VIETNAM GENERAL CONFEDERATION OF LABOUR

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**



**INFORMATION TECHNOLOGY PROJECT 2**

**CONSTRUCT STUDENT MANAGEMENT SYSTEM**

*Instructor*: **ĐẶNG MINH THẮNG**

*Student*: **NGUYỄN CHÂU THẢO QUÂN – 51600072**

Class **: 16050310**

Course  **: 20**

**HO CHI MINH CITY, 2020**

VIETNAM GENERAL CONFEDERATION OF LABOUR

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**



**INFORMATION TECHNOLOGY PROJECT 2**

**CONSTRUCT STUDENT MANAGEMENT SYSTEM**

*Instructor*: **ĐẶNG MINH THẮNG**

*Người thực hiện*: **NGUYỄN CHÂU THẢO QUÂN – 51600072**

Class **: 16050310**

Course  **: 20**

**HO CHI MINH CITY, 2020**

ACKNOWLEDGEMENT

I sincerely thank

*Ho Chi Minh city, day month year*

*Authors*

*(signature and full name)*

*Nguyễn Châu Thảo Quân*

GUARANTEES OF GRADUATION PROJECT

This graduation project was carried out at Ton Duc Thang University.

Advisor:

*(Title, fullname and signature)*

This thesis is defended at the Undergraduate Thesis Examination Committee was hold at Ton Duc Thang University on …/…/…

Confirmation of the Chairman of the Undergraduate Thesis Examination Committee and the Dean of the faculty after receiving the modified thesis (if any).

**CHAIRMAN** **DEAN OF FACULTY**

DECLARATION OF AUTHORSHIP

I hereby declare that this thesis was carried out by myself under the guidance and supervision of Mr Đặng Minh Thắng; and that work and the results contained in it are original and have not been submitted anywhere for any previous purposes. The data and figures presented in this thesis are for analysis, comments and evaluations from various resources by my own work and have been duly acknowledged in the reference part.

In addition, other comments, reviews and data used by other authors, and organizations have been acknowledged and explicitly cited.

**I will take full responsibility for any fraud detected in my thesis.** Ton Duc Thang University is unrelated to any copyright infringement caused on my work (if any).

*Ho Chi Minh city, day month year*

*Authors*

*(signature and full name)*

*Nguyễn Châu Thảo Quân*

**STUDENT MANAGEMENT SYSTEM**

ABSTRACT

In this 21st century, all is about technology. The schools and education nowadays also already changed. Not only for all high schools and primary schools, the universities also must have their own management system.

System analysis and design is an dispensable first step in the software system development process. It is realized and assumed that information system analysis and design is the most vital stage in software development process. Otherwise, a mistake in the data design process can lead to a poor quality product or no long-term value.

The management and provision of information about the educational process is an essential part of effective management of the educational process in the institutes of higher education. Nowadays, along with the development of science and technology, the demand for the informatics applications is also increasing rapidly. The construction of management system to meet above these demands is very essential. Student management is not an exception. The birth of student management system will reduce manpower, helping to manage students and even making it more convenient and especially helpful.

As a result, a system called Student Management System will be developing as a replacement of the manual management methods to solve problem that facing when was using that methods. Student Management System is a system for education that use to manage student’s information, also known as student information system, student information management system. This design of this system is web-based type, so the user can use the system directly by connect to internet.

The requirements of a reliable student management system are analysed, formed a use-case diagram of student management system, designed and implemented the architecture of the application. Regarding the implementation process, modern approaches were used to develop a reliable websites written by PHP.

TABLE OF CONTENTS

[ACKNOWLEDGEMENT i](#_Toc61045675)

[GUARANTEES OF GRADUATION PROJECT ii](#_Toc61045676)

[DECLARATION OF AUTHORSHIP iii](#_Toc61045677)

[ABSTRACT iv](#_Toc61045678)

[TABLE OF CONTENTS 1](#_Toc61045679)

[ABBREVIATIONS 3](#_Toc61045680)

[LIST OF FIGURES 4](#_Toc61045681)

[LIST OF TABLES 5](#_Toc61045682)

[CHAPTER 1 – INTRODUCTION 7](#_Toc61045683)

[1.1 Objectives and the Objects. 7](#_Toc61045684)

[1.1.1 Objectives 7](#_Toc61045685)

[1.1.2 Objects 7](#_Toc61045686)

[1.2 Scopes 7](#_Toc61045687)

[CHAPTER 2 – SYSTEM ANALYSIS AND DESIGN 8](#_Toc61045688)

[2.1 System Analysis 8](#_Toc61045689)

[2.1.1 Purpose 8](#_Toc61045690)

[2.1.2 System Perspective 8](#_Toc61045691)

[2.1.3 Actors And Use Cases Descriptions 9](#_Toc61045692)

[2.1.3.1 Diagrams 9](#_Toc61045693)

[2.1.3.2 Actors Description 10](#_Toc61045694)

[2.1.3.3 Use Case Description 10](#_Toc61045695)

[2.2 System Design 34](#_Toc61045696)

[2.2.1 Enhanced Entity Relationship Diagram 34](#_Toc61045697)

[CHAPTER 3 – INTRODUCTION TO LARAVEL 35](#_Toc61045698)

[3.1 Concept of MVC architecture 35](#_Toc61045699)

[3.1.1 Model 35](#_Toc61045700)

[3.1.2 View 36](#_Toc61045701)

[3.1.3 Controller 36](#_Toc61045702)

[3.1.4 Advantages and disadvantages 36](#_Toc61045703)

[3.1.4.1 Advantages 36](#_Toc61045704)

[3.1.4.2 Disadvantages 36](#_Toc61045705)

[3.2 Concept of Laravel 36](#_Toc61045706)

[3.2.1 Why should we use Laravel ? 36](#_Toc61045707)

[3.2.1.1 Consistency and Flexibility 37](#_Toc61045708)

[3.2.2 History of Laravel 37](#_Toc61045709)

[3.2.3 Installation. 38](#_Toc61045710)

[3.2.3.1 System requirements 38](#_Toc61045711)

[3.2.3.2 Install Laravel with composer’s create-project feature 38](#_Toc61045712)

[3.2.3.3 Install Laravel with Laravel installer tool 38](#_Toc61045713)

[3.2.4 Basic components 39](#_Toc61045714)

[3.2.4.1 Router 39](#_Toc61045715)

[3.2.4.2 Controller 41](#_Toc61045716)

