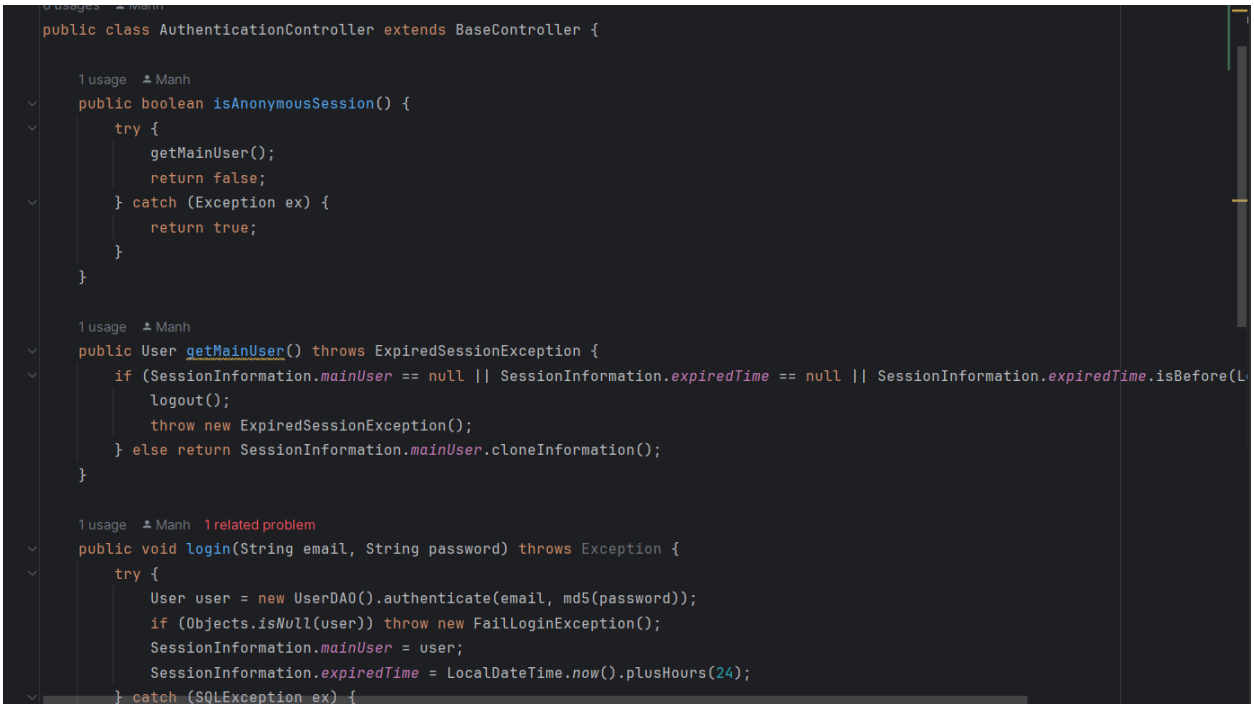
Phát hiện vi phạm LSP/ISP/DIP trong codebase

- **AuthenticationController:**

LSP: class kế thừa từ BaseController nhưng không tuân theo hành vi được mong đợi từ lớp cha. BaseController liên quan tới xử lý giỏ hàng trong khi class này tập trung vào xác thực người dùng và quản lý phiên.



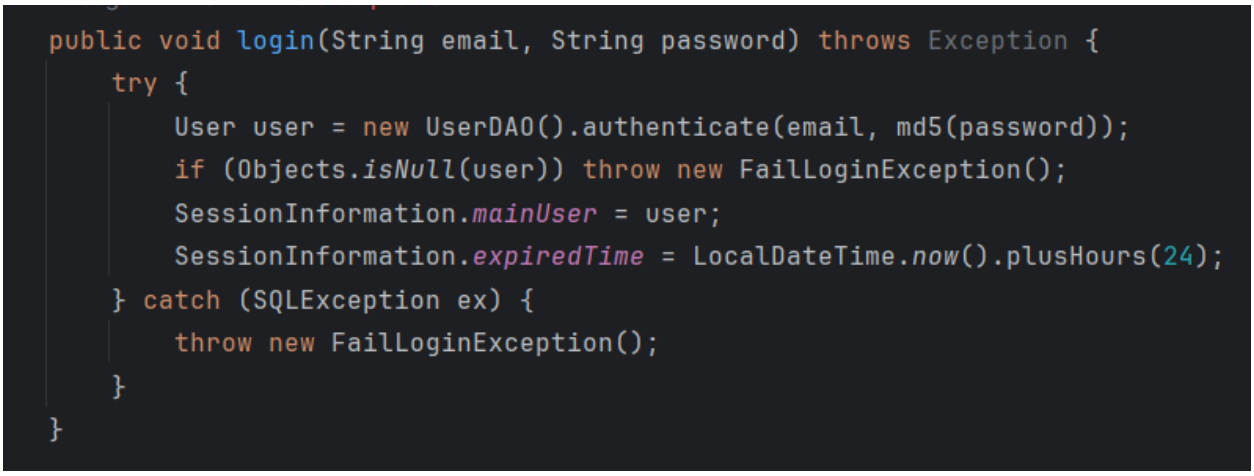
```
public class AuthenticationController extends BaseController {

    1 usage  Manh
    public boolean isAnonymousSession() {
        try {
            getMainUser();
            return false;
        } catch (Exception ex) {
            return true;
        }
    }

    1 usage  Manh
    public User getMainUser() throws ExpiredSessionException {
        if (SessionInformation.mainUser == null || SessionInformation.expiredTime == null || SessionInformation.expiredTime.isBefore(L
            logout();
            throw new ExpiredSessionException();
        } else return SessionInformation.mainUser.cloneInformation();
    }

    1 usage  Manh  1 related problem
    public void login(String email, String password) throws Exception {
        try {
            User user = new UserDAO().authenticate(email, md5(password));
            if (Objects.isNull(user)) throw new FailLoginException();
            SessionInformation.mainUser = user;
            SessionInformation.expiredTime = LocalDateTime.now().plusHours(24);
        } catch (SQLException ex) {
```

