# Describe the Run-time Architecture

for

# Book Rental and Sales System

Version 1.6 approved

Prepared by Trinh Thai Linh
Le Quang Chinh
Pham Tung Duong
Do Tien Dung
Nguyen Hoang Anh Dung

Group 7

November 30, 2023

# Table of Contents

# Table of Figures

## Revision History

| Name | Date | Reason For Changes | Version |
|---|---|---|---|
| Trinh Thai Linh | 30-Nov-23 | Create Document Template. | 1.0 |
| Pham Tung Duong | 04-Dec-23 | Add *Buy Book, Cancel Order process models.*<br>Add changes to *Rental Plan Registration process model.*<br>Add process element descriptions. | 1.1 |
| Do Tien Dung | 04-Dec-23 | Add *Sign In, Sign Up, Edit Profile process models.*<br>Add process element descriptions. | 1.2 |
| Le Quang Chinh | 04-Dec-23 | Add *Forum Service, Chat box process models.*<br>Add process element descriptions. | 1.3 |
| Nguyen Hoang Anh Dung | 04-Dec-23 | Add *Rent Book, Review process models.*<br>Add process element descriptions. | 1.4 |
| Trinh Thai Linh | 04-Dec-23 | Add the remaining process models. Add process element descriptions. | 1.5 |
| Trinh Thai Linh | 04-Dec-23 | Finalize changes. | 1.5 |

# 1. Introduction

## 1.1 Purpose

This is a report prepared by Group 7 (Object-oriented Analysis and Design Class 20, 22-23) to Describe the Run-time Architecture Solution for Book Rental and Sales System and is written based on the reporting format "IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications".

## 1.2 Intended Audience and Reading Suggestions

The different types of reader that the document is intended for are:

- **Designers:** Design the classes and the subsystems architecture that satisfies the requirements specified in this SRS.
- **Developers:** Implement codes from the design and these documents. They use this document as a blueprint, and it makes sure that, firstly, their code works properly, and secondly, it is in sync with others' codes.
- **Tester:** Use this document to understand the requirements. They have to create a test plan that outlines the scope of testing, testing objectives, test environment setup, resources required, and a schedule.
- **Documentation writers:** Ensure that the documentation aligns with the software's requirements as specified in the Software Requirements Specification (SRS). Verify that the documented information is accurate and complete.

## 1.3 Product Scope

The software, "Book Rental and Sales System", is designed to provide a platform for users to rent, purchase, read books in both physical and electrical form. Additionally, it includes a forum where users can discuss with other readers about the books they purchased, rented or simply ones that captured their interest. The system aims to enhance the reading experience and promote the reading community engagements.

## 1.4 References

[1] Form of presentation IEEE. IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications. IEEE Computer Society, 1998.
[2] System Requirements Specification Content and Format Standard, which specifies the content and format of this specification.
[3] Previous Project of Object-Oriented Analysis and Design Course, The Cargo Carriage System.

# 2. Describe Partly

## 2.1 Sign Up

### *2.1.1 Process Model*

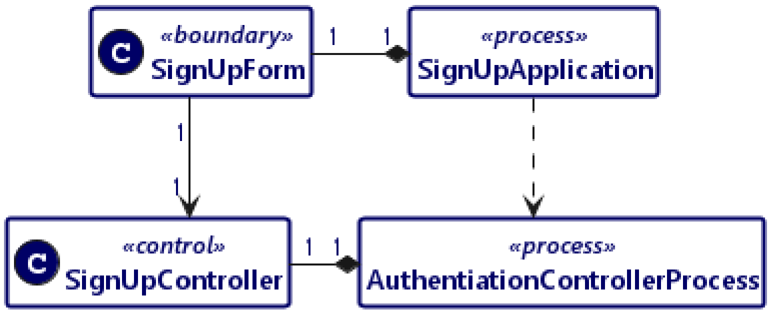


*Figure 2-1: Sign Up process model.*

### *2.1.2 Process Element Descriptions*

- **SignUpApplication:** Controls the user interfaces of the Sign Up application. Controls the family of forms that the users use to sign up. There is one instance of this process which is SignUpForm (a boundary class).
- **AuthenticationControllerProcess:** Manage the authentication processes. This process has an instance which is SignUpController.

## 2.2 Sign In and Reset Password

### *2.2.1 Process Model*

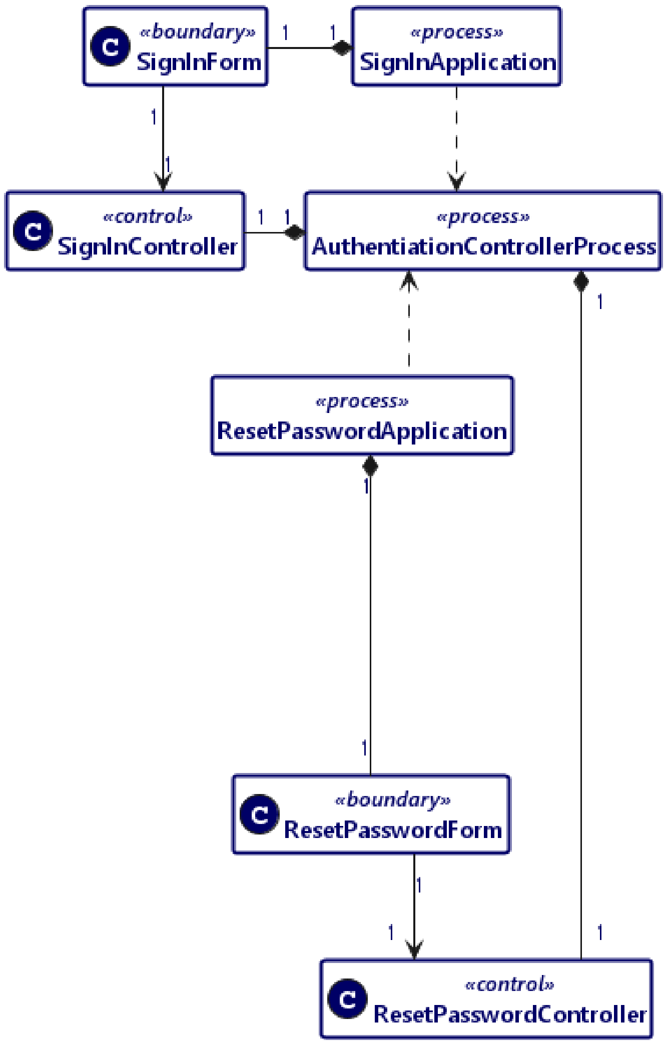


*Figure 2-2: Sign In process model.*

### *2.2.2 Process Element Descriptions*

- **SignInApplication:** Control the interfaces of the Sign In Application. Control the form that the user used to sign in. This process has only one instance which is the SignInForm (a boundary class).
- **ResetPasswordApplication:** Control the interface of the Reset Password Application. Control the form that the user used to reset their password. This process has only one instance which is the ResetPasswordForm (a boundary class).
- **AuthenticationControllerProcess:** Manage the authentication processes. This process has two instances which are ResetPasswordController and SignInController.