[3.2.4.3 Blade template 43](#_Toc61045717)

[3.3 Product 46](#_Toc61045718)

[CHAPTER 4 – CONCLUSION 47](#_Toc61045719)

ABBREVIATIONS

GUI Graphic User Interface

IoC Inversion of Control

MVC Model – View – Controller

PHP Hypertext Preprocessor

LIST OF FIGURES

[Figure 1.1: Student Information and University Management 8](#_Toc60872580)

[Figure 1.2: Entity Relationship Diagram. 33](#_Toc60872581)

[Figure 2.1: MVC architecture. 34](#_Toc60872582)

[Figure 2.2: Example of route definition. 38](#_Toc60872583)

[Figure 2.3: Example of route definition with parameter. 38](#_Toc60872584)

[Figure 2.4: Basic example of grouping routes. 39](#_Toc60872585)

[Figure 2.5: Path prefixing example 39](#_Toc60872586)

[Figure 2.6: Restricting routes 40](#_Toc60872587)

[Figure 2.7: Default generated controller. 40](#_Toc60872588)

[Figure 2.8: Simple definition method of controller. 41](#_Toc60872589)

[Figure 2.9: Route for Controller. 41](#_Toc60872590)

[Figure 2.10: Load a view with controller. 41](#_Toc60872591)

[Figure 2.11: Example of echoing variable. 44](#_Toc60872592)

[Figure 2.12: Basic example of conditional directives. 44](#_Toc60872593)

[Figure 2.13: Example of loop directive. 44](#_Toc60872594)

LIST OF TABLES

[Table 2.1: Actors description table 10](#_Toc60872595)

[Table 2.2: Use Case “Add Student”. 11](#_Toc60872596)

[Table 2.3: Use Case “Update Student”. 12](#_Toc60872597)

[Table 2.4: Use case “View Student's Detail”. 13](#_Toc60872598)

[Table 2.5: Use Case “Delete Student”. 14](#_Toc60872599)

[Table 2.6: Use Case “Add Teacher”. 15](#_Toc60872600)

[Table 2.7: Use Case “Update Teacher”. 16](#_Toc60872601)

[Table 2.8: Use Case “View Teacher’s Detail”. 17](#_Toc60872602)

[Table 2.9: Use Case “Delete Teacher”. 18](#_Toc60872603)

[Table 2.10: Use Case “Add Faculty”. 19](#_Toc60872604)

[Table 2.11: Use Case “Update Faculty”. 20](#_Toc60872605)

[Table 2.12: Use Case “View Faculty's Detail”. 21](#_Toc60872606)

[Table 2.13: Use Case “Delete Faculty”. 22](#_Toc60872607)

[Table 2.14: Use Case “Add Subject”. 23](#_Toc60872608)

[Table 2.15: Use Case “Update Subject”. 24](#_Toc60872609)

[Table 2.16: Use Case “View Subject's Detail”. 25](#_Toc60872610)

[Table 2.17: Use Case “Delete Subject”. 26](#_Toc60872611)

[Table 2.18: Use Case “Add Training Program”. 27](#_Toc60872612)

[Table 2.19: Use Case “Update Training Program”. 28](#_Toc60872613)

[Table 2.20: Use Case “Delete Training Program”. 29](#_Toc60872614)

[Table 2.21: Use Case “Grant Access” 30](#_Toc60872615)

[Table 2.22: Use Case “Add Major”. 31](#_Toc60872616)

[Table 2.23: Use Case “Update Major”. 32](#_Toc60872617)

[Table 2.24: Use Case “Delete Major”. 33](#_Toc60872618)

[Table 3.1: Comparision of writing styles between PHP and Blades. 45](#_Toc60872619)

# INTRODUCTION

## Objectives and the Objects.

### Objectives

To design and develop a Student Management System for an university.

To record all the student’s information for future reference and to manage student’s information include personal information and the academic information.

### Objects

The users of this system are given to three groups which are administrators and students. Administrator can manage all student’s data and information easily and students can check their information after login successful.

## Scopes

The main target of this system is an university, it is important because to make sure the system meet their requirements.

.

# SYSTEM ANALYSIS AND DESIGN

## System Analysis

The Student Management System can handle all the details about a student, student personal details, academic details, college details etc., The Student Manage System is an automated version of manual student management process.

### Purpose

This System Requirements Specification contains the complete system requirements for the Student Management System and describes the design decisions, architectural design and the detailed design needed to implement the system. It provides the visibility in the design and information needed for software support.

### System Perspective

The proposed system will be developed using MVC architecture and be compatible with any platform. The front end of the system will be developed using Laravel’s blade engine and backend will be developed using Laravel.

