

VIETNAMESE MEDICINAL NETWORK

Architecture Design

**Project Code: VMN**

Document Code: VMN\_ Architecture Design\_v1.0\_EN

**Ha Noi, 23/01/2016**

Record of change

\*A - Added M - Modified D - Deleted

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Effective Date | Changed Items | A\*M, D | Change Description | New Version |
| 23/01/2016 |  | A | New | v1.0 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Table 1: Record of change

SIGNATURE PAGE

**ORIGINATOR:** Nguyen Minh Tien 23/01/2016

Technical leader

**REVIEWERS:** Tran Binh Khanh ––/––/2016

Developer

**APPROVAL:**  Nguyen Van Sang ––/––/2016

Supervisor

TABLE OF CONTENTS

[1 Introduction 5](#_Toc441618377)

[1.1 Purpose 5](#_Toc441618378)

[1.2 Scope 5](#_Toc441618379)

[1.3 Definitions, Acronyms and Abbreviations 5](#_Toc441618380)

[1.4 References 6](#_Toc441618381)

[1.5 Overview 6](#_Toc441618382)

[2 Choice of Architecture design 6](#_Toc441618383)

[2.1 MVC Model 6](#_Toc441618384)

[**1.2.1** **MVC Model overview** 6](#_Toc441618385)

[2.2 Laravel 8](#_Toc441618386)

[3 Architectural Representation 10](#_Toc441618387)

[4 Architectural Goals and Constraints 11](#_Toc441618388)

[5 Use-Case View 11](#_Toc441618389)

[5.1 User module 12](#_Toc441618390)

[5.1.1 Authentication **module** 12](#_Toc441618391)

[5.2 Medicinal plant module 13](#_Toc441618392)

[5.2.1 Medicinal plant information 13](#_Toc441618393)

[5.2.2 Medicinal plant content management 14](#_Toc441618394)

[Figure 8: Medicinal plant content management 14](#_Toc441618395)

[6 Logical View 15](#_Toc441618396)

[6.1 Overview 15](#_Toc441618397)

[6.2 Architecturally Significant Design Packages 16](#_Toc441618398)

[7 Process View 19](#_Toc441618399)

[8 Deployment view 21](#_Toc441618400)

[9 Quality 21](#_Toc441618401)

# Introduction

## 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 Vietnamese Medicinal Plant website created by VMN capstone project team.

## Definitions, Acronyms and Abbreviations

|  |  |  |
| --- | --- | --- |
| Acronym | Definition | Note |
| VMN | Vietnamese Medicinal Network |  |
| MVC | Model view control |  |
| IDE | Integrated development environment |  |
| Q&A | Question and answer |  |
| GUI | Graphic user interface |  |
| User | A person who can use feature of VMN | Include Guest Member, Mod, Admin, |
| Guest | A person who does not have account |  |
| Member | Who has account and can contribute information to system | Include Common member and Mod |
| Mod | Content manager. |  |
| Admin | System manager |  |

Table 2: Definitions

## References

* VMN\_ Software Requirements Specification\_v1.0\_EN.docx
* VMN \_Data Design\_v1.0\_EN.docx
* Sample Design at Introduction to software engineering (I2SE) course in cms.fpt.edu.vn
* Software Architecture Design Illuminated Book
* <http://en.wikipedia.org/wiki/Model-view-controller>

## 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 VMN is developing a website that allow users about search and share information about Vietnamese medicine. The system of VMN is structured based on MVC combined with layered architecture and Laravel framework

### **MVC Model overview**

MVC is a software architecture pattern that separates the representation of information from user’s interaction with it. The model consists of application data, business rules, logic and functions. A view can be any output representation of data, such as a chart or a diagram. Multiple views of the same data are possible, such as bar chart for management and a tabular view for accountants. The controller mediates input, converting it to commands for the model or view.

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 sometimes referred to as a layer—for example, model layer.



Figure 1: 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.
* Very effective for deployment if graphic, programming and database development professionals are working in a team in a designed project.
* 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 reason of choosing MVC model**

* Easily manage the complexity of application by dividing the application into three components: model, view and controller.
* 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.
* VMN system is not complete system, now. We built the system that towards extensibility and maintainability in the future.

## Laravel

#### **Laravel overview**

Laravel is an open source rapid development web application framework, for use in building dynamic web sites with PHP. CodeIgniter is loosely based on the popular MVC development pattern. While view and controller classes are a necessary part of development under CodeIgniter, models are optional.