## 2.3 Edit Profile

### *2.3.1 Process Model*

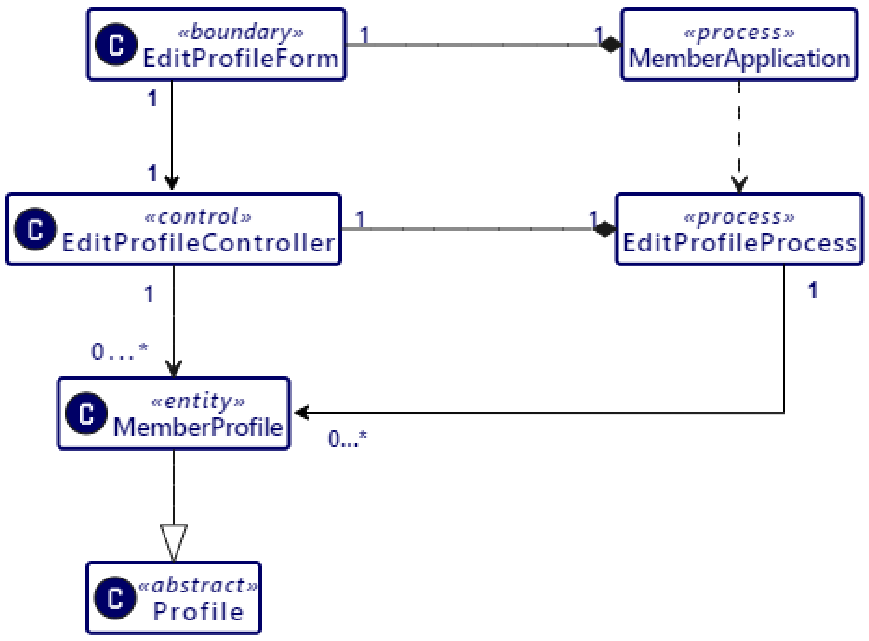

«boundary» EditProfileForm
«process» MemberApplication
«control» EditProfileController
«process» EditProfileProcess
«entity» MemberProfile
«abstract» Profile
1
1
1
1
1
1
1
0…*
1
0…*

*Figure 2-3: Member Edit Profile process model.*

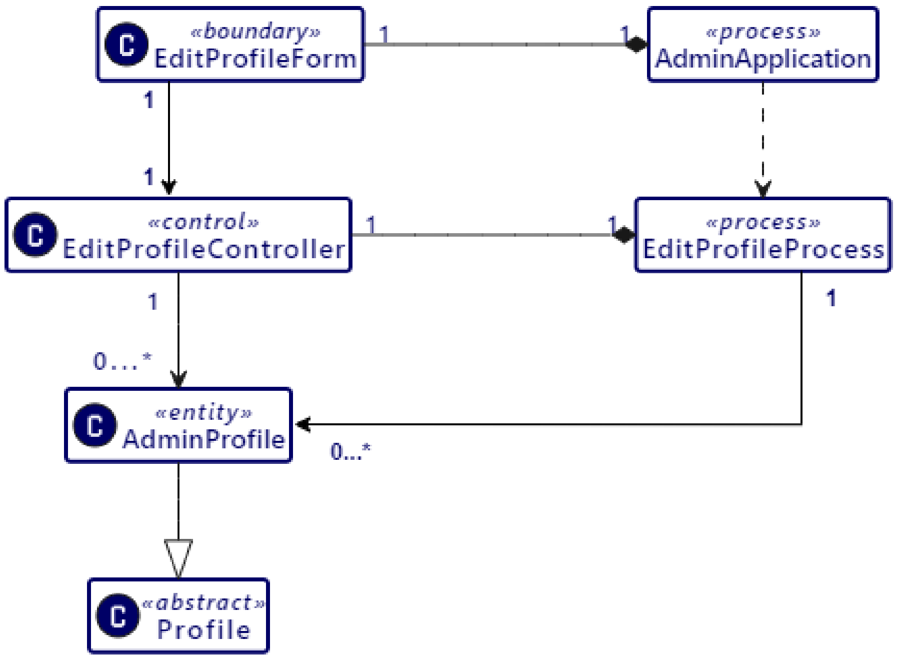

«boundary» EditProfileForm
«process» AdminApplication
«control» EditProfileController
«process» EditProfileProcess
«entity» AdminProfile
«abstract» Profile
1
1
1
1
1
1
1
0…*
1
0…*

*Figure 2-4: Admin Edit Profile process model.*

### *2.3.2 Process Element Descriptions*

- **MemberApplication:** Manages all user interfacer (UI), forms for Member, allows them to interacts with the System. In this process, MemberApplication has a EditProfileForm (a boundary class) allows Members to edit their profiles on the System.
- **EditProfileProcess:** Manage edit profile process on the System. This process has a instance of EditProfileController (a control class).
- **AdminApplication:** Manages all user interfacer (UI), forms for Administrators, allows them to interacts with the System. In this process, AdminApplication has a EditProfileForm (a boundary class) allows Admins to edit their profiles on the System.

