Component Diagrams

While other UML diagrams, which describe the functionality of a system, component diagrams are used to model the components that help make those functionalities.

In this component diagram tutorial, we will look at what a component diagram is, component diagram symbols, and how to draw one. You can use a component diagram example below to get a quick start.

What is Component Diagram

Component diagrams are used to visualize the organization of system components and the dependency relationships between them. They provide a high-level view of the components within a system.

The components can be a software component such as a database or user interface; or a hardware component such as a circuit, microchip or device; or a business unit such as supplier, payroll or shipping.

Component diagrams

- Are used in Component-Based-Development to describe systems with Service-Oriented-Architecture
- Show the structure of the code itself
- Can be used to focus on the relationship between components while hiding specification detail
- Help communicate and explain the functions of the system being built to stakeholders

Component-based development (CBD) and object-oriented development go hand-in-hand, and it is generally recognized that object technology is the preferred foundation from which to build components. The Unified Modeling Language (UML) includes a component diagram that shows the dependencies among software components, including the classifiers that specify them (for example implementation classes) and the artifacts that implement them; such as source code files, binary code files, executable files, scripts and tables.

Component diagrams, along with UML Activity diagrams, are arguably one of the “forgotten" UML diagrams. Few books invest much time discussing them, I suspect the primary reason for this is because most methodologists appear to relegate them to low-level design diagrams for specifying the configuration of your software. UML Deployment diagrams are preferred by most modelers for this task, not only can you define what you intend to deploy you can also indicate where you intend to deploy it with deployment diagrams, and most programmers prefer to use their configuration management system to define configurations. UML Component diagrams become much more useful when used as architectural-level artifacts, perhaps used to model the logical architecture of your technical or business/domain infrastructures.

There are guidelines for:

1. Components
2. Interfaces
3. Dependencies and Inheritance

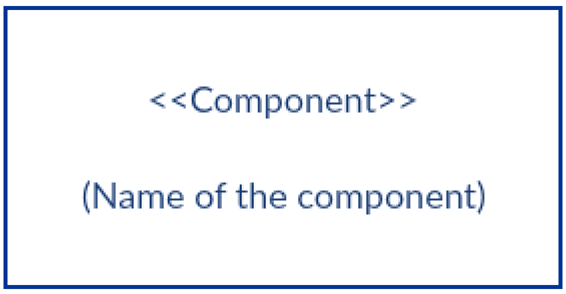
Component Diagram Symbols

We have explained below the common component diagram notations that are used to draw a component diagram.

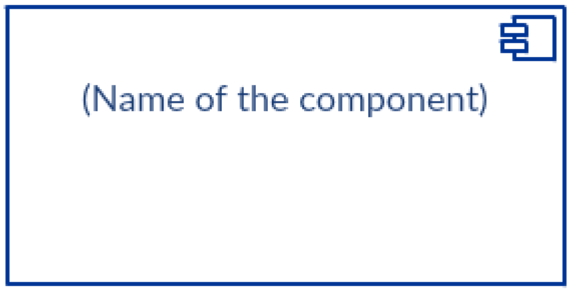
Component

There are three ways the component symbol can be used.

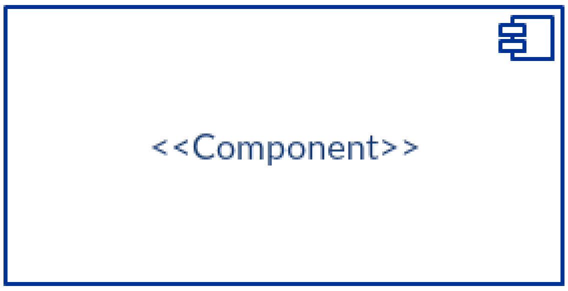
1) Rectangle with the component stereotype (the text <<component>>). The component stereotype is usually used above the component name to avoid confusing the shape with a class icon.



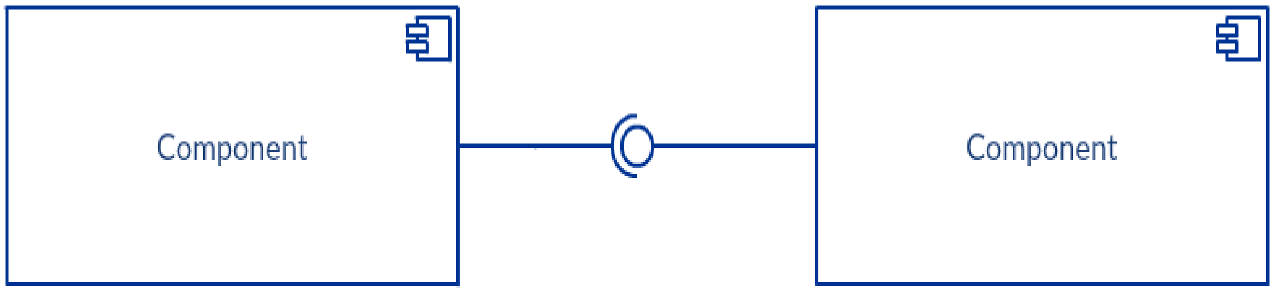
2) Rectangle with the component icon in the top right corner and the name of the component.



