HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

School of Information and communications technology

Software Design Document

Version 1.3

AIMS: An Internet Media Store

Subject: ITSS Software Development

Group 2

Dang Van Nhan – 20225990

Nguyen Lan Nhi – 20225991

Tran Cao Phong – 20226061

Ha Viet Khanh – 20225979

Ho Bao Thu - 20226003

*Hanoi,* *May 2025*

*<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**

**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*>

## 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.*

>

## 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>*

## 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>*

### 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*

*>*

### 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>*

## 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 . 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 . Example of attribute design

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

Table . 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)*>

## 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*

>

## 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 . 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 . Coupling & other SOLID of AIMS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Problem* | *PIC* | *Location* | *Solution* |
| 1 | Common Coupling | Đặng Văn Nhân | ProductInfo wrapping Product | Remove ProductInfo if it only wraps Product; merge into a single cohesive entity or enrich ProductInfo with meaningful data |
| 2 | Content Coupling | Đặng Văn Nhân | CreateProductController, Product | Encapsulate validation logic inside Product subclasses or validators instead of exposing all internals |
| 3 | Stamp Coupling | Đặng Văn Nhân | BookValidator, CDValidator, DVDValidator, LPValidator, Product | Instead of downcasting Product, use polymorphism or visitor pattern to invoke product-type-specific validations |
| 4 | Control Coupling | Đặng Văn Nhân | CreateProductController, ProductValidatorFactory | Create a new interface and inject it via constructor to decouple control logic; Apply strategy pattern for validation dispatch |
| 5 | Stamp Coupling Passing the Cart object into calculateShippingFee(), but only using some fields | Nguyễn Lan Nhi | Class "PlaceRushOrderController" → Class "Cart" | Refactor calculateShippingFee() to accept only the needed data as parameters. |
| 5 | Stamp Coupling  Passing the DeliveryInfo object into calculateShippingFee() but only using some fields | Nguyễn Lan Nhi | Class "PlaceRushOrderController" → Class "DeliveryInfo" | Extract required fields before calling method; or create ShippingAddressDTO with only needed fields. |
| 5 | Stamp Coupling sendApprovedEmail(email, Order) passes the full Order object, though it likely only requires partial data. | Nguyễn Lan Nhi | Class "OrderManagementController" → Class "Order" | Pass only essential fields to sendApprovedEmail(). |
| 5 | Control Coupling Logic in approveOrder() branches depending on Order.Status | Nguyễn Lan Nhi | Class "OrderManagementController" | Create subclasses like ApprovedOrder, RejectedOrder, PendingOrder that extend a base Order class or implement an OrderState interface. |
| 6 | Stamp Coupling DeliveryInfo is passed as a whole, but only .getProvince() is used. | Nguyễn Lan Nhi | Class "RushOrderService" | Extract getProvince() before calling isAddressInHanoiInnerCity(), or pass province directly. |
| 10 | Content Coupling ViewProductDetailsService handles validation, retrieval, and transformation in one place. | Trần Cao Phong | ViewProductDetailsService | Refactor by extracting validation to ProductValidator, data access to ProductRepository, and transformation to a mapper or factory. Service now only coordinates logic. |
| 11 | Stamp Coupling  Passes full Product into ProductInfo constructor, but only uses some fields. | Trần Cao Phong | ViewProductDetailsService → ProductInfo | Refactor ProductInfo to accept only required fields or use a static method to map needed data. Avoid unnecessary object dependency. |
| 12 | Control + Content Coupling CreateUserController handles validation, persistence, email, and logging | Hà Việt Khánh | CreateUserController | Refactor by extracting validation, email, persistence, and logging into separate services. Controller now only coordinates the process. |
| 13 | Control Coupling Passing role string to controller/service to decide flow (e.g., "admin", "productManager") instead of separating behaviors | Hà Việt Khánh | CreateUserController → validation & logic branching | Split into specialized services or use polymorphism (e.g., separate creation flows for each role). |
| 14 | Stamp Coupling Cart is passed to ProductAvailabilityService.checkCartAvailability(), but only product name and quantity are used. | Hồ Bảo Thư | PlaceOrderController → Cart | Replace Cart with a DTO containing only necessary info (productId, quantity). |
| 15 | Stamp Coupling Order is passed to DeliveryFormScreen.promptUserInput(), but only orderId is used. | Hồ Bảo Thư | PlaceOrderController → DeliveryFormScreen | Pass only orderId or use lightweight DTO like OrderSummary. |
| 16 | Stamp Coupling Order passed to OrderService.attachDeliveryAndInvoice() but only orderId is accessed internally. | Hồ Bảo Thư | PlaceOrderController → OrderService | Refactor to pass orderId only if no other order data is used. |
| 17 | Stamp Coupling DeliveryInformation passed to ShippingFeeCalculator.calculate() but only partial data may be used. | Hồ Bảo Thư | PlaceOrderController → ShippingFeeCalculator | Consider redesigning method to accept only required fields. |
| 18 | Stamp Coupling Order passed to OrderService.saveTransaction(), but only orderId is needed. | Hồ Bảo Thư | PlaceOrderController → OrderService | Replace with just orderId if possible. |

## 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>*