### Actors And Use Cases Descriptions

#### Diagrams

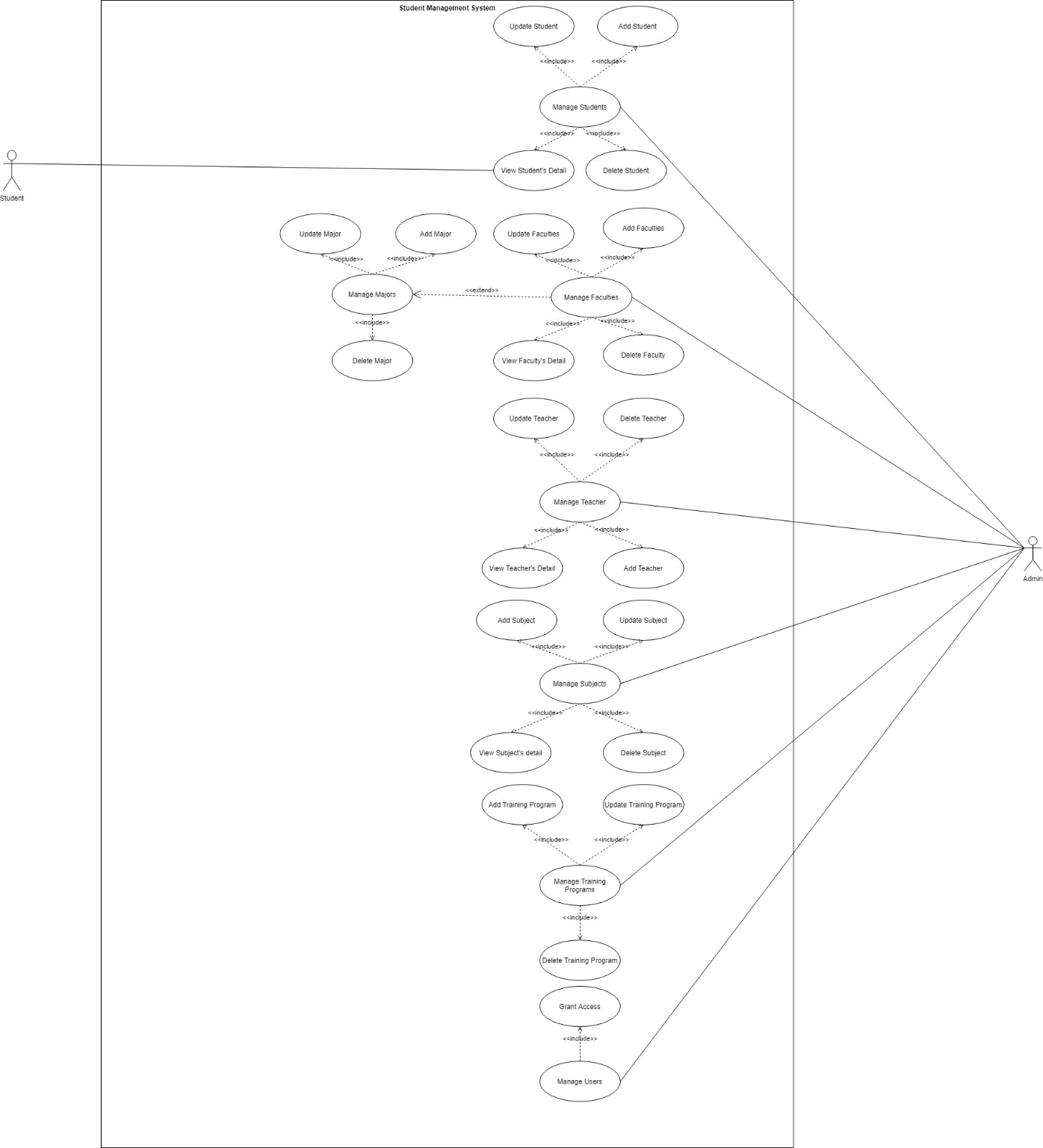


Figure .: Student Information and University Management

#### Actors Description

|  |  |  |
| --- | --- | --- |
| **#** | **Actors** | **Definitions** |
| 1 | Admin |  |
| 2 | Teacher |  |
| 3 | Student |  |

Table .: Actors description table

#### Use Case Description

|  |  |
| --- | --- |
| **Use Case ID:** | UC01 |
| **Use Case Name:** | Add Student |
| **Description:** | This use case allow user to add the new informations of student. |
| **Actor(s):** | Admin |
| **Pre-conditions:** | User is permitted to access to the dashboard. |
| **Post-conditions:** | A new student is enlisted. |
| **Main Flow:** | 1. User clicks on button “Quản Lý Sinh Viên” on the left sidebar.  2. System displays a list of student.  3. User clicks on button “+” on the top of the list.  4. System displays a form which allows user to add a new student.  5. User enters student’s information and submits.  6. System validates these information.  7. System displays an added student to the list.  8. Use Case ends. |
| **Alternative Flow:** | 6a. Missing the required information or duplicating the information.  1. Error messages display under these fields.  2. Use case resumes at main flow step 5. |
| **Exception Flow:** |  |

Table .: Use Case “Add Student”.

|  |  |
| --- | --- |
| **Use Case ID:** | UC02 |
| **Use Case Name:** | Update Student |
| **Description:** | This use case allow user to update student’s informations. |
| **Actor(s):** | Admin |
| **Pre-conditions:** | User is permitted to access to the dashboard. |
| **Post-conditions:** | Any student’s information is updated and a list of student displays. |
| **Main Flow:** | 1. User click the button “Quản Lý Sinh Viên” on the left sidebar.  2. System displays a list of students.  3. User click on button “✍” on the right of a specific student on the list.  4. System displays a form which allows user to update student’s information.  5. User enters new informations which are needed to update and submits.  6. System displays student’s profile with updated informations.  7. Use case ends. |
| **Alternative Flow:** |  |
| **Exception Flow:** |  |

Table .: Use Case “Update Student”.

|  |  |
| --- | --- |
| **Use Case ID:** | UC03 |
| **Use Case Name:** | View Student’s Detail |
| **Description:** | This use case allow user to view a detailed information of specific student including their personal information, the information at university. |
| **Actor(s):** | Admin |
| **Pre-conditions:** | User is permitted to access to the dashboard. |
| **Post-conditions:** | A student’s profile displays. |
| **Main Flow:** | 1. User clicks the button “Quản Lý Sinh Viên” on the left sidebar.  2. System displays a list of student.  3. User clicks on student’s name.  4. System displays a student’s profile.  5. Use case ends. |
| **Alternative Flow:** |  |
| **Exception Flow:** |  |

Table .: Use case “View Student's Detail”.

|  |  |
| --- | --- |
| **Use Case ID:** | UC04 |
| **Use Case Name:** | Delete Student |
| **Description:** | This use case allow user to delete a specific student from a list. |
| **Actor(s):** | Admin |
| **Pre-conditions:** | User is permitted to access to the dashboard. |
| **Post-conditions:** | A deleted student is removed from list. |
| **Main Flow:** | 1. User click the button “Quản Lý Sinh Viên” on the left sidebar.  2. System displays a list of student.  3. User click on button “–” on the right of a specific student in the list.  4. System refreshes a list and removes a deleted student from a list.  5. User can click on 🗑 to view all deleted students.  6. System will display a list of deleted students.  7. Use case ends. |
| **Alternative Flow:** |  |
| **Exception Flow:** |  |

Table .: Use Case “Delete Student”.

|  |  |
| --- | --- |
| **Use Case ID:** | UC05 |
| **Use Case Name:** | Add Teacher |
| **Description:** | This use case allow user to add the new informations of a new teacher. |
| **Actor(s):** | Admin |
| **Pre-conditions:** | User is permitted to access to the dashboard. |
| **Post-conditions:** | A new teacher is enlisted. |
| **Main Flow:** | 1. User click on button “Quản Lý Giảng Viên” on the left sidebar.  2. System displays a list of teacher.  3. User click on button “+” on the top of the list.  4. System displays a form which allows user to add a new teacher.  5. User enters all teacher’s information and submits.  6. System validates these information.  7. System displays an added teacher into a list.  8. Use Case ends. |
| **Alternative Flow:** | 6a. Missing the required information or duplicating information.  1. Error messages display under these fields.  2. Use case resumes at main flow step 5. |
| **Exception Flow:** |  |

