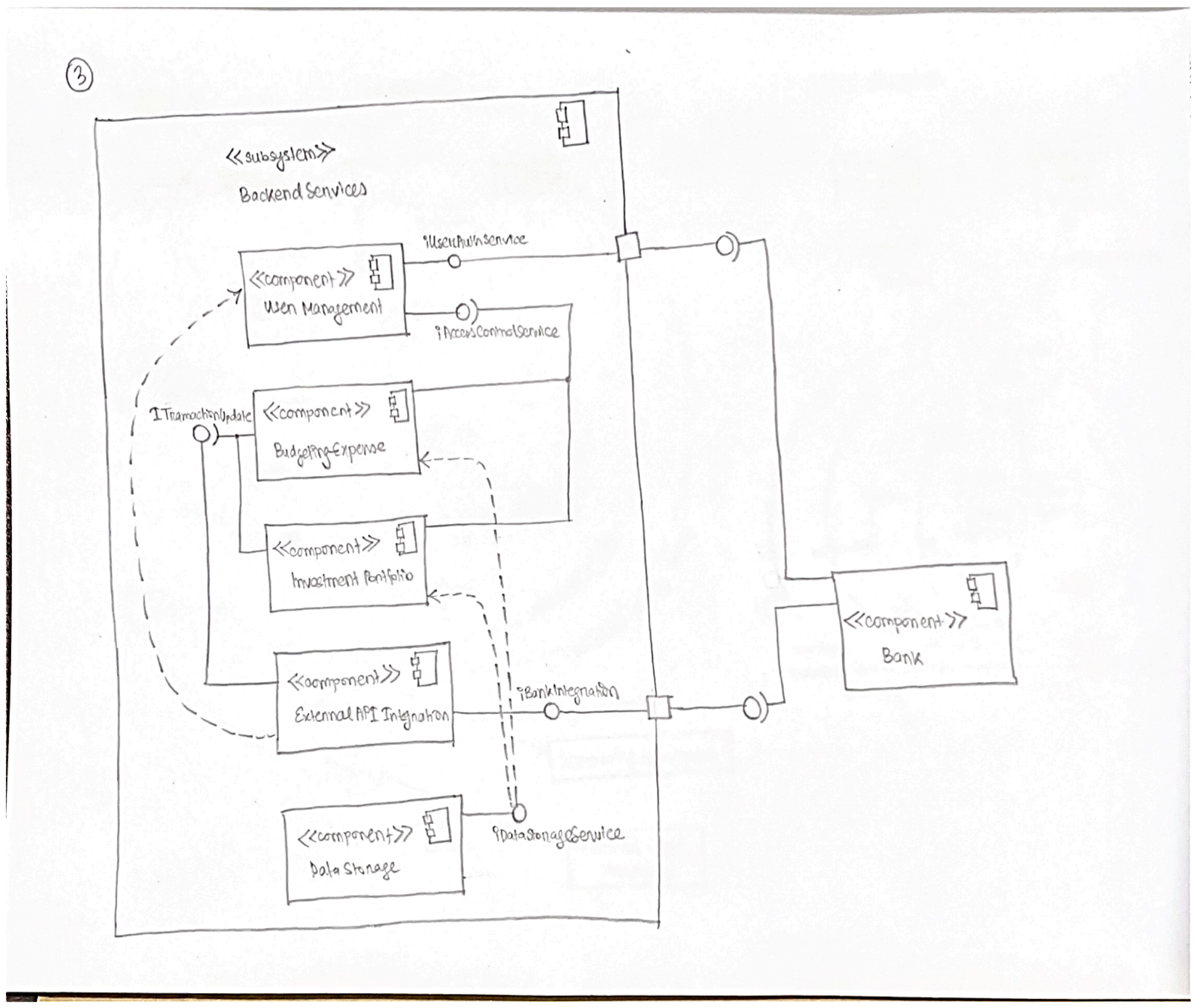
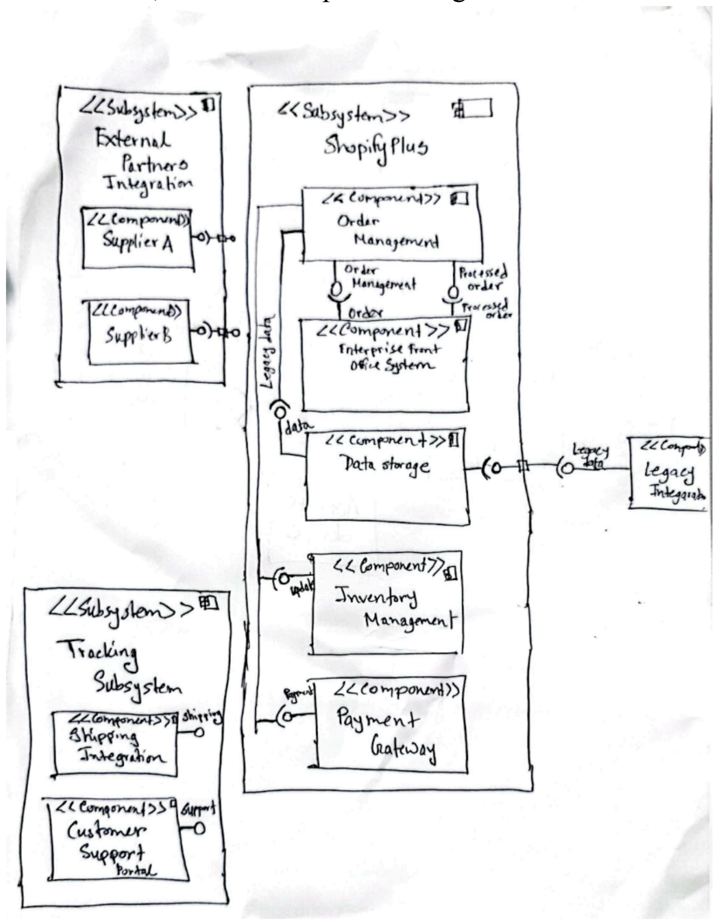
Suppose, you are working as a software architect for a mid-sized financial technology company. The company is developing a new platform called FinMaster, which aims to provide personal finance management solutions to end-users.
Backend Services subsystem has five internal components: User Management, Budgeting-Expense, Investment Portfolio, External API Integration, and Data Storage. The User Management component has two provided interfaces: iUserAuthService for user authentication and iAccessControlService for access control and role management. The budgeting-Expense component interacts with User Management component to retrieve user information and access control using required interface iAccessControlService. This component also depends on the iDataStorageService interface which is provided by the Data Storage component to store and retrieve budget data. Similar to the Budgeting-Expense component, the Investment Portfolio component retrieves user information from the User Management component using the required interface iAccessControlService and depends on the iDataStorageService interface for persisting portfolio information. External API Integration component communicates with external Bank component to retrieve transaction data using its required interface iBankIntegration. The Bank component has another required interface iUserAuthService. Delegation connectors link these external contracts of the subsystem to the realization of the contracts by User Management and External API Integration components. Both Budgeting-Expense and Investment Portfolio components use the ITransactionUpdate interface provided by the External API Integration component for retrieving transaction updates that are used in budgeting and portfolio analysis. The User Management component has a dependency on the External API Integration component to authenticate a user or verify their identity.



