Movie Recommendation System

Presenter:
Vipra Gupta
Chaitra Ramachandra

Team Number: 32

Vision

To build a robust movie recommendation system that can recommend movies to its users by considering their movie movie preferences.

Project Description

- Web application that will recommend movies to its users by analysing their genre preferences and watch patterns.
- Provides an interface through which the users can login into the system, and build their movie preference profile.
- The application analyses the user’s movie watching patterns and the ratings they gave to them to recommend movies according to the his/her taste.

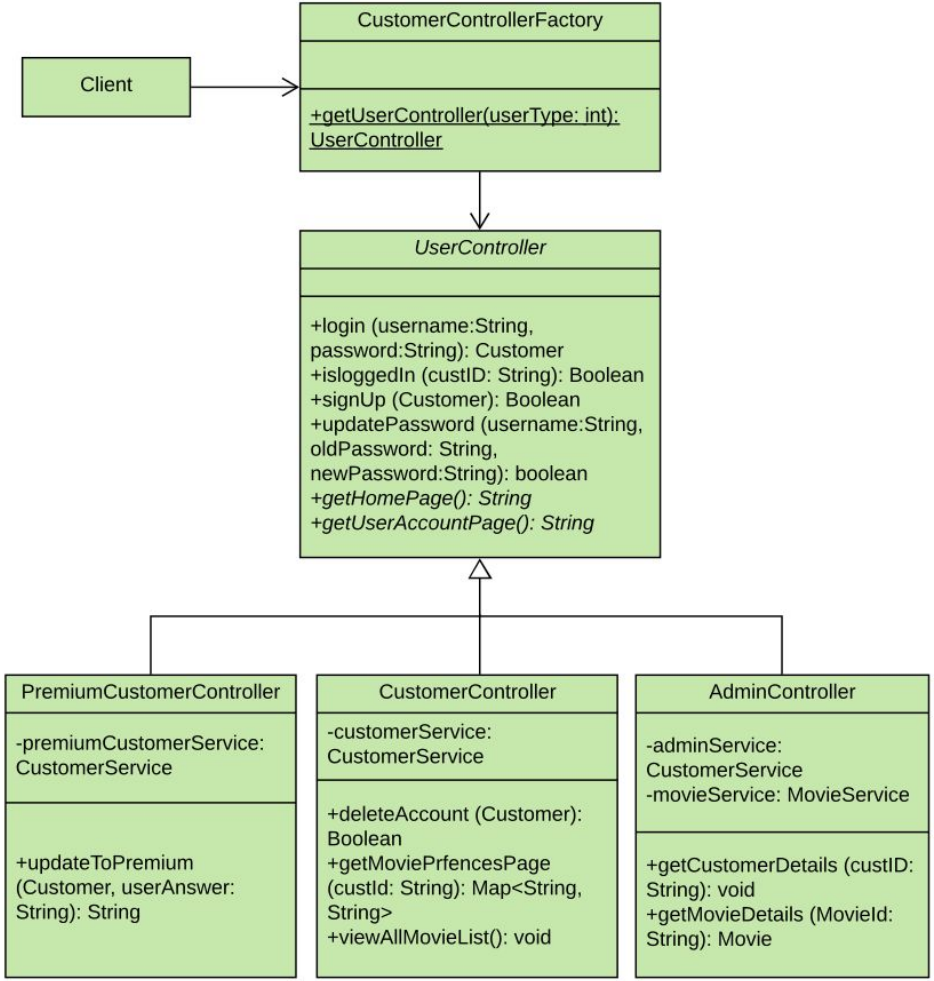
Factory Design Pattern

- When different objects based on context.
- Provides an abstraction, which object to be created.

