

VIETNAMESE MEDICINAL PLANTS 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

Team leader

**REVIEWERS:** Nguyen Hai Dang 02/02/2016

Team member

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

Supervisor

TABLE OF CONTENTS

1 Introduction 6

1.1 Purpose 6

1.2 Scope 6

1.3 Definitions, Acronyms and Abbreviations 6

1.4 References 7

1.5 Overview 7

2 Choice of Architecture design 7

2.1 MVC Model 7

**2.1.1** **MVC Model overview** 7

2.2 Laravel 9

3 Architectural Representation 11

4 Architectural Goals and Constraints 12

5 Use-Case View 12

5.1 User module 13

5.1.1 Authentication **module** 13

5.2 Medicinal plant module 14

5.2.1 Medicinal plant information 14

5.2.2 Medicinal plant content management 15

Figure 5: Medicinal plant content management 15

5.3 Remedy module 16

5.3.1 Remedy Information 16

5.3.2 Remedy content management 17

5.4 Herbal medicine store module 18

5.4.1 Herbal medicine store module 18

5.5 Member module 19

5.5.1 Member profile 19

5.5.2 Member management 20

6 Logical View 21

6.1 Overview 21

6.2 Architecturally Significant Design Packages 22

7 Process View 25

8 Deployment view 26

9 Quality 27

# 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 Plants Network website created by VMN capstone project team.

## Definitions, Acronyms and Abbreviations

|  |  |  |
| --- | --- | --- |
| Acronym | Definition | Note |
| VMN | Vietnamese Medicinal Plants 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