Table .: Use Case “Add Teacher”.

|  |  |
| --- | --- |
| **Use Case ID:** | UC06 |
| **Use Case Name:** | Update Teacher |
| **Description:** | This use case allow user to update a new teacher’s informations including their personal information, the information at university. |
| **Actor(s):** | Admin |
| **Pre-conditions:** | User is permitted to access to the dashboard. |
| **Post-conditions:** | Any teacher’s information is updated and a list of teacher displays. |
| **Main Flow:** | 1. User click on button “Quản Lý Giảng Viên” on the left sidebar.  2. System displays a list of teacher.  3. User click on button “✍” on the right of a specific teacher in the list.  4. System displays a form which allows user to update teacher’s information.  5. User enters the informations which are needed to update and submits these information.  6. System displays teacher’s profile with updated informations.  7. Use Case ends. |
| **Alternative Flow:** |  |
| **Exception Flow:** |  |

Table .: Use Case “Update Teacher”.

|  |  |
| --- | --- |
| **Use Case ID:** | UC07 |
| **Use Case Name:** | View Teacher’s Detail |
| **Description:** | This use case allow user to view a detailed information of specific teacher including their personal information, the information at university. |
| **Actor(s):** | Admin |
| **Pre-conditions:** | User is permitted to access to the dashboard. |
| **Post-conditions:** | Teacher’s profile displays. |
| **Main Flow:** | 1. User click the button “Quản Lý Giảng Viên” on the left sidebar.  2. System displays a list of teacher.  3. User click on teacher’s name.  4. System displays teacher’s profile.  5. Use case ends. |
| **Alternative Flow:** |  |
| **Exception Flow:** |  |

Table .: Use Case “View Teacher’s Detail”.

|  |  |
| --- | --- |
| **Use Case ID:** | UC08 |
| **Use Case Name:** | Delete Teacher |
| **Description:** | This use case allow user to delete a specific teacher from a list. |
| **Actor(s):** | Admin |
| **Pre-conditions:** | User is permitted to access to the dashboard. |
| **Post-conditions:** | A deleted teacher is removed from list. |
| **Main Flow:** | 1. User click the button “Quản Lý Giảng Viên” on the left sidebar.  2. System displays a list of teacher.  3. User click on button “–” on the right of a specific teacher in the list.  4. System refreshes a list.  5. User can click on 🗑 to view all deleted teachers.  6. System will display a list of deleted teachers.  7. Use case ends. |
| **Alternative Flow:** |  |
| **Exception Flow:** |  |

Table .: Use Case “Delete Teacher”.

|  |  |
| --- | --- |
| **Use Case ID:** | UC09 |
| **Use Case Name:** | Add Faculty |
| **Description:** | This use case allow user to add a specific faculty into a list. |
| **Actor(s):** | Admin |
| **Pre-conditions:** | User is permitted to access to the dashboard. |
| **Post-conditions:** | A new faculty is added to the list. |
| **Main Flow:** | 1. User click the button “Quản Lý Khoa” on the left sidebar.  2. System displays a list of faculty.  3. User click on button “+” on top of the list.  4. System displays a form which allows user to add a new faculty.  5. User enter faculty’s information and submits.  6. System validates these information.  7. Use Case ends. |
| **Alternative Flow:** | 6a. Missing id, name or dupplicating.  1. Error messages display under these fields.  2. Use case resumes at main flow step 5. |
| **Exception Flow:** |  |

Table .: Use Case “Add Faculty”.

|  |  |
| --- | --- |
| **Use Case ID:** | UC10 |
| **Use Case Name:** | Update Faculty |
| **Description:** | This use case allow user to update a specific faculty into a list. |
| **Actor(s):** | Admin |
| **Pre-conditions:** | User is permitted to access to the dashboard. |
| **Post-conditions:** | The updated information of faculty displays into a list. |
| **Main Flow:** | 1. User click the button “Quản Lý Khoa” on the left sidebar.  2. System displays a list of faculty.  3. User click on button “✍” on the right of a specific faculty on the list.  4. System displays a form which allows user to add a new faculty.  5. User enter faculty’s information and submits.  6. System displays a list of faculty with updated informations.  7. Use Case ends. |
| **Alternative Flow:** |  |
| **Exception Flow:** |  |

Table .: Use Case “Update Faculty”.

|  |  |
| --- | --- |
| **Use Case ID:** | UC11 |
| **Use Case Name:** | View Faculty |
| **Description:** | This use case allow user to view faculty’s detailed information. |
| **Actor(s):** | Admin |
| **Pre-conditions:** | User is permitted to access to the dashboard. |
| **Post-conditions:** | Faculty’s detail displays. |
| **Main Flow:** | 1. User click the button “Quản Lý Khoa” on the left sidebar.  2. System displays a list of faculty.  3. User click on a specific faculty’s name.  4. System displays faculty’s detailed information.  5. Use Case ends. |
| **Alternative Flow:** |  |
| **Exception Flow** |  |

Table .: Use Case “View Faculty's Detail”.

|  |  |
| --- | --- |
| **Use Case ID:** | UC12 |
| **Use Case Name:** | Delete Faculty |
| **Description:** | This use case allow user to delete a specific faculty with its information. |
| **Actor(s):** | Admin |
| **Pre-conditions:** | User is permitted to access to the dashboard. |
| **Post-conditions:** | Faculty are removed from list and so are its related information. |
| **Main Flow:** | 1. User click the button “Quản Lý Khoa” on the left sidebar.  2. System displays a list of faculty.  3. User click button “–” on the right of a specific faculty.  4. System refreshes a page.  5. User can click on 🗑 to view all deleted faculties.  6. System will display a list of deleted faculties.  7. Use case ends. |
| **Alternative Flow:** |  |
| **Exception Flow:** |  |

Table .: Use Case “Delete Faculty”.