## 2.4 Forum Services

### *2.4.1 Process Model*

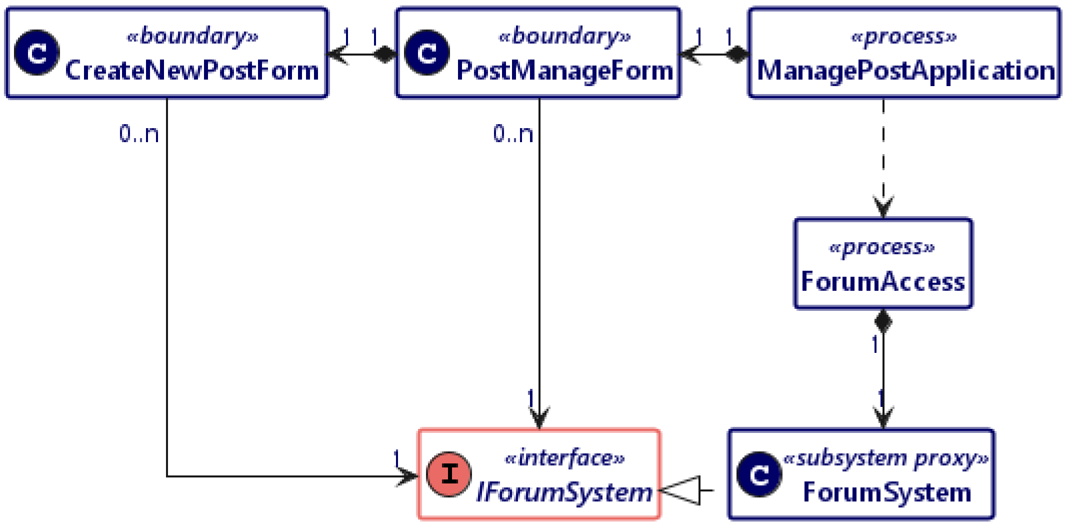


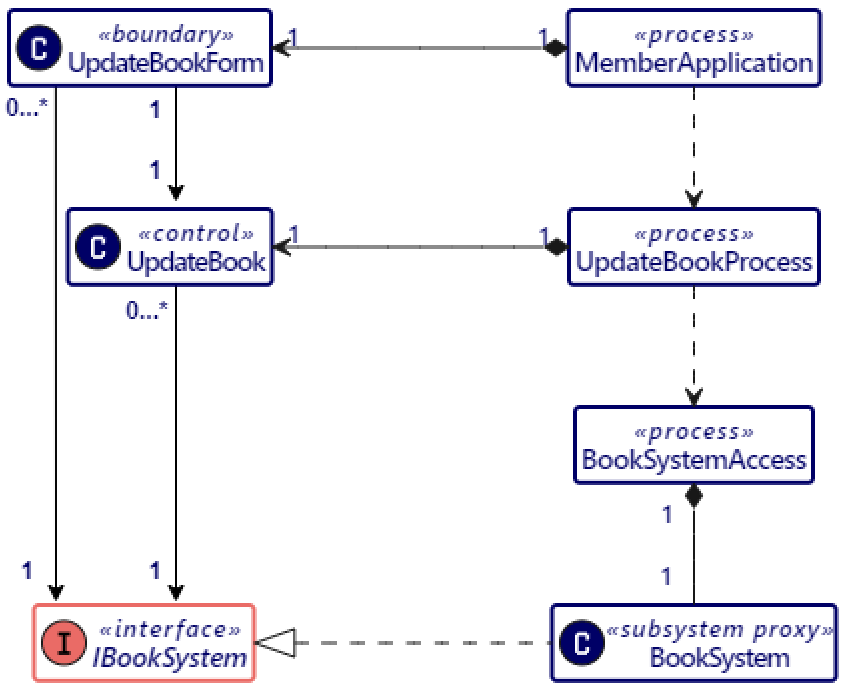
*Figure 2-5: Forum Serivce process model.*

### *2.4.2 Process Element Descriptions*

- **ManagePostApplication:** Manages the user interface (UI), forms for Member to create, edit, delete and close a post. In this process, there is an instance of PostManageForm (a boundary class) designed for closing, changing or deleting posts. The PostManageForm includes the CreateNewPostForm to create a new post. Every instances of this process indicate an action of a member to the ForumSystem.
- **ForumAccess:** Manages all accesses to the ForumSystem subsystem. Since this process is asynchronous, Member can interact with the system while waiting for responses from ForumSystem subsystem. This process also synchronizes accesses to the ForumSystem from the other system processes.

## 2.5 Update Book

### *2.5.1 Process Model*



«boundary» UpdateBookForm
«process» MemberApplication
«control» UpdateBook
«process» UpdateBookProcess
«process» BookSystemAccess
«interface» IBookSystem
«subsystem proxy» BookSystem
1
0...*

*Figure 2-6: Update Book process model.*

### *2.5.2 Process Element Descriptions*

- **MemberApplication:** Manages all user interfacer (UI), forms for Member, allows them to interacts with the System. In this process, MemberApplication has a UpdateBookForm (a boundary class) allows Members to add new books, edit book's informations or delete a book on the System.
- **UpdateBookProcess:** Manages add new book, edit book's information or delete books processes on the System. This process has a instance of UpdateBookController (a control class).
- **BookSystemAccess:** Manages all accesses to the BookSystem subsystem. Since this process is asynchronous, Member can interact with the system while waiting for responses from BookSystem subsystem. This process also synchronizes accesses to the BookSystem from the other system processes.

## 2.6 Search Book

### *2.6.1 Process Model*

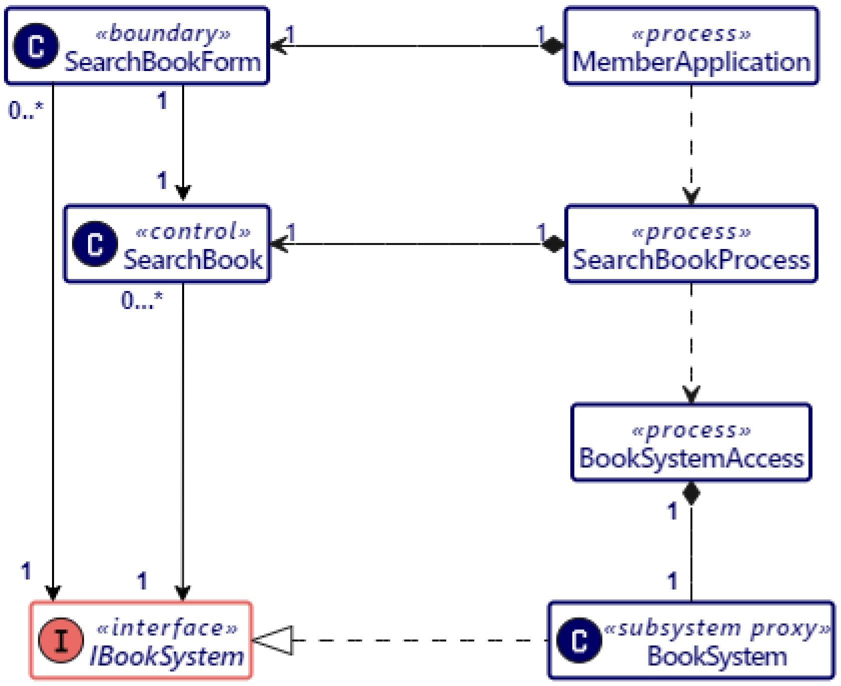

«boundary» SearchBookForm
«process» MemberApplication
«control» SearchBook
«process» SearchBookProcess
«process» BookSystemAccess
«interface» IBookSystem
«subsystem proxy» BookSystem
1
1
0..*
1
1
1
1
0...*
1
1
1
1

