HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

School of Information and communications technology

Software Design Document

Version 1.3

<AIMS>

Subject: <ITSS>

<21>

<List of participants>

*Hanoi,* *<month, year>*

*<All notations inside the angle bracket are not part of this document, for its purpose is for extra instruction. When using this document, please erase all these notations and/or replace them with corresponding content as instructed>*

*<This document, written by Prof. NGUYEN Thi Thu Trang, is used as a case study for student with related courses. Any modifications and/or utilization without the consent of the author is strictly forbidden>*

Table of Contents

Table of Contents 1

1 Introduction 5

1.1 Objective 5

1.2 Scope 5

1.3 Glossary 5

1.4 References 5

2 Overall Description 7

2.1 General Overview 7

2.2 Assumptions/Constraints/Risks 7

2.2.1 Assumptions 7

2.2.2 Constraints 7

2.2.3 Risks 8

3 System Architecture and Architecture Design 9

3.1 Architectural Patterns 9

3.2 Interaction Diagrams 9

3.3 Analysis Class Diagrams 9

3.4 Unified Analysis Class Diagram 9

3.5 Security Software Architecture 9

4 Detailed Design 10

4.1 User Interface Design 10

4.1.1 Screen Configuration Standardization 10

4.1.2 Screen Transition Diagrams 10

4.1.3 Screen Specifications 10

4.2 Data Modeling 10

4.2.1 Conceptual Data Modeling 10

4.2.2 Database Design 10

4.3 Non-Database Management System Files 11

4.4 Class Design 11

4.4.1 General Class Diagram 11

4.4.2 Class Diagrams 11

4.4.3 Class Design 11

5 Design Considerations 13

5.1 Goals and Guidelines 13

5.2 Architectural Strategies 13

5.3 Coupling and Cohesion 14

5.4 Design Principles 14

5.5 Design Patterns 15

**List of Figures**

**No table of figures entries found.**

**List of Tables**

[Table 1. Example of table design 10](#_Toc197437803)

[Table 1. Example of attribute design 12](#_Toc197437804)

[Table 1. Example of operation design 12](#_Toc197437805)

[Table 4. Example of attribute design 14](#_Toc197437806)

# Introduction

*<The following subsections of the Software Design Document (SDD) document should provide an overview of the entire SDD.>*

## Objective

<*Identify the purpose of this SDD and its intended audience. In this subsection, describe the purpose of the SDD and specify the intended audience for the SDD*>

Tài liệu Thiết kế Phần mềm (Software Design Document - SDD) này nhằm mục đích mô tả chi tiết kiến trúc và thiết kế của hệ thống **AIMS (Hệ thống Quản lý Thông tin Tự động)**. Tài liệu đóng vai trò cầu nối giữa Tài liệu Đặc tả Yêu cầu Phần mềm (SRS) và các hoạt động thiết kế chi tiết, lập trình, kiểm thử trong quá trình phát triển phần mềm.

Mục tiêu của tài liệu là trình bày rõ ràng cấu trúc tổng thể của hệ thống, thiết kế dữ liệu, thiết kế các thành phần, thiết kế giao diện, cũng như các quyết định thiết kế, các mẫu thiết kế (design pattern) được áp dụng. Qua đó đảm bảo tất cả các bên liên quan có nhận thức thống nhất về cách thức hệ thống sẽ được xây dựng để đáp ứng các yêu cầu đã đặt ra.

Đối tượng sử dụng tài liệu này bao gồm:

* **Kiến trúc sư hệ thống**: Để xem xét, phê duyệt kiến trúc và thiết kế hệ thống.
* **Lập trình viên**: Tham khảo trong quá trình triển khai và lập trình phần mềm đúng theo thiết kế.
* **Nhân viên kiểm thử (Tester)**: Nắm rõ thiết kế để xây dựng các ca kiểm thử phù hợp.
* **Quản lý dự án và trưởng nhóm kỹ thuật**: Theo dõi tiến độ, đảm bảo thiết kế phù hợp với yêu cầu nghiệp vụ và mục tiêu dự án.
* **Đội ngũ bảo trì và hỗ trợ kỹ thuật**: Hiểu cấu trúc, thiết kế của hệ thống để phục vụ cho việc bảo trì, nâng cấp hoặc xử lý sự cố sau này.

## Scope

<*In this subsection:*

1. *Identify the software product(s) to be produced by name*
2. *Explain what the software product(s) will, and, if necessary, will not do*
3. *Describe the application of the software being specified, including relevant benefits, objectives, and goals*
4. *Be consistent with similar statements in higher-level specifications if they exist*

*This should be an executive-level summary. Do not enumerate the whole requirements list here*

*Note that this will be similar to what was written in the SRS.*

>

Phần mềm được mô tả trong tài liệu này là **AIMS (Hệ thống Quản lý Thông tin Tự động)**.

Hệ thống AIMS là một giải pháp phần mềm quản lý thông tin nhằm tự động hóa quy trình thu thập, lưu trữ, xử lý và báo cáo dữ liệu trong tổ chức. Hệ thống cung cấp các chức năng quản lý dữ liệu người dùng, dữ liệu nghiệp vụ, quản lý quy trình làm việc, phân quyền người dùng, và tạo các báo cáo tổng hợp phục vụ quản lý và ra quyết định.