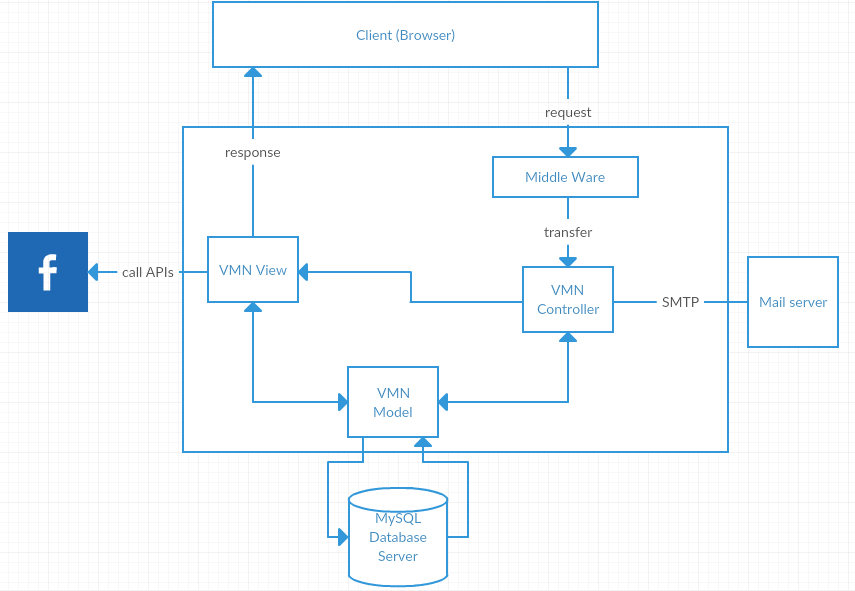


Figure 2: System overview

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**

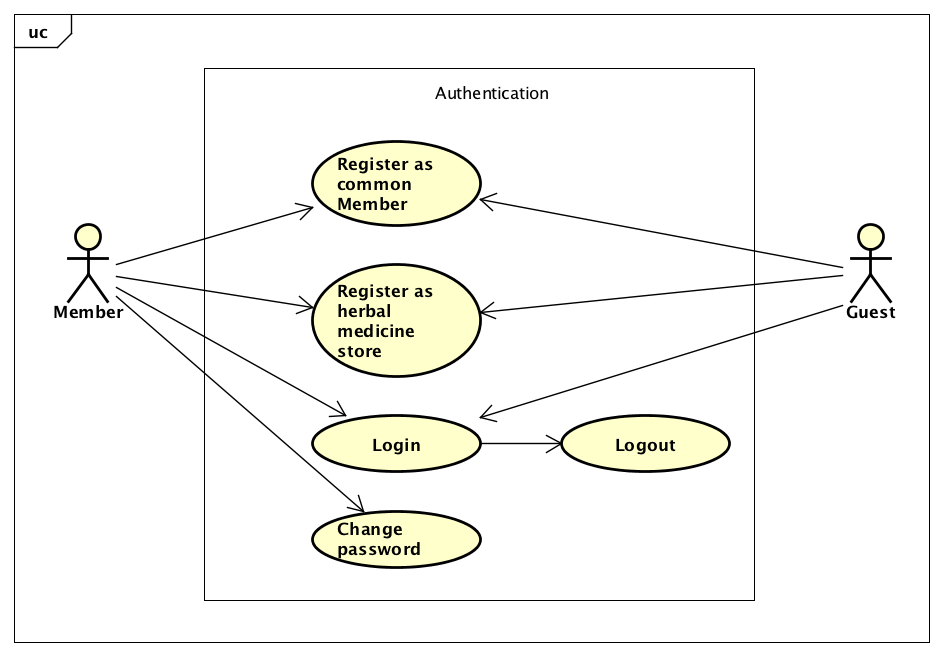


Figure 3: Authentication module

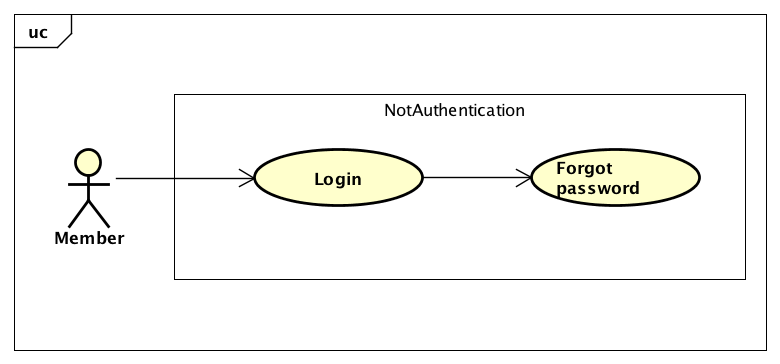


Figure 4: Not authentication module

|  |  |  |  |
| --- | --- | --- | --- |
| No | Use-case name | Actor | Description |
|  | Register as common member | Guest, member | Create an common member account to access system |
|  | Register as medicine store | Guest, member | Create a medicine store account to access system |
|  | Login | Member, Guest | Using account to access and use system’s feature |
|  | Logout | Member | Logout from system |
|  | Change password | Member | Change password of an account |
|  | Forgot password | Member | User use this function if forgot password |

