Final projects

Project 1: Xây dựng hệ thống quản lý đặt hàng online

1. Giới thiệu bài toán

Bạn được giao nhiệm vụ xây dựng một hệ thống quản lý đặt hàng cho một cửa hàng trực tuyến. Hệ thống cần hỗ trợ các chức năng như quản lý sản phẩm, xử lý đơn hàng, và thực hiện thanh toán. Để đảm bảo tính linh hoạt và mở rộng của hệ thống, bạn cần áp dụng các mẫu thiết kế phần mềm vào giải pháp của mình.

2. Yêu cầu thiết kế và cài đặt

Hệ thống hỗ trợ các chức năng sau:

- Quản lý sản phẩm: Hệ thống cần có khả năng thêm, sửa, xóa, và liệt kê các sản phẩm.
- Xử lý đơn hàng: Hệ thống phải cho phép người dùng tạo đơn hàng, thêm sản phẩm vào đơn hàng, và theo dõi trạng thái đơn hàng.
- Thanh toán: Hệ thống cần hỗ trợ nhiều phương thức thanh toán (ví dụ: thẻ tín dụng, PayPal) và xử lý thanh toán cho đơn hàng.

Mẫu thiết kế cần áp dụng:

- Singleton: Đảm bảo rằng hệ thống chỉ có một instance của lớp quản lý kết nối cơ sở dữ liệu.
- Factory Method: Tạo các đối tượng sản phẩm khác nhau (ví dụ: điện tử, thời trang) mà không cần phải biết lớp cụ thể.
- Abstract Factory: Tạo các đối tượng giao diện thanh toán khác nhau (ví dụ: thẻ tín dụng, PayPal) để hỗ trợ nhiều phương thức thanh toán.
- Observer: Cập nhật trạng thái đơn hàng cho người dùng và hệ thống khi có sự thay đổi.
- Strategy: Chọn phương pháp thanh toán phù hợp cho từng đơn hàng.
- Decorator: Thêm các tính năng bổ sung vào đơn hàng (ví dụ: quà tặng, đóng gói đặc biệt).

3. Gợi ý hướng dẫn giải quyết vấn đề

- Singleton: Tạo một lớp `DatabaseConnection` với phương thức `getInstance()` để đảm bảo chỉ có một kết nối cơ sở dữ liệu duy nhất.
- Factory Method: Tạo một lớp `ProductFactory` với phương thức `createProduct()` trả về các loại sản phẩm khác nhau dựa trên loại sản phẩm yêu cầu.
- Abstract Factory: Tạo một giao diện `PaymentFactory` và các lớp cài đặt cụ thể cho từng phương thức thanh toán. Sử dụng một lớp `PaymentFactoryProvider` để chọn và tạo factory phù hợp.
- Observer: Xây dựng một hệ thống theo dõi đơn hàng với giao diện `Observer` và lớp `Order` thực hiện việc thông báo cho các đối tượng quan tâm khi trạng thái đơn hàng thay đổi.

- Strategy: Tạo một giao diện `PaymentStrategy` với phương thức `pay()` và các lớp cài đặt cụ thể cho từng phương thức thanh toán. Sử dụng lớp `Order` để áp dụng chiến lược thanh toán thích hợp.
- Decorator: Tạo một lớp `OrderDecorator` và các lớp kế thừa để thêm các tính năng bổ sung vào đơn hàng.

4. Kết quả cần đạt

- Hệ thống phải cho phép người dùng quản lý sản phẩm, tạo đơn hàng, và thực hiện thanh toán bằng các phương thức khác nhau.
- Hệ thống phải áp dụng đúng các mẫu thiết kế để đảm bảo tính linh hoạt và mở rộng.
- Đảm bảo rằng mã nguồn được tổ chức rõ ràng, dễ hiểu và có thể mở rộng trong tương lai.

5. Hướng phát triển mở rộng bài toán

- Tích hợp với hệ thống quản lý kho: Thêm chức năng quản lý và đồng bộ kho hàng với hệ thống.
- Phân tích dữ liệu và báo cáo: Cung cấp các báo cáo về doanh thu, sản phẩm bán chạy, và phân tích đơn hàng.
- Chức năng khuyến mãi và giảm giá: Tạo hệ thống để quản lý mã giảm giá và chương trình khuyến mãi.

