# Architecture Handbook

**for**

# Book Rental and Sales System

**Version 1.2 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 |
| Le Quang Chinh | 04-Dec-23 | Add *Architectural Mechanisms* | 1.1 |
| Trinh Thai Linh | 04-Dec-23 | Add *Logical, Process, Deployment View* | 1.1 |
| Pham Tung Duong | 04-Dec-23 | Change *Deployment View* | 1.2 |
| | | | |
| | | | |
| | | | |
| | | | |

# 1. Introduction

## 1.1 Purpose

This is a report prepared by Group 7 (Object-oriented Analysis and Design Class 20, 22-23) 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". This report provides the description of the architecture that is used in the Book Rental and Sales System's development.

## 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. Architectural Mechanisms

## 2.1 Analysis Mechanisms

### *2.1.1 Persistence mechanisms*

For all classes with instances that may become persistent, you need to identify:

- **Granularity:** What is the range of size of the objects to keep persistent?
- **Volume:** How many objects (number) do you need to keep persistent?
- **Duration:** How long does the object typically need to be kept?
- **Retrieval mechanism:** How is a given object uniquely identified and retrieved?
- **Update frequency:** Are the objects more or less constant? Are they permanently updated?
- **Reliability:** Do the objects need to survive a crash of the process, the processor, or the whole system?

### *2.1.2 Communication mechanisms*

For all model elements that need to communicate with components or services that are running in other processes or threads, you need to identify:

- **Latency:** How fast must processes communicate with another?
- **Synchronicity:** Asynchronous communication.
- **Size of message:** A spectrum might be more appropriate than a single number.
- **Protocol:** Flow control, buffering, and so on.

### *2.1.3 Security mechanisms*

For all classes, packages, subsystem of the system, you need to identify:

- **Data granularity:** the level of depth represented by the data in a fact or dimension table in a data warehouse.

- **User granularity:** How many user roles does system have?
- **Security rules:** Security Rule establishes national standards to protect individuals' user data.
- **Privilege types:** What can a user role can do on the system?

## 2.2 Mapping Analysis Mechanism, Design Mechanism, Implementation Mechanism

| Analysis Mechanism | Design Mechanism | Implementation Mechanism |
|---|---|---|
| **Granularity** | MongoDB + Mongoose | MongoDB + ExpressJS |
| **Volume** | MongoDB + Mongoose | MongoDB + ExpressJS |
| **Duration** | | MongoDB + ExpressJS |
| **Retrieval mechanism** | MongoDB + Mongoose | MongoDB + ExpressJS |
| **Update frequency** | | MongoDB + ExpressJS |
| **Reliability** | MongoDB + Mongoose | MongoDB + ExpressJS |
| **Latency** | Communication | |
| **Synchronicity** | Communication | |
| **Size of message** | Communication | |
| **Protocol** | Communication | |
| **Data granularity** | Security | Secure Data Access |
| **User granularity** | Security | Secure User Setup |
| **Security rules** | Security | Secure Data Access |
| **Privilege types** | Security | Secure Data Access |
| **Redundancy** | | |
| **Error detection, handling** | | |
| **Transaction management** | | |
| **Distribution** | REST API | |

# 3. Logical View

## 3.1 High-level Layers

- **Server layer:** supports several different application servers, where "application" includes static web pages. The existence of a server is known to the network server. In general servers are managed by the network server, but application programs may partly take over this responsibility.
- **Client Layer:** The client layer is where the user accesses the application. The server layer accepts requests through the internet connection from client layer and passes these requests to the appropriate agent. The server then relays the response from the agent back to the client layer. In this cases, the client is simply a browser.

## 3.2 Architectural Layers and Their Dependencies

### *3.2.1 Layer Dependencies Diagram*

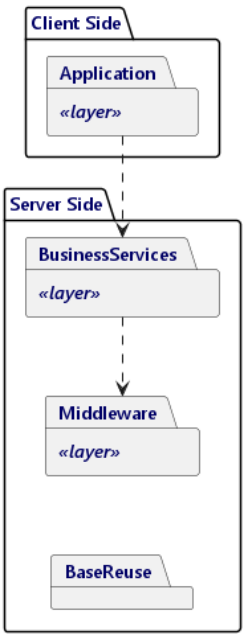


*Figure 3-1: Layer Dependencies Diagram.*

### *3.2.2 Layer Descriptions*

- **Application:** The Application layer contains application-specific design elements.
- **Business Services:** The Business Services layer contains business-specific elements that are used in several applications.
- **Middleware:** Provides utilities and platform-independent services.
- **Base Reuse:** Basic reusable design elements.