## Medicinal plant module

### 5.2.1 Medicinal plant information

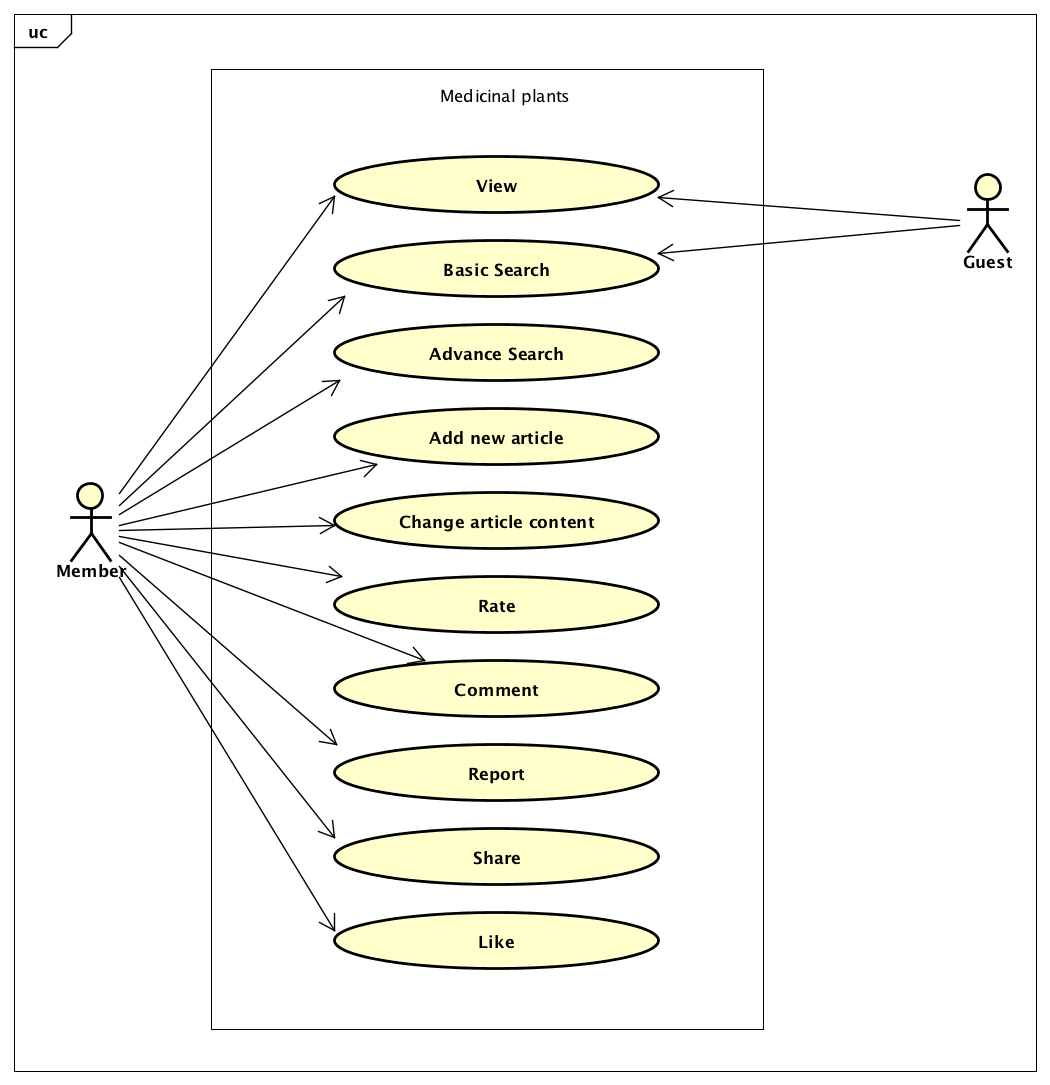


Figure 5: Medicinal plant information

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Use-case name | Actor | Description |
|  | View | Member and Guest | View detail information of a Medicinal plant |
|  | Basic search | Member and Guest | Search medicinal plant by name only |
|  | Advance search | Member and Guest | Search Medicinal plant with multi condition |
|  | Add new article | Member | Create new Medicinal plant |
|  | Change article content | Member | Author can change content |
|  | Report | Member | Report to Mod invalid medicinal plant |
|  | Rate information | Member | Rate information of Medicinal plant |
|  | Comment | Member | Comment on Medicinal plant post to give personal opinion |
|  | Share | Member | Users share article in Facebook |
|  | Like | Member | Users like article in Facebook |

### 5.2.2 Medicinal plant content management

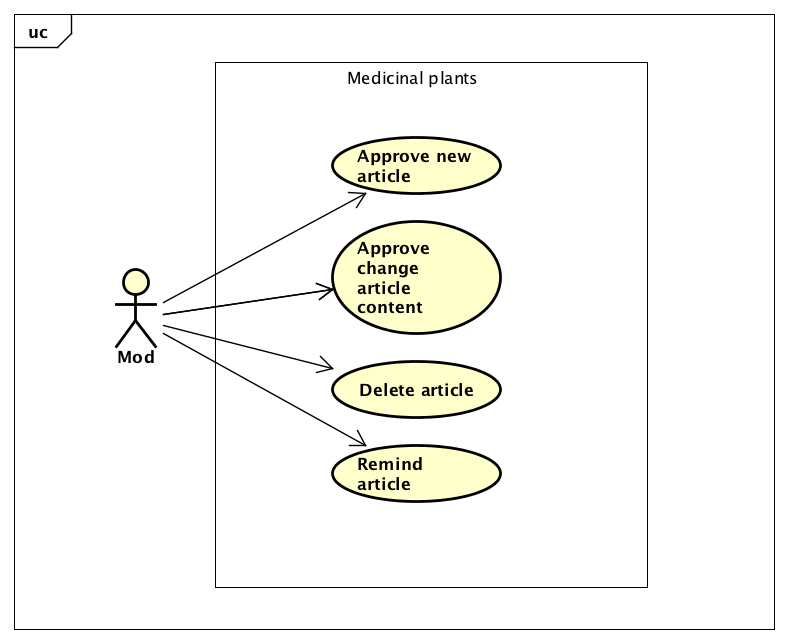


Figure 6: Medicinal plant content management

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Use-case name | Actor | Description |
|  | Approve new article | Mod | Approve new medicinal plant request from member |
|  | Approve change article content | Mod | Approve change medicinal plant request from member |
|  | Delete article | Mod | Delete invalid medicinal plant which is reported by member |
|  | Remind article | Mod | Send remind article message to author of this article |