6. Hướng dẫn đánh giá

- Đúng đắn về thiết kế: Đảm bảo các mẫu thiết kế được áp dụng đúng cách và hệ thống hoạt động theo yêu cầu.
- Chất lượng mã nguồn: Kiểm tra mã nguồn có sạch, dễ hiểu, và tuân thủ các nguyên tắc lập trình tốt.
- Tính năng và khả năng mở rộng: Đánh giá khả năng mở rộng và dễ dàng bảo trì của hệ thống.
- Kiểm tra và debug: Kiểm tra tính chính xác của hệ thống và khả năng xử lý lỗi.

Project 2: Xây dựng hệ thống quản lý thư viện

Bài tập này giúp người học thực hành việc áp dụng các mẫu thiết kế phần mềm trong một bài toán quản lý thư viện thực tế và cải thiện khả năng lập trình và thiết kế phần mềm của họ.

1. Giới thiệu bài toán

Bạn được giao nhiệm vụ phát triển một hệ thống quản lý thư viện. Hệ thống này cần hỗ trợ quản lý sách, thành viên, và các giao dịch mượn trả sách. Bạn cũng cần phải áp dụng các mẫu thiết kế phần mềm để giải quyết bài toán này.

2. Yêu cầu thiết kế và cài đặt

Hệ thống phải hỗ trợ các chức năng sau:

- Thêm, xóa, và cập nhật thông tin sách.
- Đăng ký, xóa, và cập nhật thông tin thành viên.
- Mượn và trả sách.
- Quản lý hạn trả sách và phí quá hạn.
- Cung cấp báo cáo thống kê về sách và thành viên.

Yêu cầu về mẫu thiết kế:

- Singleton: Đảm bảo rằng hệ thống chỉ có một phiên bản duy nhất của quản lý sách và quản lý thành viên.
- Factory Method: Tạo các loại sách khác nhau (ví dụ: sách giấy, sách điện tử) mà không cần chỉ định lớp sách cụ thể.
- Abstract Factory: Cung cấp các nhà máy sản xuất khác nhau cho các loại sách (ví dụ: nhà máy sản xuất sách học thuật và sách giải trí).
- Observer: Cập nhật giao diện người dùng khi có sự thay đổi trong thông tin sách hoặc thành viên.
- Decorator: Thêm các tính năng bổ sung vào sách (ví dụ: đánh dấu sách yêu thích) mà không làm thay đổi cấu trúc sách gốc.
- Strategy: Cung cấp các phương pháp tính phí quá hạn khác nhau và cho phép người dùng chọn phương pháp tính phí.

3. Gợi ý hướng dẫn giải quyết vấn đề

- Thiết kế lớp: Xác định các lớp chính của hệ thống như `Book`, `Member`, `Loan`, `FeeCalculator`, và các lớp liên quan khác.
- Áp dụng Singleton: Tạo lớp `BookManager` và `MemberManager` để đảm bảo chỉ có một phiên bản duy nhất của các lớp này trong hệ thống.
- Factory Method: Tạo lớp `BookFactory` với phương thức tạo sách tùy thuộc vào loại sách.
- Abstract Factory: Tạo các lớp `AcademicBookFactory` và `EntertainmentBookFactory` kế thừa từ `AbstractFactory` để sản xuất sách học thuật và sách giải trí.
- Observer: Cài đặt một hệ thống thông báo để cập nhật thông tin sách và thành viên khi có sự thay đổi.
- Decorator: Tạo lớp `FavoriteBookDecorator` để thêm tính năng đánh dấu sách yêu thích cho sách.
- Strategy: Tạo các lớp `LateFeeCalculator` cho các phương pháp tính phí quá hạn khác nhau và một lớp `FeeContext` để chọn phương pháp tính phí.

4. Kết quả cần đạt

- Chức năng cơ bản: Hệ thống phải hoạt động đúng theo yêu cầu, với các chức năng như thêm, xóa, cập nhật sách và thành viên, mượn và trả sách.
- Áp dụng mẫu thiết kế: Các mẫu thiết kế được áp dụng đúng cách và có thể mở rộng dễ dàng.
- Tính năng mở rộng: Hệ thống có khả năng mở rộng và bảo trì dễ dàng, ví dụ như thêm loại sách mới hoặc phương pháp tính phí mới.