**Hệ thống AIMS sẽ bao gồm các chức năng chính sau:**

* Quản lý thông tin người dùng và phân quyền.
* Quản lý dữ liệu nghiệp vụ theo từng module.
* Hỗ trợ quy trình làm việc tự động.
* Cung cấp báo cáo thống kê, báo cáo tùy chỉnh.
* Hỗ trợ tìm kiếm và truy xuất dữ liệu nhanh chóng.
* Cung cấp giao diện web thân thiện, dễ sử dụng.

**Hệ thống AIMS sẽ không bao gồm:**

* Các chức năng kế toán tài chính chi tiết.
* Các chức năng phân tích dữ liệu nâng cao (BI, AI) ngoài phạm vi báo cáo cơ bản.
* Các tích hợp với hệ thống bên ngoài nếu không được mô tả trong yêu cầu chi tiết.

**Ứng dụng của phần mềm:**  
Hệ thống AIMS được sử dụng nhằm hỗ trợ các tổ chức/doanh nghiệp nâng cao hiệu quả quản lý thông tin, giảm thiểu thao tác thủ công, tăng tính chính xác trong xử lý dữ liệu, đồng thời hỗ trợ lãnh đạo có đầy đủ thông tin để đưa ra quyết định kịp thời và chính xác.

**Lợi ích, mục tiêu và mong đợi:**

* Tự động hóa tối đa các nghiệp vụ liên quan đến quản lý thông tin.
* Đảm bảo tính chính xác, bảo mật và toàn vẹn dữ liệu.
* Giảm chi phí vận hành và nhân sự.
* Nâng cao khả năng truy xuất, phân tích và khai thác dữ liệu.
* Hỗ trợ đa nền tảng, dễ dàng mở rộng và tích hợp trong tương lai.

## Glossary

*<Listing and explaining the terms appearing in the software’s profession and this document. Any assumption of the reader’s prior knowledge or experience on the subject is ill advised>*

## References

Centers for Medicare & Medicaid Services. (n.d.). *System Design Document Template.* Retrieved from Centers for Medicare & Medicaid Services: https://www.cms.gov/Research-Statistics-Data-and-Systems/CMS-Information-Technology/XLC/Downloads/SystemDesignDocument.docx

*<Listing the referenced material used in this document, including the one related to the project>*

# Overall Description

<*This section describes the principles and strategies to be used as guidelines when designing and implementing the system.>*

## General Overview

*<Briefly introduce the system context and the basic design approach or organization. Provide a brief overview of the system and software architectures and the design goals. Include the high-level context diagram(s) for the system and subsystems provided in previous documents like SRS (e.g., general use case diagram, lower-level use case diagrams, activity diagrams), updated as necessary to reflect any changes that have been made based on more current information or understanding. If the high-level context diagram has been updated, identify the changes that were made and why>*

Hệ thống **AIMS (Hệ thống Quản lý Thông tin Tự động)** được xây dựng nhằm hỗ trợ các tổ chức/doanh nghiệp trong việc quản lý thông tin nội bộ một cách hiệu quả, chính xác và tự động hóa cao. AIMS đóng vai trò trung tâm trong việc thu thập, lưu trữ, xử lý và cung cấp dữ liệu phục vụ các hoạt động vận hành và quản lý.

Hệ thống tương tác với các nhóm người dùng khác nhau như: quản trị viên, người dùng nội bộ, nhân viên nhập liệu, cán bộ quản lý... thông qua giao diện web và các API tích hợp.

#### Phương pháp và cách tiếp cận thiết kế

Thiết kế hệ thống AIMS được thực hiện theo phương pháp hướng đối tượng (Object-Oriented Design - OOD), áp dụng các nguyên tắc thiết kế phần mềm chuẩn như phân tách thành các module độc lập, tái sử dụng cao, dễ bảo trì và dễ mở rộng.

Kiến trúc hệ thống áp dụng mô hình nhiều lớp (Multi-layer Architecture), gồm các lớp cơ bản:

* **Lớp trình bày (Presentation Layer)**: Giao diện web và API.
* **Lớp nghiệp vụ (Business Logic Layer)**: Xử lý logic nghiệp vụ.
* **Lớp dữ liệu (Data Access Layer)**: Quản lý truy cập và lưu trữ dữ liệu trong cơ sở dữ liệu.

#### Mục tiêu thiết kế

* Đảm bảo tính **modular**, dễ mở rộng.
* Đảm bảo tính **bảo mật cao** trong quản lý dữ liệu và phân quyền.
* Đảm bảo hiệu suất, khả năng xử lý dữ liệu lớn.
* Hỗ trợ khả năng tích hợp với các hệ thống bên ngoài thông qua API.
* Dễ dàng bảo trì, vận hành.

## Assumptions/Constraints/Risks

### Assumptions

*<Describe any assumptions or dependencies regarding the system, software and its use. These may concern such issues as: related software or hardware, operating systems, end-user characteristics, and possible and/or probable changes in functionality>*

Trong quá trình thiết kế và xây dựng hệ thống **AIMS**, các giả định và phụ thuộc sau đây được đưa ra nhằm xác định phạm vi và điều kiện môi trường triển khai, vận hành hệ thống:

#### Giả định về phần mềm và phần cứng liên quan

* Hệ thống sẽ được triển khai trên nền tảng **máy chủ web sử dụng hệ điều hành Linux hoặc Windows Server**.
* Hệ thống sử dụng **Cơ sở dữ liệu quan hệ (PostgreSQL hoặc SQL Server)**.
* Hệ thống yêu cầu **trình duyệt web hiện đại** (Google Chrome, Firefox, Edge) để truy cập giao diện người dùng.
* Hệ thống có thể yêu cầu **máy chủ API độc lập** nếu tích hợp với các hệ thống bên ngoài.

