HANU Training Management System

Software Architecture Document

Version 1.0

*[Note: The following template is provided for use with the Rational Unified Process. Text enclosed in square brackets and displayed in blue italics (style=InfoBlue) is included to provide guidance to the author and should be deleted before publishing the document. A paragraph entered following this style will automatically be set to normal (style=Body Text).]*

*[To customize automatic fields in Microsoft Word (which display a gray background when selected), select File>Properties and replace the Title, Subject and Company fields with the appropriate information for this document. After closing the dialog, automatic fields may be updated throughout the document by selecting Edit>Select All (or Ctrl-A) and pressing F9, or simply click on the field and press F9. This must be done separately for Headers and Footers. Alt-F9 will toggle between displaying the field names and the field contents. See Word help for more information on working with fields.]*

Revision History

| **Date** | **Version** | **Description** | **Author** |
| --- | --- | --- | --- |
| 08/12/2021 | 1.0 | Draft | Nguyen Ngoc Huyen  Lai Hanh Van |
| 09/12/2021 | 1.0 | Final | Group |
|  |  |  |  |
|  |  |  |  |

Table of Contents

1. Introduction [2](#_heading=h.gjdgxs)

1.1 Purpose [2](#_heading=h.30j0zll)

1.2 Scope [2](#_heading=h.1fob9te)

1.3 Definitions, Acronyms, and Abbreviations [2](#_heading=h.2et92p0)

1.4 References [2](#_heading=h.tyjcwt)

1.5 Overview [2](#_heading=h.3dy6vkm)

2. Architectural Representation [2](#_heading=h.1t3h5sf)

3. Architectural Goals and Constraints [2](#_heading=h.4d34og8)

4. Use-Case View [2](#_heading=h.2s8eyo1)

4.1 Use-Case Realizations [2](#_heading=h.17dp8vu)

5. Logical View [2](#_heading=h.3rdcrjn)

5.1 Overview [2](#_heading=h.26in1rg)

5.2 Architecturally Significant Design Packages [2](#_heading=h.lnxbz9)

6. Process View [2](#_heading=h.35nkun2)

7. Deployment View [2](#_heading=h.1ksv4uv)

8. Implementation View [2](#_heading=h.44sinio)

8.1 Overview [2](#_heading=h.2jxsxqh)

8.2 Layers [2](#_heading=h.z337ya)

9. Data View (optional) [2](#_heading=h.3j2qqm3)

10. Size and Performance [2](#_heading=h.1y810tw)

11. Quality [2](#_heading=h.4i7ojhp)

Software Architecture Document

# Introduction

## Purpose

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

## Scope

This Software Architecture Document provides an architectural overview of the HANU Training Management System (old version) – a system developed by Hanoi University.

## Definitions, Acronyms, and Abbreviations

## References

[1] ***Supplementary Specification,*** Nguyen Thu Hien, 06/12/2021, HANU Training Management System

[2] ***Use-Case Realization Specification:* Course Registration (Student)**, Nguyen Thi Thu Hien, 06/12/2021, HANU Training Management System

[3] ***Use-Case Realization Specification:* View Grade (Student)**, Nguyen Thi Thu Hien, 06/12/2021, HANU Training Management System

[4] ***Use-Case Realization Specification:* View Schedule (Student)**, Nguyen Thi Thu Hien, 06/12/2021, HANU Training Management System

[5] ***Use-Case Realization Specification:* Change Email (Student)**, Nguyen Ngoc Huyen, 08/12/2021, HANU Training Management System

[6] ***Use-Case Realization Specification:* Dormitory Registration (Student)**, Nguyen Ngoc Huyen, 08/12/2021, HANU Training Management System

[7] ***Use-Case Realization Specification:* Student Comments (Student)**, Nguyen Ngoc Huyen, 08/12/2021, HANU Training Management System

[8] ***Use-Case Realization Specification:* Certification Registration (Student)**, Nguyen Dieu Huong Ly, 07/12/2021, HANU Training Management System

[9] ***Use-Case Realization Specification:* View Digital Invoice (Student)**, Nguyen Dieu Huong Ly, 07/12/2021, HANU Training Management System

[10] ***Use-Case Realization Specification:* View Prerequisite Courses (Student)**, Nguyen Dieu Huong Ly, 07/12/2021, HANU Training Management System

[11] ***Use-Case Realization Specification:* Upload Announcement (Student)**, Lai Hanh Van, 08/12/2021, HANU Training Management System

[12] ***Use-Case Realization Specification:* View Tuition Fee (Student)**, Lai Hanh Van, 08/12/2021, HANU Training Management System

[13] ***Use-Case Realization Specification:* Dormitory Payment (Student),** Lai Hanh Van, 08/12/2021, HANU Training Management System.