#### **Advantages and disadvantages of Laravel**

Advantages:

* Laravel is the most common PHP framework with beautiful code
* Laravel is a PHP framework have the fast speed
* Laravel provides libraries cater to the most common tasks in web programming, such as database access, email, data checking, session management, image processing ... to the high function as XML-RPC, encryption, security ...
* Mechanism test data is closely, preventing XSS and SQL Injection of Laravel helps minimize the security risk to the system.
* Disadvantages:
* No supports about some common module
* Amateur developers face problems while extending codes and classes.
* Many methods included in the reverse routing are complex

#### **The reason of choosing Laravel**

* Consistent with VMN system
* Some modules need to reuse shared business logic layer.
* Designed according to the MVC Model
* Modern toolkit. Pinch of magic.
* Tailored for team.
* “Love beautiful code? We do too.”

# Architectural Representation

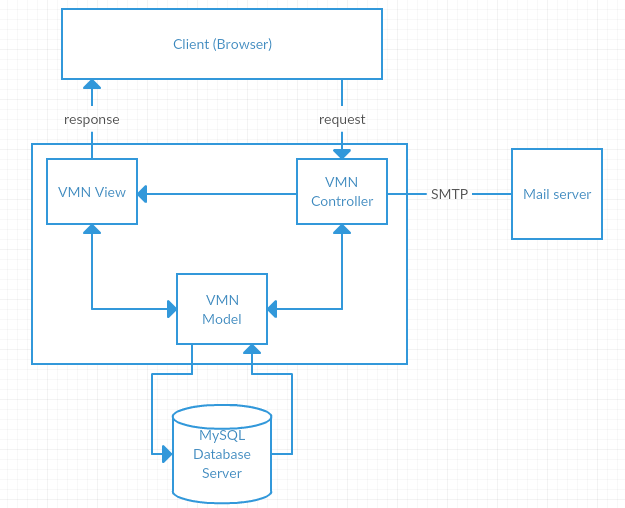


Figure 2: System overview

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

**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. The model object does not talk directly to a View, instead is made available to a controller, which accesses it when needed. When a model changes, typically it will notify its observers that a change has occurred. As with any data object it contains instance variables and getter/setter methods.

**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 VMN, the view 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.

# 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:** Usability Requirements support the following from the perspective of its primary users:

**Capacity and scalability:** throughput, storage and growth requirements.

# Use-Case View

This application includes two parts:

* The first part is Authentication service. Authentication service permit user to access and use system’s feature.
* Next part is Admin module. In admin side includes system manager who manage user account and ensure system is available.

## User module

### 5.1.1 **Authentic**ation **module**

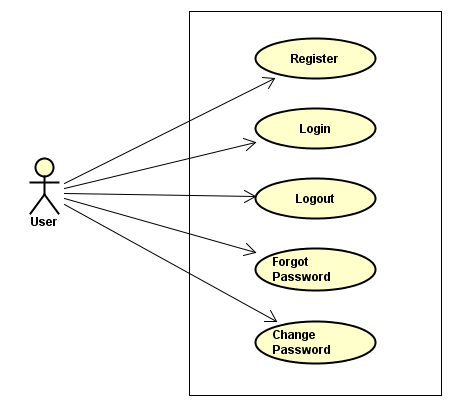


Figure 3: Authentication module

|  |  |  |  |
| --- | --- | --- | --- |
| No | Use-case name | Actor | Description |
| 1 | Register | Guest | Create an account to access system |
| 2 | Login | User | Using account to access and use system’s feature |
| 3 | Logout | User (Member) | Logout from system |
| 4 | Forgot password | User (Member) | Support member who forgot password can receive new password |
| 5 | Change password | Member | Change password of an account |