#### Giả định về môi trường vận hành

* Môi trường mạng nơi triển khai hệ thống có độ ổn định và bảo mật cao, đảm bảo kết nối liên tục giữa các thành phần hệ thống.
* Có hệ thống tường lửa, phân vùng mạng, và các cơ chế bảo mật dữ liệu phù hợp.

#### Giả định về người dùng cuối

* Người dùng cuối có kiến thức cơ bản về sử dụng máy tính, trình duyệt web.
* Người dùng được phân quyền rõ ràng theo vai trò (quản trị viên, nhân viên, cán bộ quản lý...).
* Hệ thống không yêu cầu người dùng phải có kiến thức về công nghệ thông tin nâng cao.

#### Giả định về sự thay đổi chức năng

* Trong giai đoạn đầu, hệ thống tập trung vào các chức năng cơ bản như quản lý dữ liệu, phân quyền, báo cáo.
* Các chức năng nâng cao như phân tích dữ liệu nâng cao (BI), trí tuệ nhân tạo (AI) hoặc tích hợp các hệ thống bên ngoài sẽ được xem xét triển khai ở các giai đoạn sau tùy theo nhu cầu.
* Hệ thống được thiết kế có khả năng **mở rộng module và chức năng linh hoạt** trong tương lai.

#### Giả định về các yếu tố phụ thuộc bên ngoài

* Các dữ liệu nhập vào hệ thống được giả định là đã được kiểm tra, đảm bảo tính chính xác từ phía người dùng nhập liệu.
* Các hệ thống tích hợp ngoài (nếu có) sẽ cung cấp API ổn định, có tài liệu mô tả rõ ràng.

### Constraints

*<Describe any global limitations or constraints that have a significant impact on the design of the system’s hardware, software and/or communications, and describe the associated impact. Such constraints may be imposed by any of the following (the list is not exhaustive):*

* *Hardware or software environment*
* *End-user environment*
* *Availability or volatility of resources*
* *Standards compliance*
* *Interoperability requirements*
* *Interface/protocol requirements*
* *Licensing requirements*
* *Data repository and distribution requirements*
* *Security requirements (or other such regulations)*
* *Memory or other capacity limitations*
* *Performance requirements*
* *Network communications*
* *Verification and validation requirements (testing)*
* *Other means of addressing quality goals*
* *Other requirements described in the Requirements Document*

*>*

Trong quá trình thiết kế và triển khai hệ thống **AIMS**, các ràng buộc sau đây cần được xem xét do có ảnh hưởng lớn đến thiết kế phần cứng, phần mềm và truyền thông:

#### Ràng buộc phần cứng, phần mềm và môi trường người dùng

* **Phần cứng hạn chế**: Hệ thống phải hoạt động tốt trên hạ tầng máy chủ có cấu hình trung bình, phù hợp với ngân sách của tổ chức.
* **Môi trường người dùng**: Giao diện người dùng phải tương thích với nhiều trình duyệt hiện đại, không yêu cầu cài đặt phần mềm riêng biệt.

#### Ràng buộc tài nguyên

* Hệ thống phải đảm bảo hiệu suất khi xử lý dữ liệu lớn, đồng thời giới hạn mức sử dụng bộ nhớ và CPU để tránh ảnh hưởng đến các hệ thống khác.
* Cơ sở dữ liệu phải được tối ưu hóa để đảm bảo khả năng xử lý nhanh với số lượng lớn bản ghi.

#### Tuân thủ tiêu chuẩn và giao thức

* Hệ thống phải tuân thủ các chuẩn bảo mật dữ liệu theo quy định nội bộ và luật pháp.
* Giao tiếp API phải sử dụng chuẩn RESTful hoặc tương đương, bảo đảm tính mở rộng và tương thích với các hệ thống bên ngoài.
* Yêu cầu hỗ trợ giao thức HTTPS cho tất cả các giao tiếp mạng.

#### Yêu cầu bảo mật và an toàn dữ liệu

* Hệ thống phải đảm bảo an toàn dữ liệu ở mức cao, bao gồm mã hóa, kiểm soát phân quyền chi tiết, ghi nhật ký truy cập.
* Phải tuân thủ các quy định về bảo vệ dữ liệu cá nhân (nếu áp dụng).

#### Các hạn chế về hiệu suất và khả năng mở rộng

* Hệ thống cần bảo đảm khả năng mở rộng ngang (scale-out) nếu có nhu cầu tăng tải.
* Thời gian phản hồi cho các thao tác nghiệp vụ quan trọng không được vượt quá 3 giây.

#### Các yêu cầu về xác minh, kiểm thử

* Hệ thống phải hỗ trợ triển khai các kịch bản kiểm thử tự động (unit test, integration test).
* Các lỗi phải được ghi nhật ký đầy đủ và có thông báo rõ ràng cho người dùng.