## Remedy module

### 5.3.1 Remedy Information

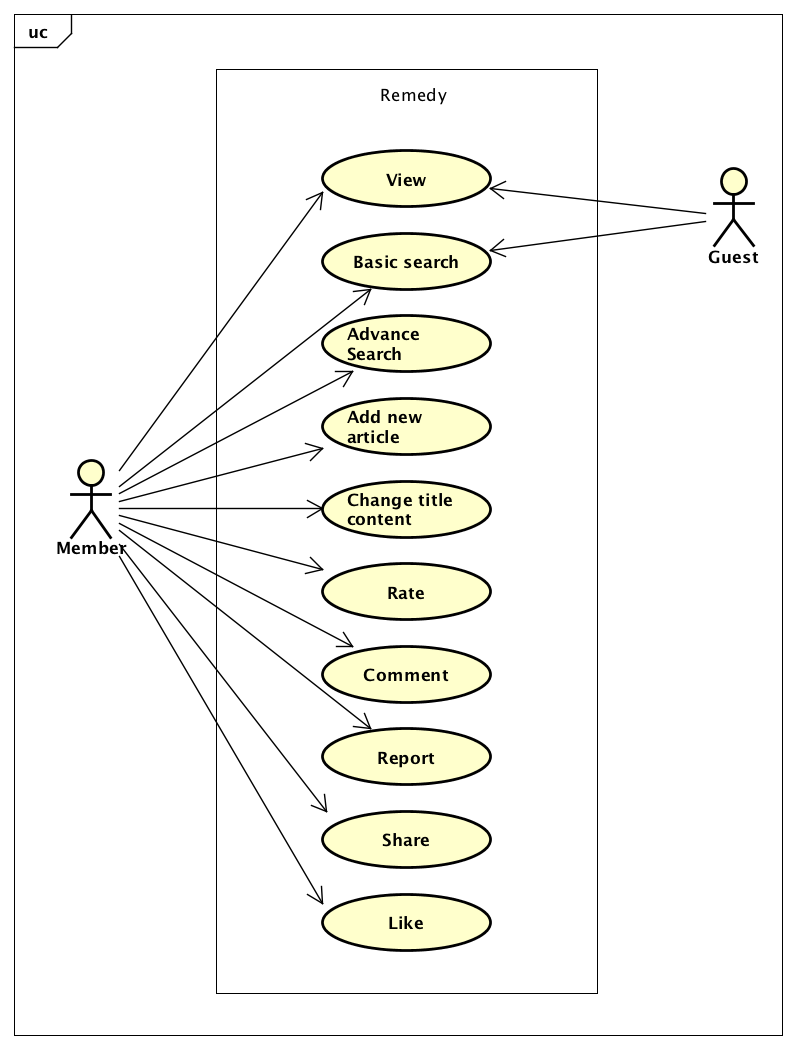


Figure 7: Remedy information

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Use-case name | Actor | Description |
|  | View | Member and Guest | View detail information of a Remedy |
|  | Basic search | Member and Guest | Search remedy by name only |
|  | Advance search | Member and Guest | Search Remedy with multi condition |
|  | Add new article | Member | Create new Remedy |
|  | Change content | Member | Author can change content |
|  | Rate | Member | Rate information of Remedy |
|  | Comment | Member | Comment on Remedy post to give personal opinion |
|  | Report | Member | Report to Mod invalid remedy |
|  | Share | Member | Users share this article in Facebook |
|  | Like | Member | Users like this article in Facebook |

