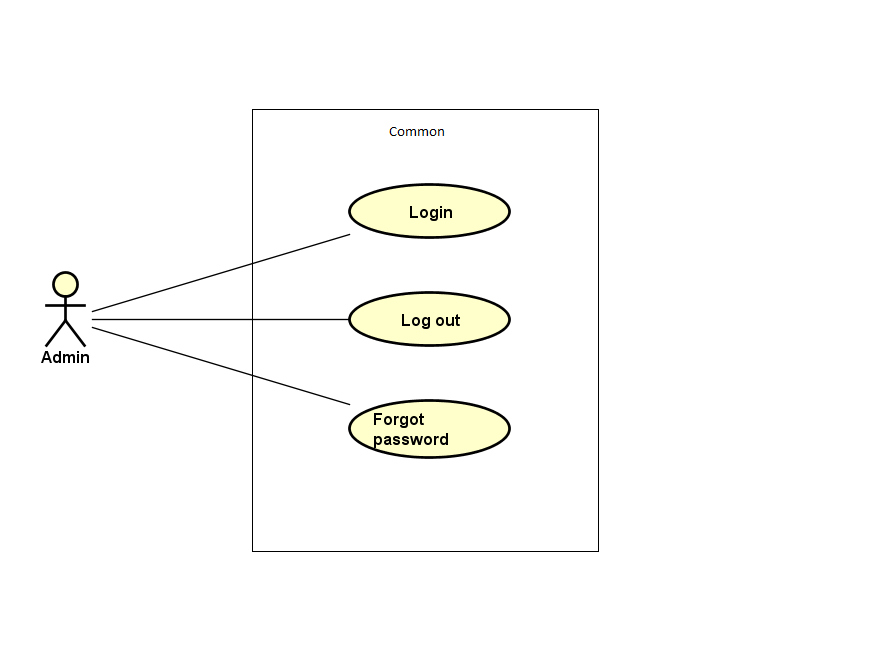


Figure ‑10: Common module

|  |  |  |  |
| --- | --- | --- | --- |
| No | Use-case name | Actor | Description |
| 1 | Admin login | Administrator | This function allows Admin logins into website |
| 2 | Admin logout | Administrator | This function allows Admin logouts of WS website |
| 3 | Admin forgot password | Administrator | This function allows Admin receiving her/ his password to email |

Table ‑10: Common module UC

### Manage Event

#### Manage Event

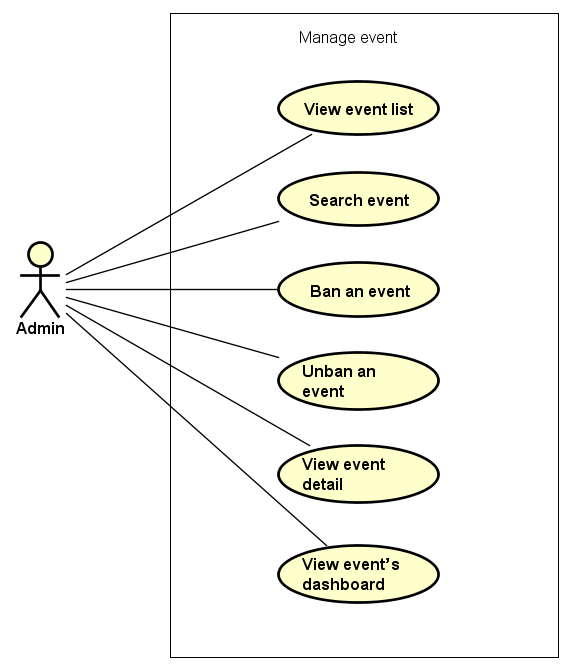


Figure ‑: Manage event

|  |  |  |  |
| --- | --- | --- | --- |
| No | Use-case name | Actor | Description |
| 1 | View event list | Administrator | This function allows Administrator to view project list of the event |
| 2 | Search event | Administrator | This function allows Administrator to search an event |
| 3 | Ban an event | Administrator | This function allows Administrator ban an event |
| 4 | Unban an event | Administrator | This function allows Administrator to Unban an event |
| 5 | View event detail | Administrator | This function allows Administrator to view information of an event |
| 6 | View event’s dashboard | Administrator | This function allows Administrator to view dashboard event |