### Risks

*<Describe any risks associated with the system design and proposed mitigation strategies.>*

# System Architecture and Architecture Design

<*Briefly describe the architectural design steps*>

## Architectural Patterns

*<Specify and briefly describe the chosen architectural patterns and the reasons why they were chosen>*

Hệ thống AIMS áp dụng mô hình Kiến trúc nhiều lớp (Multi-layer Architecture) kết hợp với mô hình Client-Server và RESTful API, với các lý do cụ thể như sau:

| Mô hình kiến trúc | Mô tả | Lý do lựa chọn |
| --- | --- | --- |
| Kiến trúc nhiều lớp (Multi-layer) | Hệ thống được chia thành các lớp riêng biệt: Lớp trình bày (UI), lớp xử lý nghiệp vụ (BLL), lớp truy cập dữ liệu (DAL) | - Tách biệt rõ ràng các chức năng. - Dễ bảo trì, mở rộng, nâng cấp. - Tăng khả năng tái sử dụng và kiểm soát lỗi hiệu quả. |
| Client-Server Architecture | Giao diện người dùng (Client) và phần mềm xử lý (Server) hoạt động độc lập và giao tiếp qua giao thức HTTP/HTTPS | - Hỗ trợ nhiều loại client (Web, mobile...). - Giảm tải cho client. - Dễ mở rộng quy mô hệ thống. |
| RESTful API | Giao tiếp giữa các thành phần được thực hiện qua API chuẩn RESTful | - Tương thích cao, dễ tích hợp với các hệ thống khác. - Chuẩn hóa giao tiếp. - Tăng khả năng mở rộng và tái sử dụng dịch vụ. |
| MVC (Model - View - Controller) cho UI | Lớp giao diện người dùng tuân theo mô hình MVC, tách biệt phần giao diện, xử lý sự kiện và dữ liệu | - Dễ phát triển, bảo trì UI. - Giảm thiểu lỗi trong giao diện. - Dễ thay đổi hoặc nâng cấp UI khi cần. |

## Interaction Diagrams

## Analysis Class Diagrams

## Unified Analysis Class Diagram

## Security Software Architecture

*<Describe the software components and configuration supporting the security and privacy of the system. Specify the architecture for (1) authentication to validate user identity before allowing access to the system;(2) authorization of users to perform functional activity once logged into the system, (3) encryption protocol to support the business risks and the nature of information, and (4) logging and auditing design, if required.>*