## Medicinal plant module

### 5.2.1 Medicinal plant information

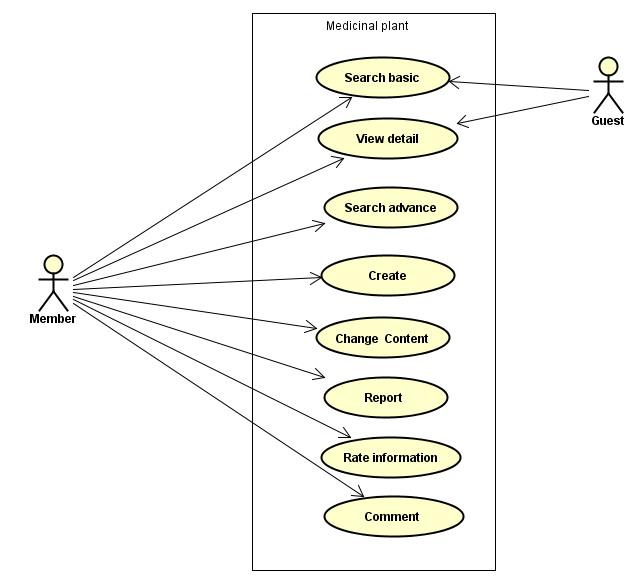
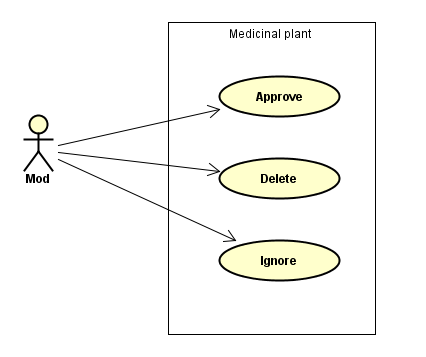


Figure 7: Medicinal plant information

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Use-case name | Actor | Description |
| 1 | Search basic | Member and Guest | Search medicinal plant by name only |
| 2 | View detail | Member and Guest | View detail information of a Medicinal plant |
| 3 | Search advance | Member and Guest | Search Medicinal plant with multi condition |
| 4 | Create | Member | Create new Medicinal plant |
| 5 | Change content | Member | Author can change content |
| 6 | Report | Member | Report to Mod invalid medicinal plant |
| 7 | Rate information | Member | Rate information of Medicinal plant |
| 8 | Comment | Member | Comment on Medicinal plant post to give personal opinion |

### 5.2.2 Medicinal plant content management



### Figure 8: Medicinal plant content management

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Use-case name | Actor | Description |
| 1 | Approve | Mod | Approve new medicinal plant or change content request from member |
| 2 | Delete | Mod | Delete invalid medicinal plant which is reported by member |
| 3 | Ignore | Mod | Ignore new plant, change content or report from user |

## Remedy module

### 5.3.1 Remedy Information

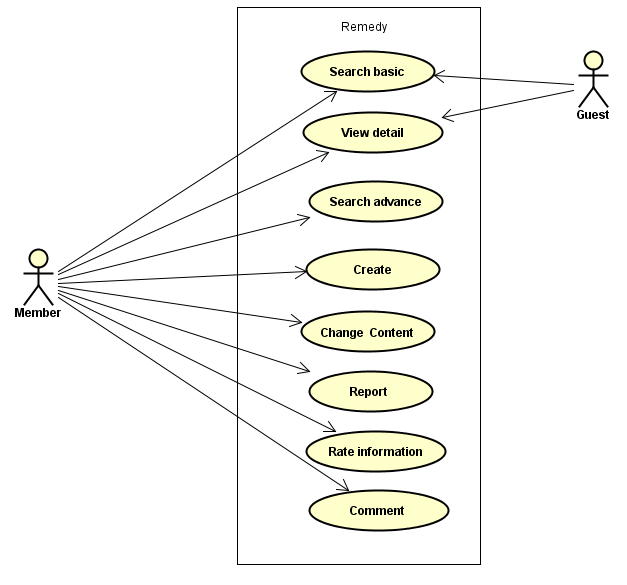


Figure 9: Remedy information

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Use-case name | Actor | Description |
| 1 | Search basic | Member and Guest | Search remedy by name only |
| 2 | View detail | Member and Guest | View detail information of a Remedy |
| 3 | Search advance | Member and Guest | Search Remedy with multi condition |
| 4 | Create | Member | Create new Remedy |
| 5 | Change content | Member | Author can change content |
| 6 | Report | Member | Report to Mod invalid remedy |
| 7 | Rate information | Member | Rate information of Remedy |
| 8 | Comment | Member | Comment on Remedy post to give personal opinion |