3) Rectangle with the component icon and the component stereotype.

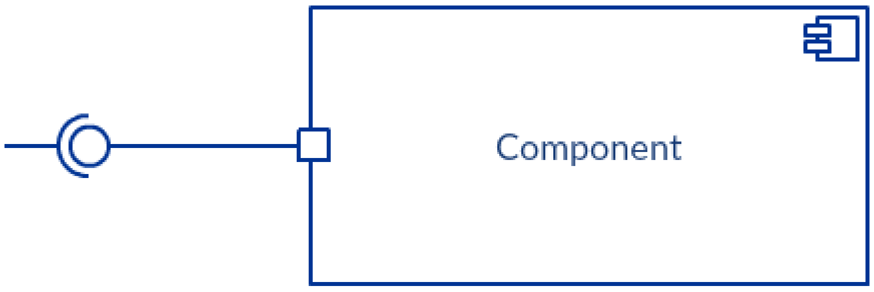


Provided Interface and the Required Interface



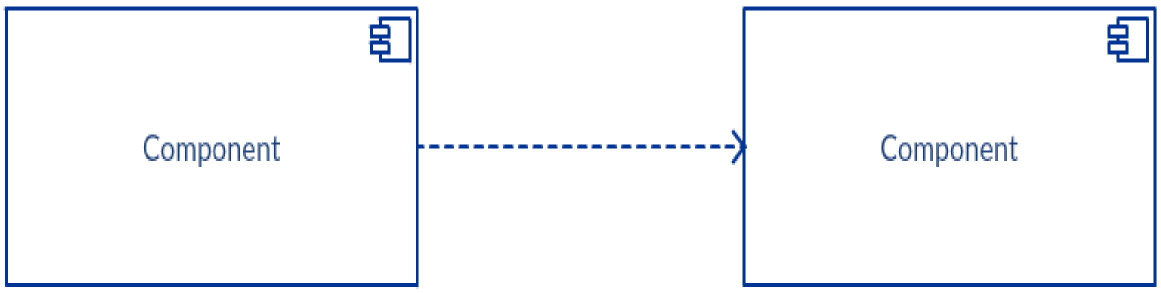
Interfaces in component diagrams show how components are wired together and interact with each other. The assembly connector allows linking the component's required interface (represented with a semi-circle and a solid line) with the provided interface (represented with a circle and solid line) of another component. This shows that one component is providing the service that the other is requiring.

Port



Port (represented by the small square at the end of a required interface or provided interface) is used when the component delegates the interfaces to an internal class.

Dependencies



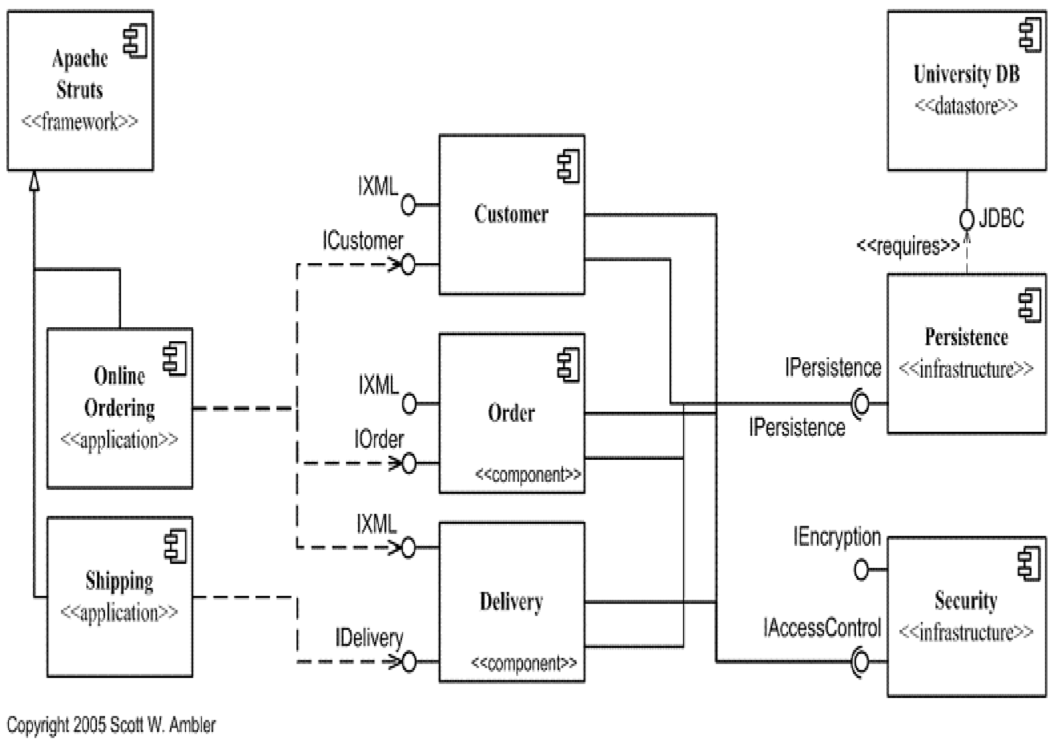
Although you can show more detail about the relationship between two components using the ball-and-socket notation (provided interface and required interface), you can just as well use a dependency arrow to show the relationship between two components.