### 5.3.2 Remedy content management

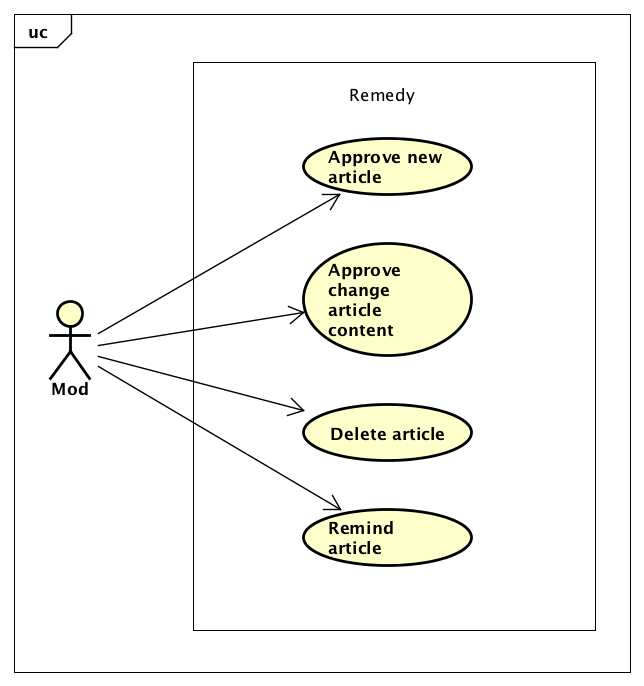


Figure 8: Remedy content management

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Use-case name | Actor | Description |
|  | Approve new article | Mod | Approve new remedy request from member |
|  | Approve change article content | Mod | Approve change article content request from member |
|  | Delete article | Mod | Delete invalid remedy which is reported by member |
|  | Remind article | Mod | Mod send remind message to author of this article |

## Herbal medicine store module

### 5.4.1 Herbal medicine store module

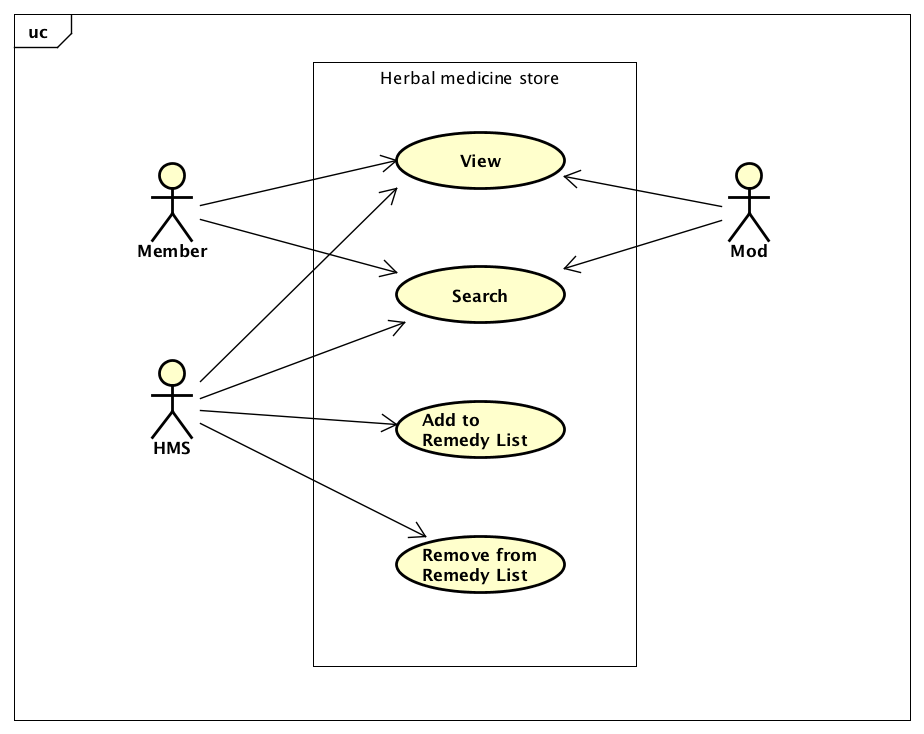


Figure 9: Herbal medicine store module

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Use-case name | Actor | Description |
| 1 | View | Member | Member, HMS, Mode can view information of a store |
| 2 | Search | Member | Member can search herb medicine store in system |
| 3 | Add to remedy list | Member | HMS can add article to remedy list |
| 4 | Remove from remedy list | Member | HMS can remove from remedy list |