# Detailed Design

## User Interface Design

*<Suppose that you design a Graphical User Interface (GUI)>*

### Screen Configuration Standardization

### Screen Transition Diagrams

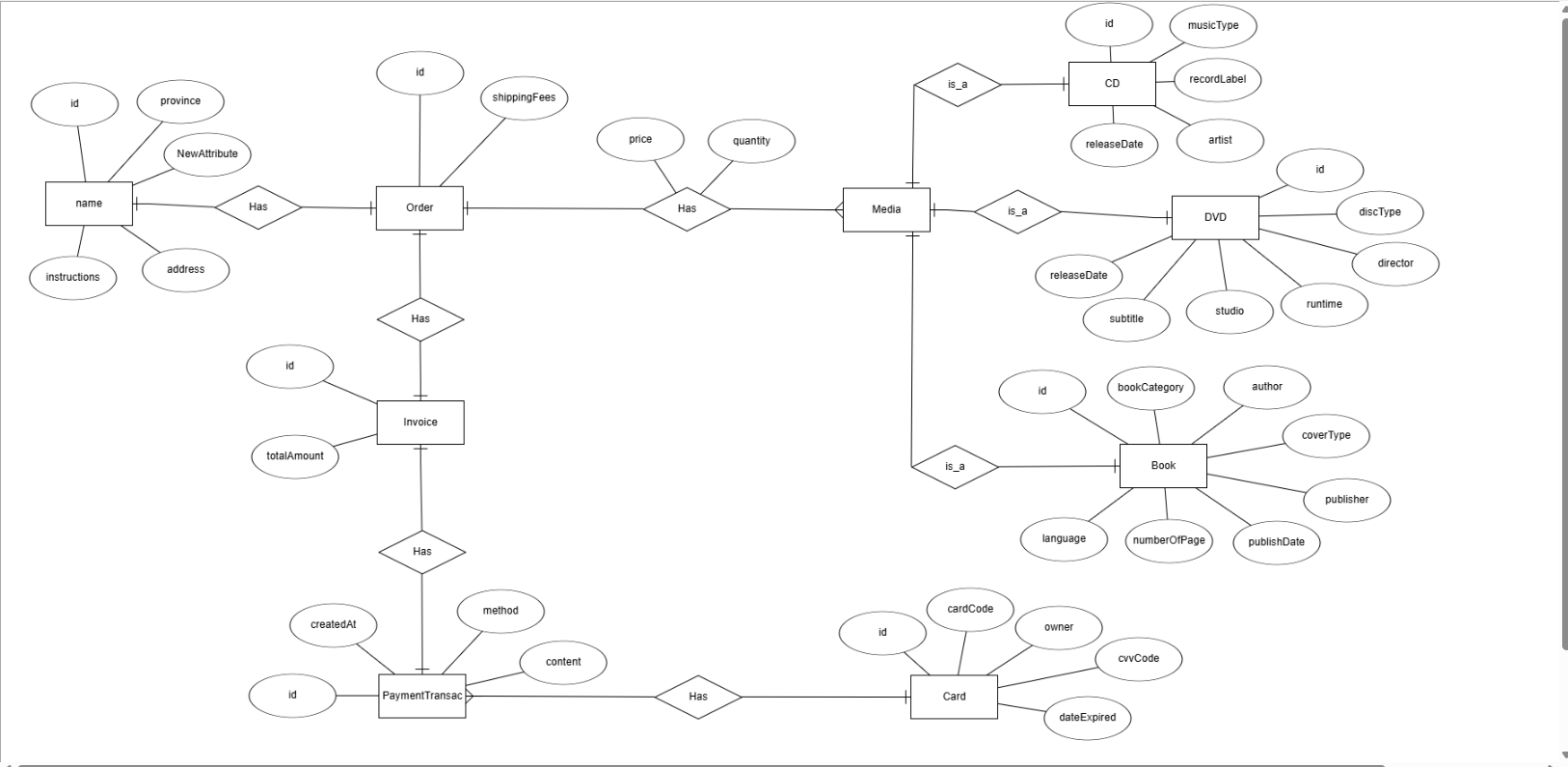
### Screen Specifications

*<Screen images should be included in the screen specifications>*

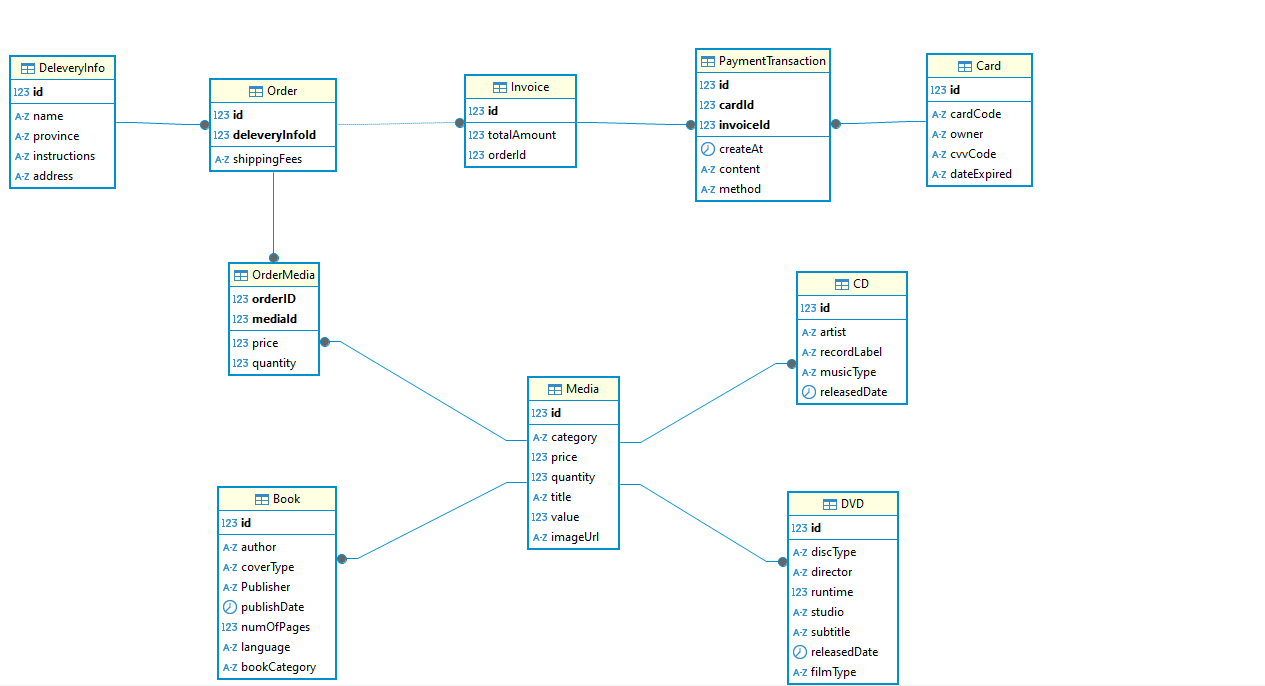
## Data Modeling

### Conceptual Data Modeling

*<E-R Diagram image and description of entities and relationships>*

**

### Database Design



#### Database Management System

*<Specify what is the decision of Database Management System (DBMS) and give some description of the DBMS>*

#### Database Diagram

<

* *Show the process to design database from E-R diagram*
* *Show the diagram of DB design*

*>*

#### Database Detail Design

<

*Give a detail design of each element in the DB diagram. For instance, in a Relational DBMS, give a detail design for each Table and their constraints, illustrated in below table (PK: Primary Key, FK: Foreign Key).*

Table 1. Example of table design

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *#* | *PK* | *FK* | *Column name* | *Data type* | *Default value* | *Mandatory* | *Description* |
| 1 | x |  | ProductID |  |  |  |  |
| 2 |  | x | CategoryID |  |  |  |  |

*You may add indexing, trigger, view, etc.*

*Give the database script*>

## Non-Database Management System Files

*<Provide the detailed description of all non-DBMS files if any and include a narrative description of the usage of each file that identifies if the file is used for input, output, or both, and if the file is a temporary file. Also provide an indication of which modules read and write the file and include file structures (refer to the data dictionary). As appropriate, the file structure information should include the following:*

*• Record structures, record keys or indexes, and data elements referenced within the records*

*• Record length (fixed or maximum variable length) and blocking factors*

*• Access method (e.g., index sequential, virtual sequential, random access, etc.)*

*• Estimate of the file size or volume of data within the file, including overhead resulting from file access methods*

