#	Class Name	PIC	Cohesion	Single Responsibility? Yes or No? Why?	Solution
1	RejectOrderController	Hồ Bảo Thư	Functional – All methods execute the reject order process.	Yes, this controller is responsible for coordinating the reject order workflow.	No change needed.
2	OrderRejectionService	Hồ Bảo Thư	Functional – Single method updates the status and reason of an order.	Yes, this service is responsible for updating the status and reason for order rejection.	No change needed.
3	PlaceOrderController	Hồ Bảo Thư	Functional – All methods contribute to executing the place order process.	Yes, this controller is responsible for coordinating the place order workflow across multiple steps.	No change needed.
4	ProductAvailabilityService	Hồ Bảo Thư	Funtional All methods and fields in the class contribute to a single task: checking the availability of products in the cart using ProductService	Yes, this service is responsible for validating the availability of all products in a cart before placing an order.	No change needed.
5	ProductService	Hồ Bảo Thư	Funtional All methods and fields in the class focus on managing a single product's availability	Yes, this service is responsible for managing inventory-related operations of a product.	No change needed.
6	CreateUserController	Hà Việt Khánh	Functional. All the functions in the class work together to support a single responsibility - coordinating the user creation process.	Yes. It coordinates the user creation process by delegating to separate services for validation, persistence, email, and logging. It should only change if the flow changes.	No change needed.
7	UserValidator	Hà Việt Khánh	Functional. All methods are focused on validating different aspects of a user.	Yes. Its responsibility is to validate the user's information (email, phone number, password, role, etc.) before user creation.	No change needed.
8	UserRepository	Hà Việt Khánh	Functional. Handles all logic related to storing and retrieving user data.	Yes. It is responsible for storing and retrieving user data and generating user IDs.	No change needed.
9	User	Hà Việt Khánh	Functional. All methods and attributes are directly related to managing a user's data (e.g., name, email, password, role) and behavior (e.g., update role, reset password).	Yes. It encapsulates all user-related data and operations.	No change needed.
10	Product	Đặng Văn Nhân	Informational Cohesion As it groups related data and behavior for a product	Yes as it only represents a product entity	No need to change
11	CD	Đặng Văn Nhân	Informational Cohesion As it groups related data and behavior for a CD	Yes, as it only represents a CD entity	No need to change
12	DVD	Đặng Văn Nhân	Informational Cohesion As it groups related data and behavior for a DVD	Yes, as it only represents a DVD entity	No need to change
13	LP	Đặng Văn Nhân	Informational Cohesion As it groups related data and behavior for a LP	Yes, as it only represents a LP entity	No need to change
14	ProductInfo	Đặng Văn Nhân	Coincidental Cohesion Low cohesion since it's just a wrapper around Product	No, since it doesn't add significant value beyond wrapping Product	Consider removing this class entirely and using Product directly
15	BookValidator	Đặng Văn Nhân	Functional Cohesion As it has a single responsibility of validating Book-specific fields	Yes, as it only handles Book validation	No need to change
16	CDValidator	Đặng Văn Nhân	Functional Cohesion As it has a single responsibility of validating CD-specific fields	Yes, as it only handles CD validation	No need to change
17	DVDValidator	Đặng Văn Nhân	Functional Cohesion As it has a single responsibility of validating DVD-specific fields	Yes, as it only handles DVD validation	No need to change
18	LPValidator	Đặng Văn Nhân	Functional Cohesion As it has a single responsibility of validating LP-specific fields	Yes, as it only handles LP validation	No need to change
19	ProductValidator	Đặng Văn Nhân	Functional Cohesion As it defines a single responsibility for product validation	Yes as it has a single responsibility of defining validation contract	No need to change
20	CreateProductController	Đặng Văn Nhân	Functional Cohesion As all methods are related to product creation and validation	No: 1. It handles both validation and creation logic 2. It contains error display functionality which could be separated	Split into separate classes: - ProductValidator: Handle all validation logic - ProductCreator: Handle product creation and database operations - ErrorHandler: Handle error display and logging
21	PlaceRushOrderController	Nguyễn Lan Nhi	Functional Cohesion - All methods (checkValidAddressAndAvailableItem, submitForm, validateForm, calculateShippingFee) work together toward the goal of handling rush orders.	Yes – This class only handles rush order requests and coordinates actions such as address check, product availability, shipping fee calculation, and form validation.	No change needed.

#	Class Name	PIC	Cohesion	Single Responsibility? Yes or No? Why?	Solution
22	RushOrderForm	Nguyễn Lan Nhi	Functional Cohesion – All attributes (deliveryTime, deliveryInstruction) and methods (submitForm()) support a single purpose: capturing and submitting rush order delivery information.	Yes – This class is responsible only for the rush order form and its related data handling.	No change needed.
23	DeliveryInfo	Nguyễn Lan Nhi	Communicational Cohesion – All components focus on storing and retrieving delivery-related data (name, address, province, email, phone number, etc.).	Yes – This class only handles storage and access of delivery information.	No change needed.
24	OrderManagementController	Nguyễn Lan Nhi	Functional Cohesion – Methods (approveOrder, sendApprovedEmail, sendRejectedEmail) serve the single purpose of processing and communicating order approval decisions.	Yes – This class only coordinates the logic for managing orders and sending relevant notifications.	No change needed.
25	Order	Nguyễn Lan Nhi	Functional Cohesion – Contains related data (orderID, status, mediaList) and methods (updateOrderStatusToApproved, updateOrderStatusToRejected) tied to order state.	Yes – This class manages the state of an business order only (status, items, date, etc)	No change needed.
26	RushOrderService	Nguyễn Lan Nhi	Functional Cohesion – All methods focus on determining rush order eligibility logically.	Yes – This class only handles eligibility checking logic, no unrelated concerns like payment or UI.	Consider extracting address check logic to a LocationValidator if reused.
27	OrderDetailScreen	Nguyễn Lan Nhi	Logical Cohesion – UI class containing approval and reject logic may mix responsibilities (display + trigger logic).	No – If the UI directly calls approval logic, it breaks SRP and tightly couples UI with business logic.	Separate UI from logic: keep OrderDetailScreen for presentation only, and delegate logic to OrderManagementController.
28	ViewProductDetailsService	Trần Cao Phong	Procedural Cohesion (original) → Functional Cohesion (after refactor	Originally No, but now split across dedicated classes	Moved validation to ProductValidator and data persistence to ProductRepository. Service now only orchestrates, aligning with SRP and higher cohesion.
29	ProductRepository	Trần Cao Phong	Functional Cohesion – manages only database operations for products	Yes – encapsulates product persistence logic	Abstracts data access, enabling cleaner separation of concerns from business logic.
30	ProductValidatorFactory	Trần Cao Phong	Functional Cohesion – manages only database operations for products	Yes (with minor caveat) – single responsibility to return correct validator	Could be improved using polymorphism (e.g., factory method pattern within product subclasses), but acceptable in this context.
31					
32					
33					
34					