## Member module

### Member profile

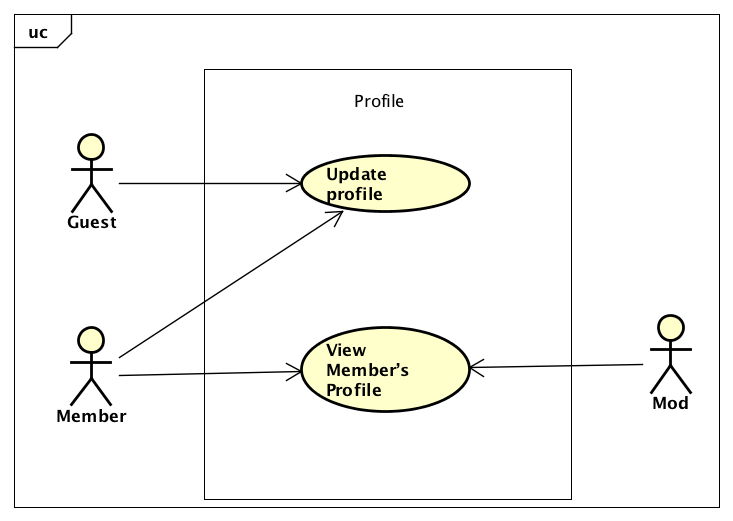


Figure 10: Member profile

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

### Member management

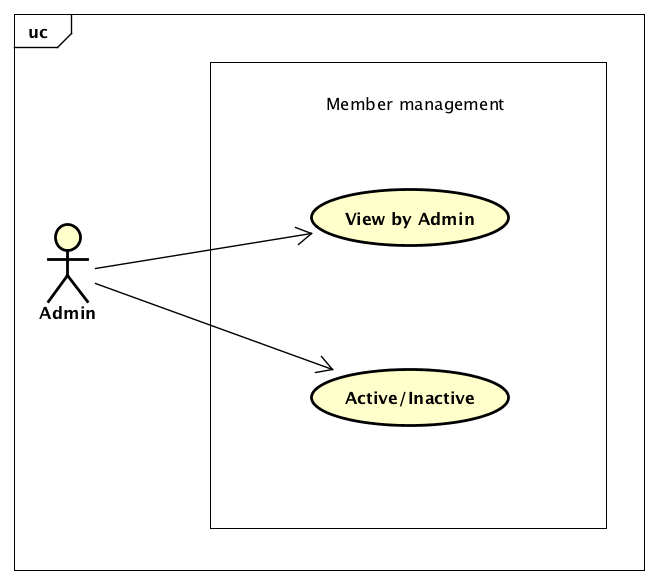


Figure 11: Member management

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Use-case name | Actor | Description |
| 1 | View by admin | Admin | View detail information of a Member (except password) |
| 2 | Active/Inactive | Admin | Active and inactive member in system |

