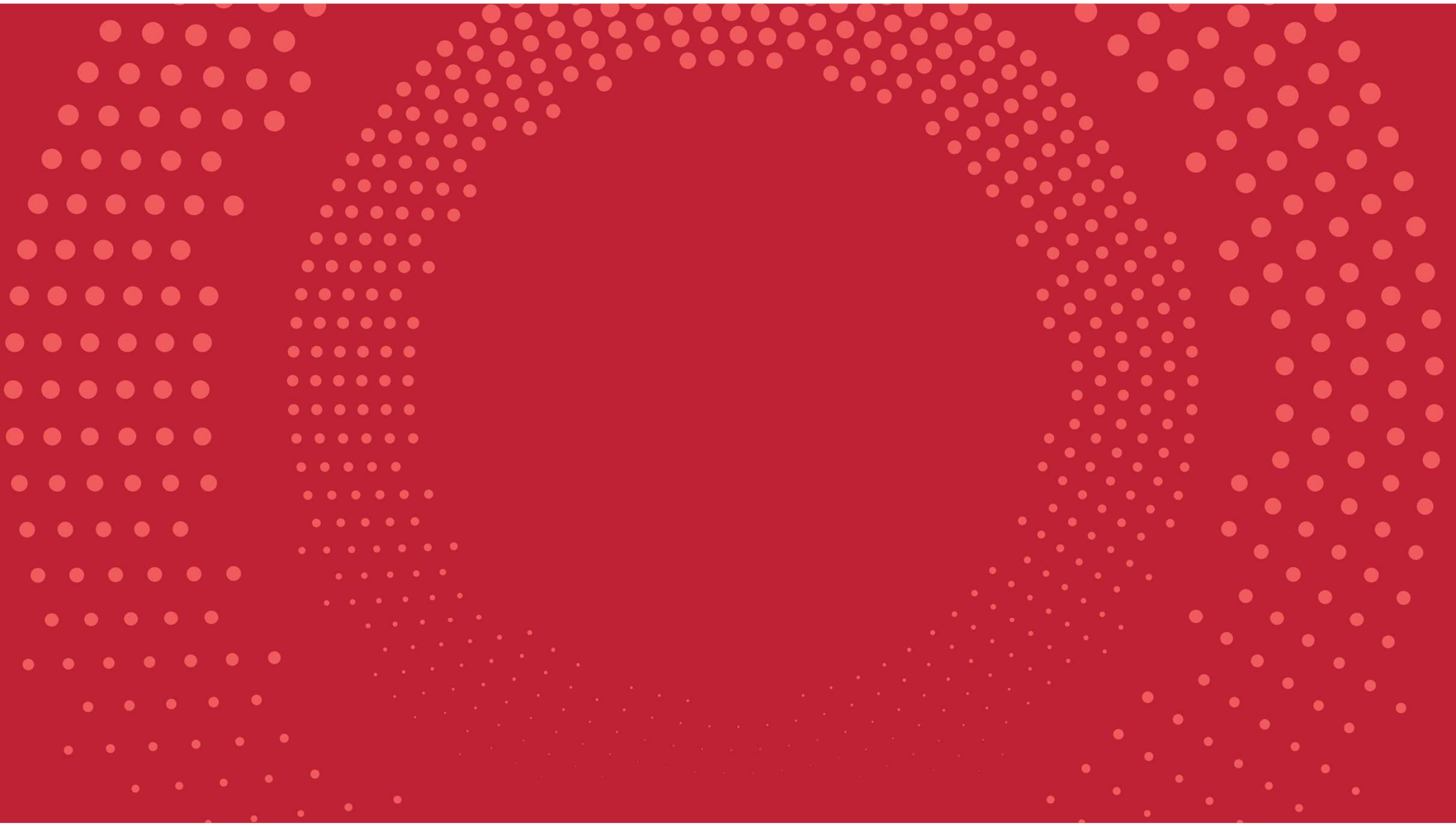
HUST

ĐẠI HỌC BÁCH KHOA HÀ NỘI

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

ONE LOVE. ONE FUTURE.



ĐẠI HỌC
BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY
OF SCIENCE AND TECHNOLOGY

AIMS: An Internet Media Store

IT4549E - ITSS Software Development
Instructor: Ph.D. Nguyen Thi Thu Trang

ONE LOVE. ONE FUTURE.

INTRODUCTION

1. Scope

The AIMS Project is a desktop e-commerce software designed to facilitate the selling of media products such as books, LP records, CDs, and DVDs.

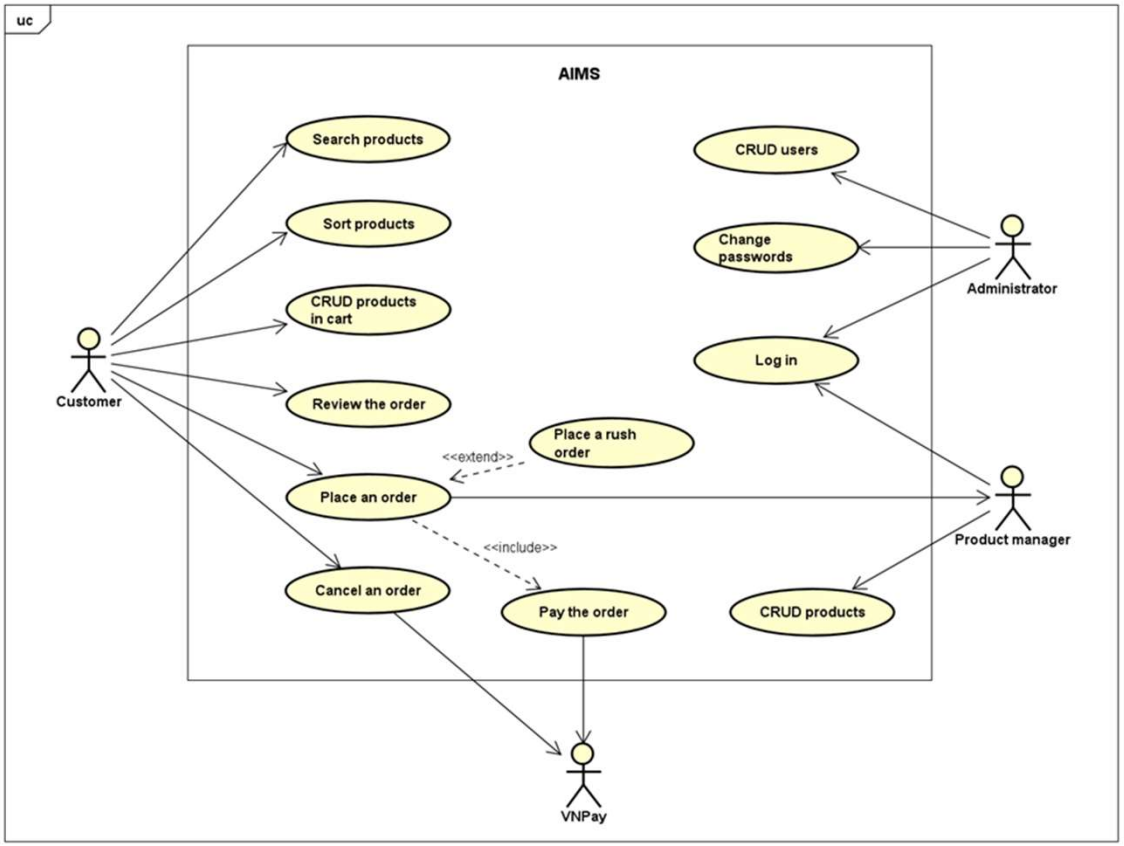
Guests: Search and purchase necessary items

Administrators: manage users

Product managers: manage and make changes to product and order

OVERALL DESCRIPTION

1. General Overview



OVERALL DESCRIPTION

2. Assumptions/Constraints/Risks

a. Assumptions

- Stable Hardware Environment

Utilizes specified operating systems

- User Proficiency

End-users have basic technical skills

- Functionality Changes

Future changes will be minimal and manageable

- Software Integration

Assumes compatibility and availability of related software components

OVERALL DESCRIPTION

2. Assumptions/Constraints/Risks

b. Assumptions

- Hardware Limitations
- User Environment
- Resource Availability
- Data Repository