*Figure 2-7: Search Book process model.*

### *2.6.2 Process Element Descriptions*

- **MemberApplication:** Manages all user interfacer (UI), forms for Member, allows them to interacts with the System. In this process, MemberApplication has a SearchBookForm (a boundary class) allows Members to search for books by keywords or by suggestions on the System.
- **SearchBookProcess:** Manages search books process on the System. This process has a instance of SearchBookController (a control class).
- **BookSystemAccess:** Manages all accesses to the BookSystem subsystem. Since this process is asynchronous, Member can interact with the system while waiting for responses from BookSystem subsystem. This process also synchronizes accesses to the BookSystem from the other system processes.

## 2.7 Sell Used Book

### *2.7.1 Process Model*

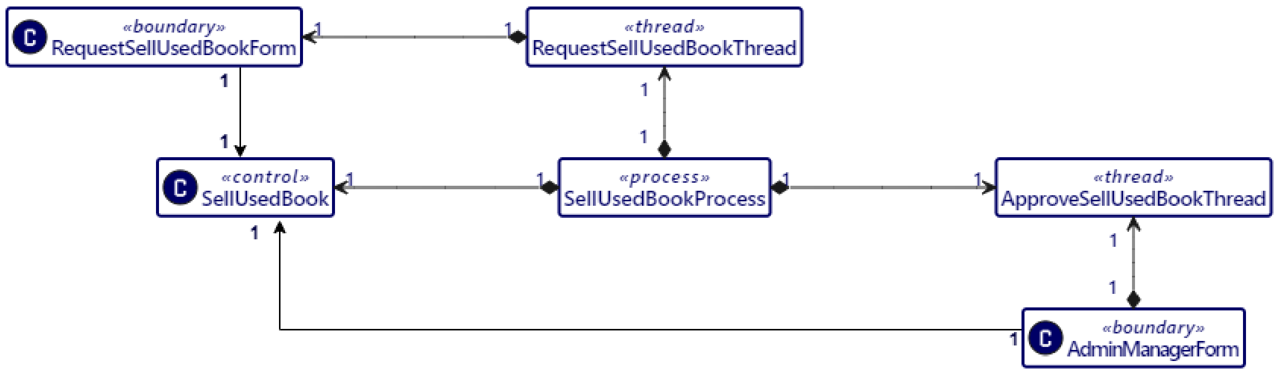

«boundary» RequestSellUsedBookForm
«thread» RequestSellUsedBookThread
«control» SellUsedBook
«process» SellUsedBookProcess
«thread» ApproveSellUsedBookThread
«boundary» AdminManagerForm
1
1
1
1
1
1
1
1
1
1
1
1
1
1

*Figure 2-8: Sell Used Book process model.*

### *2.7.2 Process Element Descriptions*

- **RequestSellUsedBookThread**: Manages the user interfaces (UI), forms for a Member to interact with System. This process has an instance of

RequestSellUsedBookForm (a boundary class) request to sell his used book on the System.

- **SellUsedBook:** Manages sell used books process on the System. This process contains two threads which are RequestSellUsedBookThread, ApproveSellUsedBookThread.
- **ApproveSellUsedBookThread:** Manages the user interfaces (UI), forms for a Administrator to interact with System. This process has an instance of AdminManagerForm (a boundary class) which has the responsibility of approving the sell used book requests from Members.

## 2.8 Chat Services

### *2.8.1 Process Model*

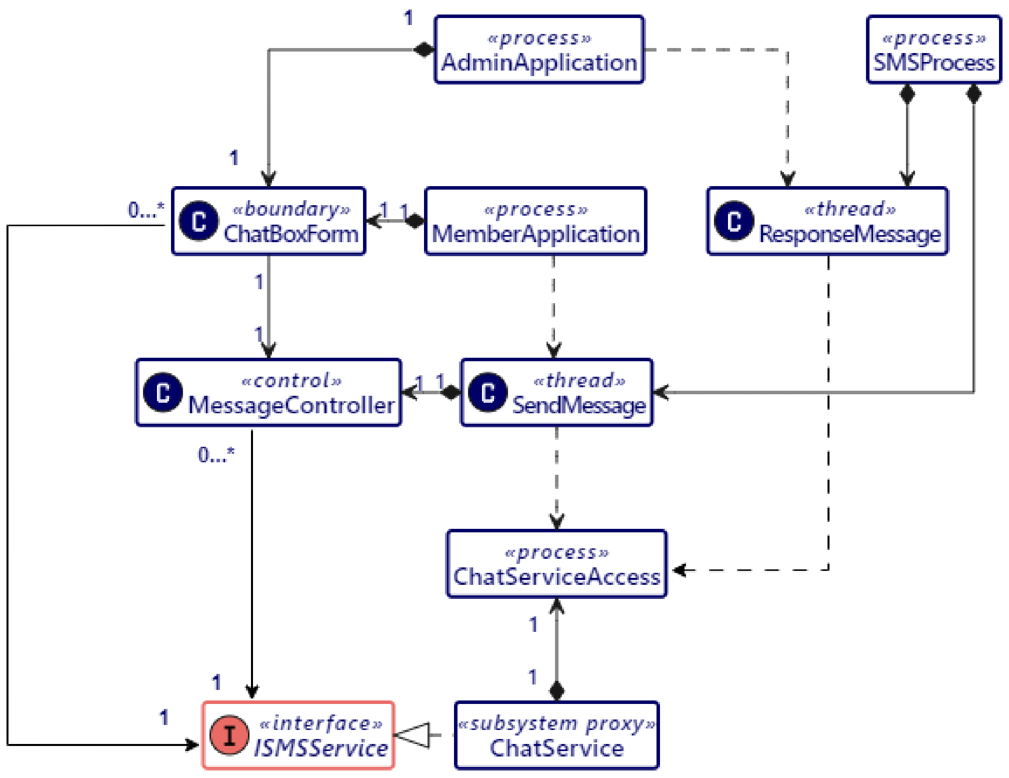

«process» AdminApplication
«process» SMSProcess
«boundary» ChatBoxForm
«process» MemberApplication
«thread» ResponseMessage
«control» MessageController
«thread» SendMessage
«process» ChatServiceAccess
«interface» ISMSService
«subsystem proxy» ChatService
1
0...*
0...*