### 5.3.2 Remedy content management

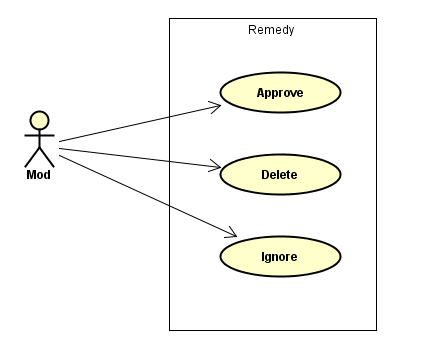
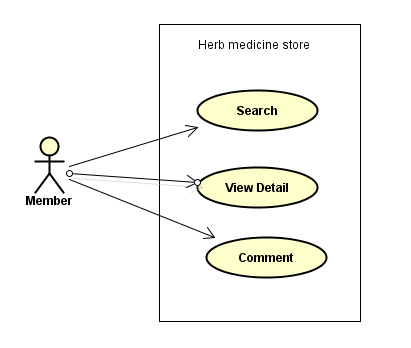


Figure 10: Remedy content management

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Use-case name | Actor | Description |
| 1 | Approve | Mod | Approve new remedy or change content request from member |
| 2 | Delete | Mod | Delete invalid remedy which is reported by member |
| 3 | Ignore | Mod | Ignore new remedy, change content or report from user |

## Herb medicine store module

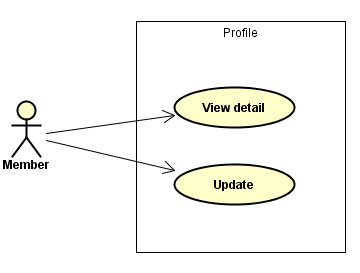
### 5.4.1 Herb medicine store module



|  |  |  |  |
| --- | --- | --- | --- |
| No. | Use-case name | Actor | Description |
| 1 | Search | Member | Member can search herb medicine store in system |
| 2 | View detail | Member | View information of a store |
| 3 | Comment | Member | Comment on information of store |

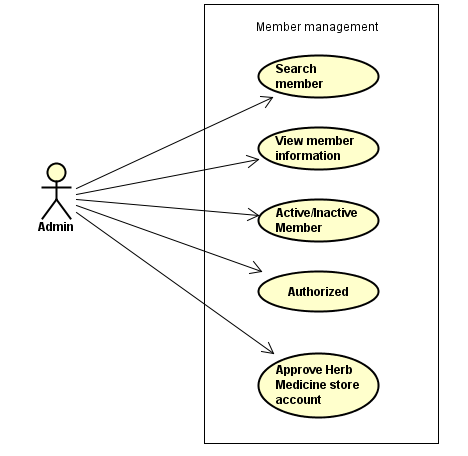
## Member module

### Member profile



|  |  |  |  |
| --- | --- | --- | --- |
| No. | Use-case name | Actor | Description |
| 1 | View detail | Member | View detail profile of a member |
| 2 | Update | Member | Member can change information of own profile |

### Member management



|  |  |  |  |
| --- | --- | --- | --- |
| No. | Use-case name | Actor | Description |
| 1 | Search Member | Admin | Search member in system |
| 2 | View detail | Admin | View detail information of a Member (except password) |
| 3 | Active/Inactive member | Admin | Active and inactive member in system |
| 4 | Authorized | Admin | Change a member’s role (common member or mod) |
| 5 | Approve Herb Medicine store account | Admin | With account is registered as Herb medicine plant admin decides this account can be approved or not |