c. Risks

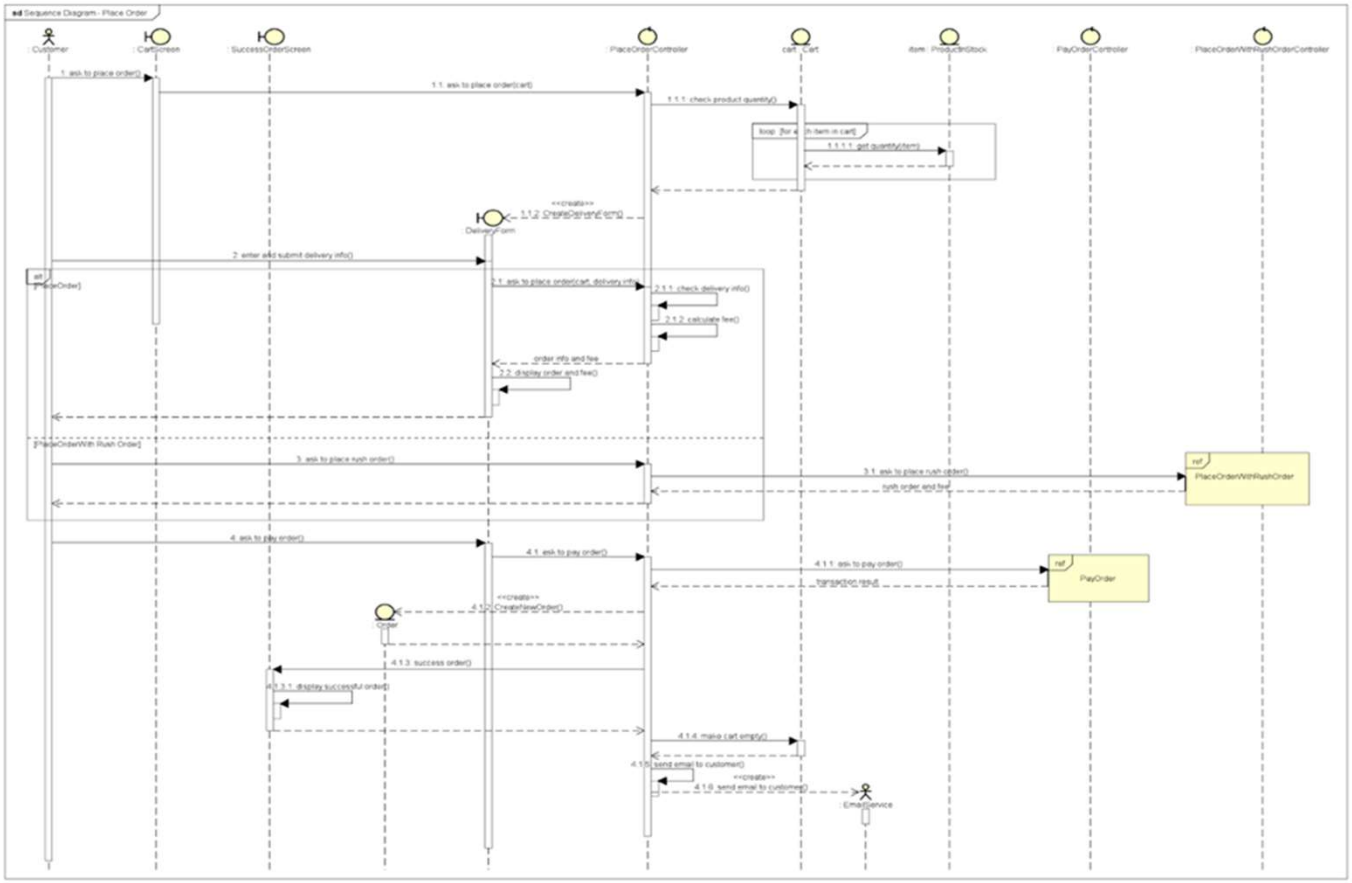
- Security Risks: Mitigated by encryption and authentication
- Resource Limitations: Addressed through efficient coding and optimization

1. Architectural Patterns
   - **Pattern Overview:**
     - **Entities**: Core business logic and data, reusable and independent. Examples: Account, Customer, Transaction.
     - **Controls**: Coordinate interactions between entities and boundaries, manage use cases. Example: PlaceOrder control.
     - **Boundaries**: Interfaces for external interactions (UI components, APIs).
   - **Advantages:**
     - **Separation of Concerns**: Modular and easier to understand.
     - **Maintainability**: Isolated changes reduce risk.
     - **Scalability**: Add features without altering existing entities.
     - **Testability**: Independent testing of business logic and interfaces.

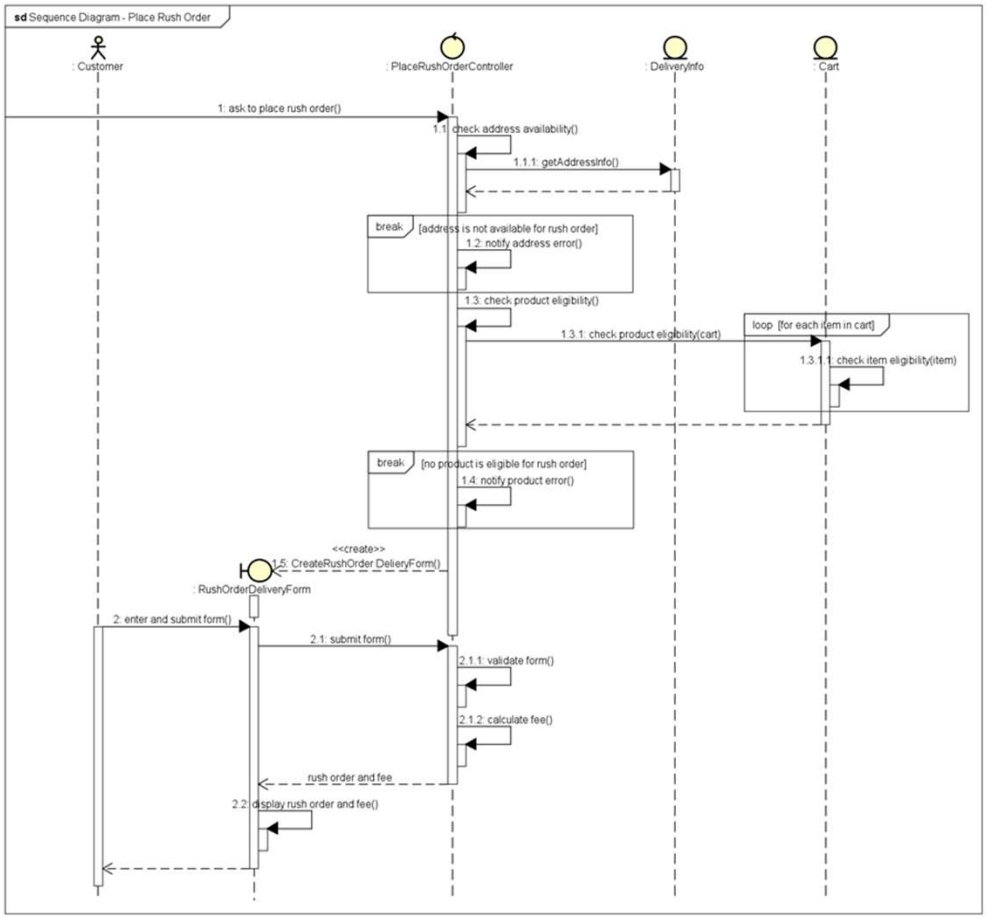
SYSTEM ARCHITECTURE & ARCHITECTURE DESIGN