Table ‑: Manage events UC

#### Manage Event Categories

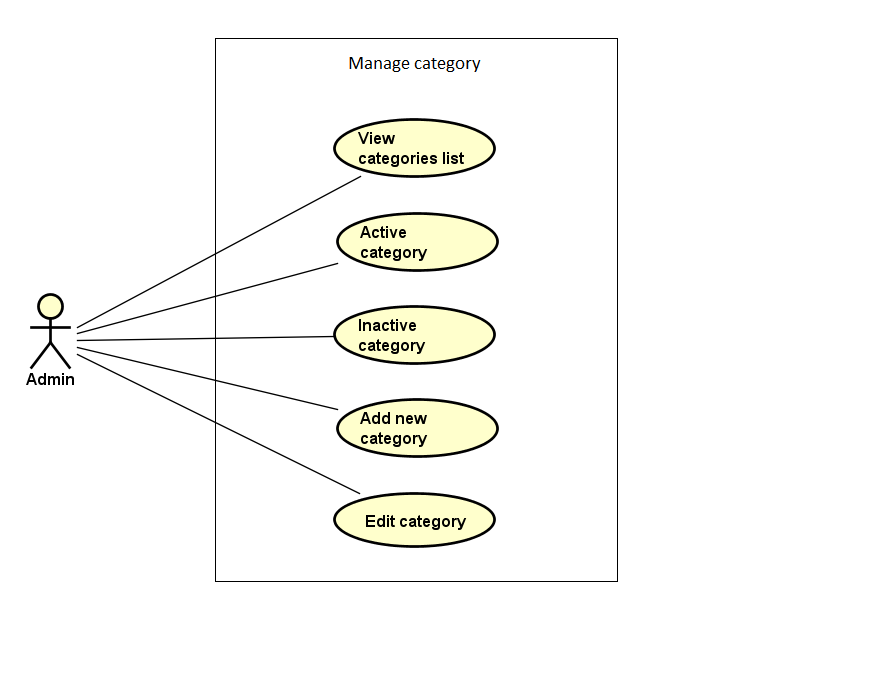


Figure ‑: Manage Categories

|  |  |  |  |
| --- | --- | --- | --- |
| No | Use-case name | Actor | Description |
| 1 | View category list | Administrator | This function allows Administrator to view Category list of the event |
| 2 | Activate category | Administrator | This function allows Administrator to activate a category |
| 3 | Deactivate category | Administrator | This function allows Administrator to deactivate a category |
| 4 | Add new category | Administrator | This function allows Administrator to add new a category |
| 5 | Edit category | Administrator | This function allows Administrator to edit a category |

Table ‑: Manage Categories

### Manage Thread

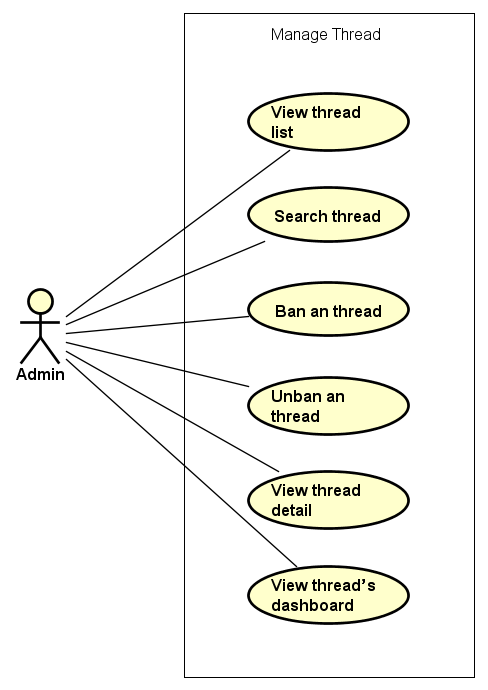


Figure ‑: Manage thread

|  |  |  |  |
| --- | --- | --- | --- |
| No | Use-case name | Actor | Description |
| 1 | View thread list | Administrator | This function allows Administrator to view project list of the event |
| 2 | Search thread | Administrator | This function allows Administrator to search an event |
| 3 | Ban an thread | Administrator | This function allows Administrator ban an event |
| 4 | Unban an thread | Administrator | This function allows Administrator to Unban an event |
| 5 | View thread detail | Administrator | This function allows Administrator to view information of an event |
| 6 | View thread’s dashboard | Administrator | This function allows Administrator to view dashboard about thread |