[14] **HANU Training Management**, <http://qldt.hanu.vn/Default.aspx?page=gioithieu>, old version.

## Overview

Section Two denotes how the architecture is presented. The goals and constraints of the representation given in Section Two are described in Section Three. Then, the Use-case View, Logical View, Process View, Deployment View are briefly presented in relation to the architecture employed in the project. Finally, criteria concerning size, performance and quality of the system will be proposed.

# Architectural Representation

This document presents the architecture as a series of views; use case view, logical view, process view and deployment view. There is no separate implementation view described in this document. These are views on an underlying Unified Modeling Language (UML) model developed using Visual Paradigm and Drawio.

# Architectural Goals and Constraints

There are some key requirements and system constraints that have a significant bearing on the architecture. They are:

* The existing website provides most of the content for display. An interface to this system must be capable of handling large traffic volumes.
* The system must be available both within the university’s intranet and on the Net outside the university.
* The system must ensure complete protection of data from unauthorized access. All accesses are subject to user identification and password control.
* All performance and loading requirements, as stipulated in the Supplementary Specification, must be taken into consideration as the architecture is being developed.

# Use-Case View

This system will be built upon Model-View-Controller architecture; therefore, every use-case is equally architecturally covered. The use cases in this system are listed below. A description of these use cases can be found later in this section.

* Course Registration
* View Grade
* View Schedule
* Change Email
* Dormitory Registration
* Student Comments
* Certificate Registration
* View Digital Invoice
* View Prerequisite Courses
* Upload Announcement
* View Tuition Fee
* Dormitory Payment

## Use-Case Realizations

### Course Registration

Every semester students have to register for different subjects (subjects they have to learn in the following semester or subjects they have to relearn in the previous semester because they did not pass). Students will register on the training management system of the school at the fixed time frame required by the school. This use-case lets students register their subjects smoothly.

### View Grade

At the end of the semester, the management will update students’ scores in the system. This use case lets students view their grade.

### View Schedule

When the new semester begins, students have to look at schedules to keep up with the start/end time of the class to join their class in time. This use-case lets students view their own timetable in a quick manner.

### Change Email

When students change their email, they need to update a new email on the system to be able to catch the information in time. Therefore, this use case shows how to change or update an email.

### Dormitory Registration

To support students who cannot go to the university to register directly for the dormitory, they can register online on the system. So, this use case defines the steps of registering a dormitory.

### Student Comments

When we want to send a comment to something, or a question to be answered, we can use the comment function of the system. This use case shows how to send comments to school management.

### Certificate Registration

When students need to apply for a certificate, except for doing the procedures directly, students can register online by registering on the school's training management system and wait for the handling process. Through this use-case, students can give their register certificate request.

### View Digital Invoice

After paying the tuition fee, the system will update the electronic invoice so that students can keep track of their receipts. Through this use-case, students can view all information related to their digital invoice and download ones they need.

### View Prerequisite Courses

Students have to set up a suitable schedule before conducting course registration, students need to check the prerequisite courses which help the process of choosing subjects and registering for courses easier. Through this use-case, students can view a list of prerequisite courses and print or export this list as an Excel file.

### Upload Announcement

This use case narrative describes how an administrator can upload a new announcement so users (students, teachers...) can read it.

### View Tuition Fee

This Use-case Realization Specification gives an overview of how the View Tuition Fee use-case is realized in terms of collaborating items within the design model.

### Dormitory Payment

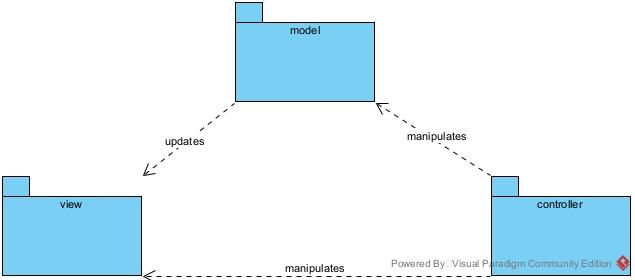
This use-case narrative describes how an user (student or teacher) can make dormitory payment in the system.