*Figure 2-9: Chat Service process model.*

### *2.8.2 Process Element Descriptions*

- **MemberApplication:** Manages all user interfacer (UI), forms for Member, allows them to interacts with the System. In this process, MemberApplication has a ChatBoxForm (a boundary class) allows Members to type and send messages on the System.
- **AdminApplication:** Manages all user interfacer (UI), forms for Administrator, allows them to interacts with the System. In this process, AdminApplication has a ChatBoxForm (a boundary class) allows Administrator to type and send messages on the System.
- **SendMessage:** A thread allows Members to send messages to the System. There is one instance of MessageController class which is used in this thread.
- **RespondMessage:** A thread allows Admin to respond to the messages which the Memebers has sent. There is one instance of MessageController class which is used in this thread.
- **SMSProcess:** This process allows the System to access to a third-party app (for example: Facebook's Messenger) to handle messages and contains two threads which are SendMessage and RespondMessage.
- **ChatServiceAccess:** Manages all accesses to the ChatSerivce subsystem. Since this process is asynchronous, the users can interact with the system while waiting for responses from ChatService subsystem. This process also synchronizes accesses to the ChatService from the other system processes.

## 2.9 Buy Book

### *2.9.1 Process Model*

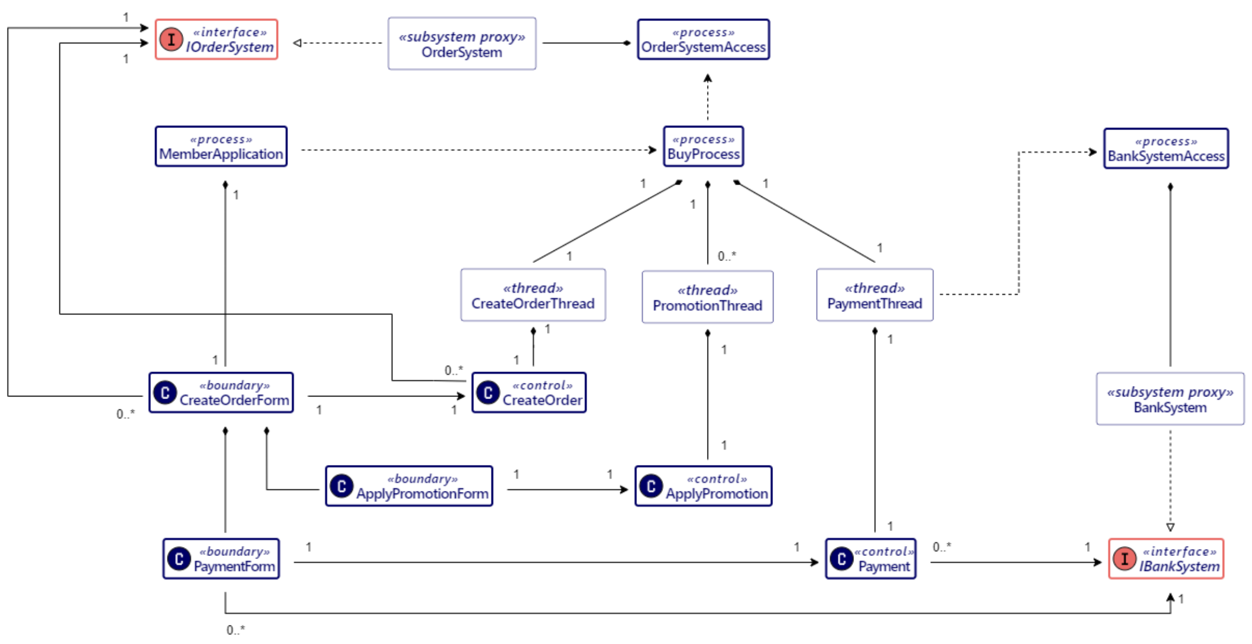

«interface» IOrderSystem
«subsystem proxy» OrderSystem
«process» OrderSystemAccess
«process» MemberApplication
«process» BuyProcess
«process» BankSystemAccess
«thread» CreateOrderThread
«thread» PromotionThread
«thread» PaymentThread
«boundary» CreateOrderForm
«control» CreateOrder
«subsystem proxy» BankSystem
«boundary» ApplyPromotionForm
«control» ApplyPromotion
«boundary» PaymentForm
«control» Payment
«interface» IBankSystem

*Figure 2-10: Buy Book process model.*

### *2.9.2 Process Element Descriptions*

- **MemberApplication**: Manages all user interfacer (UI), forms for Member, allows them to interacts with the System. In this process, the CreateOrderForm class is responsible for creating an order for purchasing items, the ApplyPromotionForm class has responsibility of allowing Members to apply discounts to their orders, and the PaymentForm class handles displaying payment information and completing the order transaction.
- **BuyProcess:** Manages the initiating of an order to completing the payment for that order. There is an instance of this process corresponding to the registration for participation in an event on the system. This process is divided into three threads which are CreateOrderThread, PromotionThread and PaymentThread.
- **CreateOrderThread:** Manages the flow of creating a new order for users.
- **PromotionThread:** Manages the flow of using a promotional code when paying for a buyer's order.
- **PaymentThread:** Manages the payment flow for the user's order value.

- **BankSystemAccess:** Manages all accesses to the BankSystem subsystem. This process also synchronizes access permissions to the BankSystem subsystem from other system processes. There is only one instance of the BankSystemAccess process.