### *3.2.3 Packages and Their Dependencies*

#### *3.2.3.1 Diagram*

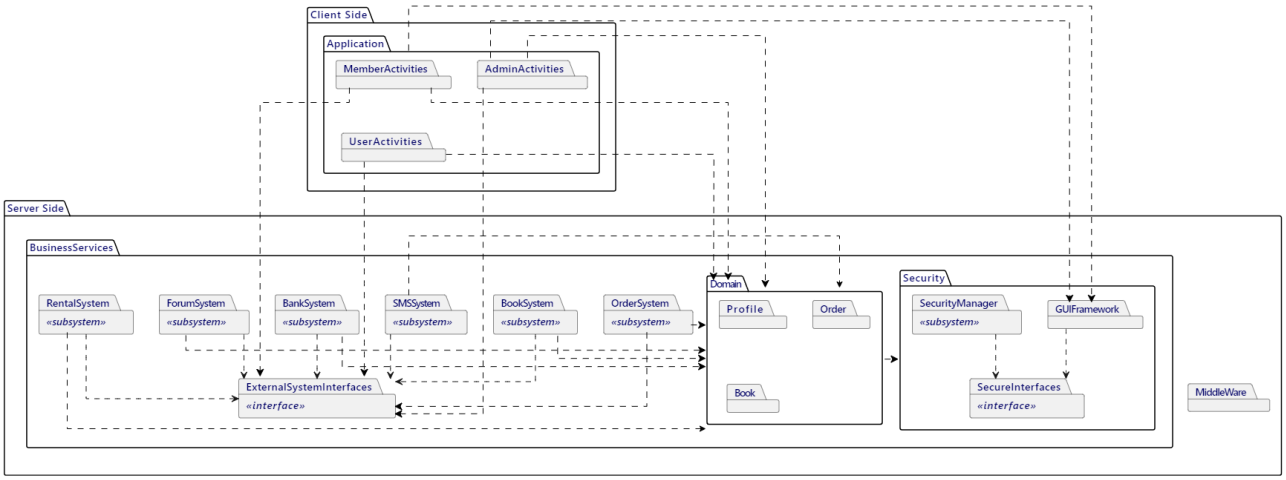


*Figure 3-2: Packages and Their Dependencies Diagram.*

#### *3.2.3.2 Package Descriptions*

- **MemberActivities:** Contains the design elements that support the Member's applications.
- **AdminActivities:** Contains the design elements that support the Admin's applications.
- **UserActivites:** Contains the design elements that support both Member and Admin's applications. (for example: Cancel Order, Edit Profile, SignIn)
- **BankSystem subsystem:** Encapsulates communication with all external bank systems.

- **SMSSystem subsystem:** Encapsulates communication with the external SMS system.
- **BookSystem subsystem:** Encapsulates behaviours to support Books related applications.
- **OrderSystem subsystem:** Encapsulates behaviours to support Orders related applications.
- **External System Interfaces:** Contains the interfaces that support access to external systems. This is so that the external system interface classes can be version controlled independently from the subsystems that realize them.
- **Base Reuse:** Basic reusable design elements.
- **Domain:** Contain the core Book Rental and Sales System's abstractions.
- **SecurityManager Subsystem:** Provides the implementation for the core security services.
- **SecureInterface:** Contains the interfaces that provide clients access to security services.
- **GUI Framework:** This package comprises a whole framework for user interface management. This framework is security-aware, it has a login window that will create a server-resident user context object.