Table ‑: Manage thread

### Manage Message

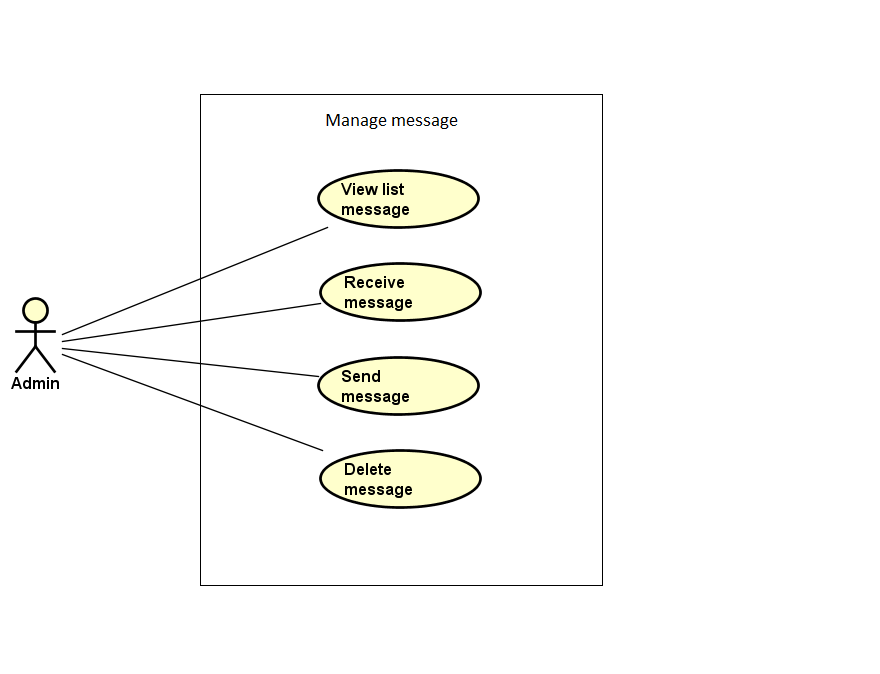


Figure ‑: Manage Message

|  |  |  |  |
| --- | --- | --- | --- |
| No | Use-case name | Actor | Description |
| 1 | View message list | Administrator | This function allows Administrator to view Message list of the project |
| 2 | Send message | Administrator | This function allows Administrator to send a message |
| 3 | Receive message | Administrator | This function allows Administrator to view a message |
| 4 | Delete message | Administrator | This function allows Administrator to delete a message |

Table ‑: Manage Message

### Manage Report

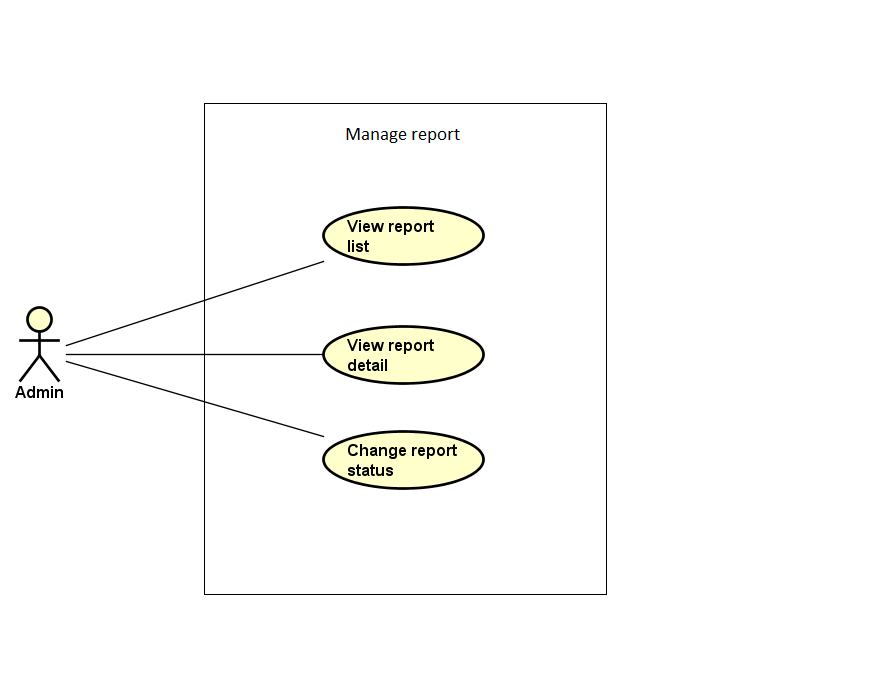


Figure ‑: Manage Report

|  |  |  |  |
| --- | --- | --- | --- |
| No | Use-case name | Actor | Description |
| 1 | View report list | Administrator | This function allows Administrator to view Report list of the event, thread, user or organization |
| 2 | View report detail | Administrator | This function allows Administrator to view information of a report |
| 3 | Change report's status | Administrator | This function allows Administrator to change status of a report if it has been solved |

