HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

School of Information and communications technology

Software Design Document

Version 1.3

<Project Name>

Subject: <Name of subject>

<Group 23>

<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 14

**List of Figures**

**No table of figures entries found.**

**List of Tables**

No table of figures entries found.

# Introduction

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

## Objective

The purpose of this Software Design Document (SDD) is to provide a comprehensive design framework for the AIMS desktop e-commerce platform. The intended audience includes software developers, project stakeholders, and instructors responsible for assessing the project’s technical quality and completeness. This document aims to outline the system’s design to facilitate implementation, ensure maintainability, and achieve the goals outlined in the project’s requirements.

## Scope

AIMS (An Internet Media Store) is a desktop e-commerce platform designed to provide users with a seamless shopping experience for physical media products, including books, CDs, DVDs, and LP records. The platform supports functionalities such as product management, cart operations, order placement, and payment processing.

The key features of AIMS include:

1. Efficient management of product inventory by administrators and product managers.
2. A user-friendly interface for customers to browse, search, and purchase media items.
3. Support for advanced functionalities such as rush order delivery within inner-city Hanoi and integration with VNPay for payment processing.
4. Robust operational parameters ensuring high performance and reliability, including the ability to serve up to 1,000 simultaneous users and operate continuously for 300 hours without failure.

The application is limited to desktop operations and does not currently support digital product sales. Future enhancements may include expanding payment options and supporting new product categories.

## Glossary

* **AIMS**: An Internet Media Store.
* **VNPay**: A payment gateway used for processing transactions.
* **Rush Order**: A delivery option allowing expedited delivery within two hours in certain locations.
* **Customer**: An end-user who interacts with the platform to browse and purchase products.
* **Cart**: A virtual container used by customers to select products for purchase.
* **Invoice**: A document summarizing the details of a purchase, including items, pricing, and total amount.

## References

|  |  |
| --- | --- |
| [1] | Centers for Medicare & Medicaid Services, "System Design Document Template," [Online]. Available: 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

*AIMS is a robust e-commerce platform for physical media products, combining an intuitive user interface with efficient backend management capabilities. The system’s architecture is designed to support scalability, performance, and reliability. The key design approach is modular, allowing the seamless addition of new features and subsystems while maintaining system integrity.*

*Key components include:*

* ***User Interface****: Designed with JavaFX to ensure responsiveness and ease of navigation.*
* ***Database Management****: Utilizes SQLite for lightweight and efficient local data storage.*
* ***Subsystem Design****: Incorporates clearly defined modules for cart, payment, and product management.*
* ***Performance Metrics****: Ensures response times of under 2 seconds under normal conditions and up to 5 seconds during peak hours.*

## Assumptions/Constraints/Risks

### Assumptions

* *Users will access the software through a desktop interface.*
* *The system’s performance and reliability metrics are based on average user behavior and concurrent usage scenarios*

### 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/Software Environment****: The application is designed for desktop environments and requires a minimum of 4GB RAM and 2GHz dual-core processor to run efficiently. It is not compatible with mobile devices or web browsers.*
* ***End-User Environment****: Limited to users with desktop systems. No support for mobile or remote multi-device synchronization.*
* ***Availability of Resources****: SQLite is used for database management. The application cannot handle extensive write operations simultaneously and does not support distributed databases.*
* ***Standards Compliance****: The application adheres to SQLite standards but does not implement advanced database features like sharding or replication.*
* ***Interoperability Requirements****: The system integrates only with VNPay for payment processing, limiting flexibility with alternative payment gateways.*
* ***Licensing Requirements****: The use of SQLite requires compliance with its open-source license terms. Additionally, VNPay's integration is subject to its API terms.*
* ***Data Repository and Distribution****: The database is stored locally on the user's machine, restricting data distribution and centralized management.*
* ***Security Requirements****: Data encryption is limited to payment data during VNPay transactions. Local database files lack encryption, posing risks in case of system compromise.*
* ***Memory/Capacity Limitations****: SQLite's limitations affect handling large datasets and multi-threaded operations, which might lead to performance degradation under heavy loads.*
* ***Performance Requirements****: The system must ensure response times of 2 seconds under normal conditions but may degrade during peak usage.*
* ***Network Communications****: VNPay integration requires a stable internet connection. Lack of connection disrupts payment processing.*
* ***Verification and Validation****: Unit and integration testing are conducted locally, limiting the scope for testing in distributed environments.*
* ***Other Quality Goals****: The design prioritizes simplicity and local deployment, which may impact extensibility for larger-scale applications.*
* ***Other Requirements****: Rush order delivery is restricted to certain locations (inner-city Hanoi). Delivery fees and timelines are subject to local logistical constraints.*

*>*

### Risks

* *Potential downtime or delays during payment processing via VNPay.*
* *Risk of exceeding SQLite’s capacity under unexpected high usage.*
* *Dependence on accurate input from administrators and users for effective operations.*

# 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 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 1. Example of attribute design

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

Table 1. 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 / 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*

>

## Coupling and Cohesion

*<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>*

## Design Principles

*<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>*

## 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>*