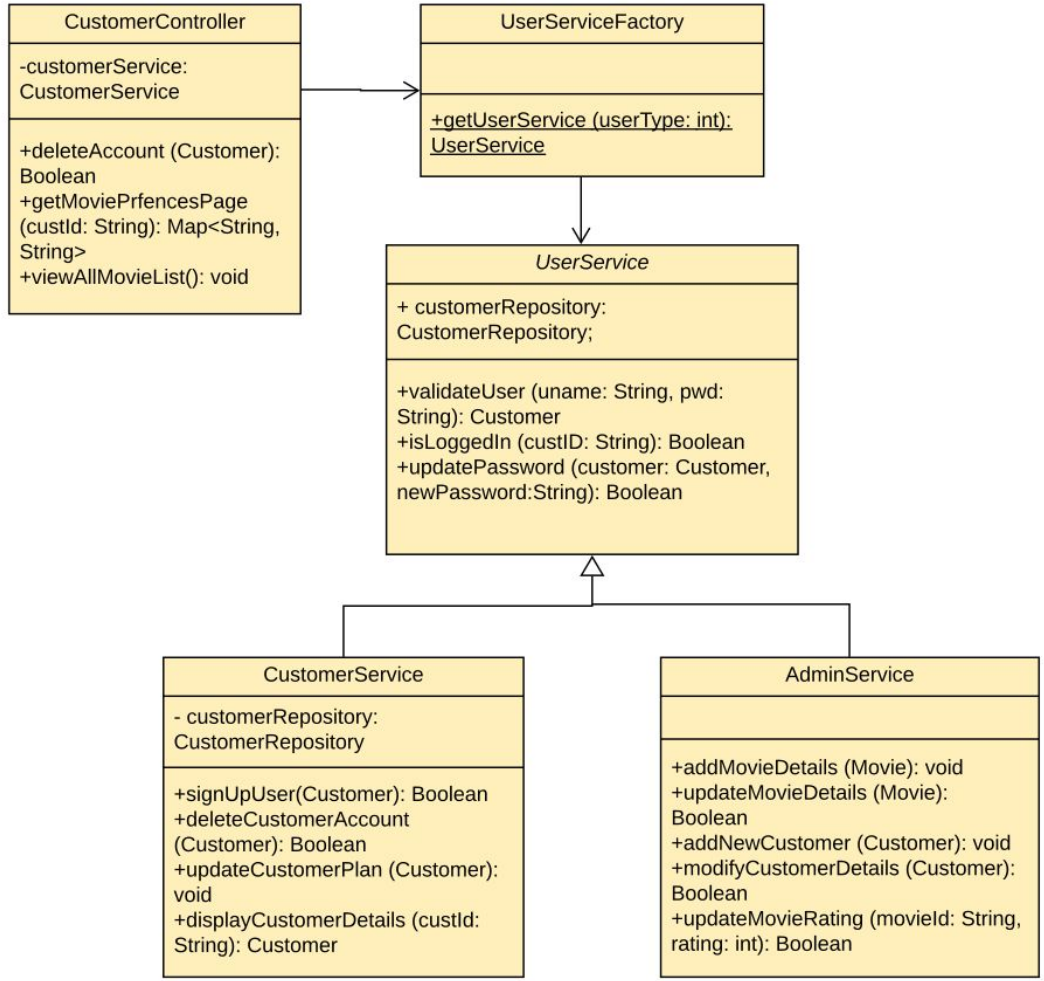
```
if (type.equals("cheese")) {
   pizza = new CheesePizza();
} else if (type.equals("greek")) {
   pizza = new GreekPizza();
} else if (type.equals("pepperoni")) {
   pizza = new PepperoniPizza();
}
```

Reference: Class Slides on Factory Design Pattern

Class Diagram To get User Controller Object



Class Diagram
To get User Service Object



```java
public class UserServiceFactory {

    public static UserService getUserService(int userType) {

        UserService userService = null;

        if (userType == 0 || userType == 1) {
            userService = new CustomerService();
        } else if (userType == 2) {
            userService = new AdminService();
        }

        return userService;
    }
}
```

```java
public class CustomerController extends UserController {

    private CustomerService customerService =
            (CustomerService) UserServiceFactory.getUserService(1);
```