Table ‑: Manage Report

# Logical View

## Overview

Logical View includes Package diagram and Class diagram. Package diagram describes the organization of packages and elements. Class Diagram provides an overview of the target system by describing the objects and classes inside the system and the relationships between them. It provides a wide variety of usages; from modeling the domain-specific data structure to detailed design of the target system

* + Controller contain the interface between:
    - Associated models
    - Associated views
    - The input devices (e.g., keyboard, pointing device, time).
    - Send commands to the model to update the model's state.
  + Model is:
    - the domain-specific software simulation
    - Or implementation of the application's central structure.
  + View: deal with everything graphical:
    - Requests data from their model
    - Display the data.
  + Repository:
    - Create queries to DB.
    - Process data.
    - Return to controller.

## Architecturally Significant Design Packages

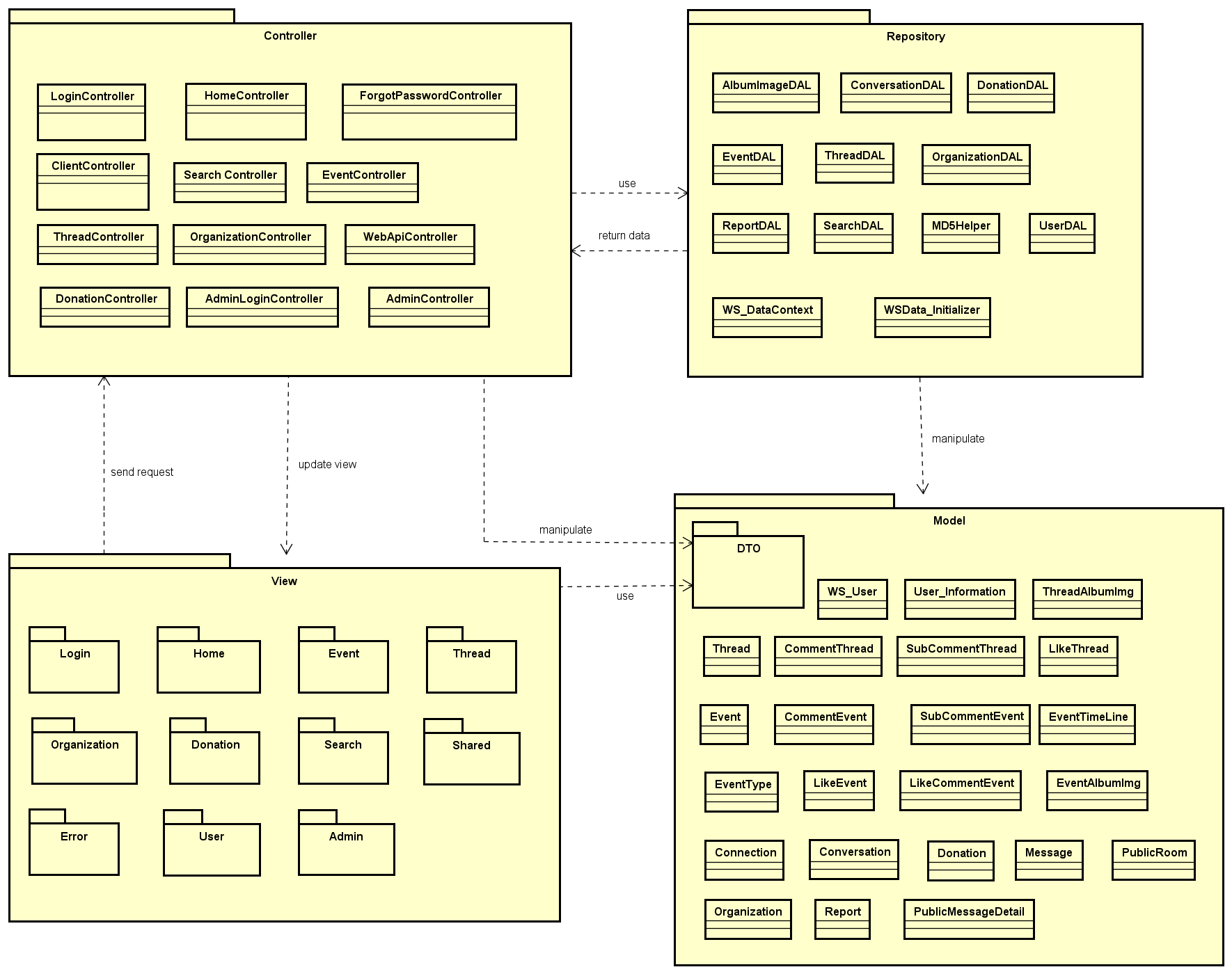


Figure ‑: Package Diagram

* **Model:**

|  |  |  |
| --- | --- | --- |
| No | Model class | Role |
|  | WS\_User | Description entity of User account information in database |
|  | User\_Information | Description entity of User information in database |
|  | Thread | Description entity of Thread in database |
|  | ThreadAlbumImg | Description entity of Album image of event in database |
|  | CommentThread | Description entity of Comment in thread in database |
|  | SubCommentThread | Description entity of SubComment in thread in database |
|  | LikeThread | Description entity of Like in thread in database |
|  | Event | Description entity of Event in database |
|  | EventTimeLine | Description entity of the schedule of event in database |
|  | CommentEvent | Description entity of Comment in event in database |
|  | SubCommentEvent | Description entity of SubComment in event in database |
|  | EventType | Description entity of Event’s category in database |
|  | LikeEvent | Description entity of Like in event in database |
|  | EventAlbumImg | Description entity of Album image of event in database |
|  | Connection | Description entity of Connection of user when connect to system in database |
|  | Conversation | Description entity of Conversation of event chatroom in database |
|  | Donation | Description entity of donate information in database |
|  | Message | Description entity of Message of user in database |
|  | PublicRoom | Description entity of Public room chat in database |
|  | Organization | Description entity of Organization in database |
|  | Report | Description entity of Report in database |
|  | PublicMessageDetail | Description entity of Public Message Detail of chatroom in database |

Table ‑: Model list

* **Controller:**

|  |  |  |
| --- | --- | --- |
| No | Controller class | Role |