ARRANGEMENT OF COMPONENTS

1. Components

As you can see in Figure 1 UML 2.X components are modeled as rectangles with either a visual stereotype in the top left corner or the textual stereotype of <>. Components realize one or more interfaces, modeled using the lollipop notation in Figure 1, and may have dependencies on other components - as you can see the *Persistence* component has a dependency on the *Corporate DB* component.

Figure 1. A UML Component 2.x diagram representing the logical architecture of a simple e-commerce system.


Apache Struts <<framework>>
University DB <<datastore>>
JDBC
<<requires>>
IXML
Customer
ICustomer
Persistence <<infrastructure>>
IPersistence
Online Ordering <<application>>
IXML
Order
IOrder
<<component>>
IPersistence
IXML
Delivery
<<component>>
IDelivery
Shipping <<application>>
IEncryption
Security <<infrastructure>>
IAccessControl
Copyright 2005 Scott W. Ambler

1. Use Descriptive Names for Architectural Components
2. Use Environment-Specific Naming Conventions for Detailed Design Components
3. Apply Textual Stereotypes to Components Consistently
4. Avoid Modeling Data and User Interface Components

2. Interfaces

1. Prefer Lollipop Notation To Indicate Realization of Interfaces By Components
2. Prefer the Left-Hand Side of A Component for Interface Lollipops
3. Show Only Relevant Interfaces

3. Dependencies and Inheritance

Components will have dependencies either on other components or better yet on the interfaces of other components. As you can see in Figure 1 and Figure 2 dependencies are modeled using a dashed line with an open arrowhead.

1. Model Dependencies From Left To Right
2. Place Child Components Below Parent Components
3. Components Should Only Depend on Interfaces
4. Avoid Modeling Compilation Dependencies

How to Draw a Component Diagram

You can use a component diagram when you want to represent your system as components and want to show their interrelationships through interfaces. It helps you get an idea of the implementation of the system. Following are the steps you can follow when drawing a component diagram.

Step 1: figure out the purpose of the diagram and identify the artifacts such as the files, documents etc. in your system or application that you need to represent in your diagram.

Step 2: As you figure out the relationships between the elements you identified earlier, create a mental layout of your component diagram