DIP: class này phụ thuộc trực tiếp vào class UserDAO qua việc gọi trực tiếp phương thức authenticate của lớp UserDAO



```
public void login(String email, String password) throws Exception {
    try {
        User user = new UserDAO().authenticate(email, md5(password));
        if (Objects.isNull(user)) throw new FailLoginException();
        SessionInformation.mainUser = user;
        SessionInformation.expiredTime = LocalDateTime.now().plusHours(24);
    } catch (SQLException ex) {
        throw new FailLoginException();
    }
}
```

- **HomeController:**

LSP: class kế thừa từ BaseController nhưng không tuân theo hành vi được mong đợi từ lớp cha**.**

DIP: class này phụ thuộc trực tiếp vào class MediaDAO qua vệc gọi trực tiếp phương thức getAllMedia của lớp UserDAO

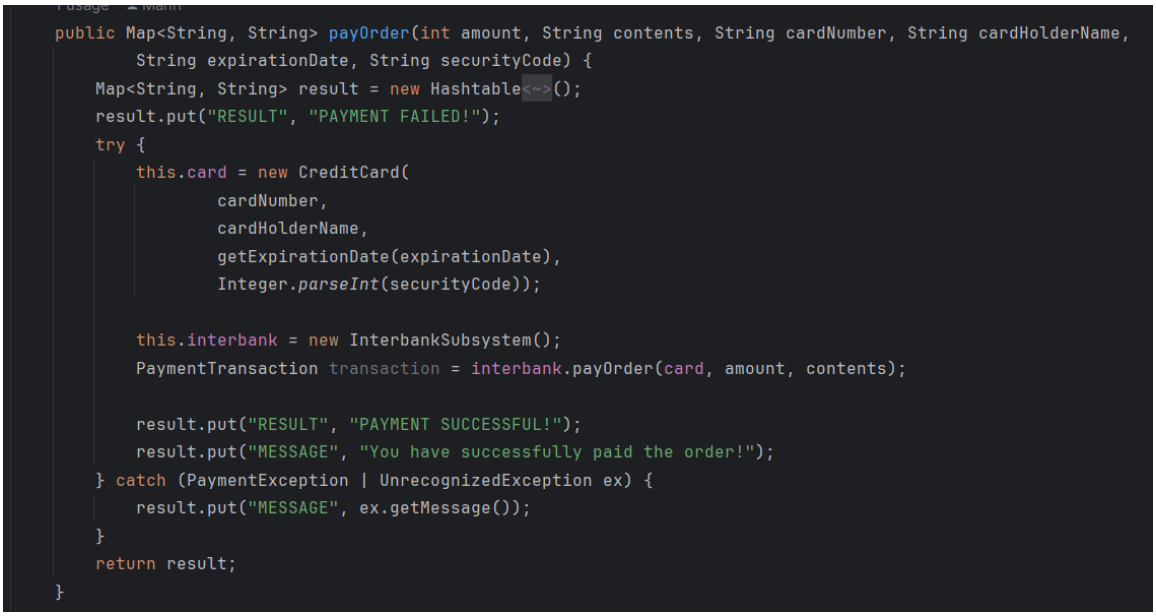
```
public class HomeController extends BaseController {


    /**
     * this method gets all Media in DB and return back to home to display
     * @return List[Media]
     * @throws SQLException
     */
    1 usage   Manh
    public static List getAllMedia() throws SQLException{
        return new MediaDAO().getAllMedia();
    }
}
```

- **PaymentController:**

LSP: class kế thừa từ BaseController nhưng không tuân theo hành vi được mong đợi từ lớp cha.