Suppose you are developing an e-commerce platform called "ShopifyPlus," aimed at providing a robust and scalable solution for businesses of all sizes. The platform leverages various components across the enterprise, and as an architect, you utilize UML component diagrams to describe the application architecture. The "Order Management" component is a component of the ShopifyPlus subsystem which serves as a backbone, orchestrating the entire order processing workflow. The component offers two crucial interfaces: "OrderEntry" and "ProcessedOrder." Both interfaces are required by the "Enterprise Front Office System" component, enabling seamless integration with the front-end systems of businesses. Additionally, the "Order Management" component relies on external services to fulfill its functionalities: An external "Legacy Integration" component provides the "LegacyData" interface, allowing ShopifyPlus Data Storage Component to integrate with legacy systems and retrieve historical data. An internal "Inventory Management" component offers the "Inventory" interface, enabling real-time inventory updates and stock management. A "Payment Gateway" component facilitates secure payment processing by providing the "Payment" interface. Furthermore, the platform interfaces with external companies through the "External Partners Integration" subsystem, consisting of two components: "Supplier A" component provides the "SAIntegration" interface. "Supplier B" component provides the "SBIntegration" interface. Delegation connectors link the external contracts of the External Partners Integration subsystem. Moreover, two critical components are integrated into the tracking subsystem: The "Shipping Integration" component provides shipping-related functionalities through the "Shipping" interface. The "Customer Support Portal" component enables seamless customer support interactions by offering the "Support" interface. Finally, the "Data Warehouse" component aggregates and stores order-related data for reporting and analytics purposes, leveraging the "DataStore" interface provided by the "Data Storage" component through a delegation connector.