5. Hướng phát triển mở rộng bài toán

- Tích hợp với hệ thống quản lý thư viện trực tuyến: Kết nối với các dịch vụ quản lý thư viện trực tuyến để đồng bộ dữ liệu.
- Giao diện người dùng: Xây dựng giao diện người dùng thân thiện và hỗ trợ trên các thiết bị di động.
- Tính năng tìm kiếm nâng cao: Thêm chức năng tìm kiếm nâng cao cho sách và thành viên.
- Quản lý tồn kho: Thêm chức năng quản lý tồn kho sách và thông báo khi sách gần hết hàng.

6. Hướng dẫn đánh giá

- Đánh giá chức năng: Kiểm tra các chức năng của hệ thống để đảm bảo chúng hoạt động đúng như yêu cầu.
- Đánh giá áp dụng mẫu thiết kế: Xem xét cách người học áp dụng các mẫu thiết kế để đảm bảo chúng được sử dụng hiệu quả và đúng cách.
- Đánh giá mã nguồn: Đánh giá chất lượng mã nguồn, bao gồm cấu trúc, tổ chức, và khả năng đọc hiểu của mã.
- Đánh giá khả năng mở rộng: Kiểm tra khả năng mở rộng của hệ thống và tính dễ bảo trì của mã nguồn.

Project 3: Xây dựng hệ thống quản lý bệnh viện

Bài tập này sẽ giúp người học làm quen với việc áp dụng các mẫu thiết kế trong một tình huống thực tế và cải thiện khả năng lập trình và thiết kế phần mềm của họ.

1. Giới thiệu bài toán

Bạn được giao nhiệm vụ phát triển một hệ thống quản lý bệnh viện để giúp quản lý thông tin bệnh nhân, bác sĩ, lịch hẹn, và hóa đơn thanh toán. Hệ thống này cần phải hỗ trợ việc đăng ký bệnh nhân, tạo và quản lý lịch hẹn, theo dõi lịch sử bệnh nhân và phát hành hóa đơn thanh toán. Hãy áp dụng các mẫu thiết kế phần mềm để giải quyết các yêu cầu sau:

2. Yêu cầu thiết kế và cài đặt

Hệ thống phải hỗ trợ các chức năng sau:

- Đăng ký và quản lý thông tin bệnh nhân.
- Đăng ký và quản lý thông tin bác sĩ.
- Tạo và quản lý lịch hẹn của bệnh nhân với bác sĩ.
- Theo dõi và cập nhật lịch sử bệnh án của bệnh nhân.
- Tạo và quản lý hóa đơn thanh toán cho các dịch vụ y tế.

Yêu cầu về mẫu thiết kế:

- Singleton: Đảm bảo rằng hệ thống chỉ có một phiên bản duy nhất của quản lý lịch hẹn và hóa đơn thanh toán.
- Factory Method: Tạo ra các loại dịch vụ y tế khác nhau (ví dụ: khám bệnh, xét nghiệm) mà không cần chỉ định lớp dịch vụ cụ thể.

- Abstract Factory: Cung cấp các nhà máy dịch vụ khác nhau cho các loại dịch vụ y tế (ví dụ: nhà máy khám bệnh, nhà máy xét nghiệm).
- Observer: Cập nhật giao diện người dùng khi có thay đổi trong lịch hẹn hoặc trạng thái bệnh nhân.
- Decorator: Thêm các tính năng bổ sung vào hóa đơn (ví dụ: giảm giá cho các dịch vụ) mà không làm thay đổi cấu trúc hóa đơn gốc.
- Strategy: Cung cấp các phương pháp thanh toán khác nhau (ví dụ: thanh toán bằng thẻ tín dụng, thanh toán qua bảo hiểm) và cho phép người dùng chọn phương pháp thanh toán.

3. Gợi ý hướng dẫn giải quyết vấn đề

- Thiết kế lớp: Xác định các lớp chính của hệ thống như `Patient`, `Doctor`, `Appointment`, `MedicalRecord`, `Invoice`, và các lớp liên quan khác.
- Áp dụng Singleton: Tạo lớp `AppointmentManager` và `InvoiceManager` để đảm bảo chỉ có một phiên bản duy nhất của các lớp này trong hệ thống.
- Factory Method: Tạo một lớp `MedicalServiceFactory` với phương thức tạo dịch vụ y tế tùy thuộc vào loại dịch vụ.
- Abstract Factory: Tạo các lớp `ConsultationFactory` và `TestFactory` kế thừa từ `AbstractFactory` để cung cấp dịch vụ khám bệnh và xét nghiệm.
- Observer: Cài đặt một hệ thống thông báo để cập nhật giao diện người dùng khi có sự thay đổi trong lịch hẹn hoặc thông tin bệnh nhân.
- Decorator: Tạo lớp `DiscountDecorator` để thêm tính năng giảm giá cho hóa đơn mà không làm thay đổi cấu trúc hóa đơn gốc.
- Strategy: Tạo các lớp `CreditCardPayment`, `InsurancePayment` và một lớp `PaymentContext` để chọn phương pháp thanh toán.