*• Definition of the update frequency of the file (If the file is part of an online transaction-based system, provide the estimated number of transactions per unit of time, and the statistical mean, mode, and distribution of those transactions.)*

*• Backup and recovery specifications>*

## Class Design

### General Class Diagram

<General class diagram which shows the whole class diagram of the software. This diagram may have packages, subsystems and classes. Classes in this diagram may not have all attributes and operations>

### Class Diagrams

<Detail class diagram with full attributes and operations>

#### Class Diagram for Package A

#### Class Diagram for Subsystem B

…

### Class Design

<Detail design for each class>

#### Class “SampleClass1”

<SampleClass1 class image in UML>

Table 2. Example of attribute design

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Name* | *Data type* | *Default value* | *Description* |
| 1 |  |  |  |  |
| 2 |  |  |  |  |

Table 3. Example of operation design

|  |  |  |  |
| --- | --- | --- | --- |
| *#* | *Name* | *Return type* | *Description (purpose)* |
| 1 |  |  |  |
| 2 |  |  |  |

*Parameter*:

* x: Default value, description
* y: Default value, description

*Exception*:

* AException if …
* BException if …

**Method**

How to use parameters / attributes

Flowchart / activity diagram / sequence diagram if the method has a complex/special algorithm

**State**

State diagram if any

#### Class “SampleClass2”

…

# Design Considerations

***<Describe issues which need to be addressed or resolved before attempting to devise a complete design solution. Remember that, you have to refactor your source code to strictly follow the final design>***

## Goals and Guidelines

*<Describe any goals, guidelines, principles, or priorities which dominate or embody the design of the system and its software.*

*Examples of such goals might be: an emphasis on speed versus memory use; or working, looking, or “feeling” like an existing product.*

*Guidelines include coding guidelines and conventions.*

*For each such goal or guideline, describe the reason for its desirability unless it is implicitly obvious.*

*Describe any design policies and/or tactics that do not have sweeping architectural implications (meaning they would not significantly affect the overall organization of the system and its high-level structures), but which nonetheless affect the details of the interface and/or implementation of various aspects of the system (e.g., choice of which specific product to use)*>

#### Mục tiêu thiết kế

1. **Tối ưu hóa hiệu suất và tốc độ**:
   * Hệ thống phải đảm bảo hiệu suất tối ưu trong việc xử lý các tác vụ người dùng như tìm kiếm, cập nhật và báo cáo.
   * **Lý do**: Các tác vụ này có thể được thực hiện nhiều lần và với số lượng lớn, vì vậy yêu cầu về hiệu suất là cực kỳ quan trọng, đặc biệt trong môi trường doanh nghiệp có lượng dữ liệu lớn.
2. **Đảm bảo tính bảo mật cao**:
   * Hệ thống phải áp dụng các cơ chế bảo mật tiên tiến như mã hóa SSL cho giao thức mạng, phân quyền người dùng chặt chẽ và bảo vệ dữ liệu nhạy cảm.
   * **Lý do**: Bảo mật thông tin là ưu tiên hàng đầu, đặc biệt khi xử lý dữ liệu nhạy cảm và cá nhân của người dùng hoặc khách hàng.
3. **Khả năng mở rộng và tính linh hoạt**:
   * Hệ thống được thiết kế để dễ dàng mở rộng, thêm chức năng mới mà không cần thay đổi nhiều cấu trúc hiện có.
   * **Lý do**: Môi trường và yêu cầu của hệ thống có thể thay đổi theo thời gian, vì vậy hệ thống phải có khả năng mở rộng mà không ảnh hưởng đến hiệu suất và độ ổn định.
4. **Tính dễ sử dụng và thân thiện với người dùng**:
   * Giao diện người dùng (UI) phải trực quan, dễ hiểu, giúp người dùng thực hiện công việc mà không gặp phải sự phức tạp không cần thiết.
   * **Lý do**: Hệ thống phục vụ nhiều đối tượng người dùng, từ nhân viên đến quản lý, do đó UI cần phải đơn giản, dễ sử dụng để đảm bảo hiệu quả công việc cao nhất.

#### Hướng dẫn thiết kế

1. **Hướng dẫn về mã nguồn**:
   * **Sử dụng quy tắc đặt tên thống nhất**: Các tên biến, hàm và lớp phải mang tính mô tả cao và dễ hiểu, tuân thủ quy ước đặt tên như camelCase cho biến và PascalCase cho lớp.
   * **Tuân thủ chuẩn lập trình hướng đối tượng (OOP)**: Mã nguồn phải tối thiểu hóa sự phụ thuộc giữa các lớp và dễ dàng tái sử dụng. Đặc biệt, phải chú trọng đến nguyên lý SOLID trong OOP.
   * **Lý do**: Điều này giúp mã nguồn dễ bảo trì, dễ đọc và dễ dàng mở rộng hoặc thay đổi mà không ảnh hưởng đến toàn bộ hệ thống.
2. **Hướng dẫn về giao diện người dùng (UI)**:
   * **Sử dụng các thành phần giao diện chuẩn**: Các button, form, table phải được thiết kế đồng bộ và dễ nhìn, tránh sử dụng các kiểu thiết kế phức tạp hoặc gây khó khăn cho người sử dụng.
   * **Tuân thủ nguyên tắc thiết kế Material Design**: Các yếu tố như không gian, sự phân biệt các thành phần và màu sắc phải nhất quán.
   * **Lý do**: Đảm bảo người dùng có trải nghiệm tốt nhất và dễ dàng tương tác với hệ thống.
3. **Hướng dẫn về hiệu suất và tối ưu hóa**:
   * **Tối ưu hóa truy vấn cơ sở dữ liệu (SQL)**: Tránh việc sử dụng các truy vấn không tối ưu, giảm thiểu việc truy xuất dữ liệu không cần thiết từ cơ sở dữ liệu.
   * **Lý do**: Các truy vấn không tối ưu có thể dẫn đến giảm hiệu suất và khả năng xử lý chậm, đặc biệt khi dữ liệu lớn.
4. **Hướng dẫn về an toàn và bảo mật**:
   * **Đảm bảo bảo mật dữ liệu đầu vào và đầu ra**: Kiểm tra và xử lý tất cả các dữ liệu đầu vào từ người dùng để tránh các cuộc tấn công như SQL injection, cross-site scripting (XSS), và cross-site request forgery (CSRF).
   * **Lý do**: Hệ thống cần phải bảo vệ dữ liệu nhạy cảm và tránh các lỗ hổng bảo mật có thể gây rủi ro cho thông tin người dùng và tổ chức.