2. Interaction Diagram:

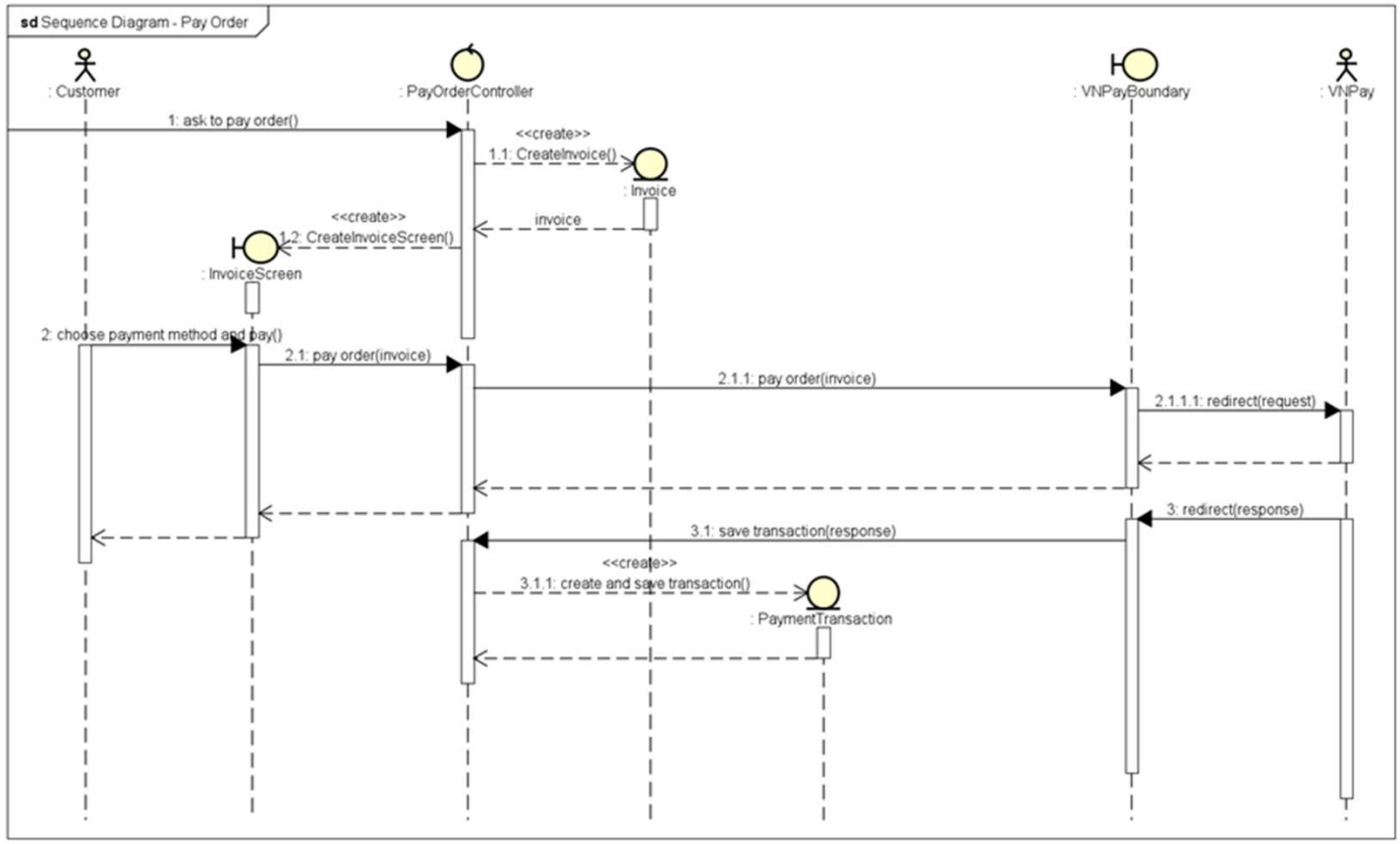
a) Place order



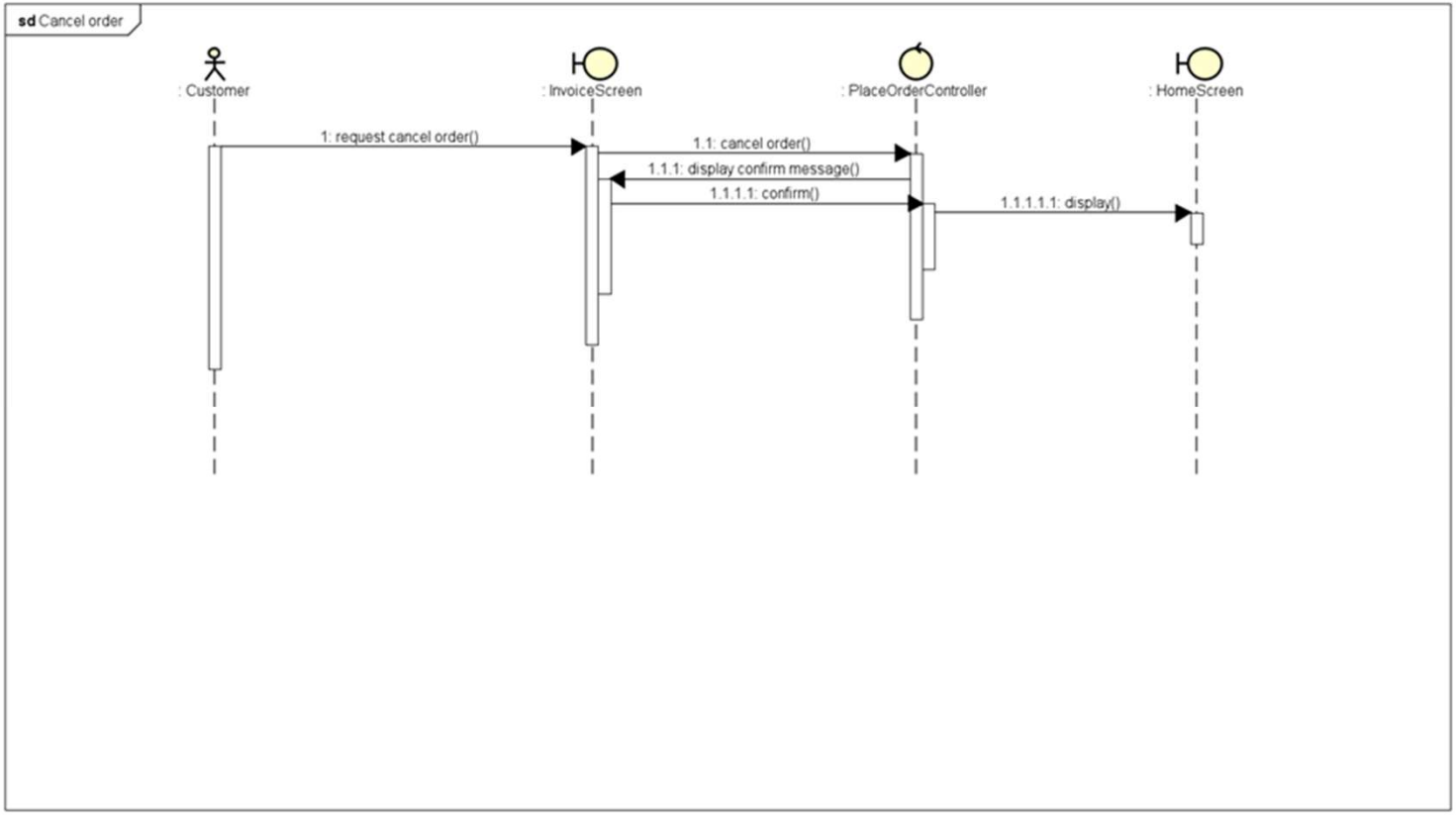
b) Place rush order



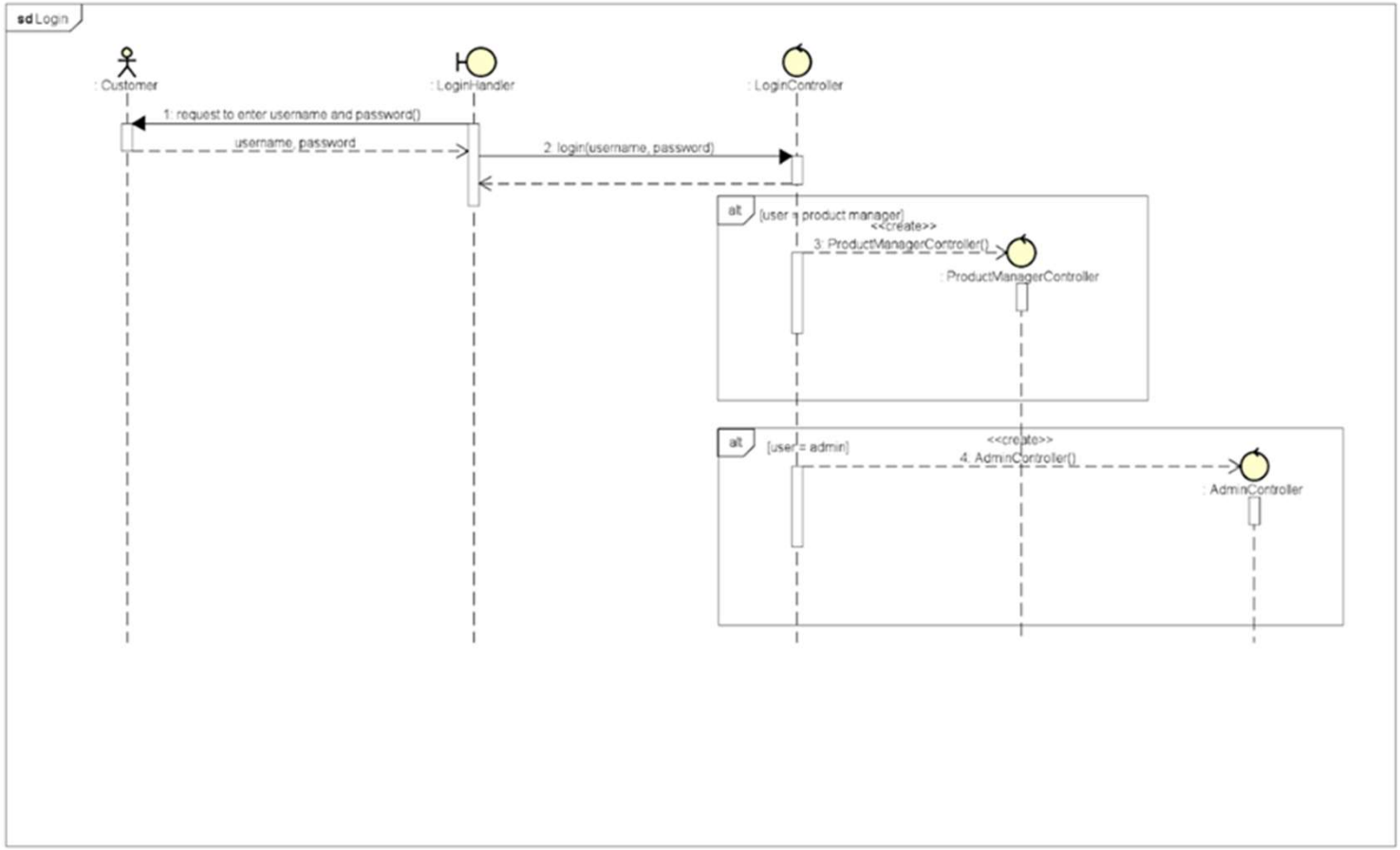
c) Pay order


sd Sequence Diagram - Pay Order
: Customer
: PayOrderController
: VNPayBoundary
: VNPay
1: ask to pay order()
<<create>>
1.1: CreateInvoice()
: Invoice
invoice
<<create>>
1.2: CreateInvoiceScreen()
: InvoiceScreen
2: choose payment method and pay()
2.1: pay order(invoice)
2.1.1: pay order(invoice)
2.1.1.1: redirect(request)
3: redirect(response)
3.1: save transaction(response)
<<create>>
3.1.1: create and save transaction()
: PaymentTransaction