4. Kết quả cần đạt

- Chức năng cơ bản: Hệ thống phải hoạt động đúng theo yêu cầu, với các chức năng như đăng ký bệnh nhân, tạo và quản lý lịch hẹn, và thanh toán hóa đơn.
- Áp dụng mẫu thiết kế: Các mẫu thiết kế được áp dụng đúng cách và có thể mở rộng dễ dàng.
- Tính năng mở rộng: Hệ thống có khả năng mở rộng và bảo trì dễ dàng, ví dụ như thêm các dịch vụ y tế mới hoặc phương pháp thanh toán mới.

5. Hướng phát triển mở rộng bài toán

- Tích hợp với hệ thống y tế điện tử: Kết nối với hệ thống hồ sơ y tế điện tử của bệnh viện hoặc phòng khám.
- Giao diện người dùng: Xây dựng giao diện người dùng thân thiện và hỗ trợ trên các thiết bị di động.
- Tính năng nhắc nhở: Thêm tính năng nhắc nhở tự động qua email hoặc SMS cho các lịch hẹn sắp tới.
- Quản lý thuốc: Thêm chức năng quản lý thông tin thuốc và đơn thuốc cho bệnh nhân.

6. Hướng dẫn đánh giá

- Đánh giá chức năng: Kiểm tra các chức năng của hệ thống để đảm bảo chúng hoạt động đúng như yêu cầu.
- Đánh giá áp dụng mẫu thiết kế: Xem xét cách người học áp dụng các mẫu thiết kế để đảm bảo chúng được sử dụng hiệu quả và đúng cách.
- Đánh giá mã nguồn: Đánh giá chất lượng mã nguồn, bao gồm cấu trúc, tổ chức, và khả năng đọc hiểu của mã.
- Đánh giá khả năng mở rộng: Kiểm tra khả năng mở rộng của hệ thống và tính dễ bảo trì của mã nguồn.

Project 4: Xây dựng hệ thống quản lý đặt phòng khách sạn

Bài tập này sẽ giúp người học làm quen với việc áp dụng các mẫu thiết kế trong một tình huống thực tế và cải thiện khả năng lập trình và thiết kế phần mềm của họ.

1. Giới thiệu bài toán

Bạn được yêu cầu phát triển một hệ thống quản lý đặt phòng cho một khách sạn. Hệ thống này sẽ cho phép người dùng kiểm tra tình trạng phòng, đặt phòng, và thanh toán. Hệ thống cũng cần hỗ trợ quản lý thông tin phòng, khách hàng, và các đơn đặt phòng. Hãy áp dụng các mẫu thiết kế phần mềm để giải quyết các yêu cầu sau:

2. Yêu cầu thiết kế và cài đặt

Hệ thống phải hỗ trợ các chức năng sau:

- Xem danh sách phòng trống và tình trạng phòng.
- Đặt phòng và hủy đặt phòng.
- Quản lý thông tin khách hàng.
- Theo dõi lịch sử đặt phòng và thanh toán.
- Tạo và quản lý hóa đơn thanh toán cho các đặt phòng.

Yêu cầu về mẫu thiết kế:

- Singleton: Đảm bảo rằng hệ thống chỉ có một phiên bản duy nhất của quản lý đặt phòng và hóa đơn thanh toán.
- Factory Method: Tạo ra các loại phòng khác nhau (ví dụ: phòng đơn, phòng đôi, phòng suite) mà không cần chỉ định lớp phòng cụ thể.
- Abstract Factory: Cung cấp các nhà máy phòng khác nhau cho các loại phòng (ví dụ: nhà máy phòng đơn, nhà máy phòng suite).
- Observer: Cập nhật giao diện người dùng khi có thay đổi trong tình trạng phòng hoặc đơn đặt phòng.
- Decorator: Thêm các dịch vụ bổ sung cho phòng (ví dụ: dịch vụ phòng, minibar) mà không làm thay đổi cấu trúc phòng gốc.
- Strategy: Cung cấp các phương pháp thanh toán khác nhau (ví dụ: thanh toán bằng thẻ tín dụng, thanh toán qua chuyển khoản) và cho phép người dùng chọn phương pháp thanh toán.