|  |  |
| --- | --- |
| **Use Case ID:** | UC13 |
| **Use Case Name:** | Add Subject |
| **Description:** | This use case allow user to add a specific subject into a list. |
| **Actor(s):** | Admin |
| **Pre-conditions:** | User is permitted to access to the dashboard. |
| **Post-conditions:** | A new subject is enlisted. |
| **Main Flow:** | 1. User click the button “Quản Lý Môn Học” on the left sidebar.  2. System displays a list of subjects.  3. User click on button “+” on top of the list.  4. System displays a form which allows user to add a new subject.  5. User enter subject’s information and submits.  6. System validates these information.  7. Use Case ends. |
| **Alternative Flow:** | 6a. Missing required information or dupplicating.  1. Error messages display under these fields.  2. Use case resumes at main flow step 5. |
| **Exception Flow:** |  |

Table .: Use Case “Add Subject”.

|  |  |
| --- | --- |
| **Use Case ID:** | UC14 |
| **Use Case Name:** | Update Subject |
| **Description:** | This use case allow user to update a specific subject from a list. |
| **Actor(s):** | Admin |
| **Pre-conditions:** | User is permitted to access to the dashboard. |
| **Post-conditions:** | Updated informations of subject display. |
| **Main Flow:** | 1. User click the button “Quản Lý Môn Học” on the left sidebar.  2. System displays a list of subject.  3. User click on button “✍” on the right of a specific subject on the list.  4. System displays a form which allows user to update a new subject.  5. User enter faculty’s information which are needed to updated and submits.  6. System displays subject’s detail with updated informations.  7. Use Case ends. |
| **Alternative Flow:** |  |

Table .: Use Case “Update Subject”.

|  |  |
| --- | --- |
| **Use Case ID:** | UC15 |
| **Use Case Name:** | View Subject |
| **Description:** | This use case allow user to view subject’s detailed information. |
| **Actor(s):** | Admin |
| **Pre-conditions:** | User is permitted to access to the dashboard. |
| **Post-conditions:** | Subject’s detailed information displays. |
| **Main Flow:** | 1. User click the button “Quản Lý Môn Học” on the left sidebar.  2. System displays a list of subject.  3. User click on a specific subject’s name.  4. System displays subject’s detail.  5. Use Case ends. |
| **Alternative Flow:** |  |

Table .: Use Case “View Subject's Detail”.

|  |  |
| --- | --- |
| **Use Case ID:** | UC16 |
| **Use Case Name:** | Delete Subject |
| **Description:** | This use case allow user to delete a specific subject. |
| **Actor(s):** | Admin |
| **Pre-conditions:** | User is permitted to access to the dashboard. |
| **Post-conditions:** | Subject are removed from list. |
| **Main Flow:** | 1. User click the button “Quản Lý Môn Học” on the left sidebar.  2. System displays a list of subject.  3. User click button “–” on the right of a specific subject.  4. System refreshes a page and removes a deleted subject.  5. User can click on 🗑 to view all deleted subjects.  6. System will display a list of deleted subjects.  7. Use case ends. |
| **Alternative Flow:** |  |
| **Exception Flow:** |  |

Table .: Use Case “Delete Subject”.

|  |  |
| --- | --- |
| **Use Case ID:** | UC17. |
| **Use Case Name:** | Add Training Program. |
| **Description:** | This use case allow user to add a new training program. |
| **Actor(s):** | Admin |
| **Pre-conditions:** | User is permitted to access the dashboard. |
| **Post-conditions:** | A new training program is enlisted |
| **Main Flow:** | 1. User click the button “Quản Lý Chương Trình Đào Tạo” on the left sidebar.  2. System displays a list of training programs.  3. User click button “+” on top of the list.  4. System displays a form which allows user to add a new training program.  5. User enters the information and submits.  6. System validates these information.  7. Use Case ends. |
| **Alternative Flow:** | 6a. Missing the required informations  1. Error messages display under these required fields.  2. Use Case resumes in main flow step 5 |
| **Exception Flow:** |  |

Table .: Use Case “Add Training Program”.

|  |  |
| --- | --- |
| **Use Case ID:** | UC18. |
| **Use Case Name:** | Update Training Program. |
| **Description:** | This use case allow user to update the information of training program. |
| **Actor(s):** | Admin |
| **Pre-conditions:** | User is permitted to access the dashboard. |
| **Post-conditions:** | A new information of training program is updated |
| **Main Flow:** | 1. User clicks the button “Quản Lý Chương Trình Đào Tạo” on the left sidebar.  2. System displays a list of training programs.  3. User clicks button “✍” on top of the list.  4. System displays a form which allows user to update a new information of training program.  5. User enters the information and submits.  6. System validates these information.  7. Use Case ends. |
| **Alternative Flow:** |  |
| **Exception Flow:** |  |

Table .: Use Case “Update Training Program”.

|  |  |
| --- | --- |
| **Use Case ID:** | UC19. |
| **Use Case Name:** | Delete Training Program. |
| **Description:** | This use case allow user to delete a specific training program. |
| **Actor(s):** | Admin |
| **Pre-conditions:** | User is permitted to access the dashboard. |
| **Post-conditions:** | A training program is removed from list. |
| **Main Flow:** | 1. User clicks the button “Quản Lý Chương Trình Đào Tạo” on the left sidebar.  2. System displays a list of training programs.  3. User clicks button “–” on top of the list.  4. System refreshes and removes a deleted training program from list.  5. User can click on 🗑 to view all deleted training programs.  6. System will display a list of deleted training programs.  7. Use case ends. |
| **Alternative Flow:** |  |
| **Exception Flow:** |  |

Table .: Use Case “Delete Training Program”.

|  |  |
| --- | --- |
| **Use Case ID:** | UC20. |
| **Use Case Name:** | Grant Access. |
| **Description:** | This use case allow user to authorize a specific user with roles. |
| **Actor(s):** | Admin |
| **Pre-conditions:** | User is permitted to access the dashboard. |
| **Post-conditions:** | A user is granted a role. |
| **Main Flow:** | 1. User clicks the button “Quản Lý Tài Khoản” on the left sidebar.  2. System displays a list of accounts.  3. User clicks button 🡻 on right of an account to the list and clicks on “Phân Quyền”.  4. System refreshes a lists.  5. User can click on 🡻 to view all roles of an account with the checked checkbox.  6. Use case ends. |
| **Alternative Flow:** |  |
| **Exception Flow:** |  |

