# Integration Testing

The integration testing process aims to combine various modules of the system to ensure they function together seamlessly. We are following a hybrid approach for integration testing, that combines both top-down and bottom-up strategies.

This approach ensures that each module seamlessly integrates into the larger system, thereby guaranteeing smooth functionality of the entire system.

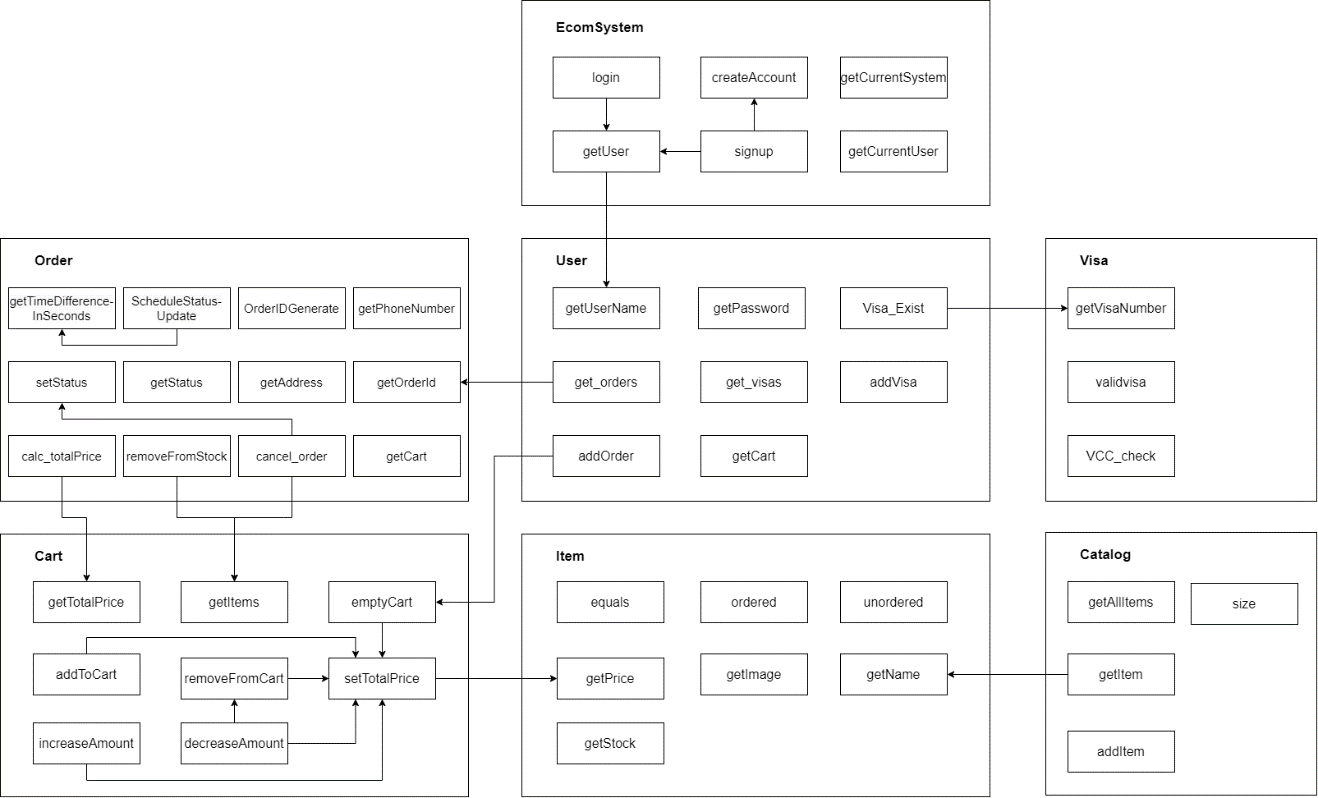
## Modules

This figure shows the structure of the system and how each class uses other classes. (e.g. E-commerce uses User)

A diagram of a product

Description automatically generated

The next figure shows how each module how each module depends on other modules. (e.g. getUser function in EcomSystem class depends on getUserName in User class)



### EcomSystem module

- Initializes user accounts and manages user-related functionalities.

- Dependencies: **User**.

### User module

- Represents a user in the system, handling user data and actions.

- Dependencies: **Order, Visa, Cart**.

### Visa module

- Handles payment information for users.

- Dependencies: None.

### Order module

- Manages orders placed by users on their carts.

- Dependencies: **Cart**.

### Cart module

- Stores items added by users for purchase.

- Dependencies: **Item**.

### Catalog module

- Stores and manages items.

- Dependencies: **Item**.

### Item module

- Represents individual items available for purchase.

- Dependencies: None.

## Implementation History

1. Implement **User** and **Order** Class
2. Implement User Controllers: SignupController and LoginController.
3. Implement Order Controllers: OrderListController, OrderRequestController, and OrderManagementController.
4. Implement **Visa** Class
5. Implement Visa Controllers: PaymentController and PaymentExistingCardController.
6. Implement **Item** Class
7. Implement **Catalog** Class
8. Implement ItemController
9. Implement CatalogController
10. Implement **Cart** Class
11. Implement CartController
12. Implement **EcomSystem** Class

## Top-down

* Most high-level module (**EcomSystem**) is implemented at the end.
* **User** is implemented before **Visa**, and **Cart**.
* **Order** uses **Cart**. **Order** is implemented first.

## Bottom-up

* Controllers are implemented after their classes.
* **Catalog** uses **Item**. **Item** is implemented first.
* **Item** is implemented before **Cart**.

## Integration Testing Approach

Integration testing ensures that the integrated modules work together correctly. conducting the following tests:

**Test the interfaces of each module with its controllers.**

Example: confirm that SignupController interacts with the signup() method.

**Test the functionality of the integrated modules, ensuring they perform as expected.**

Example: test if adding items to the cart reflects in the user's cart and updates the total price correctly.

**Test the interaction between dependent modules, ensuring seamless integration.**

Example: Test if the *Order* class correctly uses the *Cart* and *Item* classes to place orders.

## Integration Testing Result

The integration process successfully combined the e-commerce system modules. All modules of the system are integrated correctly and function smoothly together.

By employing a hybrid approach, we ensured that all modules of the system are integrated correctly and function smoothly together. That it allowed us to identify and address issues at both the component and system levels, ensuring a robust and reliable system.

Integration testing will further validate that the system functions reliably and meets the required specifications.