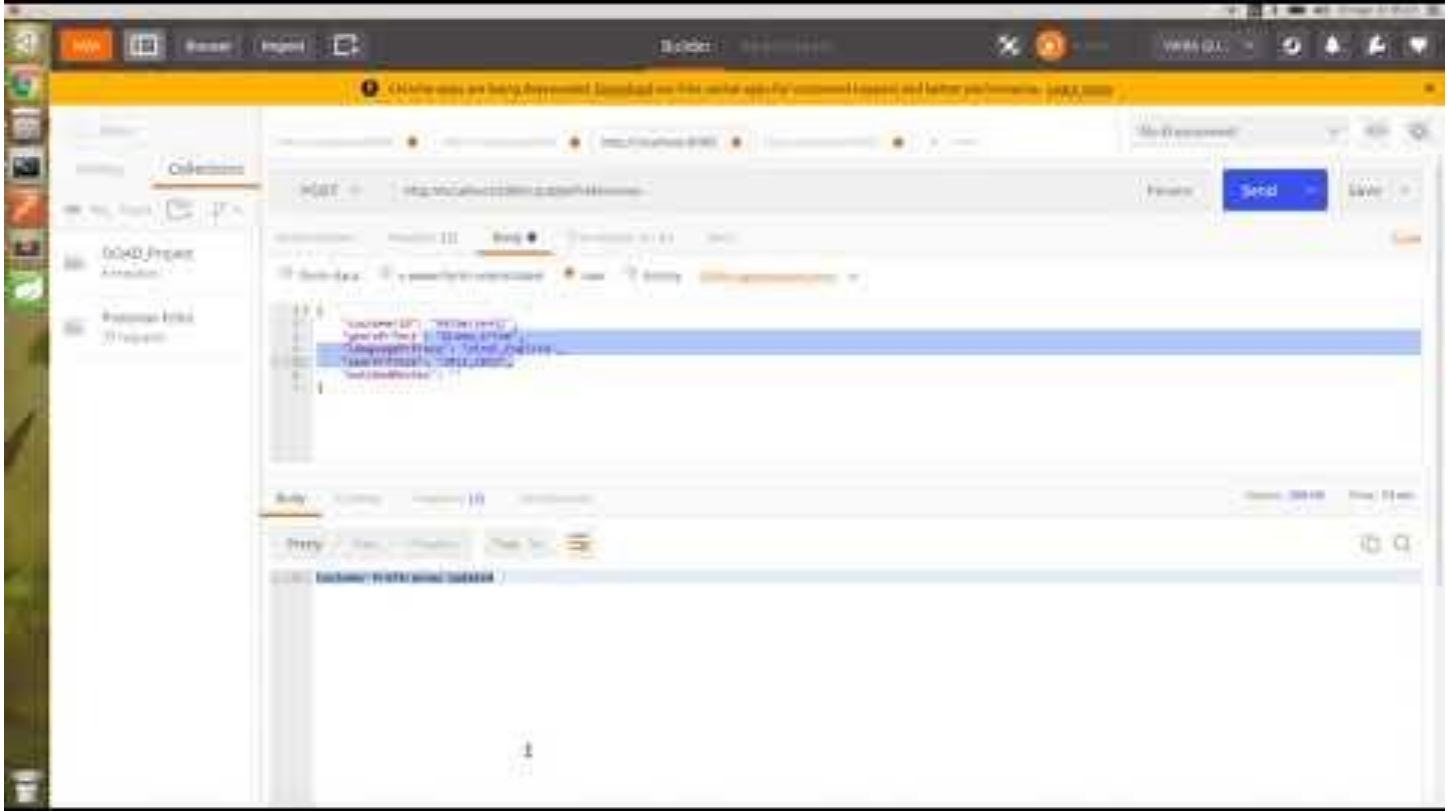
```java
public class AdminController extends UserController {

    private CustomerService adminService =
            (CustomerService) UserServiceFactory.getUserService(2);
```

```
public class CustomerControllerFactory {

    public static UserController getUserController(int userType) {
        UserController userController = null;

        if (userType == 0) {
            userController = new PremiumCustomerController();
        } else if (userType == 1) {
            userController = new CustomerController();
        } else if (userType == 2) {
            userController = new AdminController();
        }
        return userController;
    }
}
```

Use Case Demo

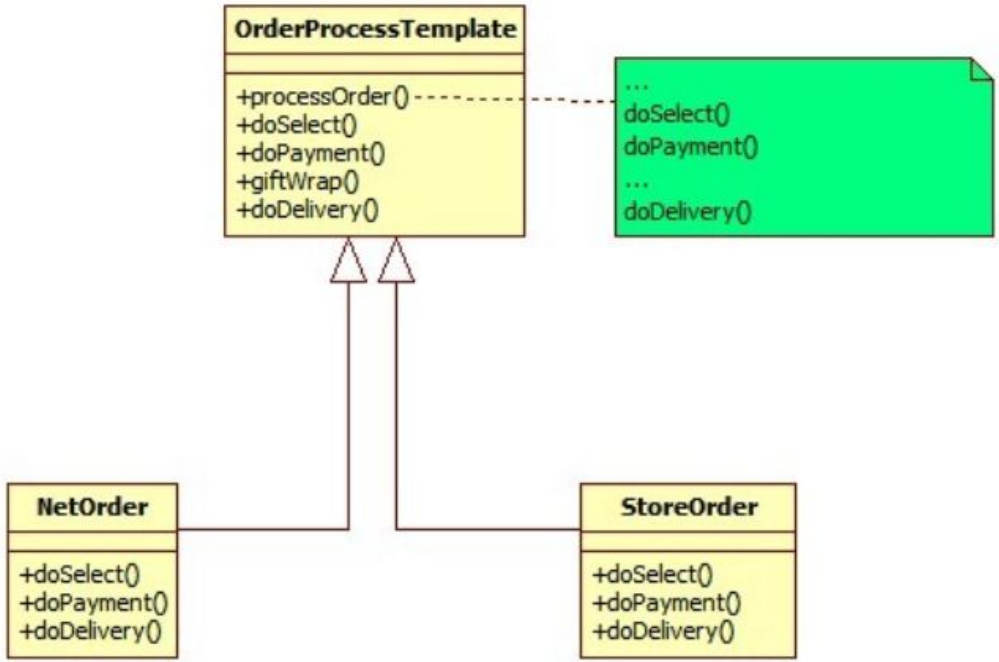
- UR-09: As a customer, I should be able to view my profile.
- UR-11: As a customer, I should be able to add my movie preferences.
  - Language
  - Year
  - Genre
- UR-19: As a customer, I should be able to view the movie recommendations