|  | LoginController | * Receive request login, logout from client. * Call method login, logout. * Respond login view and login, logout status. |
|  | HomeController | * Receive request to home page from client. * Handle request from client and call methods to get all data of home page. * Respond data back to Home View. |
|  | ClientController | * Receive request get client partial views of from client. * Generate partial views and respond to client. |
|  | ForgotPassword Controller | * Receive request get new password from client. * Call methods to get. * Respond data back. |
|  | Search Controller | * Receive request search from client. * Handle request from client and call methods to get all data which match with searching key * Respond data back to Search View |
|  | EventController | * Receive request to event page from client. * Handle request from client and call methods to get all data of event page. * Respond data back to Event view |
|  | ThreadController | * Receive request to thread page from client. * Handle request from client and call methods to get all data of thread page. * Respond data back to Thread View. |
|  | OrganizationController | * Receive request to organization page from client. * Handle request from client and call methods to get all data of organization page. * Respond data back to organization View. |
|  | WebApiController | * Receive requests (get, post, push, delete) about information from client. * Respond status, message and json data to client. |
|  | DonationController | * Receive request to Donation page from client. * Handle request from client and call methods to get all data of Donation page. * Respond data back to Donation View. |
|  | AdminLoginController | * Receive request login, logout from admin. * Call method login, logout. * Respond login view and login, logout status. |
|  | AdminController | * Receive request get admin partial views of admin user. * Generate partial views and respond to admin. |

Table ‑: Controller list

* **Repository:**

|  |  |  |
| --- | --- | --- |
| No | Repository class | Role |
|  | WS\_DataContext | * Connect to DB, create queries to select, update, deletes data. |
|  | WSData\_Initializer | * Initialize data. |
|  | AlbumImageDAL | * Process album image and return to controller. |
|  | ConversationDAL | * Process conversation data and return to controller. |
|  | DonationDAL | * Process donation data and return to controller. |
|  | EventDAL | * Process event data and return to controller. |
|  | ThreadDAL | * Process thread data and return to controller. |
|  | OrganizationDAL | * Process organization data and return to controller. |
|  | ReportDAL | * Process report data and return to controller. |
|  | SearchDAL | * Process search data and return to controller. |
|  | MD5DAL | * Process password initialize data and return to controller. |

Table ‑3: Repository list

* **DTOs:**

Include many data transfer object classes.