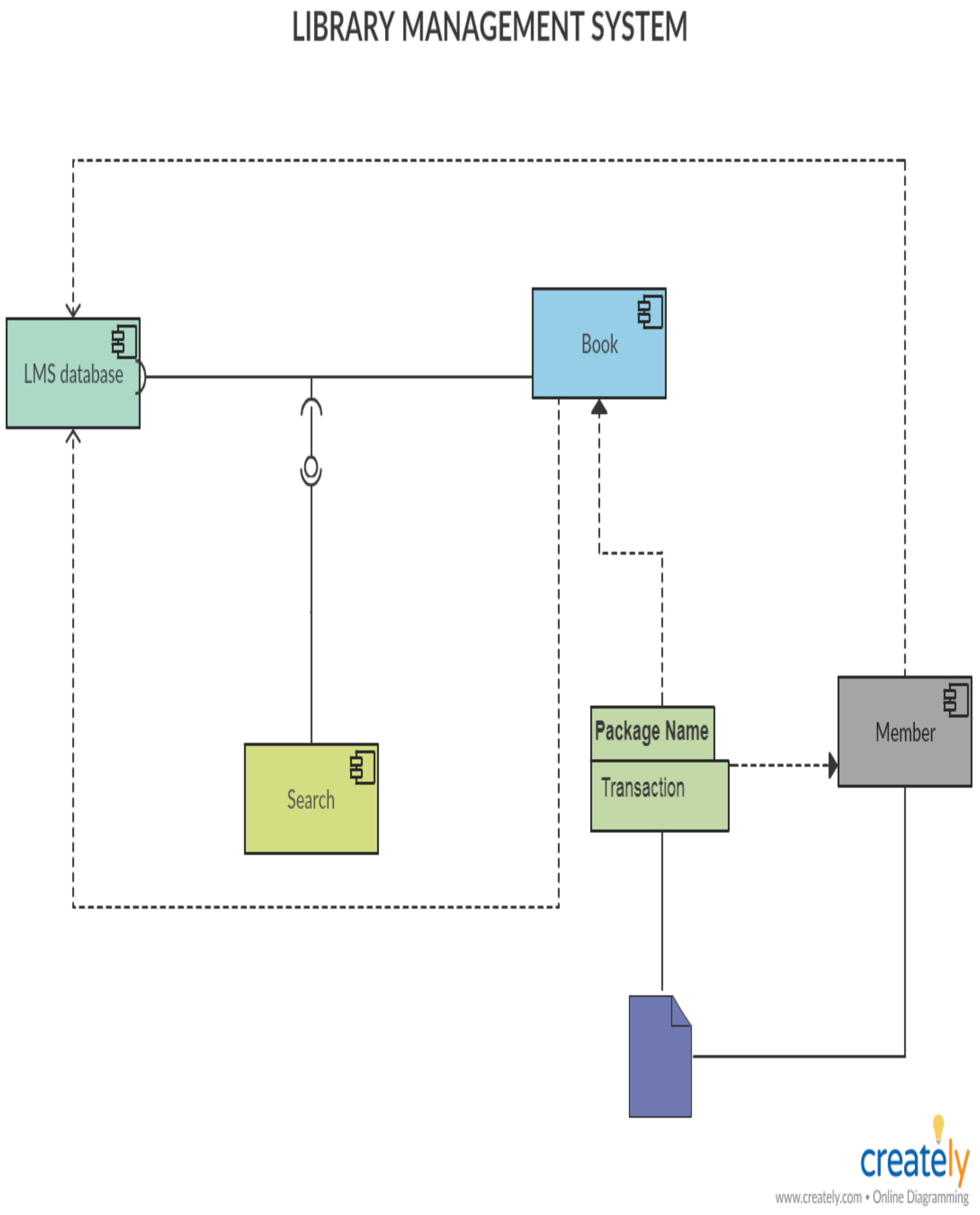
Step 3: As you draw the diagram, add components first, grouping them within other components as you see fit

Step 4: Next step is to add other elements such as interfaces, classes, objects, dependencies etc. to your component diagram and complete it.

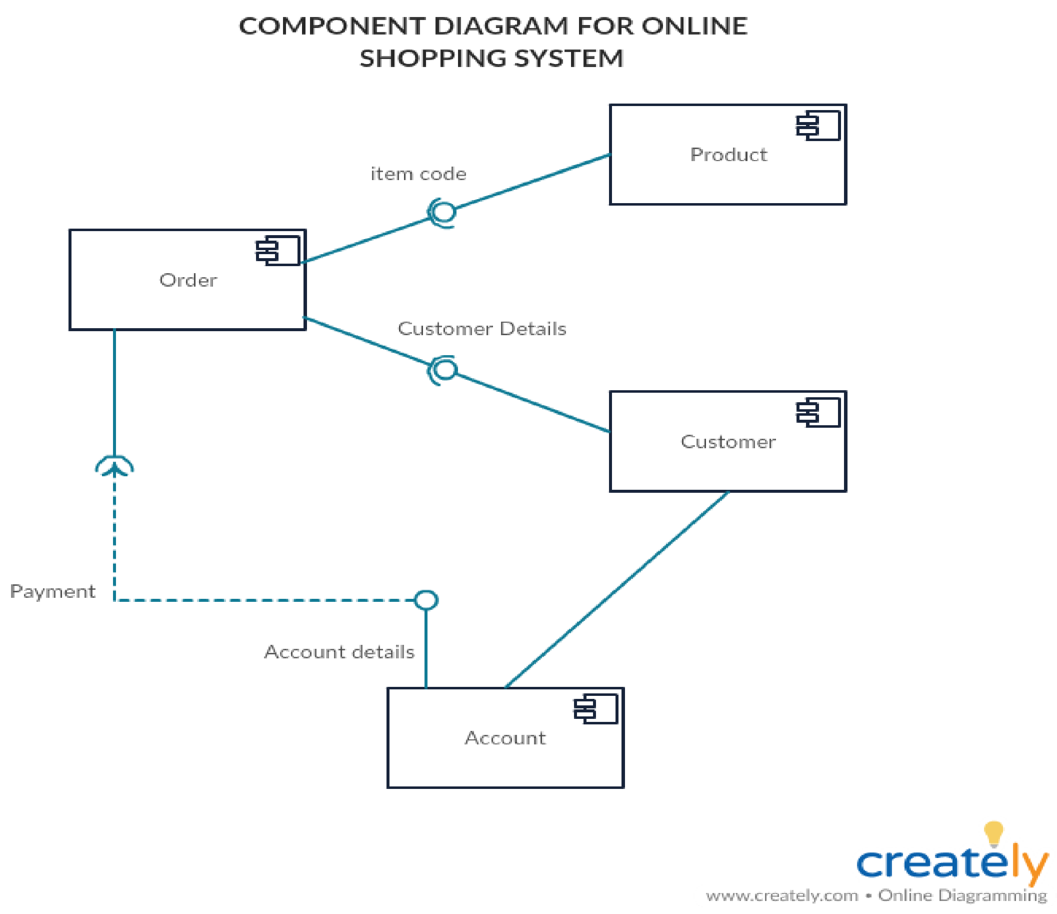
Step 5: You can attach notes on different parts of your component diagram to clarify certain details to others.