# Logical View

## Overview

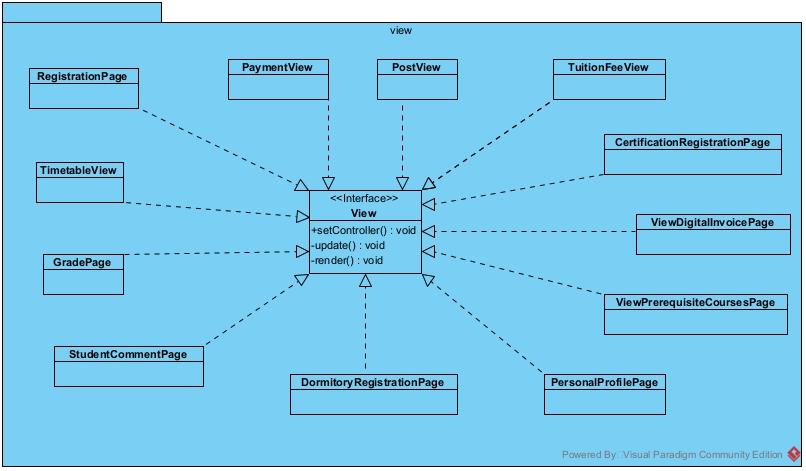
The logical view of the HANU Training Management System is comprised of 3 significant packages:

* model: contains classes that directly manages the data, logic, and rules of the HANU Training Management System and displayed in the view.
* view: contains classes that generate output representation of information to the user based on changes in the model.
* controller: contains classes that can send commands to the model to update the model’s state; it can also send commands to its associated view to change the view’s presentation of the model.



## Architecturally Significant Design Packages

### Package View

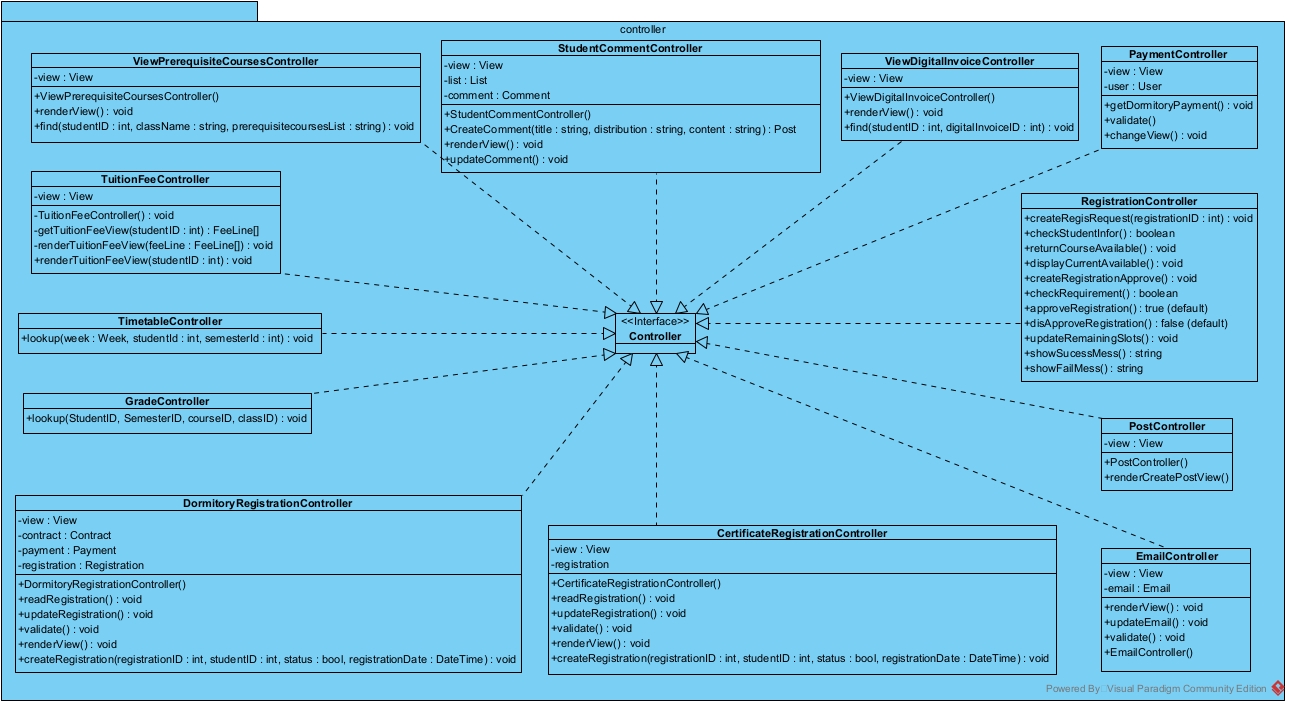


| **Name** | view |
| --- | --- |
| **Brief Description** | contains classes that generate output representation of information to the user based on changes in the model. |
| **Interfaces** | View |
| **Classes** | RegistrationPage, TimetableView, GradePage, StudentCommentPage, DormitoryRegistrationPage, PersonalProfilePage, ViewPrerequisiteCoursesPage, ViewDigitalInvoicePage, CertificationRegistrationPage, TuitionFeeView, PostView and PaymentView. |