## 2.10 Cancel Order

### *2.10.1 Process Model*

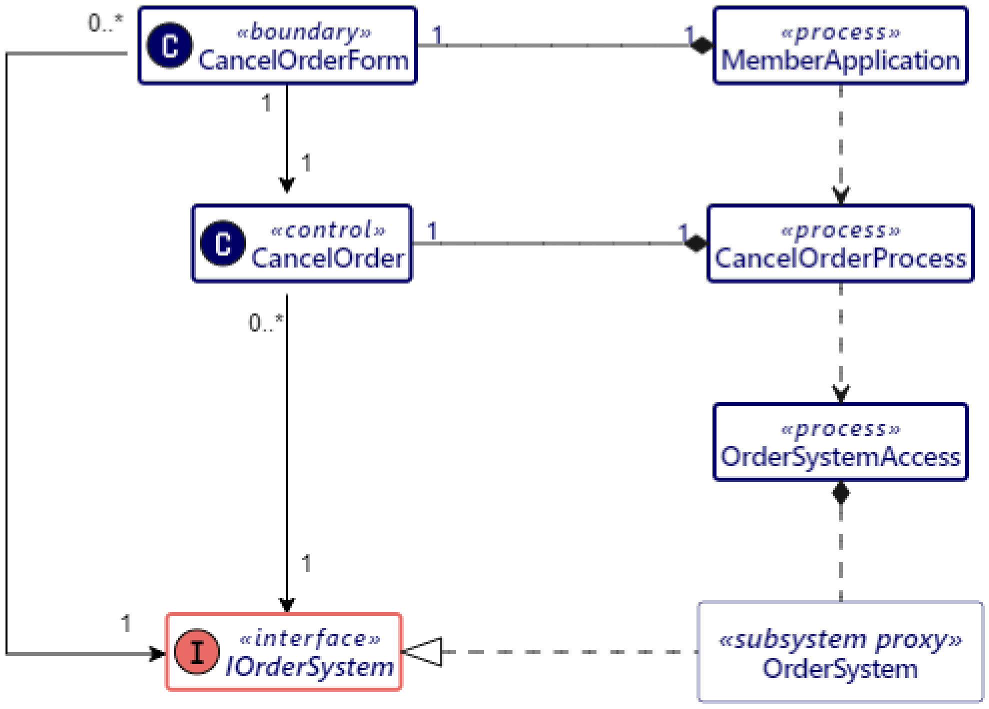

0..*
«boundary» CancelOrderForm
1
1
«process» MemberApplication
1
1
«control» CancelOrder
1
1
«process» CancelOrderProcess
0..*
«process» OrderSystemAccess
1
1
«interface» IOrderSystem
«subsystem proxy» OrderSystem

*Figure 2-11: Cancel Order process model.*

### *2.10.2 Process Element Descriptions*

- **MemberApplication**: Manages all user interfacer (UI), forms for Member, allows them to interacts with the System. This process includes the CancelOrderForm ( a boundary class) allows Members to request cancel his orders.

- **CancelOrderProcess**: Manages the process of canceling an order when a buyer decides not to make a purchase. There is an instance of this process which is CancelOrder (a control class) corresponding to each time a buyer cancels an order.
- **OrderSystemAccess**: Manages all accesses to the OrderSystem subsystem. This process also synchronizes access permissions to the OrderSystem subsystem from other system processes.

## 2.11 Review Order

### *2.11.1 Process Model*

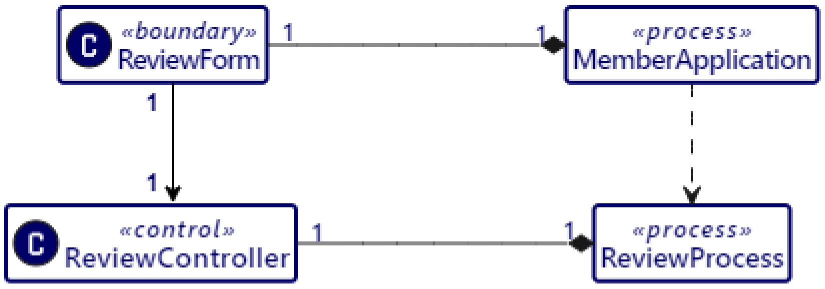


*Figure 2-12: Review Order process model.*

### *2.11.2 Process Element Descriptions*

- **MemberApplication:** Manages all user interfacer (UI), forms for Member, allows them to interacts with the System. This process includes the ReviewForm ( a boundary class) allows Members to request sending feedbacks to his successfully delivered orders.
- **ReviewProcess: M**anages the process of reviewing a successfully delivered order. There is an instance of this process which is ReviewController (a control class) corresponding to each time a Member chooses to review his orders.