#### Chính sách và chiến lược thiết kế

1. **Chọn lựa công nghệ phù hợp**:
   * Hệ thống sử dụng **Java Spring Boot** cho phần backend vì tính bảo mật, khả năng mở rộng và dễ dàng tích hợp với các dịch vụ khác.
   * **Frontend** sử dụng **ReactJS** vì tính linh hoạt, khả năng tương tác động và dễ dàng tích hợp với API RESTful.
   * **Lý do**: Các công nghệ này được chọn vì tính ổn định, cộng đồng phát triển mạnh và khả năng duy trì lâu dài.
2. **Tối ưu hóa tài nguyên và khả năng chịu tải**:
   * Cấu hình hệ thống cần tối ưu hóa tài nguyên phần cứng (RAM, CPU) để đảm bảo hệ thống có thể xử lý hàng nghìn yêu cầu đồng thời.
   * **Lý do**: Các yêu cầu về hiệu suất và khả năng chịu tải cao đảm bảo rằng hệ thống không bị sập khi có lượng truy cập lớn.

## Architectural Strategies

*<Describe any design decisions and/or strategies that affect the overall organization of the system and its higher-level structures. These strategies should provide insight into the key abstractions and mechanisms used in the system architecture. Describe the reasoning employed for each decision and/or strategy (possibly referring to previously stated design goals and principles) and how any design goals or priorities were balanced or traded-off.*

*Examples of design decisions might concern (but are not limited to) things like the following:*

*• Use of a particular type of product (programming language, database, library, commercial off-the-shelf (COTS) product, etc.)*

*• Reuse of existing software components to implement various parts/features of the system*

*• Future plans for extending or enhancing the software*

*• User interface paradigms (or system input and output models)*

*• Hardware and/or software interface paradigms*

*• Error detection and recovery*

*• Memory management policies*

*• External databases and/or data storage management and persistence*

*• Distributed data or control over a network*

*• Generalized approaches to control*

*• Concurrency and synchronization*

*• Communication mechanisms*

*• Management of other resources*

>

 **Lựa chọn ngôn ngữ lập trình và công nghệ phát triển**

* **Quyết định**: Hệ thống sử dụng C# cho phần backend và **ReactJS** cho phần frontend.
* **Lý do**: C# được chọn vì khả năng mở rộng, tính bảo mật cao, và cộng đồng phát triển mạnh mẽ. ReactJS được lựa chọn cho frontend vì tính năng động, khả năng tái sử dụng thành phần, và khả năng tương tác mượt mà với API RESTful.
* **Cân nhắc mục tiêu thiết kế**: Quyết định này đảm bảo tính linh hoạt trong việc phát triển và duy trì hệ thống, đồng thời tối ưu hóa hiệu suất và bảo mật.

 **Sử dụng cơ sở dữ liệu quan hệ (Relational Database)**

* **Quyết định**: Sử dụng **MySQL** làm cơ sở dữ liệu chính của hệ thống.
* **Lý do**: MySQL phù hợp với nhu cầu lưu trữ dữ liệu có cấu trúc và hỗ trợ các tính năng cần thiết như giao dịch, khóa ngoại và chỉ mục để tối ưu hóa truy vấn.
* **Cân nhắc mục tiêu thiết kế**: MySQL đảm bảo hiệu suất cao khi xử lý các truy vấn phức tạp và dữ liệu lớn, đồng thời dễ dàng mở rộng khi cần.

 **Tái sử dụng các thành phần phần mềm hiện có**

* **Quyết định**: Sử dụng các thư viện mã nguồn mở và các API đã có sẵn để giảm thiểu công sức phát triển từ đầu.
* **Lý do**: Việc tái sử dụng các thành phần phần mềm (như thư viện xử lý JSON, xác thực bảo mật) giúp giảm thiểu thời gian phát triển, tăng tính ổn định và dễ dàng bảo trì.
* **Cân nhắc mục tiêu thiết kế**: Tái sử dụng giúp hệ thống giảm chi phí và rủi ro, đồng thời tăng khả năng tương thích và bảo mật.