d) Cancel order

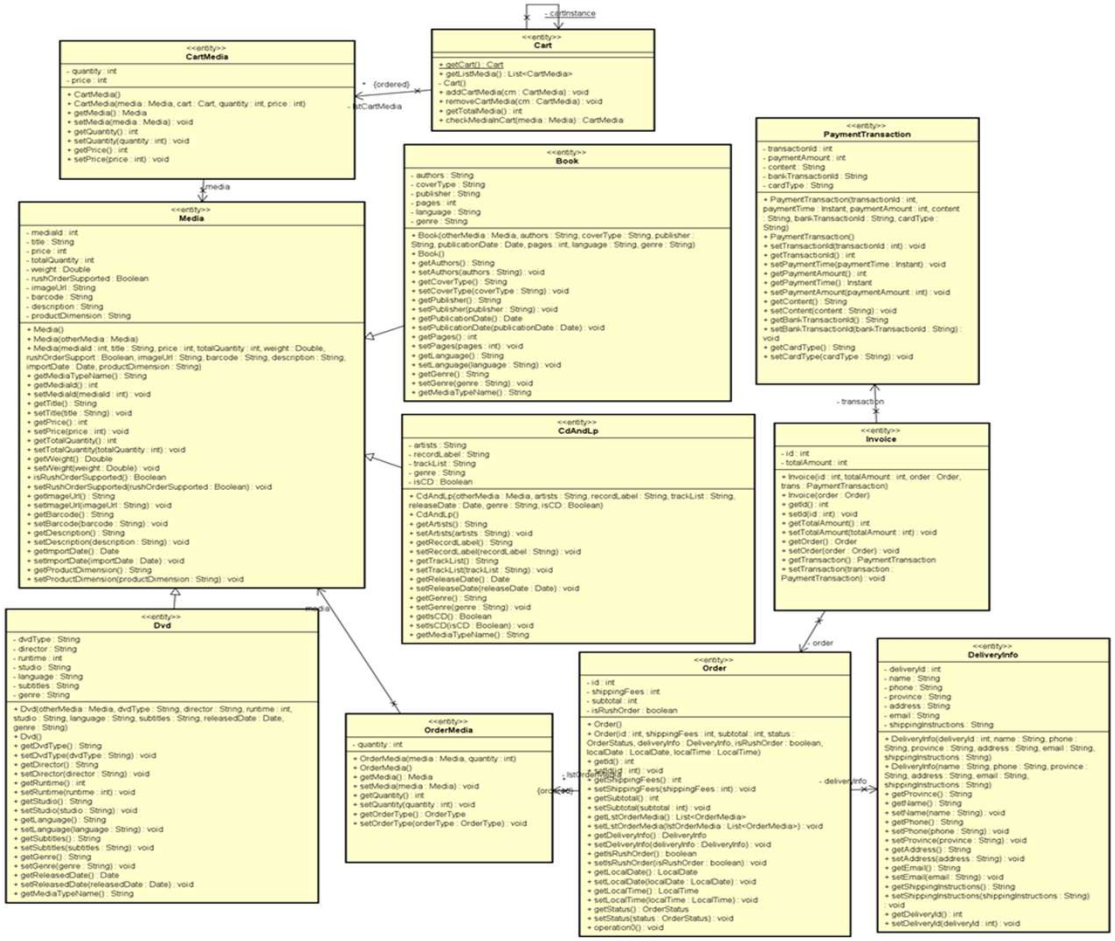


e) Login

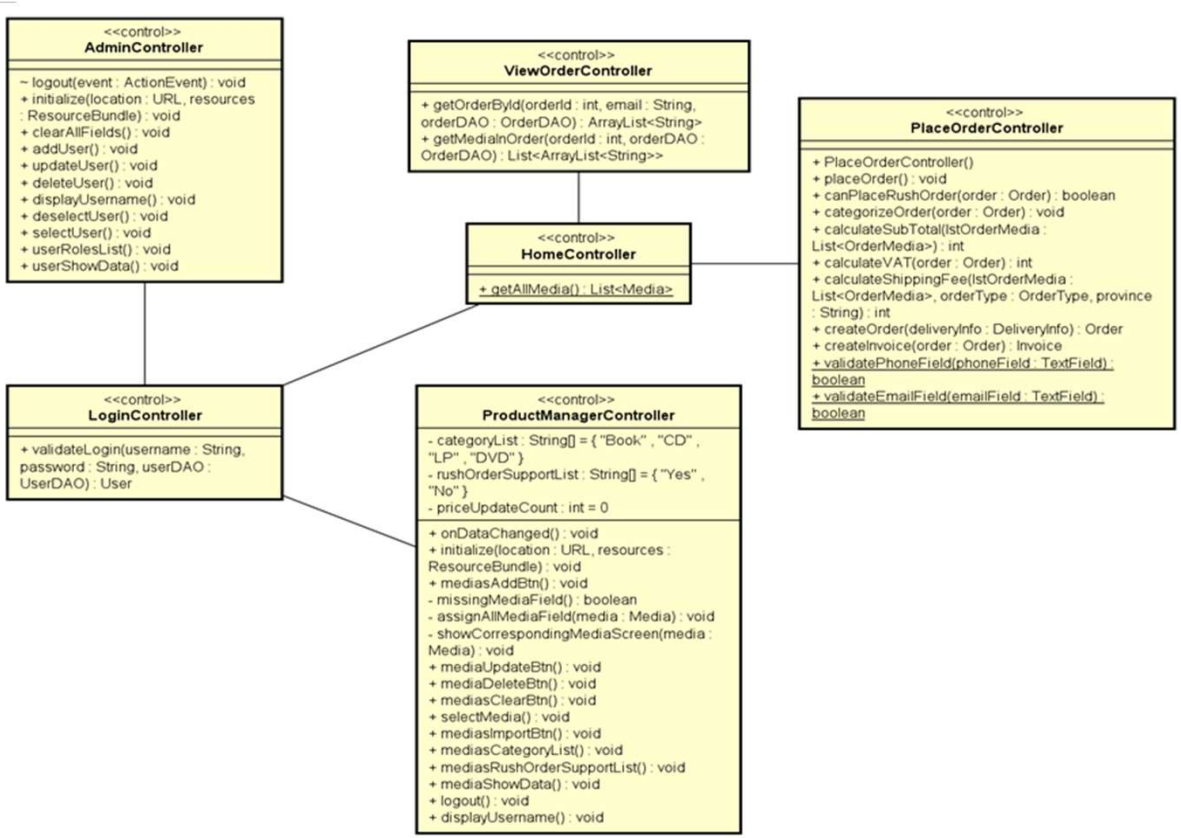

sd Login
: Customer
: LoginHandler
: LoginController
1: request to enter username and password()
username, password
2: login(username, password)
alt
[user = product manager]
<<create>>
3: ProductManagerController()
: ProductManagerController
alt
[user = admin]
<<create>>
4: AdminController()
: AdminController