**Interface View:**

| **Name** | View | | | |
| --- | --- | --- | --- | --- |
| **Brief Description** | shows what the Model is doing | | | |
| **Implementing Classes** | RegistrationPage, TimetableView, GradePage, StudentCommentPage, DormitoryRegistrationPage, PersonalProfilePage, ViewPrerequisiteCoursesPage, ViewDigitalInvoicePage, CertificationRegistrationPage, TuitionFeeView, PostView and PaymentView. | | | |
| **Operations** | | | | |
| **Header** | **Return Type** | **Access** | **Scope** | **Specification** |
| setController(Controller) | void | Public | Instance | Map this view with the specified controller. |
| update() | void | Public | Instance | Update this view based on changes in model. |
| render() | void | public | Instance | Render this view. |

### Package Controller



| **Name** | controller |
| --- | --- |
| **Brief Description** | contains classes that can send commands to the model to update the model’s state; it can also send commands to its associated view to change the view’s presentation of the model. |
| **Interfaces** | Controller |
| **Classes** | ViewPrerequisiteCoursesController, TuitionFeeController, TimetableController, GradeController, DormitoryRegistrationController, CertificateRegistrationController, EmailController, PostController, RegistrationController, PaymentController, ViewDigitalInvoiceController, StudentCommentController |

**Interface Controller:**

| **Name** | Controller |
| --- | --- |
| **Brief Description** | Get incoming data streams and control data into model object and update view when data changes |
| **Implementing Classes** | ViewPrerequisiteCoursesController, TuitionFeeController, TimetableController, GradeController, DormitoryRegistrationController, CertificateRegistrationController, EmailController, PostController, RegistrationController, PaymentController, ViewDigitalInvoiceController, StudentCommentController |

**Class ViewPrerequisiteCoursesController:**

| **Name** | ViewPrerequisiteCoursesController | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Brief Description** | Controller for the functionality of the View Prerequisite Courses | | | | | | |
| **Attributes** | | | | | | | |
| **Name** | **Type** | **Access** | **Mutable** | **Optional** | **Length** | **Min** | **Max** |
| view | View | private | true | false | N/A | N/A | N/A |
| **Operation** | | | | | | | |
| **Header** | **Return Type** | | **Access** | **Scope** | **Specification** | | |
| renderView() | void | | public | Instance | Handling the view request | | |
| find() | void | | public | Instance | Handling the finding request | | |

**Class TuitionFeeController:**

| **Name** | TuitionFeeController | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Brief Description** | Controller to handle tuition calculation and display to students. | | | | | | |
| **Attributes** | | | | | | | |
| **Name** | **Type** | **Access** | **Mutable** | **Optional** | **Length** | **Min** | **Max** |
| view | View | private | true | false | N/A | N/A | N/A |
| **Operation** | | | | | | | |
| **Header** | **Return Type** | | **Access** | **Scope** | **Specification** | | |
| GetTuitionFee(studentId: int) | void | | private | Instance | Gets the list of tuition fee lines for the selected student in the current semester's tuition fee report. | | |
| RenderTuitionFeeView(feeLines: FeeLine[]) | void | | private | Instance | Based on the fee lines fetched from persistent storage, renders the tuition fee report view. | | |
| RenderTuitionFeeView(studentId: int) | void | | public | Instance | The tuition fee view is displayed to the student using the route processing method. | | |

**Class TimetableController:**

| **Name** | TimetableController | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Brief Description** | Controller for the functionality of the Timetable View | | | | | | |
| **Attributes:** No special attribute. | | | | | | | |
| **Operation** | | | | | | | |
| **Header** | **Return Type** | | **Access** | **Scope** | **Specification** | | |
| lookup() | void | | private | Instance | Handling the lookup request | | |

**Class GradeController:**

| **Name** | GradeController | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Brief Description** | Controller for the functionality of the Grade View | | | | | | |
| **Attributes:** No special attribute. | | | | | | | |
| **Operation** | | | | | | | |
| **Header** | **Return Type** | | **Access** | **Scope** | **Specification** | | |
| lookup() | void | | private | Instance | Handling the finding request | | |