Component Diagram Examples

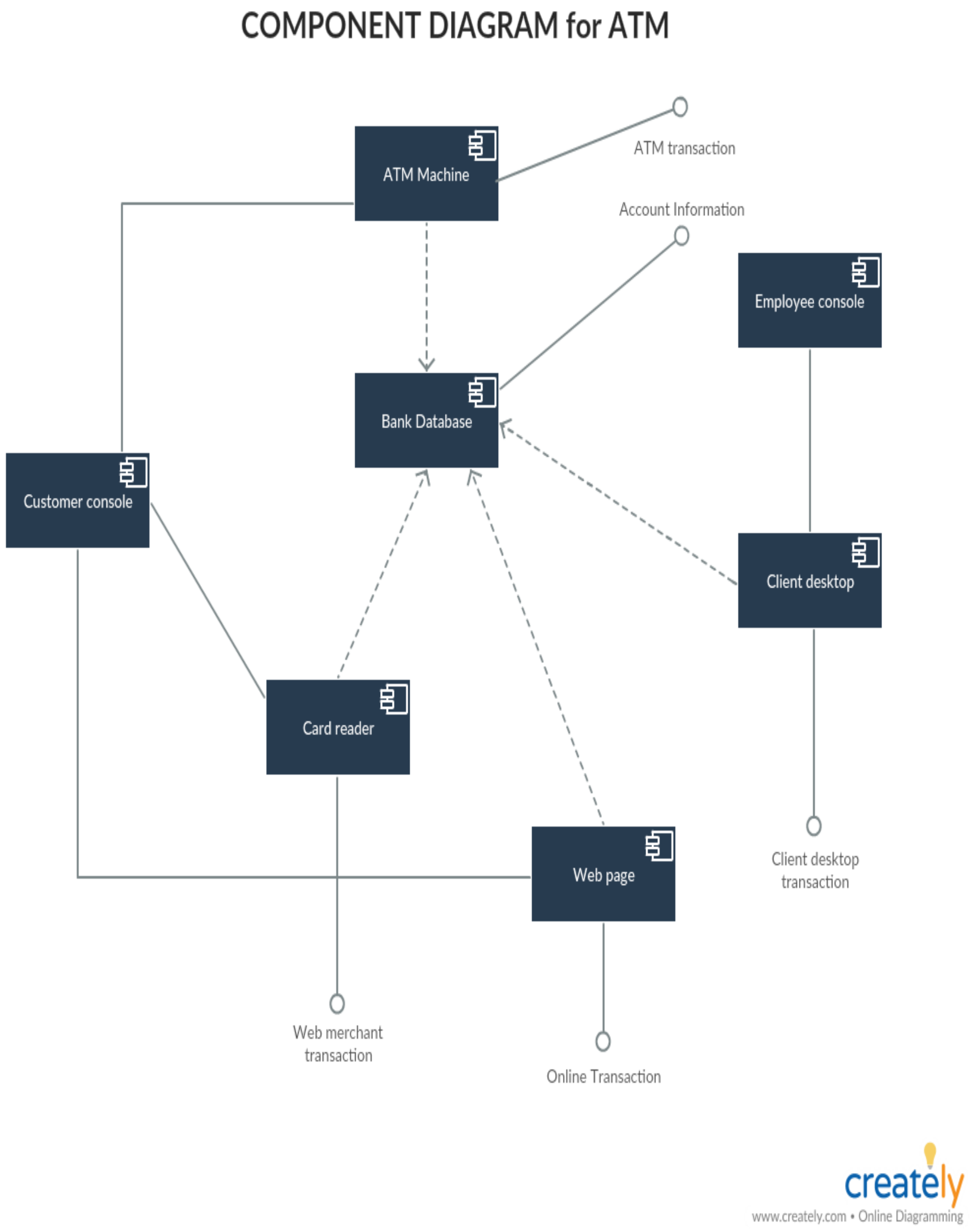
Component Diagram for Library Management System