# 4. Process View

## 4.1 Process Model

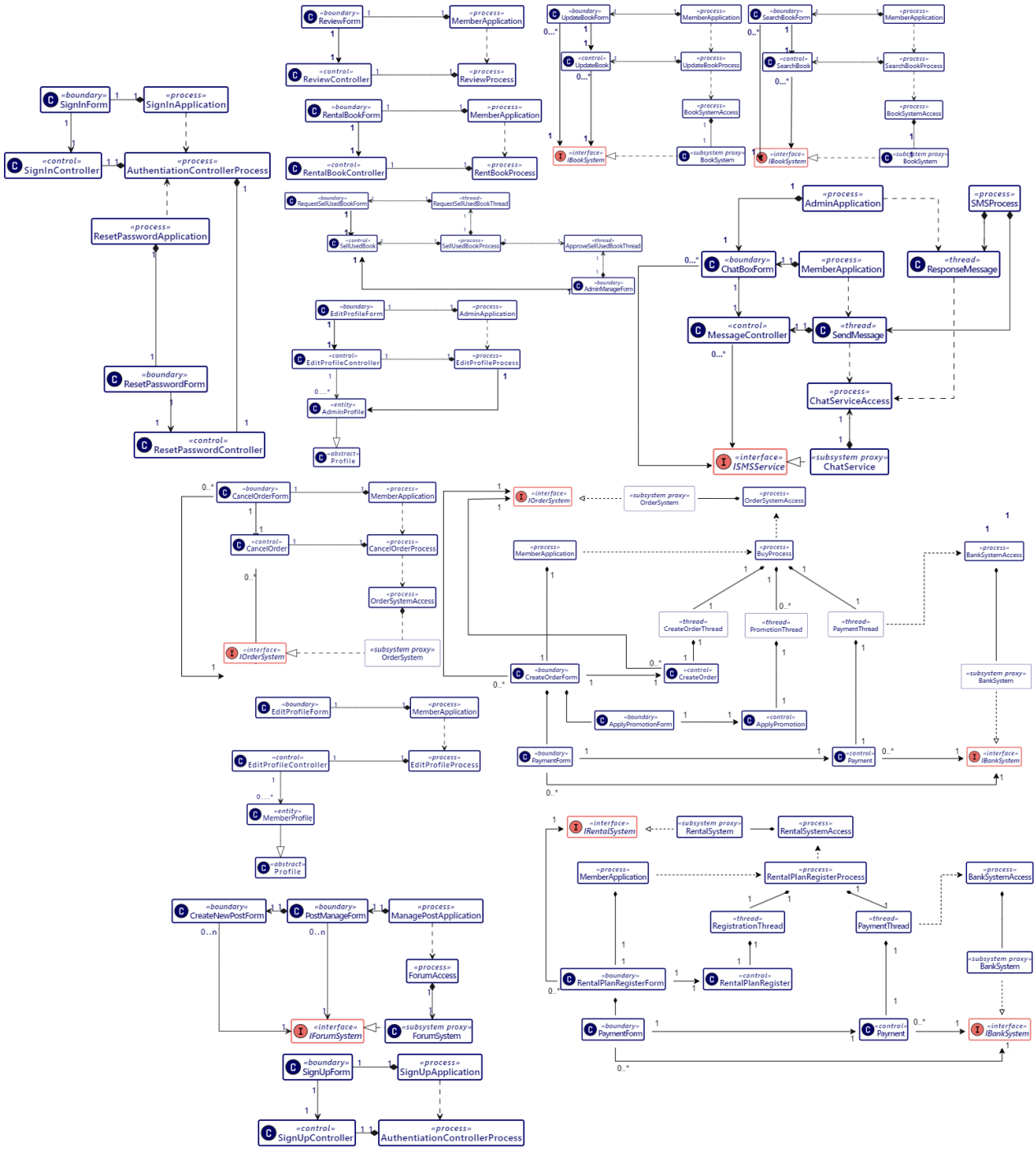


*Figure 4-1: Concurrent Processes Diagram.*

## 4.2 Process Element Descriptions

- **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.
- **Process 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: 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.
- **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.

## 4.3 Design Elements to Processes Map

| Design Elements | Process |
|---|---|
| SignUpForm | SignUpApplication |
| SignInForm | SignInApplication |
| ResetPasswordForm | ResetPasswordApplication |
| CreatePostForm | MemberApplication |
| ManagePost | MemberApplication |
| CreateOrderForm | MemberApplication |
| PaymentForm | PaymentThread |
| CancelOrderForm | MemberApplication, AdminApplication |
| ReviewForm | MemberApplication |
| EditProfileForm | MemberApplication, AdminApplication |
| UpdateCartForm | MemberApplication |
| SearchBookForm | MemberApplication |
| ViewBookForm | MemberApplication |
| UpdateBookForm | AdminApplication |
| ReadBookForm | MemberApplication |
| FavoriteBookForm | MemberApplication |
| RentalPlanRegisterForm | MemberApplication |
| AdminManagerForm | AdminApplication |

| | |
|---|---|
| SellUsedBookForm | MemberApplication |
| SearchBook | SearchBookProcess |
| UpdateBook | UpdateBookProcess |
| EditProfile | EditProfileProcess |
| CreateOrder | BuyProcess |
| ReviewController | ReviewProcess |
| CancelOrder | CancelOrderProcess |
| RentalPlanRegister | RentalPlanRegisterProcess |
| RentBook | RentBookProcess |
| SellUsedBookController | SellUsedBookProcess |
| ChatService | SMSProcess |