**Class DormitoryRegistrationController:**

| **Name** | DormitoryRegistrationController | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Brief Description** | Controller for the functionality of the Dormitory Registration | | | | | | |
| **Attributes** | | | | | | | |
| **Name** | **Type** | **Access** | **Mutable** | **Optional** | **Length** | **Min** | **Max** |
| view | View | private | true | false | N/A | N/A | N/A |
| contract | Contract | private | true | false | N/A | N/A | N/A |
| payment | Payment | private | true | false | N/A | N/A | N/A |
| registration | Registration | private | true | false | N/A | N/A | N/A |
| **Operation** | | | | | | | |
| **Header** | **Return Type** | | **Access** | **Scope** | **Specification** | | |
| readRegistration() | void | | public | Instance | Handling the reading request | | |
| updateRegistration() | void | | public | Instance | Handling the updating request | | |
| validate() | void | | public | Instance | Checking the information | | |
| renderView() | void | | public | Instance | Handling to render the registration view for the active student | | |
| createRegistartion() | void | | public | Instance | Try to create a new dormitory registration based on inputs and return a message showing whether the operation succeeded | | |

**Class CertificateRegistrationController:**

| **Name** | CertificateRegistrationController | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Brief Description** | Controller for the functionality of the Certificate Registration | | | | | | |
| **Attributes** | | | | | | | |
| **Name** | **Type** | **Access** | **Mutable** | **Optional** | **Length** | **Min** | **Max** |
| view | View | private | true | false | N/A | N/A | N/A |
| registration | Registration | private | true | false | N/A | N/A | N/A |
| **Operation** | | | | | | | |
| **Header** | **Return Type** | | **Access** | **Scope** | **Specification** | | |
| readRegistration() | void | | public | Instance | Handling the reading request | | |
| updateRegistration() | void | | public | Instance | Handling the updating request | | |
| validate() | void | | public | Instance | Checking the information | | |
| renderView() | void | | public | Instance | Handling to render the registration view for the active student | | |
| createRegistartion() | void | | public | Instance | Try to create a new certificate registration based on inputs and return a message showing whether the operation succeeded | | |

**Class EmailController:**

| **Name** | EmailController | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Brief Description** | Controller for the functionality of the Email | | | | | | |
| **Attributes** | | | | | | | |
| **Name** | **Type** | **Access** | **Mutable** | **Optional** | **Length** | **Min** | **Max** |
| view | View | private | true | false | N/A | N/A | N/A |
| email | Email | private | true | false | N/A | N/A | N/A |
| **Operation** | | | | | | | |
| **Header** | **Return Type** | | **Access** | **Scope** | **Specification** | | |
| updateEmail() | void | | public | Instance | Handling the updating request | | |
| validate() | void | | public | Instance | Checking the information | | |
| renderView() | void | | public | Instance | Handling to render the email view for the active student | | |

**Class PostController:**

| **Name** | PostController | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Brief Description** | Controller for the functionality of the Post. | | | | | | |
| **Attributes** | | | | | | | |
| **Name** | **Type** | **Access** | **Mutable** | **Optional** | **Length** | **Min** | **Max** |
| view | View | private | true | false | N/A | N/A | N/A |
| **Operation** | | | | | | | |
| **Header** | **Return Type** | | **Access** | **Scope** | **Specification** | | |
| renderCreatePostView() | void | | public | Instance | Handling to render the post view for the active student | | |

**Class RegistrationController:**

| **Name** | RegistrationController | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Brief Description** | Controller to handle course registration in the system. | | | | | | |
| **Attributes:** No special attribute. | | | | | | | |
| **Operation** | | | | | | | |
| **Header** | **Return Type** | | **Access** | **Scope** | **Specification** | | |
| checkStudentInfo() | void | | public | Instance | Checking the information. | | |
| displayCourseAvailable() | void | | public | Instance | Presenting the courses that are open to the user. | | |
| saveRegistration() | void | | public | Instance | Saving the courses that the student has enrolled in the database. | | |

**Class PaymentController:**

| **Name** | PaymentController | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Brief Description** | Controller for the functionality of the View Payment Controller. | | | | | | |
| **Attributes** | | | | | | | |
| **Name** | **Type** | **Access** | **Mutable** | **Optional** | **Length** | **Min** | **Max** |
| view | View | private | true | false | N/A | N/A | N/A |
| **Operation** | | | | | | | |
| **Header** | **Return Type** | | **Access** | **Scope** | **Specification** | | |
| save() | void | | private | Instance | Saving changes from this to the database. | | |
| validate() | void | | public | Instance | Checking the information. | | |
| changeView() | void | | public | Instance | Redirecting the user to a new page on demand. | | |