LIBRARY MANAGEMENT SYSTEM
LMS database
Book
Search
Package Name
Transaction
Member
creately
www.creately.com • Online Diagramming

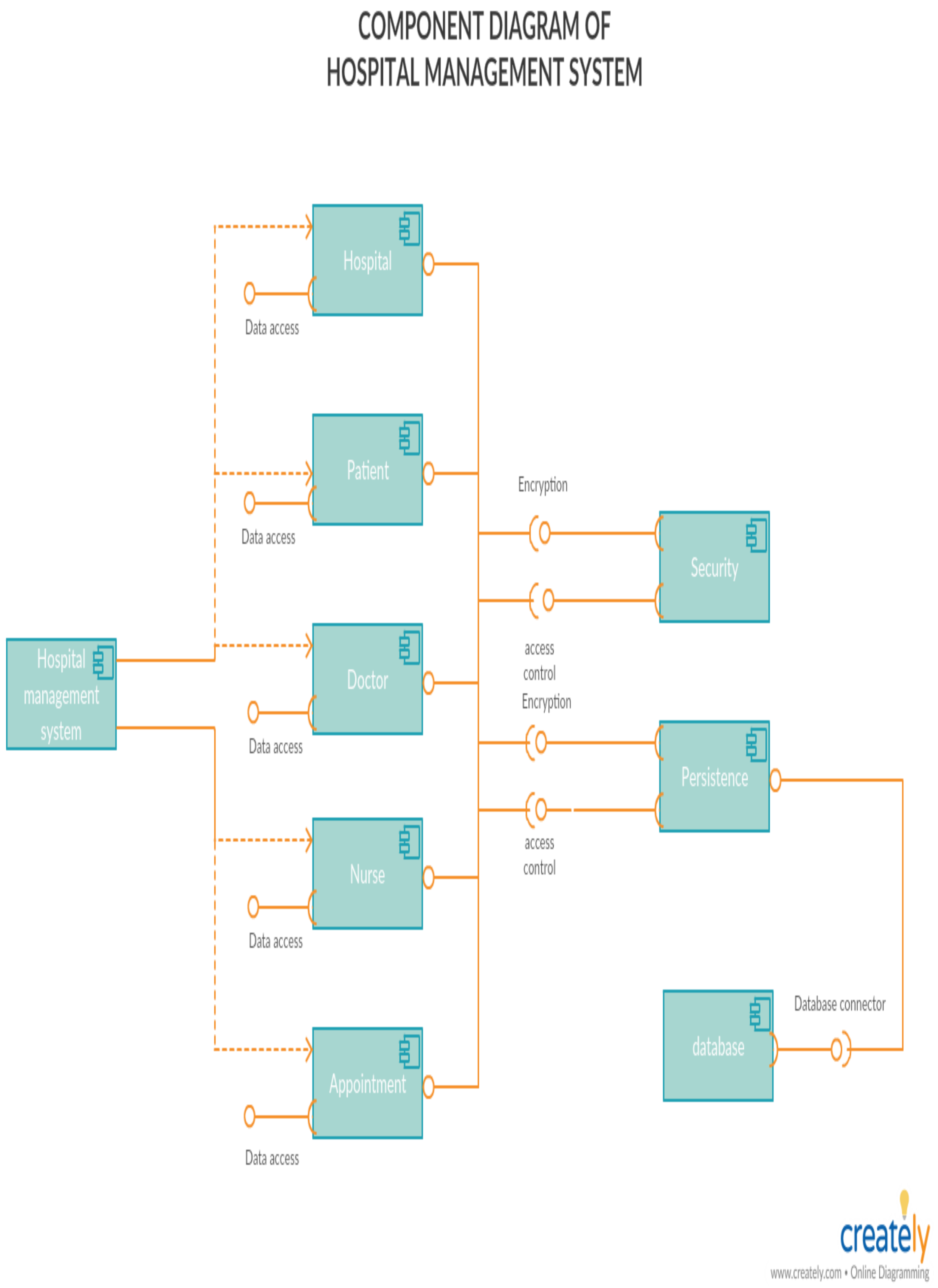
Component Diagram for Online Shopping System