DIP: class này phụ thuộc trực tiếp vào class CreditCard



```
public Map<String, String> payOrder(int amount, String contents, String cardNumber, String cardHolderName,
        String expirationDate, String securityCode) {
    Map<String, String> result = new Hashtable<~>();
    result.put("RESULT", "PAYMENT FAILED!");
    try {
        this.card = new CreditCard(
                cardNumber,
                cardHolderName,
                getExpirationDate(expirationDate),
                Integer.parseInt(securityCode));

        this.interbank = new InterbankSubsystem();
        PaymentTransaction transaction = interbank.payOrder(card, amount, contents);

        result.put("RESULT", "PAYMENT SUCCESSFUL!");
        result.put("MESSAGE", "You have successfully paid the order!");
    } catch (PaymentException | UnrecognizedException ex) {
        result.put("MESSAGE", ex.getMessage());
    }
    return result;
}
```

- **PaymentTransaction:**

DIP: class này phụ thuộc trực tiếp vào class CreditCard

```java
public class PaymentTransaction {
    2 usages
    private String errorCode;
    1 usage
    private CreditCard card;
    1 usage
    private String transactionId;
    1 usage
    private String transactionContent;
    1 usage
    private int amount;
    1 usage
    private String createdAt;

    1 usage  Manh
    public PaymentTransaction(String errorCode, CreditCard card, String transactionId, String transactionContent,
                              int amount, String createdAt) {
        super();
        this.errorCode = errorCode;
        this.card = card;
        this.transactionId = transactionId;
        this.transactionContent = transactionContent;
        this.amount = amount;
        this.createdAt = createdAt;
    }

    1 usage  Manh
    public String getErrorCode() { return errorCode; }
}
```

- **InterbankPayloadConverter:**

DIP: class này phụ thuộc trực tiếp vào class CreditCard

```java
1 usage  Manh
PaymentTransaction extractPaymentTransaction(String responseText) {
    MyMap response = convertJSONResponse(responseText);

    if (response == null)
        return null;
    MyMap transaction = (MyMap) response.get("transaction");
    CreditCard card = new CreditCard(
            (String) transaction.get("cardCode"),
            (String) transaction.get("owner"),
            (String) transaction.get("dateExpired"),
            Integer.parseInt((String) transaction.get("cvvCode")));

    PaymentTransaction trans = new PaymentTransaction(
            (String) response.get("errorCode"),
            card,
            (String) transaction.get("transactionId"),
            (String) transaction.get("transactionContent"),
            Integer.parseInt((String) transaction.get("amount")),
            (String) transaction.get("createdAt"));

    switch (trans.getErrorCode()) {
```

– **InterbankSubsystemController:**

DIP: class này phụ thuộc trực tiếp vào class CreditCard

```
public class InterbankSubsystemController {

    2 usages
    private static InterbankPayloadConverter interbankPayloadConverter = new InterbankPayloadConverter();
    1 usage
    private static InterbankBoundary interbankBoundary = new InterbankBoundary();

    1 usage  Manh
    public PaymentTransaction refund(CreditCard card, int amount, String contents) { return null; }

    1 usage  Manh
    public PaymentTransaction payOrder(CreditCard card, int amount, String contents) {
        String requestPayload = interbankPayloadConverter.convertToRequestPayload(card, amount, contents);
        String responseText = interbankBoundary.query(InterbankConfigs.PROCESS_TRANSACTION_URL, requestPayload);
        return interbankPayloadConverter.extractPaymentTransaction(responseText);
    }

}
```

– **InterbankSubsystem:**

DIP: class này phụ thuộc trực tiếp vào class InterbankSubsystemController thay vì phụ thuộc vào trừu tượng qua việc gọi trực tiếp phương thức payOrder và refund của InterbankSubsystemController

```
    private InterbankSubsystemController ctrl;

    /**
     * Initializes a newly created {@code InterbankSubsystem} object so that it
     * represents an Interbank subsystem.
     */
    1 usage  Manh
    public InterbankSubsystem() { this.ctrl = new InterbankSubsystemController(); }

    /**
     * @see InterbankInterface#payOrder(CreditCard, int,
     *      String)
     */
    2 usages  Manh
    public PaymentTransaction payOrder(CreditCard card, int amount, String contents) {
        PaymentTransaction transaction = ctrl.payOrder(card, amount, contents);
        return transaction;
    }

    /**
     * @see InterbankInterface#refund(CreditCard, int,
     *      String)
     */
    1 usage  Manh
    public PaymentTransaction refund(CreditCard card, int amount, String contents) {
        PaymentTransaction transaction = ctrl.refund(card, amount, contents);
        return transaction;
    }
}
```