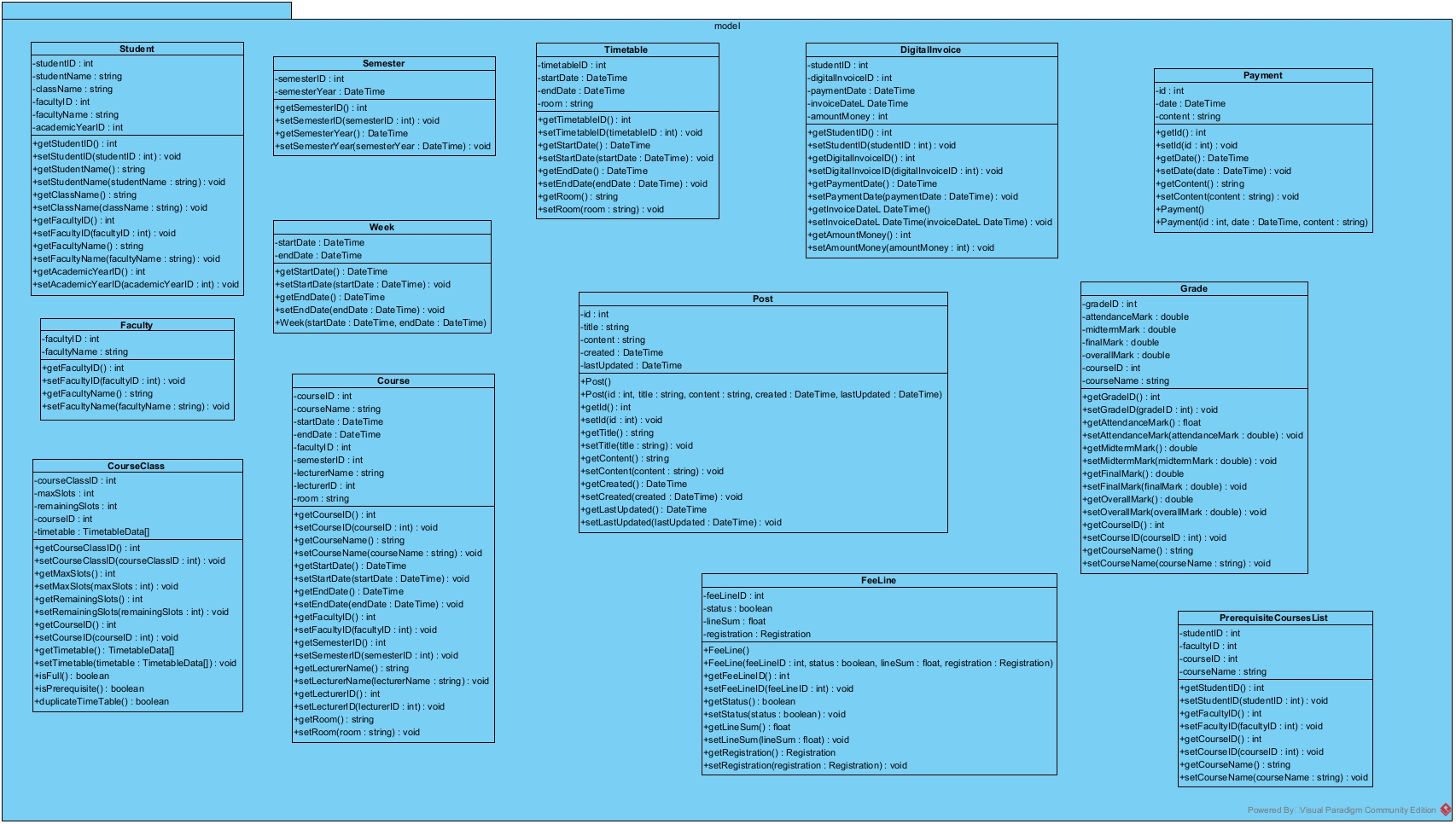
**Class ViewDigitalInvoiceController:**

| **Name** | ViewDigitalInvoiceController | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Brief Description** | Controller for the functionality of the Digital Invoice View | | | | | | |
| **Attributes** | | | | | | | |
| **Name** | **Type** | **Access** | **Mutable** | **Optional** | **Length** | **Min** | **Max** |
| view | View | private | true | false | N/A | N/A | N/A |
| **Operation** | | | | | | | |
| **Header** | **Return Type** | | **Access** | **Scope** | **Specification** | | |
| renderView() | void | | public | Instance | Handling to render the digital invoice view for the active student | | |
| find() | void | | public | Instance | Handling the finding request | | |