Table .: Use Case “Grant Access”

|  |  |
| --- | --- |
| **Use Case ID:** | UC21. |
| **Use Case Name:** | Add Major. |
| **Description:** | This use case allow user to add a new major. |
| **Actor(s):** | Admin |
| **Pre-conditions:** | 1. User is permitted to access the dashboard.  2. A new faculty is added.  3. A profile of a faculty displays. |
| **Post-conditions:** | A new major is enlisted |
| **Main Flow:** | 1. User click the button “Quản Lý Khoa” on the left sidebar.  2. System displays a list of faculties.  3. User click faculty’s name.  4. System displays faculty’s profile.  5. User clicks on button “+” on top of the list of majors.  6. System displays a form which allows user to add a new major.  7. User enters the informations and submits  8. System validates these information.  9. Use Case ends. |
| **Alternative Flow:** | 8a. Missing the required informations  1. Error messages display under these required fields.  2. Use Case resumes in main flow step 5 |
| **Exception Flow:** |  |

Table .: Use Case “Add Major”.

|  |  |
| --- | --- |
| **Use Case ID:** | UC22. |
| **Use Case Name:** | Update Major. |
| **Description:** | This use case allow user to update a specific major. |
| **Actor(s):** | Admin |
| **Pre-conditions:** | 1. User is permitted to access the dashboard.  2. A new faculty is added.  3. A profile of a faculty displays. |
| **Post-conditions:** | The informations of a major is updated. |
| **Main Flow:** | 1. User click the button “Quản Lý Khoa” on the left sidebar.  2. System displays a list of faculties.  3. User click faculty’s name.  4. System displays faculty’s profile.  5. User clicks on button “+” on top of the list of majors.  6. System displays a form which allows user to update a new information.  7. User enters the informations and submits.  8. Use Case ends. |
| **Alternative Flow:** |  |
| **Exception Flow:** |  |

Table .: Use Case “Update Major”.

|  |  |
| --- | --- |
| **Use Case ID:** | UC23. |
| **Use Case Name:** | Delete Major. |
| **Description:** | This use case allow user to delete a specific major. |
| **Actor(s):** | Admin |
| **Pre-conditions:** | 1. User is permitted to access the dashboard.  2. A new faculty is added.  3. A profile of a faculty displays. |
| **Post-conditions:** | A major is deleted. |
| **Main Flow:** | 1. User click the button “Quản Lý Khoa” on the left sidebar.  2. System displays a list of faculties.  3. User click faculty’s name.  4. System displays faculty’s profile.  5. User clicks on button “–” on right of the list of majors.  7. System refreshes a list and a deleted major is removed from list.  8. Use Case ends. |
| **Alternative Flow:** | 8a. Missing the required informations  1. Error messages display under these required fields.  2. Use Case resumes in main flow step 5 |
| **Exception Flow:** |  |

Table .: Use Case “Delete Major”.

## System Design

### Enhanced Entity Relationship Diagram

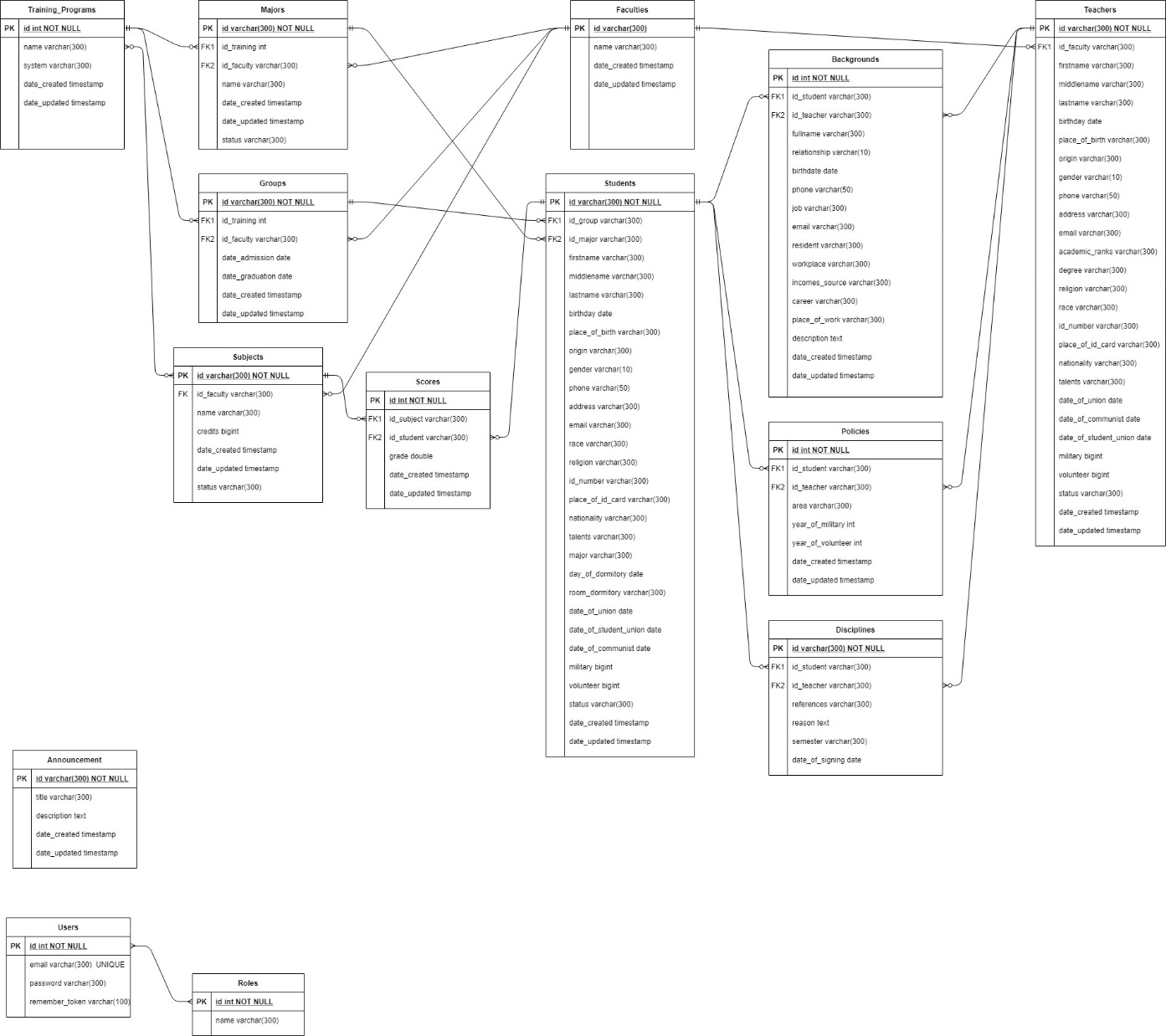


Figure .: Entity Relationship Diagram.