# 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

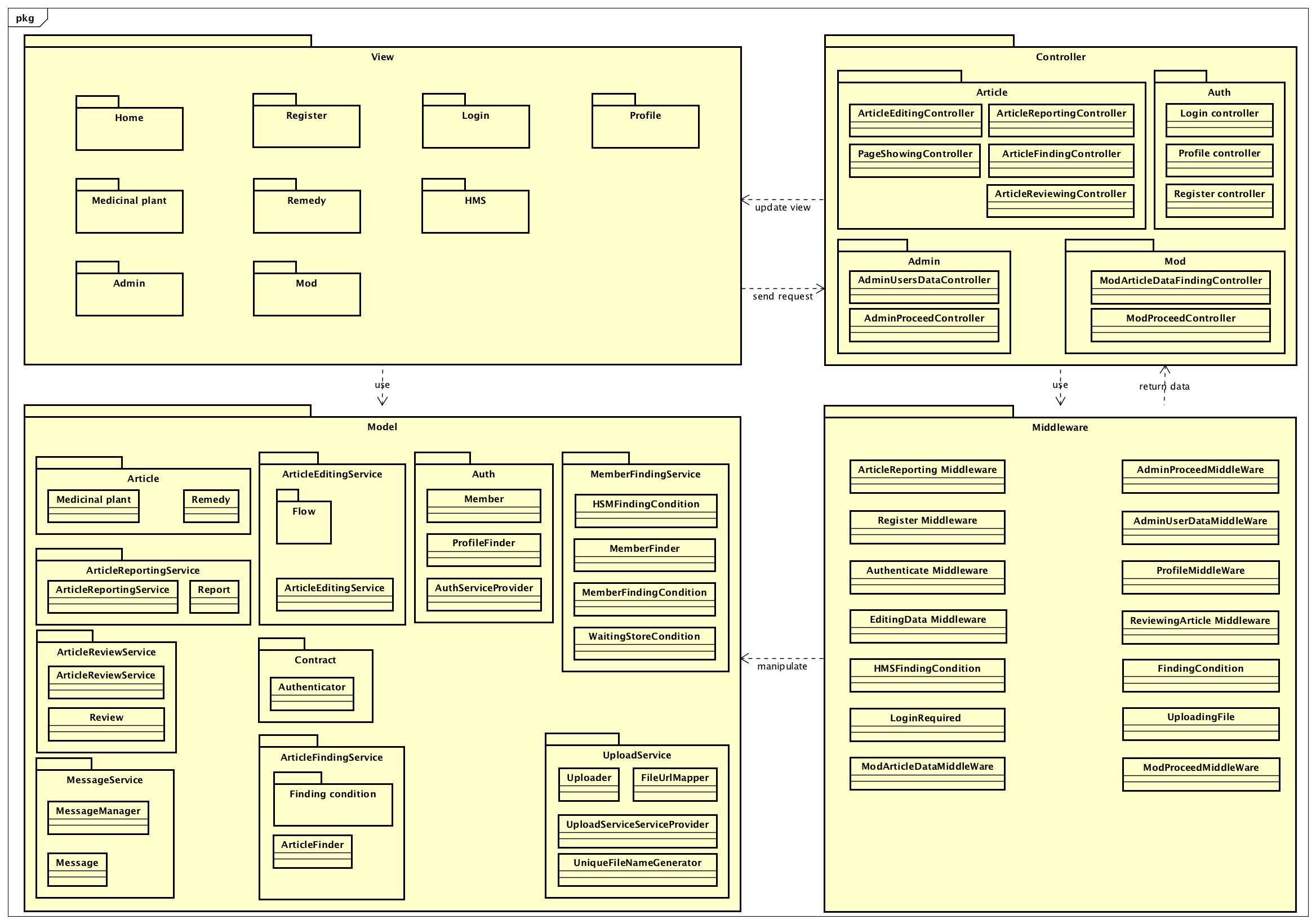


Figure 12: Package Diagram

* Model

|  |  |  |
| --- | --- | --- |
| No. | Controller class | Role |
|  | HSM | Description entity of Herbal medicine Store in database |
|  | Medicinal plant | Description entity of Medicinal Plant in database |
|  | Remedy | Description entity of Remedy in database |

* Controller

|  |  |  |
| --- | --- | --- |
| No. | Controller class | Role |
|  | RegisterController | * Receive request of register * Handle request from client and call method to do register * Respond result and data back to View. |
|  | AuthController | * Receive request of authentication * Handle request from client and call method to do authentication * Respond result and data back to View. |
|  | PasswordController | * Receive request about password (change password or fogot password) from client. * Handle request from client and call method to process * Respond result and data back to View and send mail to notice user. |
|  | HomeController | * Receive request to homepage * Call method in model to get data home page * Respond data back to View. |
|  | ArticleEditingController | * Receive request of article editing (include add new, change content, delete, approve) * Handle request from client and call method to add new, change content, delete or approve and save to Database. * Respond result and data back to View. |
|  | ArticleFindingController | * Receive request of article finding (include search, view detail) * Handle request from client and call method to get data from Database. * Respond result and data back to View. |
|  | MemberFidingController | * Receive request of member finding (include search, view detail) * Handle request from client and call method to get data from Database. * Respond result and data back to View. |
|  | MemberEditingController | * Receive request of article editing (include add new, change content, delete, approve) * Handle request from client and call method to update profile, active/inactive, delete or approve and save to Database. * Respond result and data back to View. |