## 2.12 Rental Plan Registration

### *2.12.1 Process Model*

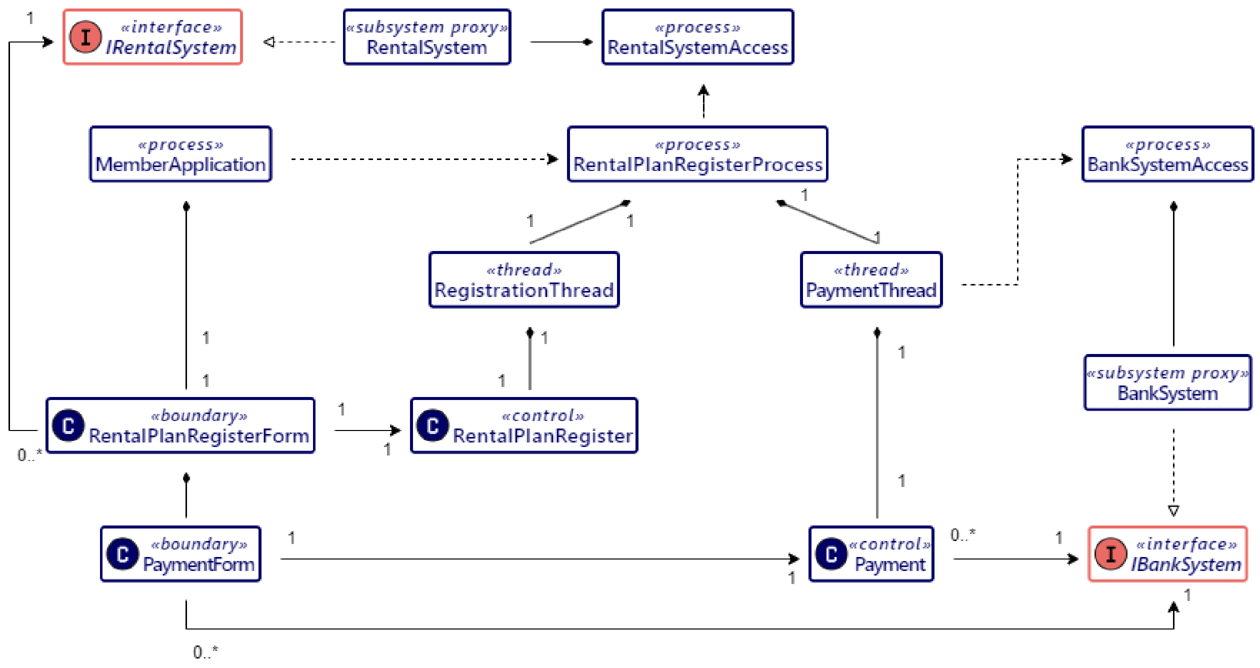

«interface» IRentalSystem
«subsystem proxy» RentalSystem
«process» RentalSystemAccess
«process» MemberApplication
«process» RentalPlanRegisterProcess
«process» BankSystemAccess
«thread» RegistrationThread
«thread» PaymentThread
«subsystem proxy» BankSystem
«boundary» RentalPlanRegisterForm
«control» RentalPlanRegister
«boundary» PaymentForm
«control» Payment
«interface» IBankSystem
1
0..*

*Figure 2-13: Rental Plan Registration process model.*

### *2.12.2 Process Element Descriptions*

- **MemberApplication**: Manages all user interfacer (UI), forms for Member, allows them to interacts with the System. In this process, the RentalPlanRegisterForm (a boundary class) is responsible for creating an order for purchasing items and the PaymentForm (a boundary class) handles displaying payment information and completing the order transaction.
- **RentalPlanRegisterProcess**: Manages the process of registration. This process contains two threads which are Registration Thread and PaymentThread.
- **RegistrationThread:** Manages the process of filling the information for the registration. There is an instance of this process which is RentalPlanRegister (a

control class) corresponding to the registration for participation in an event on the system.

- **PaymentThread:** Manages the payment flow for the user's order value.
- **BankSystemAccess:** Manages all accesses to the BankSystem subsystem. This process also synchronizes access permissions to the BankSystem subsystem from other system processes. There is only one instance of the BankSystemAccess process.

## 2.13 Rent Book

### *2.13.1 Process Model*

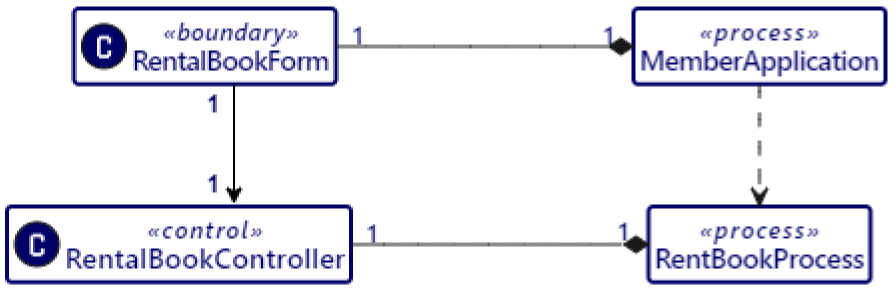


*Figure 2-14: Rent Book process model.*

### *2.13.2 Process Element Descriptions*

- **MemberApplication:** Manages all user interfacer (UI), forms for Member, allows them to interacts with the System. In this process, the RentalBookForm (a boundary class) is responsible for request renting a book on the System.
- **RentBookProcess:** Manages the process of renting a book on the System. There is only one instance of this process which is RentBookController (a control class).