# INTRODUCTION TO LARAVEL

## Concept of MVC architecture

MVC is an abbreviation of Model – View – Controller, an architecture or a software design pattern used in software engineering that makes creating huge applications easy. It does not belong to any programming language or framework, but it is a concept that can be used in creating any kind of application or software in any programming language.

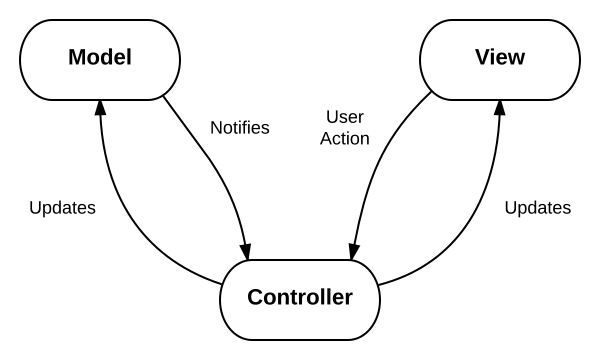


Figure .: MVC architecture.

(Source: https://www.interserver.net/tips/kb/mvc-advantages-disadvantages-mvc/)

### Model

Model works directly with the database by fetching, inserting, updating or deleting from it. It does not have to deal with user GUI or data processing.

### View

View is the GUI on which users perform some action such as showing, search data to the user on GUI and to respond to the events. In other words, it is used for displaying data and sending the events to the respective controller.

### Controller

Controller is the part in which we process the data after getting a request from View and before changing anything in database with Model. it contains the functions that we can program however we want.

### Advantages and disadvantages

#### Advantages

Reducing time exponentially because of this architecture seperation.

The development process of the application becomes fast.

Easy for a development team to collaborate and work together.

#### Disadvantages

Hard to develope the small applications.

## Concept of Laravel

Laravel is a powerful and flexible PHP framework provides a structure and starting point for creating a web application. It has a thriving community and a wide ecosystem of tools, and as a result it’s growing in appeal and reach.

### Why should we use Laravel ?

Unlike other frameworks, it is easy to see why it’s beneficial to use the individual components or packages that are available to all developers. With packages, someone else is responsible for developing and maintaining an isolated piece of code that has a well – defined job.

It prepackage a collection of third – party components together with custom framework like configuration files, service providers, prescribed directory structures. The benefit of using a framework in general is that someone has made decisions not just about the individual components, but also about the connectivity of those components together

#### Consistency and Flexibility

Unlike other frameworks, they address this issue by providing a carefully considered answer to the question “Which component should we use ?” and ensuring that the particular components chosen work well together. Additionally, they provide conventions that reduce the amount of code a developer new to the project.

### History of Laravel

Laravel is created by Taylor Otwell as an alternative solution for CodeIgniter, provide many more vital features such as authentication and authorization.

Laravel 1 was released in June 2011 and was written completely from scratch. It featured a custom ORM (Eloquent); closure – based routing; a module system for extension; and helpers for forms, validation, authentication,…

Both Laravel 2 and 3 were released in November 2011 and February 2012 respectively. They introduced controllers, unit testing, a command – line tool, an IoC container, Eloquent Relationships and migrations.

Taylor Otwell rewrote the entire framework from the ground up and developed a set of components under the code name Illuminate and released Laravel 4 in May 2013 with an entirely new structure. Instead of bundling the majority of its code as a download, Laravel now pulled the majority of its components from Symfony and the Illuminate components through composer. It also introduced queues, mailing, facades and database seeding and this is the reason why Laravel was now relying on Symfony components.

Laravel 5 was released in February 2015, as a result of significant changes to the end of the Laravel upgrade lifecycle to 4.3. In addition to the current array of new features and enhancements, Laravel 5 also introduces an internal directory tree structure for new application development.

In March 2015, programmers has voted for the most popular PHP framework, Laravel took the first place for it in 2015. It became the most popular PHP projec and the most followed on Github.

### Installation.

#### System requirements

PHP 7.1.3+ for Laravel versions 5.6 to 5.8, PHP 7.0.0+ for version 5.5, PHP 5.6.4+ for version 5.4, PHP between 5.6.4 and 7.1.\* for version 5.3 or PHP 5.5.9+ for version 5.2 and 5.1.

The required PHP extensions include OpenSSL, PDO, Mbstring, Tokenizer, XML (Laravel 5.3+), Ctype, JSON (Laravel 5.6+), BCMath (Laravel 5.7+).

Composer.

Apache, MySQL.

#### Install Laravel with composer’s create-project feature

We can create a Laravel project directly with the following steps:

1. composer create-project laravel/laravel <project\_directory>
2. cd <project\_directory>
3. php artisan serve

#### Install Laravel with Laravel installer tool

We can also create a Laravel project from Laravel installer with the following steps:

1. composer global require laravel/installer
2. laravel new <project\_directory>
3. cd <project\_directory>
4. php artisan serve

### Basic components

#### Router

In Laravel application, all web routes must be defined in routes/web.php and routes/api.php is for API routes. Web routes are the routes that will be visited by users. The most basic routes accept a URI and a closure, providing a very simple and expressive method of defining routes and behavior without complicated routing configuration files.



Figure .: Example of route definition.

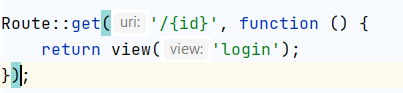


Figure .: Example of route definition with parameter.

The router allows to register routes that respond to any HTTP method:

1. Route::get($uri, $callback);
2. Route::post($uri, $callback);
3. Route::put($uri, $callback);
4. Route::patch($uri, $callback);
5. Route::delete($uri, $callback);
6. Route::options($uri, $callback);

Any routes with POST, PUT, PATCH or DELETE method should include a csrf token field:

<form method="POST" action="/profile">

@csrf

...

</form>

We can group several routes together and apply any shared configuration settings once to the entire group in order to reduce this duplication. We can use this operation for prefixing path, restricting.

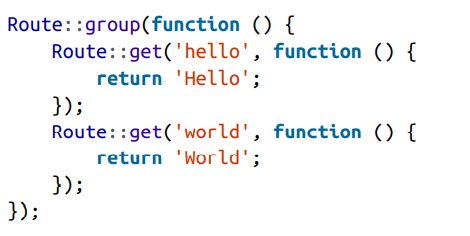


Figure .: Basic example of grouping routes.