* View

Include many blade.php files

# Process View



Figure 12: Sequence diagram

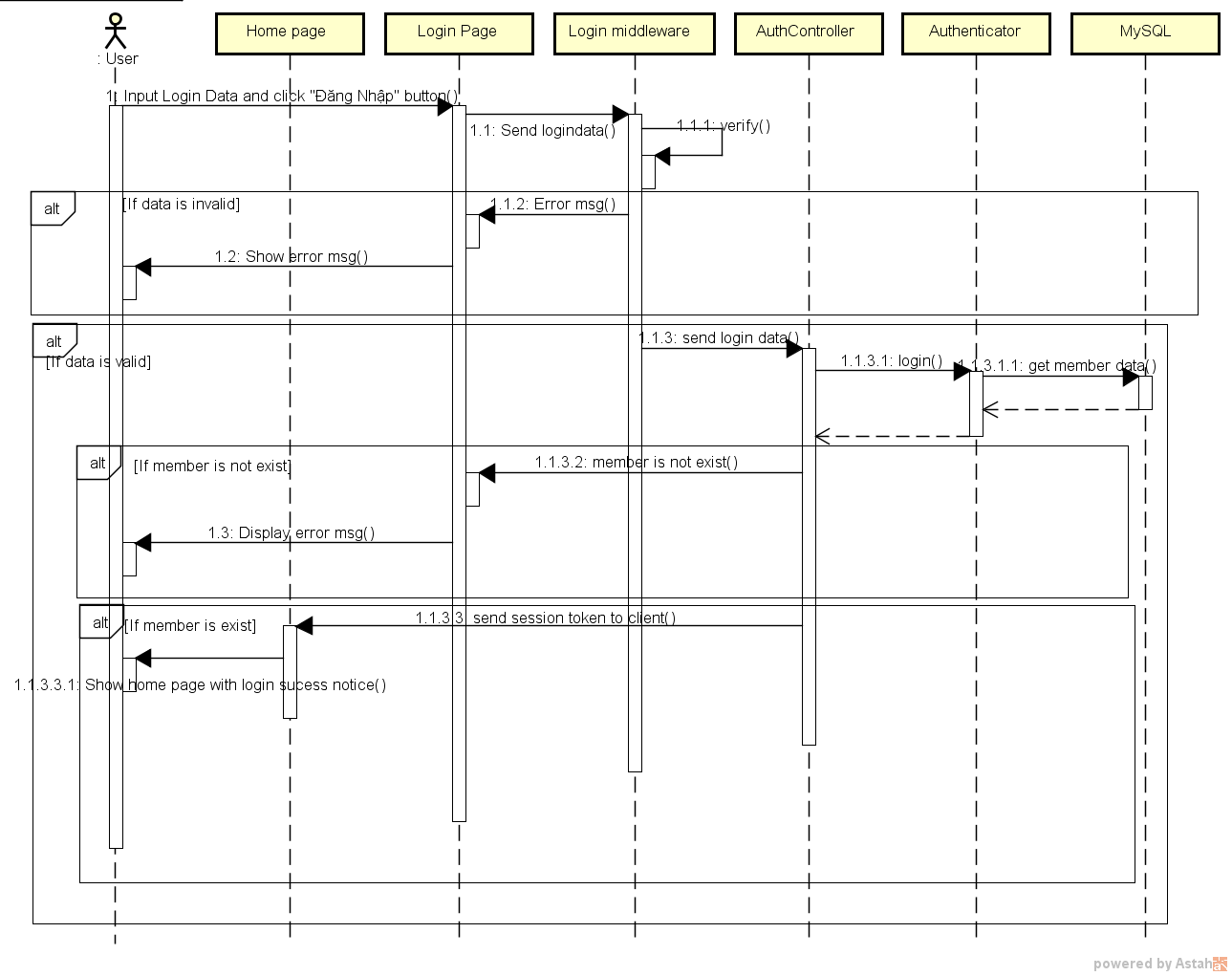


Figure 13: Login sequence

User input data(username, password) and click “Đăng nhập” button, user send a request login to server, middleware will check data

If data entered is in valid, midleware will send back an error message

If data enterd is valid, controller will handle and call login function. In login function, Authenticator get data member from DB by login data

If member is not exist (gotten data is null) error msg will be displayed in Login page

If is exist (gotten data is not null), notice login success and redirect to homepage

# Deployment view

Deployment view of website

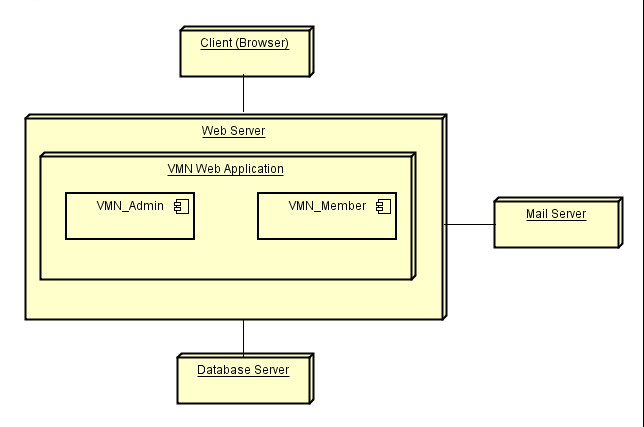


Figure 14: Deployment view

|  |  |  |
| --- | --- | --- |
| 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”