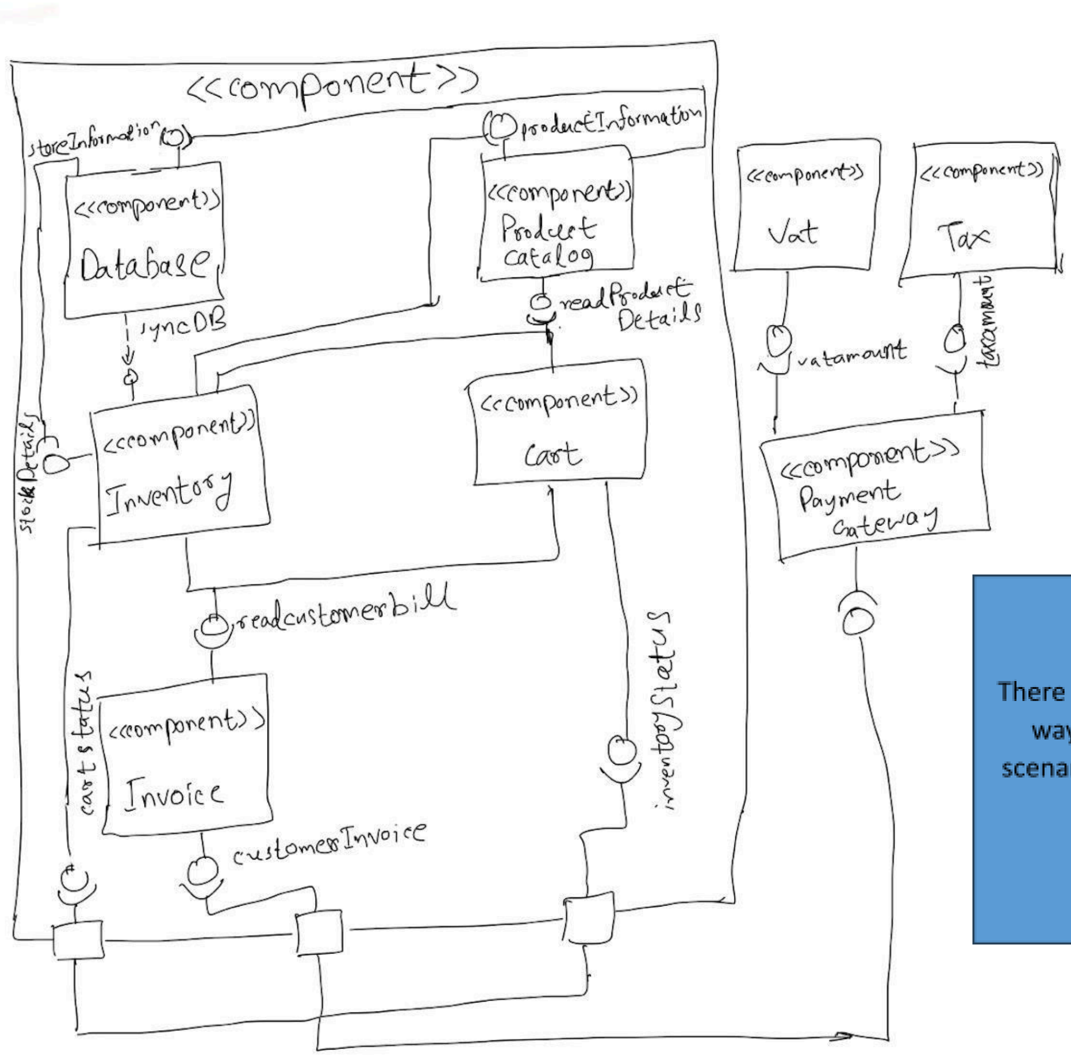
Now, **Draw** a Component Diagram based on the above scenario.



This solution has some errors. Identify them.

Supposed you are developing an e-commerce website and it contains several components. The website component contains five components that are: database, product catalog, inventory, cart, and invoice. The Database component provides an interface, storeInformation that is required by the product catalog component. Product Catalog component provides an interface, productInformation which is required by the Inventory component. Database component requires the stockDetails interface which is provided by the Inventory component. Product Catalog component provides another interface, readProductDetails which is required by both the Inventory and Cart components. The inventory component provides an interface called syncDB which is dependent on database updates from the Database component. The inventory and cart components both provide an interface called readCustomerBill which is required by the Invoice component. There are three components outside the Website component. They are payment gateway, vat, and tax. Cart, Invoice, and Inventory components provide inventoryStatus, customerInvoice, and cartStatus interfaces respectively and all of these are required by the Payment Gateway component. The Payment Gateway component also requires vatAmount from the Vat component and taxAmount from the Tax component.

Draw a component diagram based on the above scenerio.



There is an alternative way to solve this scenario. Think about that.