3. Analysis Class Diagram

a) Package Entity

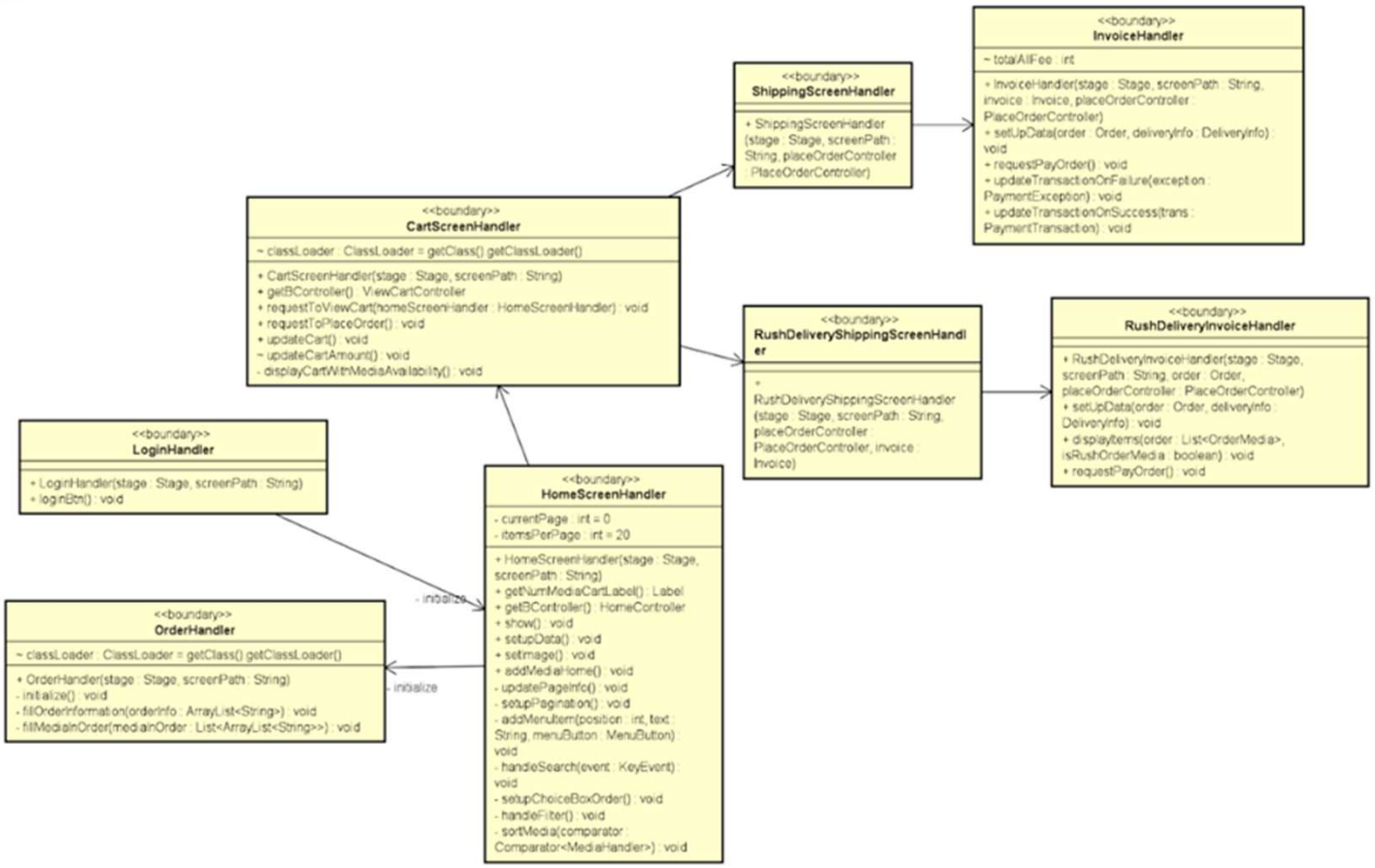


b) Package Controller

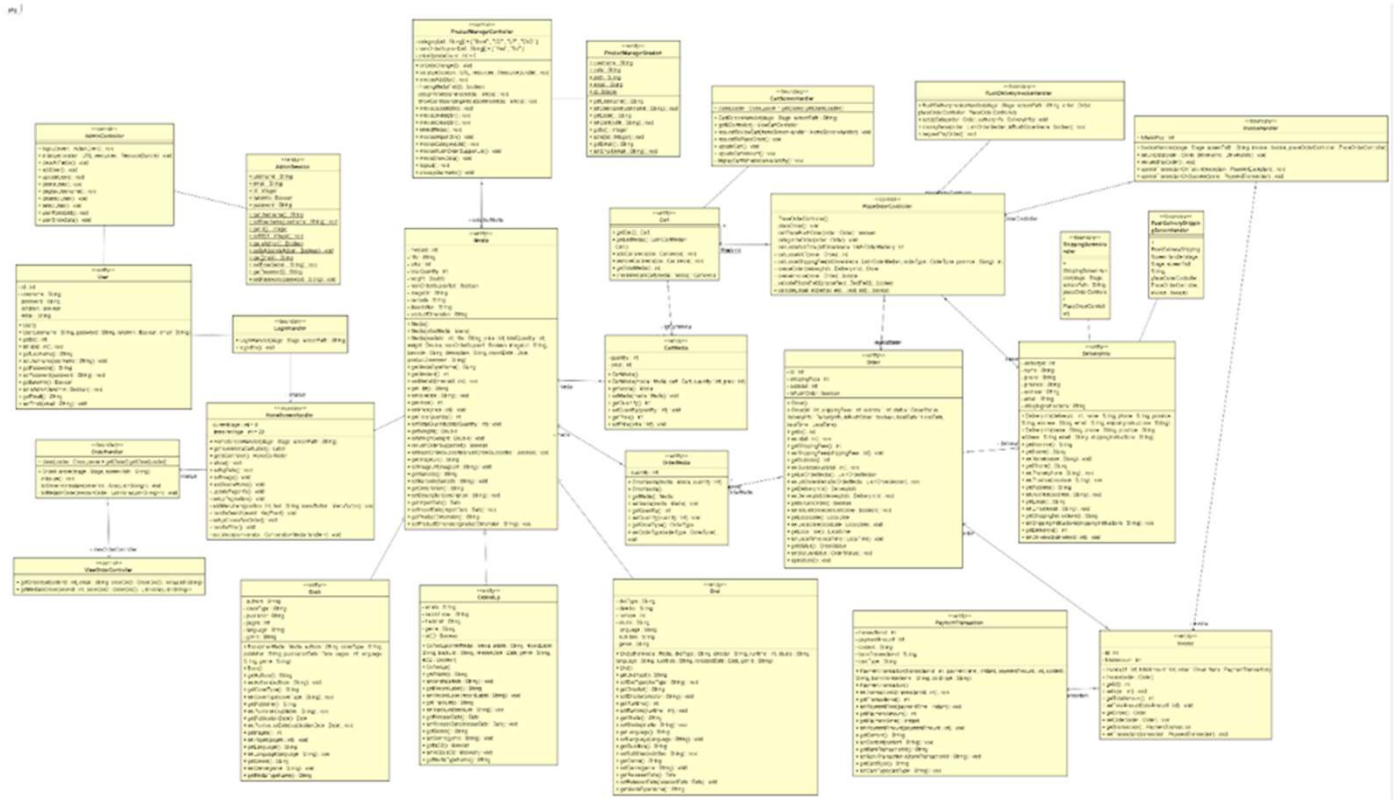


SYSTEM ARCHITECTURE & ARCHITECTURE DESIGN

c) Package Boundary



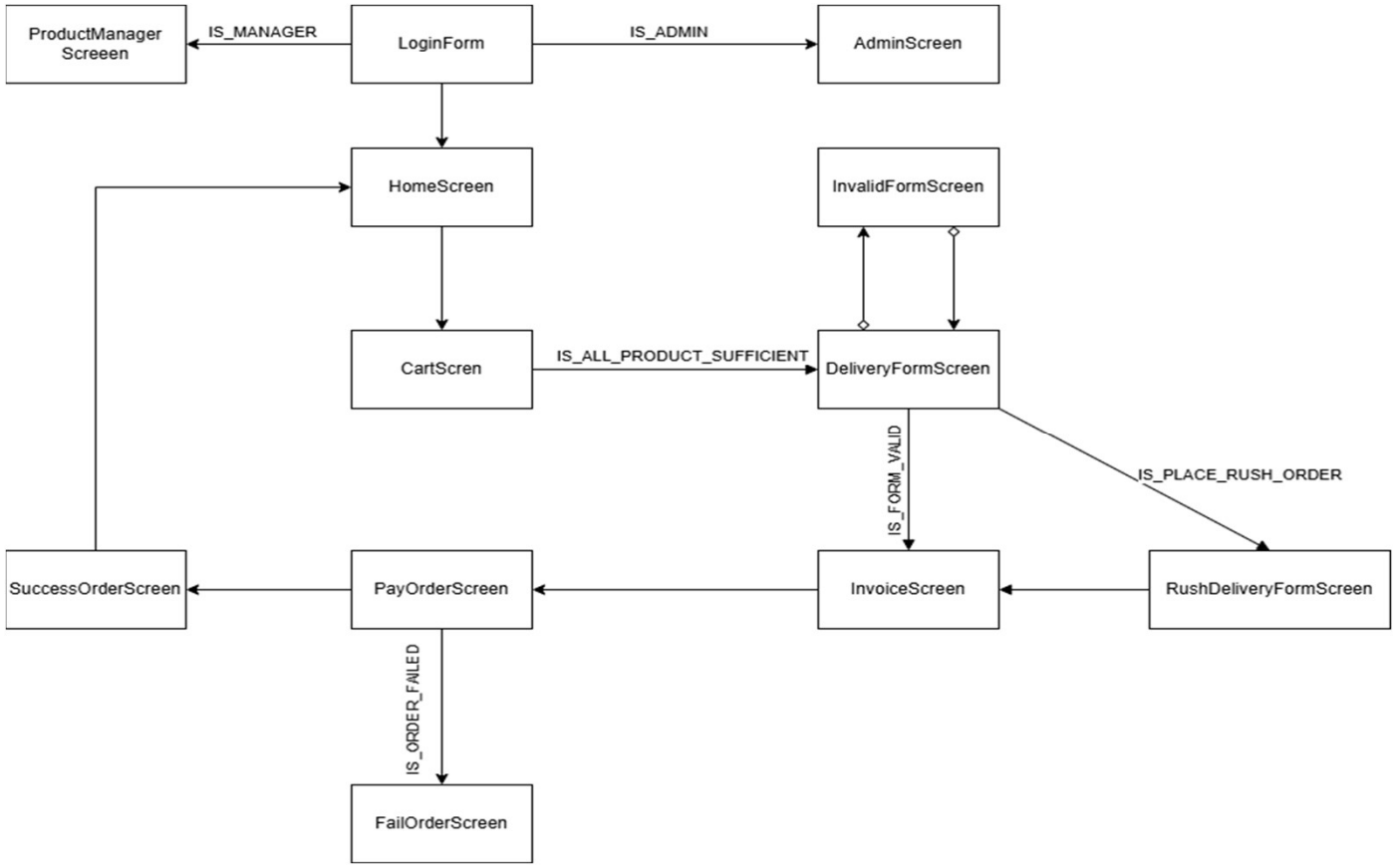
4. Unified Analysis Class Diagram



DETAILED DESIGN

1. User interface design
   - There are screens in our AIMS software: Cart Screen, Delivery Forms, Login Form, Admin Dashboard, Product Manager Dashboard, …
   - Display: resolution 1366x768.
   - Consistency of expressions: comma-separated for numbers, a limited range of valid characters for strings ([0-9],. _-)