* **View:**

Include many .cshtml file

# Process view

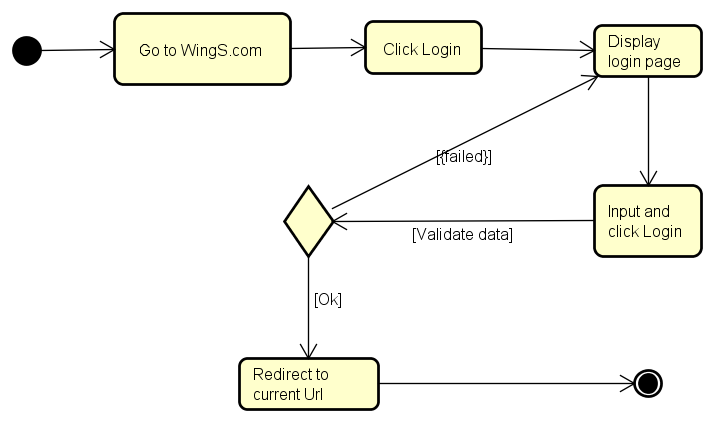


Figure ‑: Login activity diagram

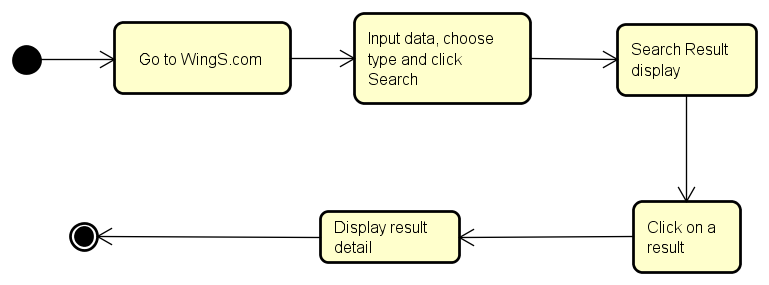


Figure ‑: Searching activity diagram

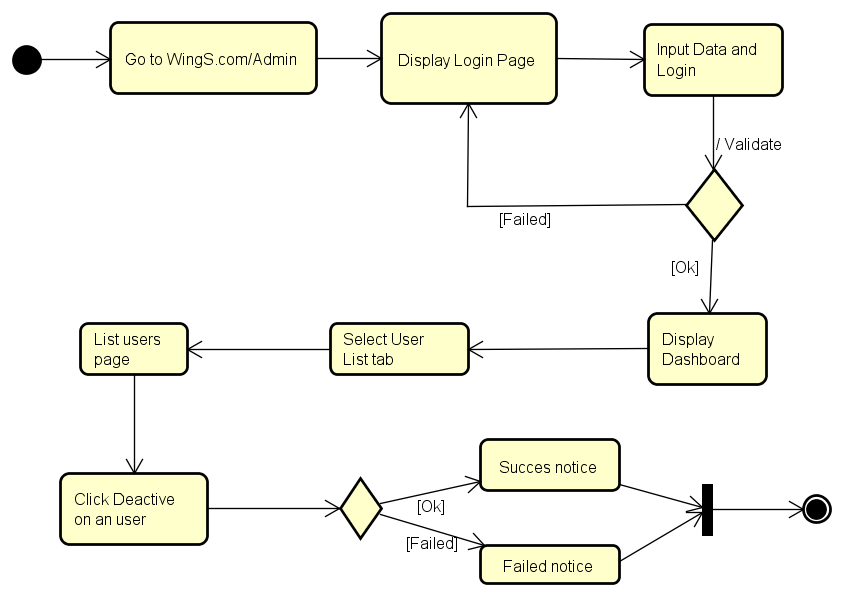
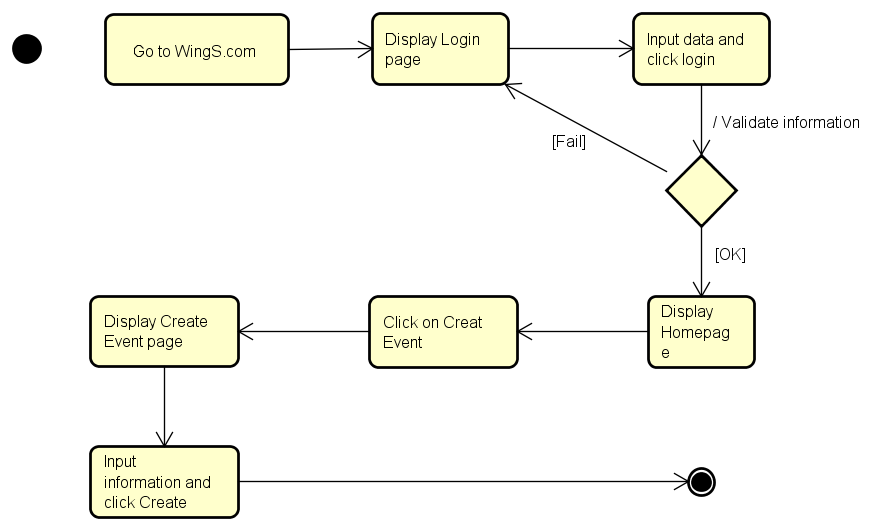