 **Hướng tới khả năng mở rộng hệ thống**

* **Quyết định**: Thiết kế hệ thống theo mô hình **microservices** trong tương lai, với các dịch vụ độc lập có thể mở rộng riêng biệt.
* **Lý do**: Kiến trúc microservices cho phép hệ thống dễ dàng mở rộng và thay đổi trong tương lai mà không làm ảnh hưởng đến các phần còn lại của hệ thống. Mỗi dịch vụ có thể phát triển độc lập và sử dụng công nghệ riêng nếu cần.
* **Cân nhắc mục tiêu thiết kế**: Đảm bảo khả năng mở rộng và duy trì lâu dài, đặc biệt khi lượng người dùng và dữ liệu tăng cao.

 **Xử lý lỗi và khôi phục hệ thống**

* **Quyết định**: Hệ thống sử dụng các cơ chế phát hiện và khôi phục lỗi tự động, bao gồm logging lỗi chi tiết và thông báo cho người dùng.
* **Lý do**: Các cơ chế phát hiện lỗi giúp phát hiện và xử lý các vấn đề nhanh chóng, giảm thiểu thời gian ngừng hoạt động của hệ thống. Việc ghi nhật ký chi tiết giúp việc khôi phục và tìm nguyên nhân lỗi dễ dàng hơn.
* **Cân nhắc mục tiêu thiết kế**: Đảm bảo tính ổn định và độ tin cậy cao, đồng thời cung cấp thông tin đầy đủ để người quản trị có thể xử lý sự cố hiệu quả.

 **Quản lý bộ nhớ và tài nguyên**

* **Quyết định**: Hệ thống sẽ sử dụng chiến lược quản lý bộ nhớ động với các cơ chế giải phóng tài nguyên không sử dụng.
* **Lý do**: Điều này giúp tránh tình trạng thiếu bộ nhớ và giảm thiểu việc sử dụng tài nguyên không cần thiết, đảm bảo hệ thống hoạt động ổn định và hiệu quả.
* **Cân nhắc mục tiêu thiết kế**: Tối ưu hóa tài nguyên hệ thống để đảm bảo hiệu suất và khả năng phục hồi trong trường hợp tài nguyên bị giới hạn.

 **Giao tiếp và đồng bộ hóa**

* **Quyết định**: Hệ thống sử dụng **RESTful API** cho giao tiếp giữa các thành phần và dịch vụ, đồng thời áp dụng các cơ chế **asynchronous processing** cho các tác vụ dài hạn.
* **Lý do**: RESTful API giúp giao tiếp giữa các phần của hệ thống dễ dàng và hiệu quả, trong khi cơ chế xử lý bất đồng bộ giúp hệ thống không bị chặn bởi các tác vụ lâu dài như tải tệp hoặc xử lý báo cáo.
* **Cân nhắc mục tiêu thiết kế**: Đảm bảo hệ thống hoạt động mượt mà và không bị gián đoạn trong khi vẫn xử lý các tác vụ yêu cầu nhiều thời gian.

 **Quản lý dữ liệu phân tán**

* **Quyết định**: Hệ thống có khả năng phân tán dữ liệu qua các máy chủ khác nhau khi cần, sử dụng cơ chế sao lưu và đồng bộ hóa tự động.
* **Lý do**: Phân tán dữ liệu giúp tăng tính khả dụng và giảm thiểu rủi ro mất mát dữ liệu. Đồng thời, việc sao lưu và đồng bộ hóa dữ liệu tự động giúp hệ thống dễ dàng phục hồi trong trường hợp sự cố.
* **Cân nhắc mục tiêu thiết kế**: Tăng tính khả dụng và bảo vệ dữ liệu trong trường hợp hệ thống gặp sự cố hoặc có yêu cầu cao về tải.

## Design and Program Evaluation

*<Evaluate your design and describe which levels of coupling and cohesion that your design is at. Give proofs for your assumptions. Explain if there is any special design or exceptions>*

*<You may show the previous design from which you made improvements to get better levels of coupling and cohesion. You should clarify how and why you did these improvements>*

*<Does your design follow the SOLID principles if there are new requirements/changing requirements in the future? Give proofs for your assumptions. Explain if there is any special design or exceptions>*

*<You may show the previous design from which you made improvements to get a better design, which follows SOLID principles in spite of additional requirements. You should clarify how and why you did these improvements>*

### Cohesion & SRP

Explain in detail for each class in the table.

Table 4. Cohesion & SRP of AIMS

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *#* | *Class Name* | *PIC* | *Cohesion* | *SRP* | *Solution* |
| 1 | Order | TuấnNH | Temporal: m1(), m2() | Yes:  2 responsibilities: r1, r2 | Separate m2() to the class XXX… |
| 2 | <class name> | <person in charge> | <which level, why, where> | <Yes/No, why, where> | <solution if necessary> |

### Coupling & Other SOLID

Explain in detail for each problem in the table. Each problem, you may provide a class diagram before and after the improvement.

Table 5. Coupling & other SOLID of AIMS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Problem* | *PIC* | *Location* | *Solution* |
| 1 | Control coupling | TuấnNH | Class A and class B | Create a new class C as a superclass for A and B… |
| 2 | <Any bad coupling level or any SOLID violation> | <person in charge> | <list of class or you may provide a class diagram if necessary> |  |

## Design Patterns

*<Do you use any design patterns for your design? If yes, describe detailly why you use those design patterns? Describe in detail on the solutions and how to implement each design pattern>*