   - Control: input format checking functionality included; no stack frames, each screen is separated.
   - Error: error message with its details displayed while encountering such one.

DETAILED DESIGN: SCREEN TRANSITION DIAGRAM

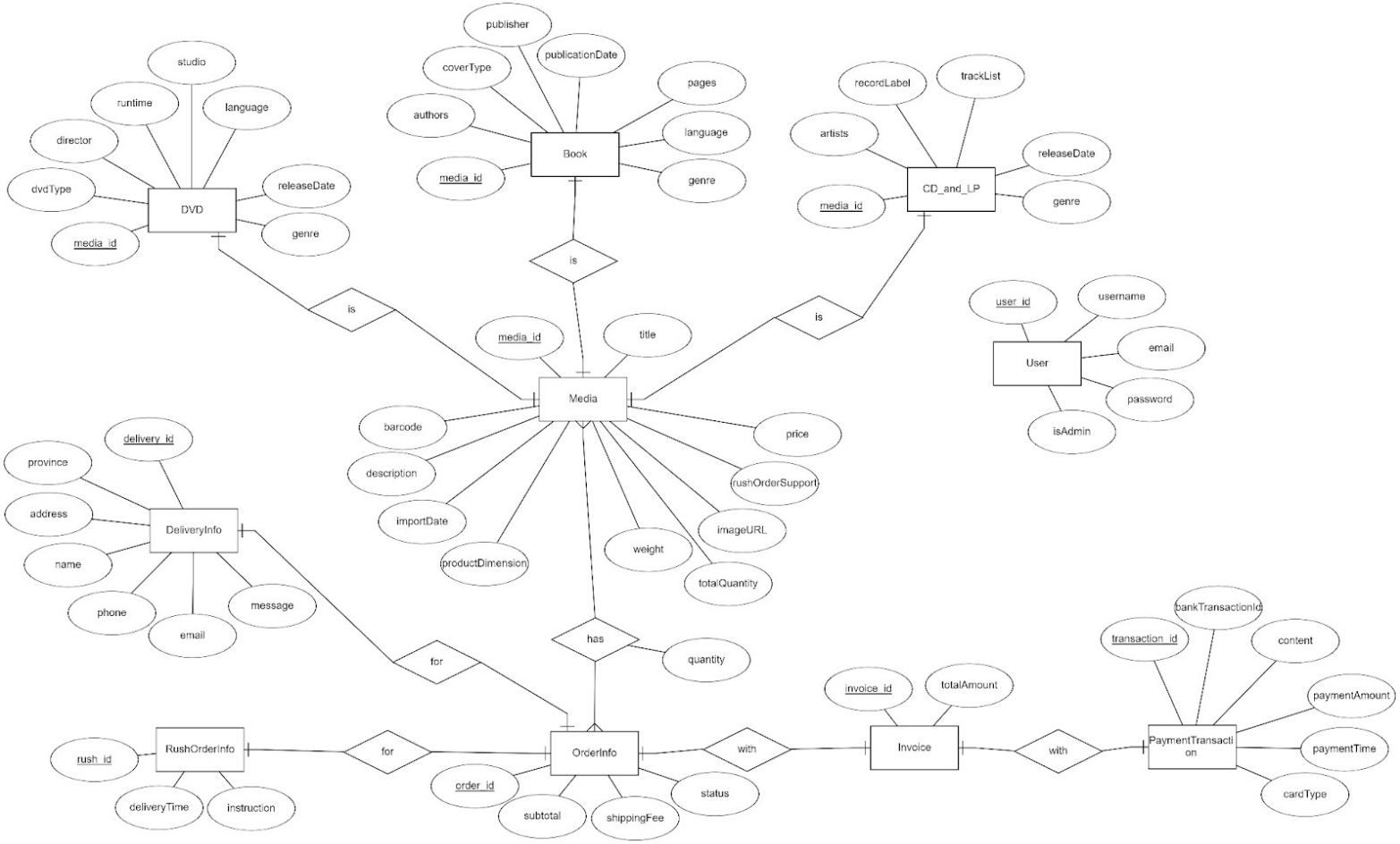


DETAILED DESIGN

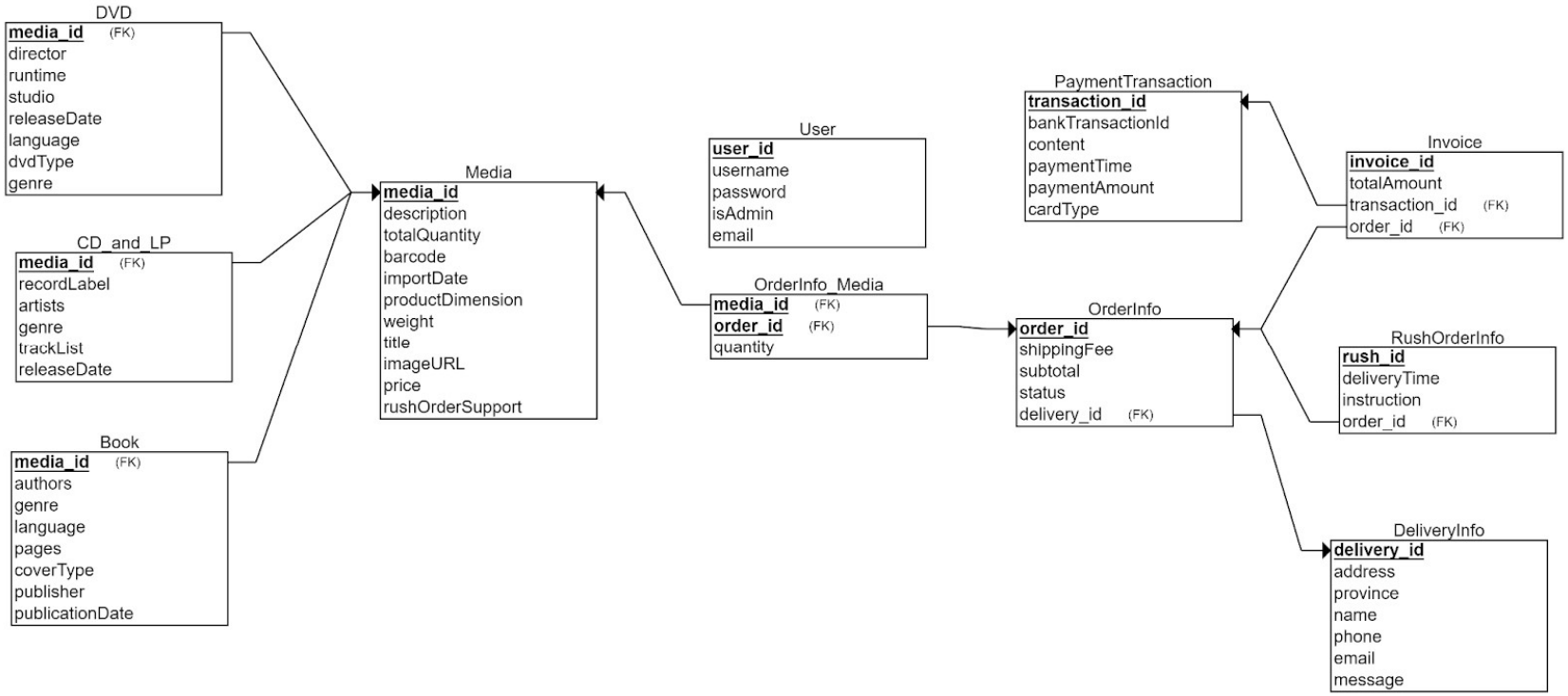
2. Data Modelling

- Relational database management system: MySQL
- Perks: High reliability, scalable, robust, secured.
- Index: index created to support rapid search on database.