**Class StudentCommentController:**

| **Name** | StudentCommentController | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Brief Description** | Controller for the functionality of the Student Comment View | | | | | | |
| **Attributes** | | | | | | | |
| **Name** | **Type** | **Access** | **Mutable** | **Optional** | **Length** | **Min** | **Max** |
| view | View | private | true | false | N/A | N/A | N/A |
| list | List | private | true | false | N/A | N/A | N/A |
| comment | Comment | private | true | false | N/A | N/A | N/A |
| **Operation** | | | | | | | |
| **Header** | **Return Type** | | **Access** | **Scope** | **Specification** | | |
| renderView() | void | | public | Instance | Handling to render the digital invoice view for the active student | | |
| createComment() | void | | public | Instance | Handling the finding request | | |
| updateComment() | void | | public | Instance | Handling and storing data changes | | |

### Package Model



In the system, all together for center functionalities to be carried out, these classes assume an indispensable part: Student, Registration, Payment, Grades, Post and Fee Line. Following are the particulars of these models.

**Class Student:**

| **Name** | Registration | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Brief Description** | Data model class for student entity in the database | | | | | | |
| **Attributes** | | | | | | | |
| **Name** | **Type** | **Access** | **Mutable** | **Optional** | **Length** | **Min** | **Max** |
| studentId | int | private | True | False | N/A | 1 | N/A |
| studentName | String | private | True | False | N/A | N/A | N/A |
| className | String | private | True | False | N/A | 1 | N/A |
| facultyId | int | private | True | False | N/A | 1 | N/A |
| facultyName | String | private | True | False | N/A | 1 | N/A |
| academicYearId | int | private | True | False | N/A | 1 | N/A |
| **Operations:** Only public get/set operations corresponding to the attributes. | | | | | | | |

**Class Registration:**

| **Name** | Registration | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Brief Description** | Data model class for registration entity in the database | | | | | | |
| **Attributes** | | | | | | | |
| **Name** | **Type** | **Access** | **Mutable** | **Optional** | **Length** | **Min** | **Max** |
| registrationId | int | private | True | False | N/A | 1 | N/A |
| status | boolean | private | True | False | N/A | N/A | N/A |
| courseId | int | private | True | False | N/A | 1 | N/A |
| studentId | int | private | True | False | N/A | 1 | N/A |
| courseClassId | int | private | True | False | N/A | 1 | N/A |
| semesterId | int | private | True | False | N/A | 1 | N/A |
| **Operations:** Only public get/set operations corresponding to the attributes. | | | | | | | |

**Class Payment:**

| **Name** | Registration | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Brief Description** | Data model class for payment entity in the database | | | | | | |
| **Attributes** | | | | | | | |
| **Name** | **Type** | **Access** | **Mutable** | **Optional** | **Length** | **Min** | **Max** |
| paymentId | int | private | True | False | N/A | 1 | N/A |
| date | DateTime | private | True | False | N/A | N/A | N/A |
| content | String | private | True | False | N/A | N/A | N/A |
| **Operations:** Only public get/set operations corresponding to the attributes. | | | | | | | |

**Class Grades:**

| **Name** | Grades | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Brief Description** | Data model class for grades entity in the database | | | | | | |
| **Attributes** | | | | | | | |
| **Name** | **Type** | **Access** | **Mutable** | **Optional** | **Length** | **Min** | **Max** |
| id | int | private | True | False | N/A | 1 | N/A |
| attendanceMark | double | private | True | False | N/A | N/A | N/A |
| midtermMark | double | private | True | False | N/A | N/A | N/A |
| finalMark | double | private | True | False | N/A | N/A | N/A |
| overallMark | double | private | True | False | N/A | N/A | N/A |
| courseName | String | private | True | False | N/A | 1 | N/A |
| studentId | int | private | True | False | N/A | 1 | N/A |
| **Operations:** Only public get/set operations corresponding to the attributes. | | | | | | | |

**Class Post:**

| **Name** | Grades | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Brief Description** | Data model class for post entity in the database | | | | | | |
| **Attributes** | | | | | | | |
| **Name** | **Type** | **Access** | **Mutable** | **Optional** | **Length** | **Min** | **Max** |
| id | int | private | True | False | N/A | 1 | N/A |
| content | String | private | True | False | N/A | N/A | N/A |
| created | DateTime | private | True | False | N/A | N/A | N/A |
| title | String | private | True | False | N/A | N/A | N/A |
| lastUpdated | DateTime | private | True | False | N/A | N/A | N/A |
| **Operations:** Only public get/set operations corresponding to the attributes. | | | | | | | |