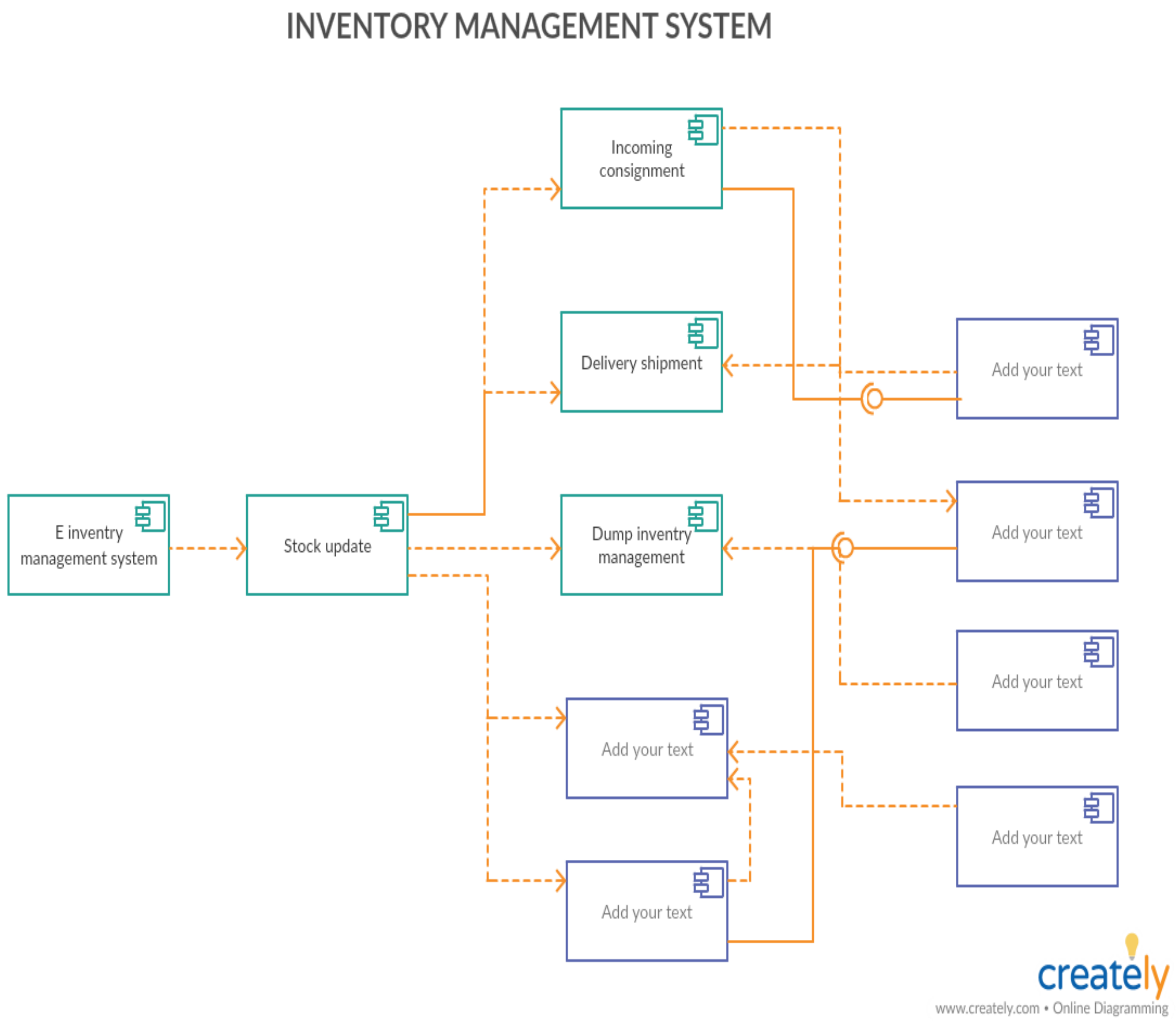
Component Diagram for ATM

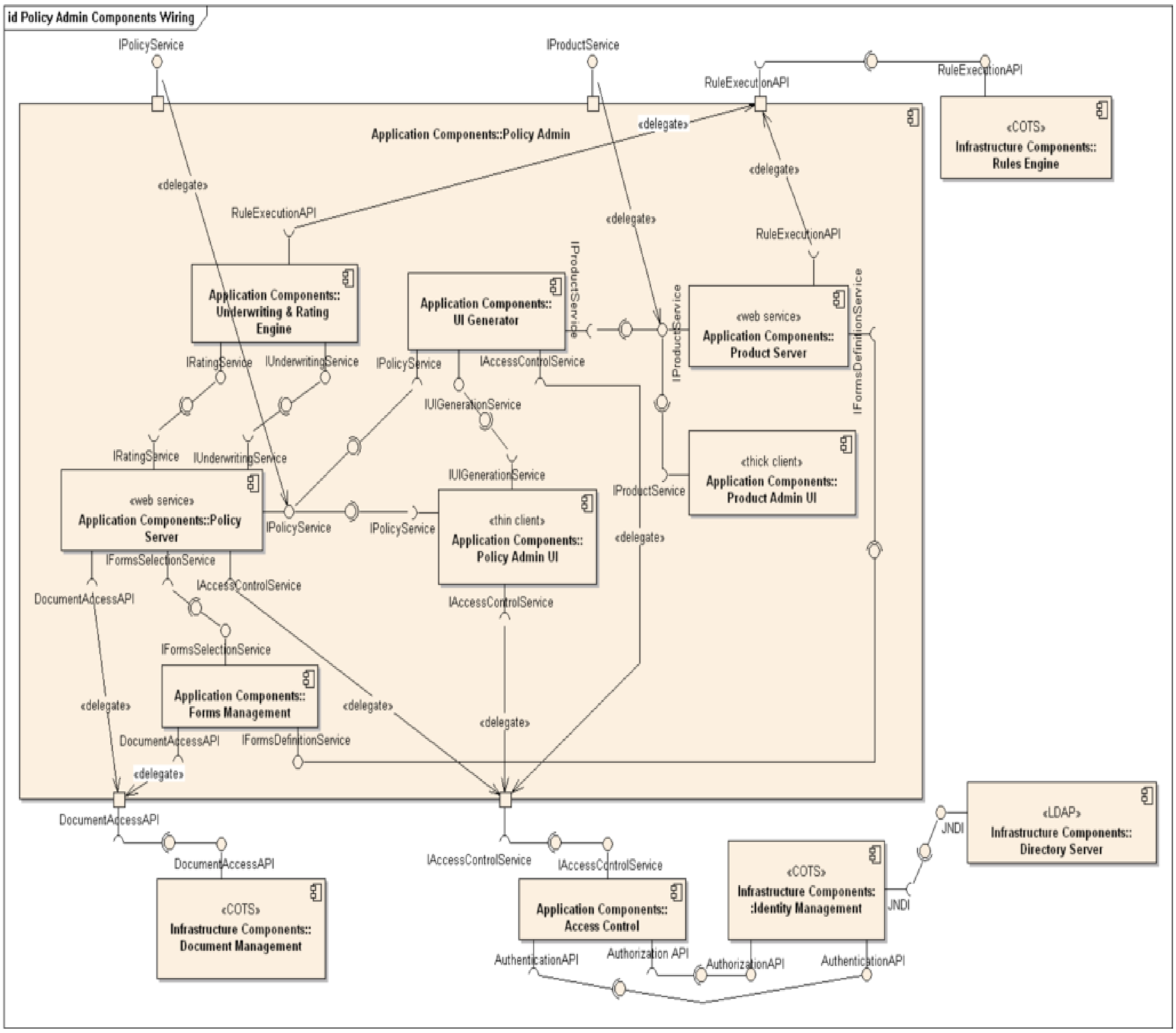


Component Diagram for Hospital Management System



Component Diagram for Inventory Management System




id Policy Admin Components Wiring
IPolicyService
IProductService
RuleExecutionAPI
RuleExecutionAPI
«COTS»
Infrastructure Components:: Rules Engine
Application Components::Policy Admin
«delegate»
«delegate»
«delegate»
«delegate»
RuleExecutionAPI
RuleExecutionAPI
Application Components:: Underwriting & Rating Engine
Application Components:: UI Generator
IProductService
IProductService
IFormsDefinitionService
«web service»
Application Components:: Product Server
IRatingService
IUnderwritingService
IPolicyService
IAccessControlService
IUIGenerationService
IRatingService
IUnderwritingService
IUIGenerationService
«thick client»
Application Components:: Product Admin UI
IProductService
«web service»
Application Components::Policy Server
IPolicyService
IPolicyService
«thin client»
Application Components:: Policy Admin UI
«delegate»
IFormsSelectionService
IAccessControlService
DocumentAccessAPI
IAccessControlService
IFormsSelectionService
Application Components:: Forms Management
«delegate»
«delegate»
«delegate»
DocumentAccessAPI
IFormsDefinitionService
«delegate»
DocumentAccessAPI
DocumentAccessAPI
«COTS»
Infrastructure Components:: Document Management
IAccessControlService
IAccessControlService
Application Components:: Access Control
«COTS»
Infrastructure Components:: :Identity Management
JNDI
JNDI
«LDAP»
Infrastructure Components:: Directory Server
AuthenticationAPI
Authorization API
AuthorizationAPI
AuthenticationAPI