DETAILED DESIGN: ENTITY RELATIONSHIP DIAGRAM



DETAILED DESIGN: DATABASE SCHEMA



DETAILED DESIGN

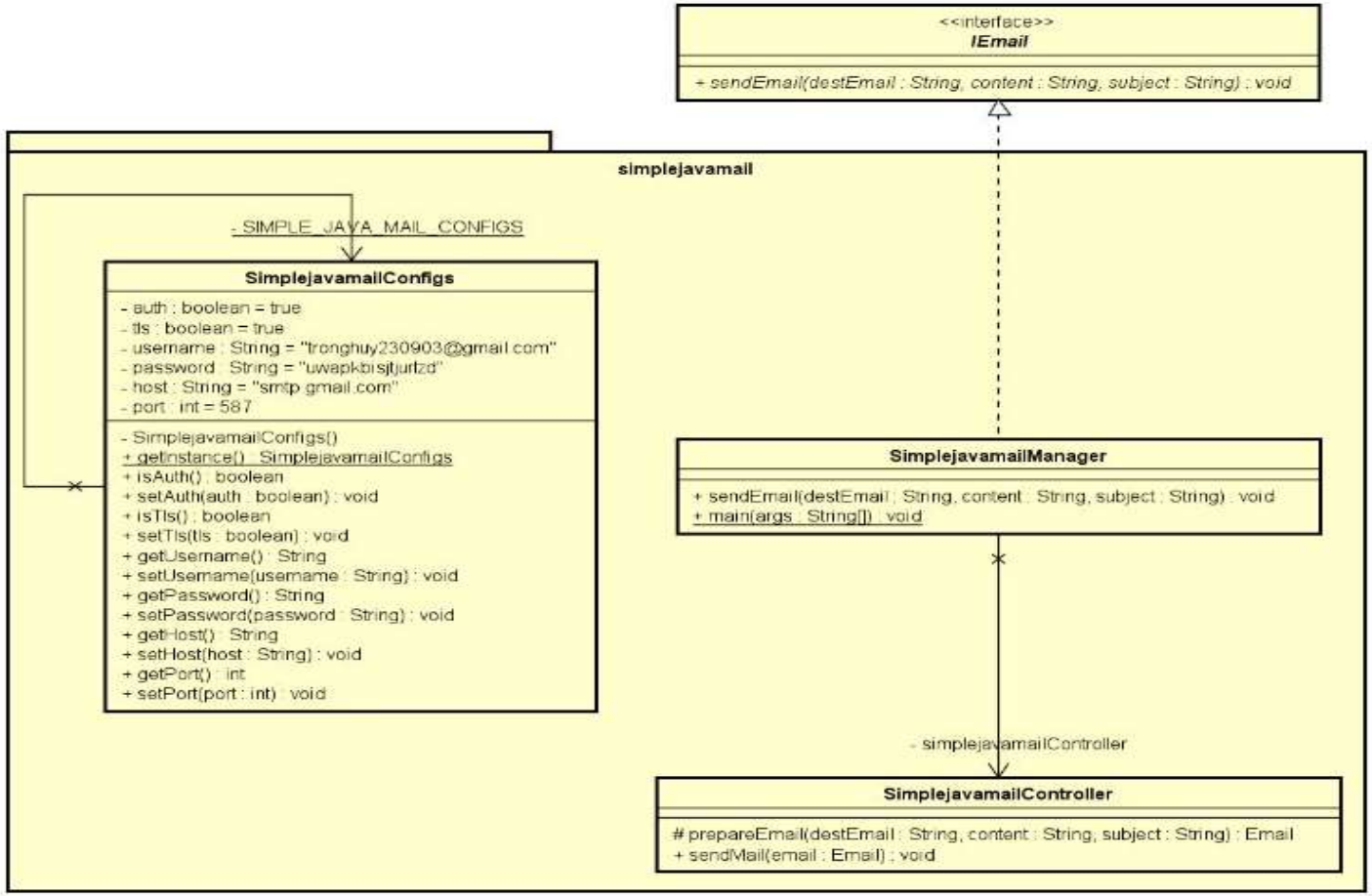
3. Class design: there are 7 main packages:
   - Persistence: 2 sub-packages: DAO(data accessible object) and database, which is responsible for storing, retrieving data from and executing operations on database.
   - Entity: this package includes necessary entities which interact with the system.
   - View: includes all screen handlers.
   - Controller: all controllers are packaged here (PlaceOrderController, ...)
   - Exception: packages all exceptions which can be encountered during execution.
   - Utils: extended utility classes which support detailed development (error message displayed, screen configuration paths,...)
   - Subsystem: 2 sub-packages email, payment, which represent email subsystem and VNPay payment subsystem respectively.

DETAILED DESIGN

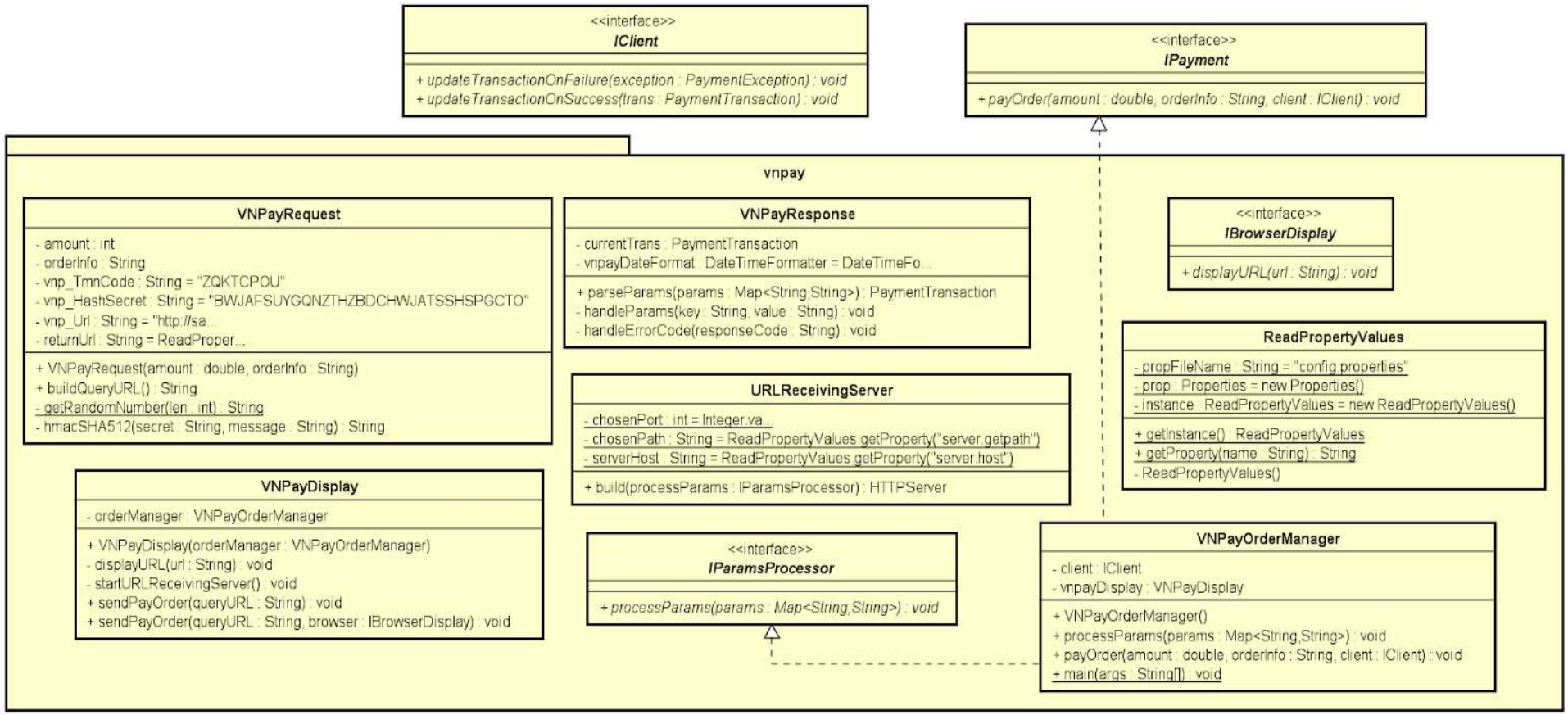
4. Class design for subsystems

- Email: This subsystem is responsible for e-mail transmissions during the execution of the system.
- Payment: Payment subsystem injects VNPay payment gateway to streamline checkout process.