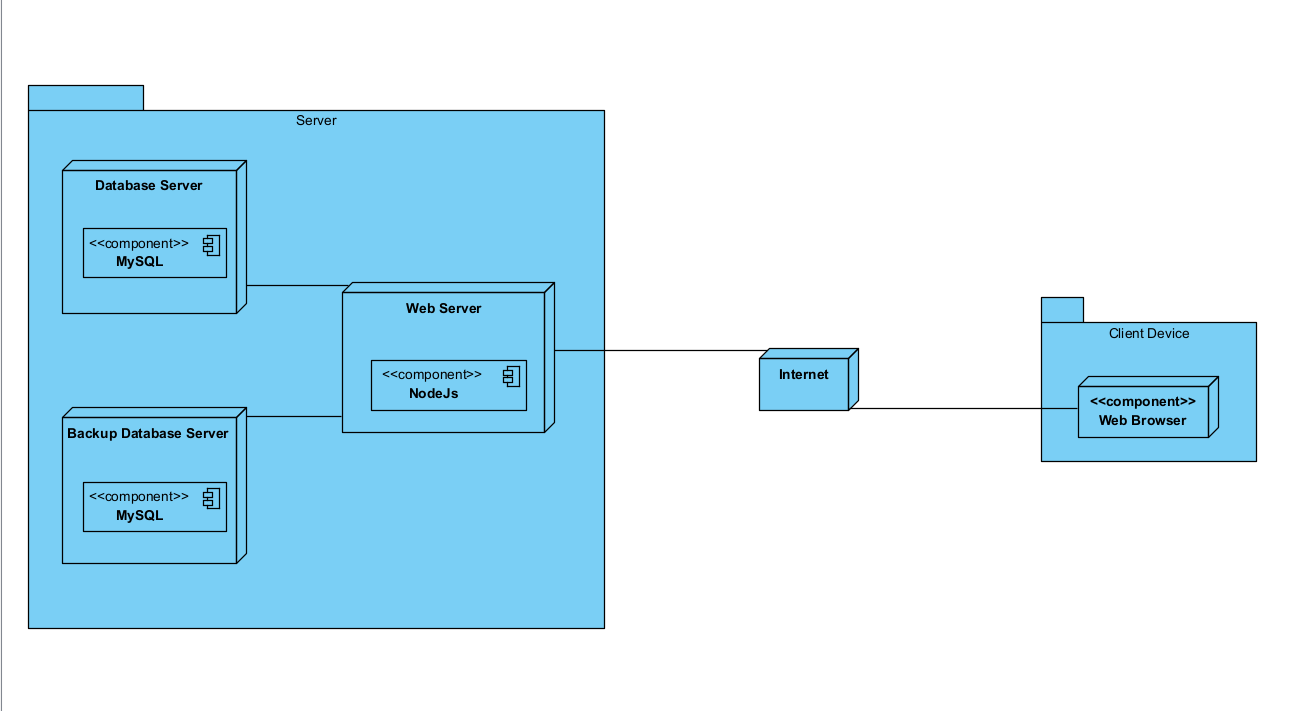
**Class FeeLine:**

| **Name** | Grades | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Brief Description** | Data model class for fee line entity in the database | | | | | | |
| **Attributes** | | | | | | | |
| **Name** | **Type** | **Access** | **Mutable** | **Optional** | **Length** | **Min** | **Max** |
| feeLineId | int | private | True | False | N/A | 1 | N/A |
| status | String | private | True | False | N/A | N/A | N/A |
| lineSum | float | private | True | False | N/A | N/A | N/A |
| registration | String | private | True | False | N/A | N/A | N/A |
| **Operations:** Only public get/set operations corresponding to the attributes. | | | | | | | |

# Process View (optional)

# Deployment View

The system will be hosted on a single server cluster. The system consists of these mandatory physical nodes: a web server, a database server and a backup database server. This also contains a NodeJs Web server on which our web application will be housed, as well as two MySQL database servers — one for active operation and one for backup. Clients can connect to the system via an Internet connection. A simplified perspective of the Training Management System implementation is provided.

**

# Implementation View

Because the system's implementation is solely based on the design, the implementation perspective will be ignored in this document.

# Data View (optional)

# Size and Performance

The major dimensioning characteristics of the software that impact the architecture and performance constraints:

* The system shall support up to 1000 concurrent users against the primary database at any given time, and up to 500 concurrent users against the local servers at any one time.
* The system must perform all functions with minimal time delays.
* The system must also accurately save all information transactions.

# Quality

The system architecture supports the quality requirements:

* In order to maintain the highest degree of system integrity, the system can ensure that all information transitions are saved.
* Databases will be backed up daily in concern with safety implications.
* The system is capable of displaying correctly on different devices web browsers of any screen size (i.e., responsive design).

All system functions are available through popular web browsers, for instance, Google Chrome, Mozilla Firefox, Opera, Safari, Microsoft Edge, Internet Explorer