# 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

## Architecturally Significant Design Packages

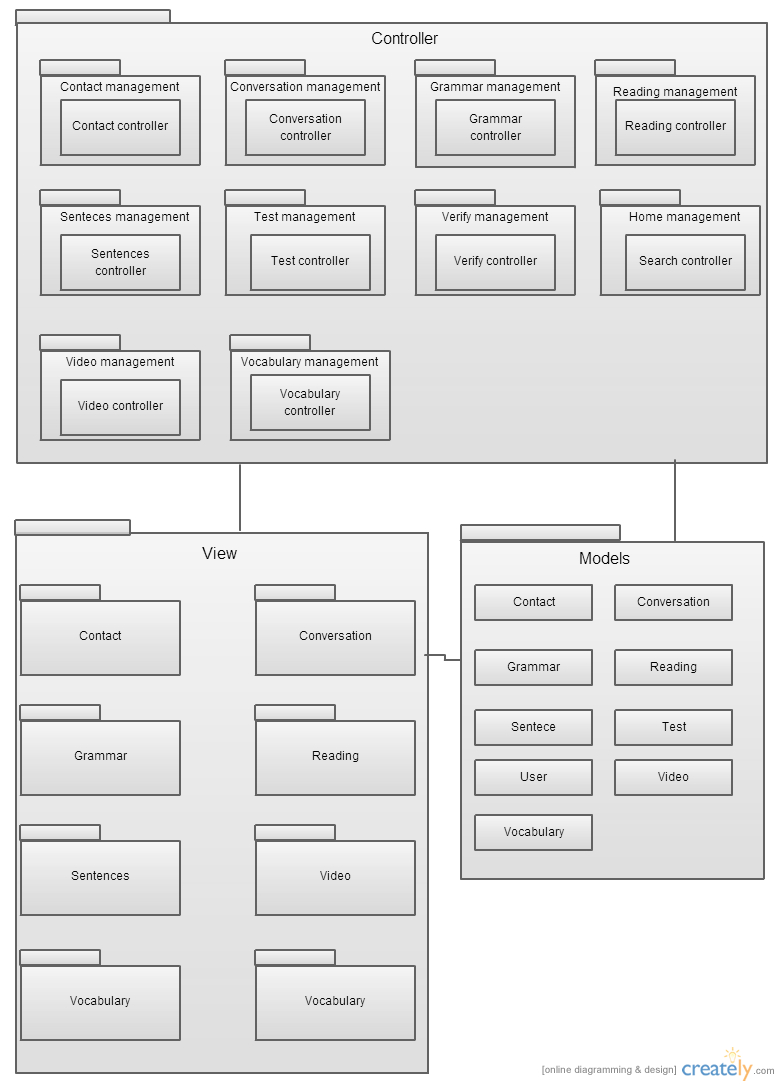


Figure 17: Package Diagram

* Model

|  |  |  |
| --- | --- | --- |
| No. | Controller class | Role |
| 1 | Contact | Description entity of Contact in database |
| 2 | Conversation | Description entity of Conversation in database |
| 3 | Grammar | Description entity of Grammar in database |
| 4 | Reading | Description entity of Reading in database |
| 5 | Sentence | Description entity of Sentence in database |
| 6 | Test | Description entity of Test in database |
| 7 | User | Description entity of User in database |
| 8 | Video | Description entity of Video in database |
| 9 | Vocabulary | Description entity of Vocabulary in database |

* Controller

|  |  |  |
| --- | --- | --- |
| No. | Controller class | Role |
| 1 | ContactController | * Receive request contact’s information from client. * Handle request from client and call method in Contact Model to get data from Database. * Respond data back to Contact View. |
| 2 | ConversationController | * Receive request conversation’s information from client. * Handle request from client and call method in Conversation Model to get data from Database. * Respond data back to Conversation View. |
| 3 | GrammarController | * Receive request grammar’s information from client. * Handle request from client and call method in Grammar Model to get data from Database. * Respond data back to Grammar View. |
| 4 | ReadingController | * Receive request reading’s information from client. * Handle request from client and call method in Reading Model to get data from Database. * Respond data back to Reading View. |
| 5 | SentenceController | * Receive request sentence’s information from client. * Handle request from client and call method in Sentence Model to get data from Database. * Respond data back to Sentence View. |
| 6 | TestController | * Receive request test’s information from client. * Handle request from client and call method in Test Model to get data from Database. * Respond data back to Test View. |
| 7 | VerifyController | * Receive request user’s information from client. * Handle request from client and call method in User Model to get data from Database. * Respond data back to Homepage View. |
| 8 | SearchController | * Receive request search’s information from client. * Handle request from client and call method in Search Model to get data from Database. * Respond data back to Search View. |
| 9 | VideoController | * Receive request video’s information from client. * Handle request from client and call method in Video Model to get data from Database. * Respond data back to Video View. |
| 10 | VocabularyController | * Receive request vocabulary’s information from client. * Handle request from client and call method in Vocabulary Model to get data from Database. * Respond data back to Vocabulary View. |