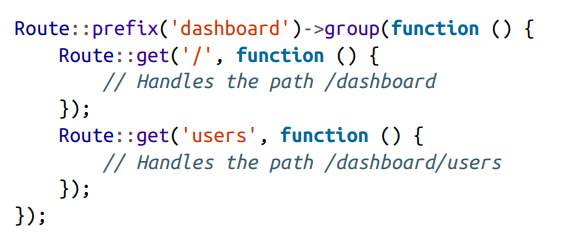


Figure .: Path prefixing example

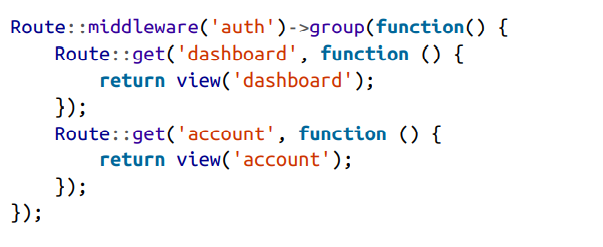


Figure .: Restricting routes

#### Controller

Create a controller: php artisan make:controller <ControllerName>. For example of creating TaskController: php artisan make:controller TaskController.

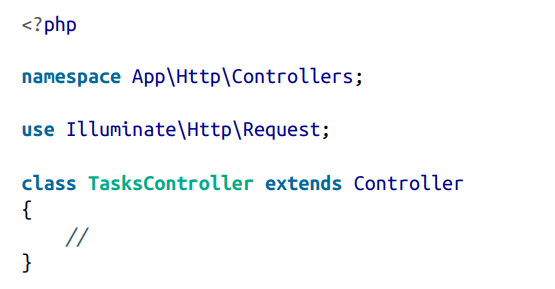


Figure .: Default generated controller.

In order to assign controller’s actions to the route, we need to define a public method to this controller.

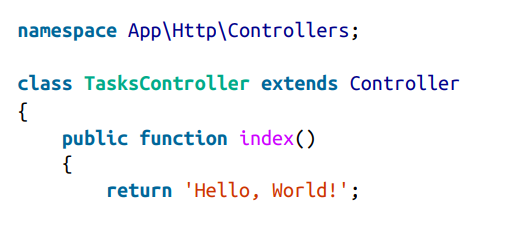


Figure .: Simple definition method of controller.



Figure .: Route for Controller.

This controller’s method loads the resources/views/announcement/view and passes it variables named id, announcement which contains the result of Eloquent method and route parameter.

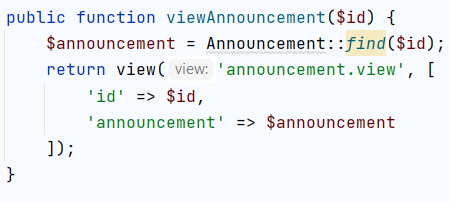


Figure .: Load a view with controller.

#### Blade template

Blade is a Laravel’s template engine which inspired by Microsoft’s Razor engine. It’s boasts a concise syntax, a shallow learning curve, a powerful and intuitive inheritance model and easy extensibility.

Blade uses curly braces for its echo and introduces a convention in which its custom tags called “directives”, are prefixed with an @. These directives are an alternative solutions to the original PHP syntax, can compiles it to the embedded PHP.

|  |  |  |
| --- | --- | --- |
| **#** | **PHP syntax** | **Blade’s directives** |
| 01 | <?php echo $a ?> | {{$a}} |
| Conditional structure | | |
| 02 | <?php  if(condition)  …  else if (condition)  …  else  …  ?> | @if (condition)  ….  @elseif (condition)  …..  @else  ……  @endif |
| 03 | <?php  switch($var)  case 1:  break;  case 2:  break;  ?> | @switch($var)  @case(1):  @break  @case(2)  @break  @endswitch |
| Loop structure | | |
| 04 | <?php  for ($i = 0; $i <= n; $i++)  ….  if (condition)  break;  if (condition)  continue;  ?> | @for ($i = 0; $i <= n; $i++)  …  @break (condition)  @continue (condition)  @endfor |
| 05 | <?php  while ($i <= n)  ….  ?> | @while ($i <= n)  …  @endwhile |
| 06 | <?php  foreach ($array as $a)  ….  ?> | @foreach ($array as $a)  …  @endforeach |
| Page customization | | |
| 07 |  | @extends() |
| 08 |  | @section() |
| 09 | <?php include(‘’) ?> | @include() |

Table .: Difference of writing styles between PHP and Blades.



Figure .: Example of echoing variable.

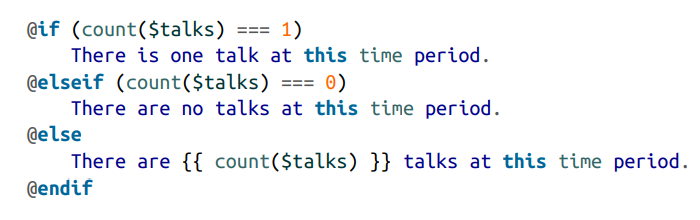


Figure .: Basic example of conditional directives.

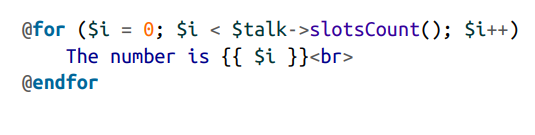


Figure .: Example of loop directive.

## Product

# CONCLUSION

**REFERENCES**

**Vietnamese**

1. Khoa Nguyễn (2015), Mô hình MVC là gì ?, <https://khoanguyen.me/tim-hieu-mo-hinh-mvc-la-gi/ />, viewed on November 4th, 2020.
2. Hoàng Tựa (2015). Laravel là gì? Giới thiệu Laravel Framework, <<http://webfaver.com/php-coding/laravel-5/tut-laravel-5-0-laravel-la-gi-gioi-thieu-laravel-framework.html/>>, November 5th, 2020.

**English**

1. Jithin (2016), < https://www.interserver.net/tips/kb/mvc-advantages-disadvantages-mvc/ />, viewd on November 4th , 2020.
2. <https://laravel.com/docs/8.x />, viewed on November 4th, 2020.
3. Matt Stauffer (2019), *Laravel Up & Running: A Framework for Building Modern PHP Apps*, O’Reilly Media, United States of America.