DETAILED DESIGN: CLASS DIAGRAM FOR EMAIL SUBSYSTEM



DETAILED DESIGN: CLASS DIAGRAM FOR PAYMENT SUBSYSTEM



DESIGN CONSIDERATIONS

1. Goals and guidelines

a) Goals

- **Performance Optimization**: Fast, responsive interactions with efficient queries, caching, and minimized data processing overhead.
- **Scalability**: Modular design, scalable technologies (e.g., MySQL), and expandable cloud infrastructure.
- **Security and Privacy**: Strong encryption, robust authentication, and comprehensive logging to protect user data.
- **User-Friendly Interface**: Intuitive, consistent UI for easy navigation and improved user satisfaction.

DESIGN CONSIDERATIONS

1. Goals and guidelines

b) Guidelines

- **Coding Standards**: Adherence to guidelines for quality and readability, including naming conventions, code formatting, and documentation.
- **Modular Design**: Discrete, self-contained modules for easier management, testing, and updates.
- **Error Handling**: Robust mechanisms for graceful recovery, diagnostic logging, and user-friendly error messages.

DESIGN CONSIDERATIONS

2. Architectural strategies
   - Programming Language and Frameworks: JavaFX(GUI), Maven.
   - Database Management System: MySQL.
   - UI paradigms: follows Rich Client Application paradigm using JavaFX.
   - Error handling: try-catch blocks are added to ensure internal error does not cause the whole program to collapse.
   - Testing: JUnit.
   - Version control: GitHub's "Feature/Release" workflow.

DESIGN CONSIDERATIONS

3. Coupling and Cohesion

a) Coupling

- **Modular Architecture**: Distinct modules for UI (JavaFX), backend (Java), and data persistence (MySQL) minimize interdependencies.
- **Interfaces and Abstraction**: Components interact through defined interfaces and abstract classes, reducing direct dependencies (e.g., DAOs using Hibernate).

DESIGN CONSIDERATIONS

3. Coupling and Cohesion

b) Cohesion

- **Single Responsibility Principle**: Classes/methods focus on a single task (e.g., UI controllers handle interactions, service classes handle business logic).
- **Well-defined Modules**: Clear purpose for each module (data handling, business logic, UI management) ensures self-contained, cohesive components.
- **Consistent Design Patterns**: Use of patterns like Entity-Boundary-Controller (EBC) promotes high cohesion.

DESIGN CONSIDERATIONS

4. Design patterns

a) Singleton design pattern

- Ensures a class has a single instance and provides a global access point to that instance.
- Useful for coordinating system-wide actions (logging, db connections, etc..).
- Reduces memory overhead, avoid conflicts from multiple instances.
- Promotes consistent access.
- Simplifies debugging and maintenance by centralizing control logic.

DESIGN CONSIDERATIONS

4. Design patterns

a) Singleton design pattern

- Ensures a class has a single instance and provides a global access point to that instance.
- Useful for coordinating system-wide actions (logging, db connections, etc..).
- Reduces memory overhead, avoid conflicts from multiple instances.
- Promotes consistent access.
- Simplifies debugging and maintenance by centralizing control logic.

DESIGN CONSIDERATIONS

4. Design patterns

a) Singleton design pattern

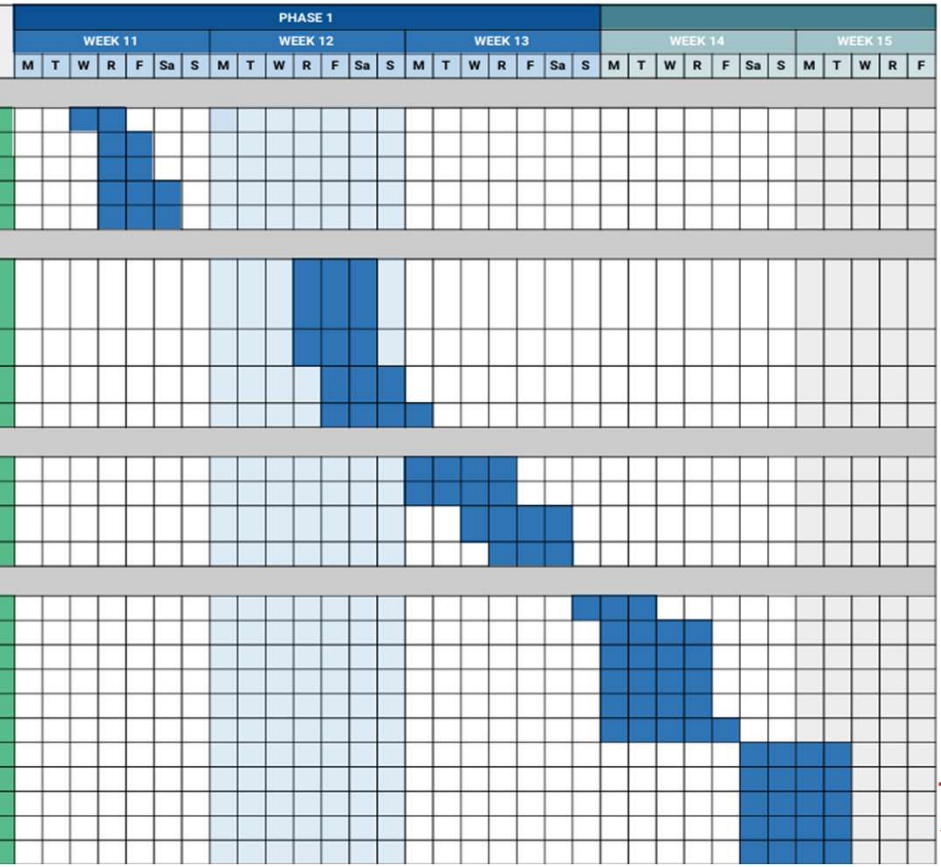
- Ensures a class has a single instance and provides a global access point to that instance.
- Useful for coordinating system-wide actions (logging, db connections, etc..).
- Reduces memory overhead, avoid conflicts from multiple instances.
- Promotes consistent access.
- Simplifies debugging and maintenance by centralizing control logic.

SPRINT BACKLOG & TEAM CONTRIBUTION

SPRINT BACKLOG

PROJECT TITLE	AIMS	COMPANY NAME	ISD.ICT.20232-10
PRODUCT OWNER	Nguyen Trong Huy	DATE	5/22/24
SCRUM MASTER	Nguyen Trong Huy		

Sprint	TASK TITLE	TASK OWNER	LEVEL OF PRIORITY	START DATE (MM/DD/YY)	DUE DATE (MM/DD/YY)	DURATION	PCT OF COMPLETE
WEEK 11							
1	Create and set up database and connection	Huy NT	HIGH	5/22/24	5/23/24	2	100
2	Implement the home screen and product view	Huy DN	HIGH	5/24/24	5/25/24	2	100
3	Implement the viewing cart	Huu	HIGH	5/24/24	5/25/24	2	100
4	Implement for VNPay interface and Pay Order	Hoang	HIGH	5/24/24	5/26/24	3	100
5	Implement Place Order, Place Rush Order function	Hung	HIGH	5/24/24	5/26/24	3	100
WEEK 12							
1	Draw the class diagram from code (just relation ship between classes, subsystem and packages not for attributes and methods) and write the report evaluating the cohesion and coupling of the program	Hoang	LOW	5/30/24	6/1/24	2	100
2	Writing the report for evaluating the good design for the whole program (SOLID)	Huy DN & Huu	LOW	5/30/24	6/1/24	2	100
3	Fix the code (or refractor the program) to meet the good design	Huy DN	MEDIUM	5/31/24	6/2/24	3	100
4	Finishing the Place Order and Place Rush Order	Hung & Huy NT	HIGH	5/31/24	6/3/24	4	100
WEEK 13							
1	Complete connect VNPay to PayOrder	Hoang	MEDIUM	6/3/24	6/6/24	3	100
2	Complete Rush Delivery Form and Phone Validation	Hung	MEDIUM	6/3/24	6/6/24	3	100
3	Implement dashboard screen for product manager and CRUD products, viewing order	Huu, Huy DN	HIGH	6/5/24	6/8/24	3	100
4	Implement Login function	Huy NT	MEDIUM	6/6/24	6/8/24	2	100
WEEK 14 & WEEK 15							
1	Issue redesign database for order	Hoang	HIGH	6/9/24	6/11/24	2	100
2	Product manager: update order status	Hoang	HIGH	6/10/24	6/13/24	3	100
3	Customer: review order by order id and email	Huu, Huy NT	HIGH	6/10/24	6/13/24	3	100
4	Email subsystem & integrate	Huy NT	HIGH	6/10/24	6/13/24	3	100
5	Admin dashboard	Hung, Huy NT	HIGH	6/10/24	6/13/24	3	100
6	Fix bug for image of media	Huy DN	MEDIUM	6/10/24	6/14/24	4	100
7	SRS documentation	Huu	LOW	6/15/24	6/18/24	3	100
8	Sequence diagram	Hung	LOW	6/15/24	6/18/24	3	100
9	ERD & DB Schema	Hoang	LOW	6/15/24	6/18/24	3	100
10	Design-level class diagram	Huy DN, Huy NT	LOW	6/15/24	6/18/24	3	100
11	Slides	Hung, Huu	MEDIUM	6/17/24	6/18/24	1	100



THANK YOU !