# 3. Describe Concurrency

## 3.1 Process Model

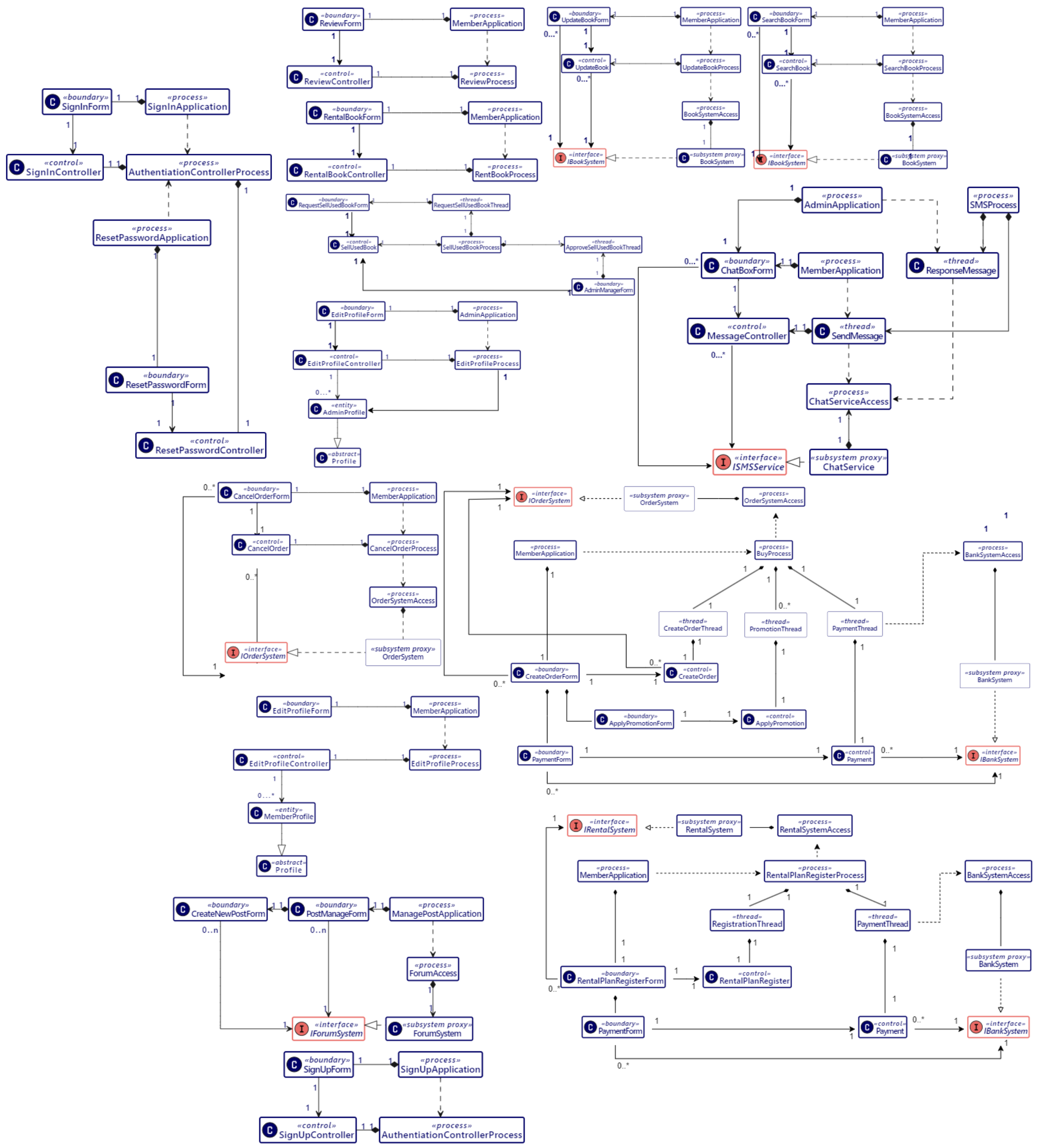


*Figure 3-1: Concurrent processes model.*

## 3.2 Process Element Descriptions

- **SignUpApplication:** Controls the user interfaces of the Sign Up application. Controls the family of forms that the users use to sign up. There is one instance of this process which is SignUpForm (a boundary class).
- **AuthenticationControllerProcess:** Manage the authentication processes.
- **SignInApplication:** Control the interfaces of the Sign In Application. Control the form that the user used to sign in. This process has only one instance which is the SignInForm (a boundary class).
- **ResetPasswordApplication:** Control the interface of the Reset Password Application. Control the form that the user used to reset their password. This process has only one instance which is the ResetPasswordForm (a boundary class).
- **MemberApplication:** Manages all user interfacer (UI), forms for Member, allows them to interacts with the System.
- **RentBookProcess:** Manages the process of renting a book on the System. There is only one instance of this process which is RentBookController (a control class).
- **RentalPlanRegisterProcess**: Manages the process of registration. This process contains two threads which are Registration Thread and PaymentThread.
- **RegistrationThread:** Manages the process of filling the information for the registration. There is an instance of this process which is RentalPlanRegister (a control class) corresponding to the registration for participation in an event on the system.
- **PaymentThread:** Manages the payment flow for the user's order value.
- **BankSystemAccess:** Manages all accesses to the BankSystem subsystem. This process also synchronizes access permissions to the BankSystem subsystem from other system processes. There is only one instance of the BankSystemAccess process.

- **MemberApplication:** Manages all user interfacer (UI), forms for Member, allows them to interacts with the System. This process includes the ReviewForm ( a boundary class) allows Members to request sending feedbacks to his successfully delivered orders.
- **ReviewProcess:** Manages the process of reviewing a successfully delivered order. There is an instance of this process which is ReviewController (a control class) corresponding to each time a Member chooses to review his orders.
- **CancelOrderProcess**: Manages the process of canceling an order when a buyer decides not to make a purchase. There is an instance of this process which is CancelOrder (a control class) corresponding to each time a buyer cancels an order.
- **OrderSystemAccess**: Manages all accesses to the OrderSystem subsystem. This process also synchronizes access permissions to the OrderSystem subsystem from other system processes.
- **BuyProcess:** Manages the initiating of an order to completing the payment for that order. There is an instance of this process corresponding to the registration for participation in an event on the system. This process is divided into three threads which are CreateOrderThread, PromotionThread and PaymentThread.
- **CreateOrderThread:** Manages the flow of creating a new order for users.
- **PromotionThread:** Manages the flow of using a promotional code when paying for a buyer's order.
- **PaymentThread:** Manages the payment flow for the user's order value.
- **BankSystemAccess:** Manages all accesses to the BankSystem subsystem. This process also synchronizes access permissions to the BankSystem subsystem from other system processes. There is only one instance of the BankSystemAccess process.

- **SendMessage:** A thread allows Members to send messages to the System. There is one instance of MessageController class which is used in this thread.
- **RespondMessage:** A thread allows Admin to respond to the messages which the Memebers has sent. There is one instance of MessageController class which is used in this thread.
- **SMSProcess:** This process allows the System to access to a third-party app (for example: Facebook's Messenger) to handle messages and contains two threads which are SendMessage and RespondMessage.
- **ChatServiceAccess:** Manages all accesses to the ChatSerivce subsystem. Since this process is asynchronous, the users can interact with the system while waiting for responses from ChatService subsystem. This process also synchronizes accesses to the ChatService from the other system processes.
- **RequestSellUsedBookThread**: Manages the user interfaces (UI), forms for a Member to interact with System. This process has an instance of RequestSellUsedBookForm (a boundary class) request to sell his used book on the System.
- **SellUsedBook:** Manages sell used books process on the System. This process contains two threads which are RequestSellUsedBookThread, ApproveSellUsedBookThread.
- **ApproveSellUsedBookThread:** Manages the user interfaces (UI), forms for a Administrator to interact with System. This process has an instance of AdminManagerForm (a boundary class) which has the responsibility of approving the sell used book requests from Members.
- **SearchBookProcess:** Manages search books process on the System. This process has a instance of SearchBookController (a control class).
- **BookSystemAccess:** Manages all accesses to the BookSystem subsystem. Since this process is asynchronous, Member can interact with the system while waiting for

responses from BookSystem subsystem. This process also synchronizes accesses to the BookSystem from the other system processes.

- **EditProfileProcess:** Manage edit profile process on the System. This process has a instance of EditProfileController (a control class).
- **AdminApplication:** Manages all user interfacer (UI), forms for Administrators, allows them to interacts with the System.
- **ManagePostApplication:** Manages the user interface (UI), forms for Member to create, edit, delete and close a post. In this process, there is an instance of PostManageForm (a boundary class) designed for closing, changing or deleting posts. The PostManageForm includes the CreateNewPostForm to create a new post. Every instances of this process indicate an action of a member to the ForumSystem.
- **ForumAccess:** Manages all accesses to the ForumSystem subsystem. Since this process is asynchronous, Member can interact with the system while waiting for responses from ForumSystem subsystem. This process also synchronizes accesses to the ForumSystem from the other system processes.
- **UpdateBookProcess:** Manages add new book, edit book's information or delete books processes on the System. This process has a instance of UpdateBookController (a control class).
- **BookSystemAccess:** Manages all accesses to the BookSystem subsystem. Since this process is asynchronous, Member can interact with the system while waiting for responses from BookSystem subsystem. This process also synchronizes accesses to the BookSystem from the other system processes.