Figure ‑3: De-active user activity diagram

Figure ‑4: Create Event activity diagram



# Deployment View

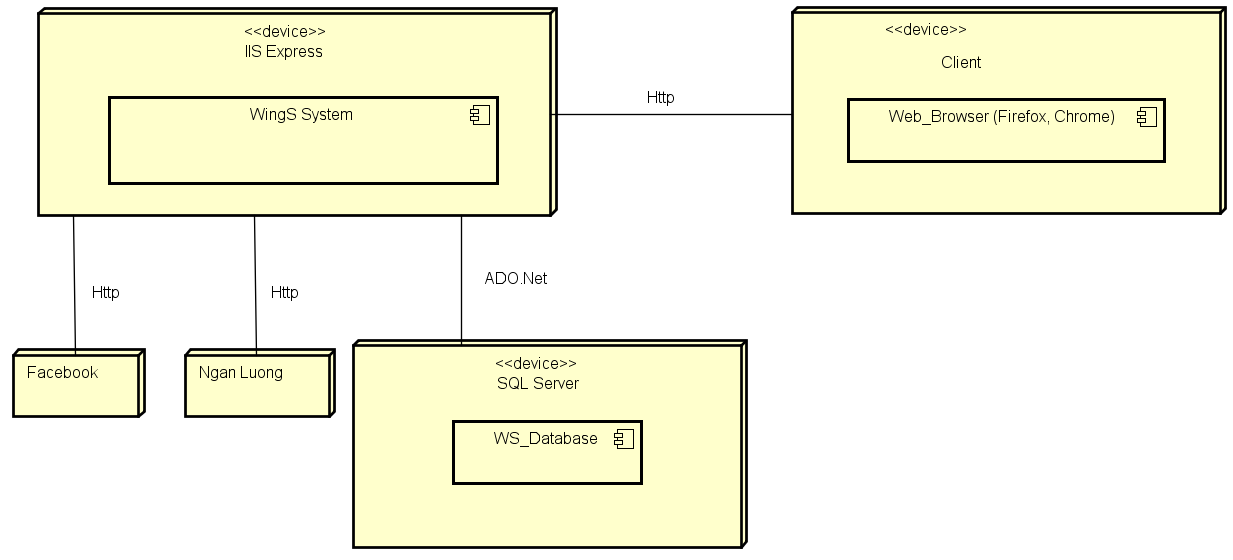


Figure ‑: Deployment Diagram

|  |  |  |
| --- | --- | --- |
| **No** | **Name** | **Description** |
| 1 | SQL Server | SQL Server use to store system’s data.  Using SQL Server 2013 or higher. |
| 2 | Facebook | Facebook Api server |
| 3 | Ngan luong | Ngan luong Api server |
| 4 | Client | Client is web browser to use system. Firefox 30, Chrome 53 or higher. |

**Table 8‑1:** Deployment Diagram Description

# Quality

Reference to: WS\_Software requirement specification\_v1.0\_EN.docx