* View

Include many .php files

# Process View



Figure 18: Sequence diagram

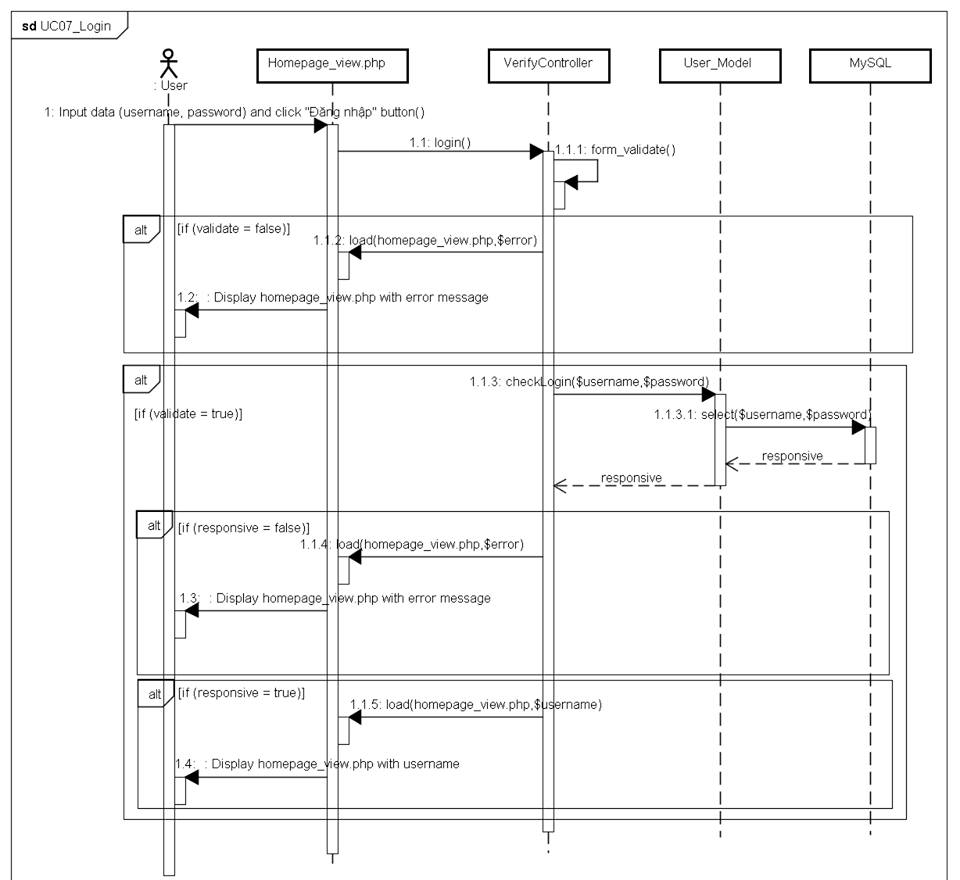


Figure 19: Login sequence

User input data(username, password) and click “Đăng nhập” button, user send a request login to verifyController class call function login()

If data entered validate is false, function login() will call function load(homepage\_view.php.$error) in verfifyController

If data enterd is validate is true, userModel will call function checkLogin($username,$password) in verifyController and send request select($username,$password) to MySQL. MySQL will response to ViewController

If responsive is false, function load(homepage\_view.php,$error) is called, display homepage\_view.php with error message to User

If responsive if tru, function load(homepage\_view.php,$username) is called, display homepage\_view.php with username

# Deployment view

Deployment view of website

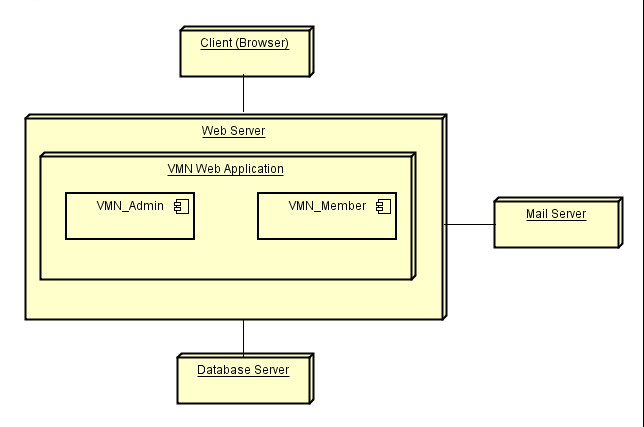


Figure 19: Deployment view

|  |  |  |
| --- | --- | --- |
| No. | Name | Description |
| 1 | Database sever | Database server use to store system’s data. MySQL5.6.16 or higher |
| 2 | Mail server | Mail server use to interact with user’s email |
| 3 | Web server | Web server is host system’s website. |
| 4 | Client | Client is web browser to use system. Firefox 20, Chrome 36 or higher. |

# Quality

Reference to: “VMN\_VN\_Software requirement specification\_v1.1\_EN”