# 5. Deployment View

## 5.1 Deployment View Diagram

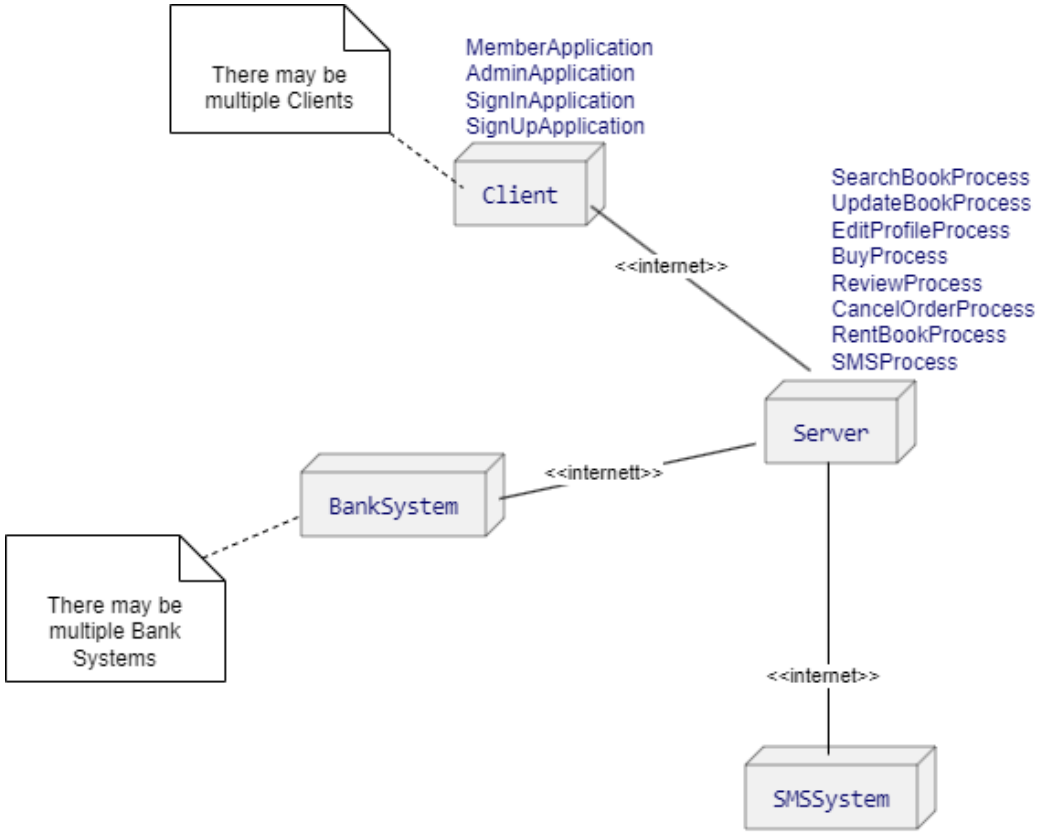


*Figure 5-1: Deployment View Diagram.*

# 6. Appendix A: Glossary

- **Analysis class:** A class used to model interaction between the system's surroudings, control, information and associate behavior specific in system( Boundary class, control class, Entity class)
- **Architectural layer:** Layer is contructed, progressed and concerned with the logical divison of components and functionality in system.
- **Bank System:** A system supports finance management of system and stakeholders for transactions.
- **Context Diagram:** A context diagram is a data flow diagram, with only one massive central progress that subsumes everything inside the scope of the system. It shows how the system will receive and send data flows to the external entities involved.
- **Dependency:** Exists between two elements if changes to one element may cause changes to the other.
- **Design Element**: The analysis classes are refined into design model elements (design classes, packages and subsystems).
- **Interface:** An interface is a model element that defines a set of behaviors (a set of operations) offered by a classifier model element (specifically, a class, subsystem, or component).
- **Layer:** A grouping of classes, packages or subsystems that together have responsibility for one major aspect of a system.
- **Package:** A general-purpose mechanism for organizing elements into groups. They provide the ability to organize the model under development. A package is represented as a tabbed folder.
- **Persistence:** refers to the characteristic of state that outlives the process that created it. This is achieved in practice by storing the state as data in computer data storage.
- **Security:** Is the degree of resistance to, or protection for data, resources of customer from harm.

- **Subsystem:** Is used as a unit of behavior in the system, which provides the ability to completely encapsulate the interactions of a number of class and/or subsystems.
- **Deployment View:** shows the configuration of run time processing nodes and the components that live on them.