3. Gợi ý hướng dẫn giải quyết vấn đề

- Thiết kế lớp: Xác định các lớp chính của hệ thống như `Room`, `Booking`, `Customer`, `Invoice`, và các lớp liên quan khác.
- Áp dụng Singleton: Tạo lớp `BookingManager` và `InvoiceManager` để đảm bảo chỉ có một phiên bản duy nhất của các lớp này trong hệ thống.
- Factory Method: Tạo một lớp `RoomFactory` với phương thức tạo phòng tùy thuộc vào loại phòng.
- Abstract Factory: Tạo các lớp `SingleRoomFactory` và `SuiteRoomFactory` kế thừa từ `AbstractFactory` để cung cấp các loại phòng khác nhau.
- Observer: Cài đặt một hệ thống thông báo để cập nhật giao diện người dùng khi có sự thay đổi trong tình trạng phòng hoặc đơn đặt phòng.
- Decorator: Tạo lớp `ServiceDecorator` để thêm các dịch vụ bổ sung cho phòng mà không làm thay đổi cấu trúc phòng gốc.
- Strategy: Tạo các lớp `CreditCardPayment`, `BankTransferPayment` và một lớp `PaymentContext` để chọn phương pháp thanh toán.

4. Kết quả cần đạt

- Chức năng cơ bản: Hệ thống phải hoạt động đúng theo yêu cầu, với các chức năng như xem tình trạng phòng, đặt phòng, và thanh toán.
- Áp dụng mẫu thiết kế: Các mẫu thiết kế được áp dụng đúng cách và có thể mở rộng dễ dàng.
- Tính năng mở rộng: Hệ thống có khả năng mở rộng và bảo trì dễ dàng, ví dụ như thêm các loại phòng mới hoặc phương pháp thanh toán mới.

5. Hướng phát triển mở rộng bài toán

- Tích hợp với hệ thống quản lý khách sạn lớn hơn: Kết nối với các hệ thống quản lý khách sạn khác để đồng bộ hóa dữ liệu đặt phòng.
- Giao diện người dùng: Xây dựng giao diện người dùng thân thiện và hỗ trợ trên các thiết bị di động.
- Tính năng ưu đãi: Thêm tính năng ưu đãi và khuyến mãi cho các đặt phòng.
- Quản lý dịch vụ bổ sung: Thêm chức năng quản lý và đặt dịch vụ bổ sung như tour du lịch hoặc dịch vụ spa.

6. Hướng dẫn đánh giá

- Đánh giá chức năng: Kiểm tra các chức năng của hệ thống để đảm bảo chúng hoạt động đúng như yêu cầu.
- Đánh giá áp dụng mẫu thiết kế: Xem xét cách người học áp dụng các mẫu thiết kế để đảm bảo chúng được sử dụng hiệu quả và đúng cách.
- Đánh giá mã nguồn: Đánh giá chất lượng mã nguồn, bao gồm cấu trúc, tổ chức, và khả năng đọc hiểu của mã.
- Đánh giá khả năng mở rộng: Kiểm tra khả năng mở rộng của hệ thống và tính dễ bảo trì của mã nguồn.

Project 5: Xây dựng hệ thống quản lý thực đơn nhà hàng

1. Giới thiệu bài toán

Bạn được yêu cầu phát triển một hệ thống quản lý thực đơn cho một nhà hàng. Hệ thống này sẽ giúp quản lý thông tin món ăn, đặt món, và thanh toán hóa đơn. Hệ thống cũng cần hỗ trợ quản lý thông tin nguyên liệu và các khuyến mãi. Hãy áp dụng các mẫu thiết kế phần mềm để giải quyết các yêu cầu sau:

2. Yêu cầu thiết kế và cài đặt

Hệ thống phải hỗ trợ các chức năng sau:

- Xem danh sách món ăn và thông tin chi tiết của chúng.
- Thêm, sửa, và xóa món ăn khỏi thực đơn.
- Đặt món và thanh toán hóa đơn.
- Quản lý thông tin nguyên liệu và các nhà cung cấp.
- Cung cấp các khuyến mãi và ưu đãi đặc biệt.

Yêu cầu về mẫu thiết kế:

- Singleton: Đảm bảo rằng hệ thống chỉ có một phiên bản duy nhất của quản lý đơn hàng và hóa đơn thanh toán.
- Factory Method: Tạo ra các loại món ăn khác nhau (ví dụ: món chính, món tráng miệng, đồ uống) mà không cần chỉ định lớp món ăn cụ thể.
- Abstract Factory: Cung cấp các nhà máy món ăn khác nhau cho các loại món ăn (ví dụ: nhà máy món chính, nhà máy món tráng miệng).
- Observer: Cập nhật giao diện người dùng khi có thay đổi trong thông tin món ăn hoặc đơn hàng.
- Decorator: Thêm các tùy chọn bổ sung cho món ăn (ví dụ: thêm gia vị, kích cỡ phần ăn) mà không làm thay đổi cấu trúc món ăn gốc.
- Strategy: Cung cấp các phương pháp thanh toán khác nhau (ví dụ: thanh toán bằng thẻ tín dụng, thanh toán qua ví điện tử) và cho phép người dùng chọn phương pháp thanh toán.

3. Gợi ý hướng dẫn giải quyết vấn đề

- Thiết kế lớp: Xác định các lớp chính của hệ thống như `Dish`, `Order`, `Invoice`, `Ingredient`, `Promotion`, và các lớp liên quan khác.
- Áp dụng Singleton: Tạo lớp `OrderManager` và `InvoiceManager` để đảm bảo chỉ có một phiên bản duy nhất của các lớp này trong hệ thống.
- Factory Method: Tạo một lớp `DishFactory` với phương thức tạo món ăn tùy thuộc vào loại món ăn.
- Abstract Factory: Tạo các lớp `MainCourseFactory` và `DessertFactory` kế thừa từ `AbstractFactory` để cung cấp các loại món ăn khác nhau.
- Observer: Cài đặt một hệ thống thông báo để cập nhật giao diện người dùng khi có sự thay đổi trong thông tin món ăn hoặc đơn hàng.
- Decorator: Tạo lớp `AddonDecorator` để thêm các tùy chọn bổ sung cho món ăn mà không làm thay đổi cấu trúc món ăn gốc.

- Strategy: Tạo các lớp `CreditCardPayment`, `EWalletPayment` và một lớp `PaymentContext` để chọn phương pháp thanh toán.

4. Kết quả cần đạt

- Chức năng cơ bản: Hệ thống phải hoạt động đúng theo yêu cầu, với các chức năng như xem thực đơn, đặt món, và thanh toán hóa đơn.
- Áp dụng mẫu thiết kế: Các mẫu thiết kế được áp dụng đúng cách và có thể mở rộng dễ dàng.
- Tính năng mở rộng: Hệ thống có khả năng mở rộng và bảo trì dễ dàng, ví dụ như thêm loại món ăn mới hoặc phương pháp thanh toán mới.

5. Hướng phát triển mở rộng bài toán

- Tích hợp với hệ thống quản lý kho: Kết nối với hệ thống quản lý kho để đồng bộ hóa thông tin nguyên liệu và nhà cung cấp.
- Giao diện người dùng: Xây dựng giao diện người dùng thân thiện và hỗ trợ trên các thiết bị di động.
- Tính năng đặt món trước: Thêm tính năng đặt món trước cho khách hàng và thông báo khi món ăn sẵn sàng.
- Quản lý khuyến mãi: Thêm chức năng quản lý và áp dụng khuyến mãi cho các đơn hàng.

6. Hướng dẫn đánh giá

- Đánh giá chức năng: Kiểm tra các chức năng của hệ thống để đảm bảo chúng hoạt động đúng như yêu cầu.
- Đánh giá áp dụng mẫu thiết kế: Xem xét cách người học áp dụng các mẫu thiết kế để đảm bảo chúng được sử dụng hiệu quả và đúng cách.
- Đánh giá mã nguồn: Đánh giá chất lượng mã nguồn, bao gồm cấu trúc, tổ chức, và khả năng đọc hiểu của mã.
- Đánh giá khả năng mở rộng: Kiểm tra khả năng mở rộng của hệ thống và tính dễ bảo trì của mã nguồn.