Link: https://youtu.be/B5qRgd4IKu8



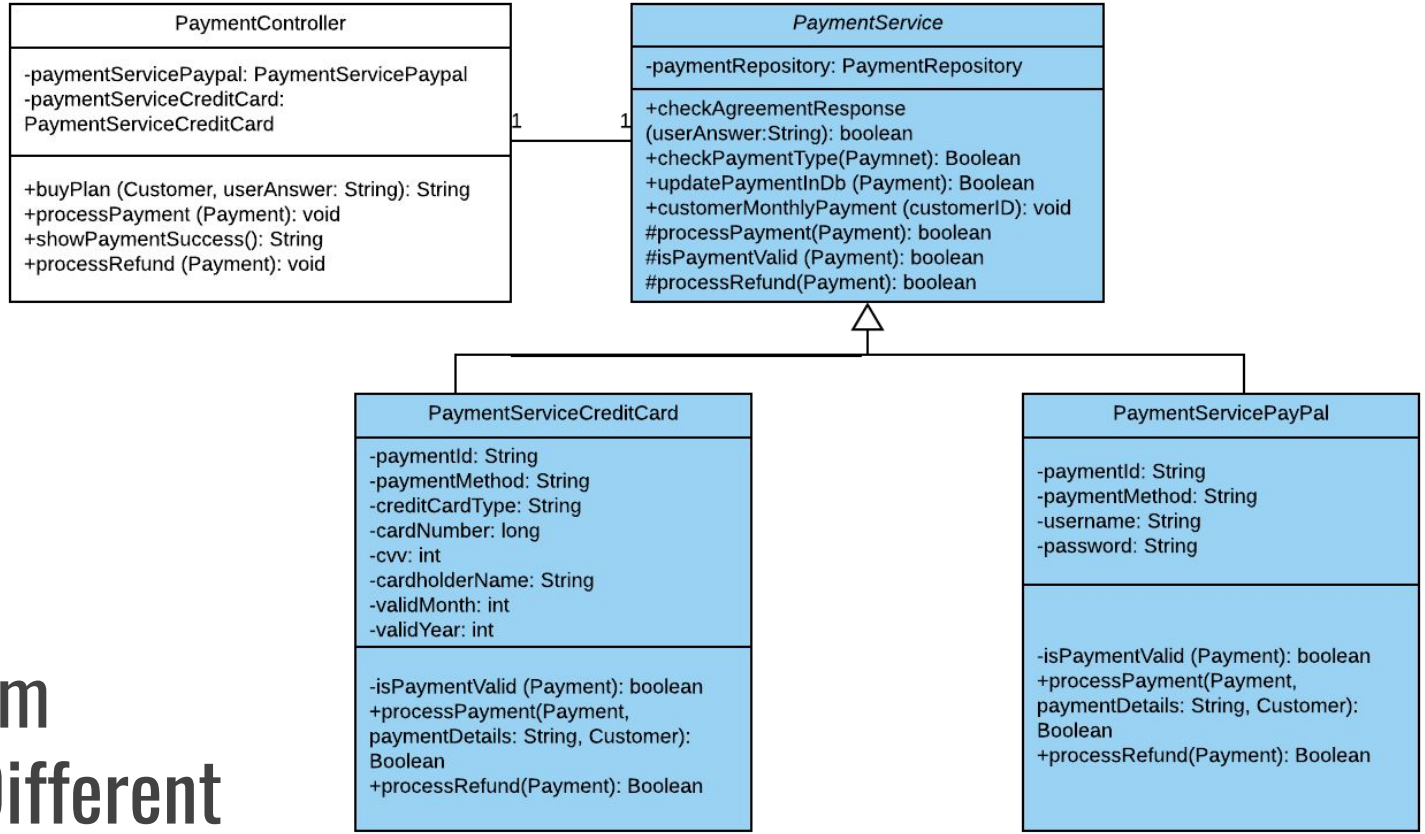
Template Design Pattern

Template Method lets the subclasses redefine certain steps of the algorithm without changing the structure.



Reference: https://javapapers.com/design-patterns/template-method-design-pattern/

Class Diagram To Process Different Types of Payment



```java
public abstract class PaymentService {
    public final void templateMethods(Payment pay) {
        processPayment(pay);
        isPaymentValid(pay);
        processRefund(pay);
    }

    //Template methods
    protected abstract boolean processPayment(Payment pay);
    protected abstract boolean isPaymentValid(Payment pay);
    protected abstract boolean processRefund(Payment pay);

    //All other concrete methods below
    public boolean checkAgreementResponse(String userAnswer) {
        if(userAnswer.toLowerCase().equals("yes"))
            return true;
        else
            return false;
    }
```

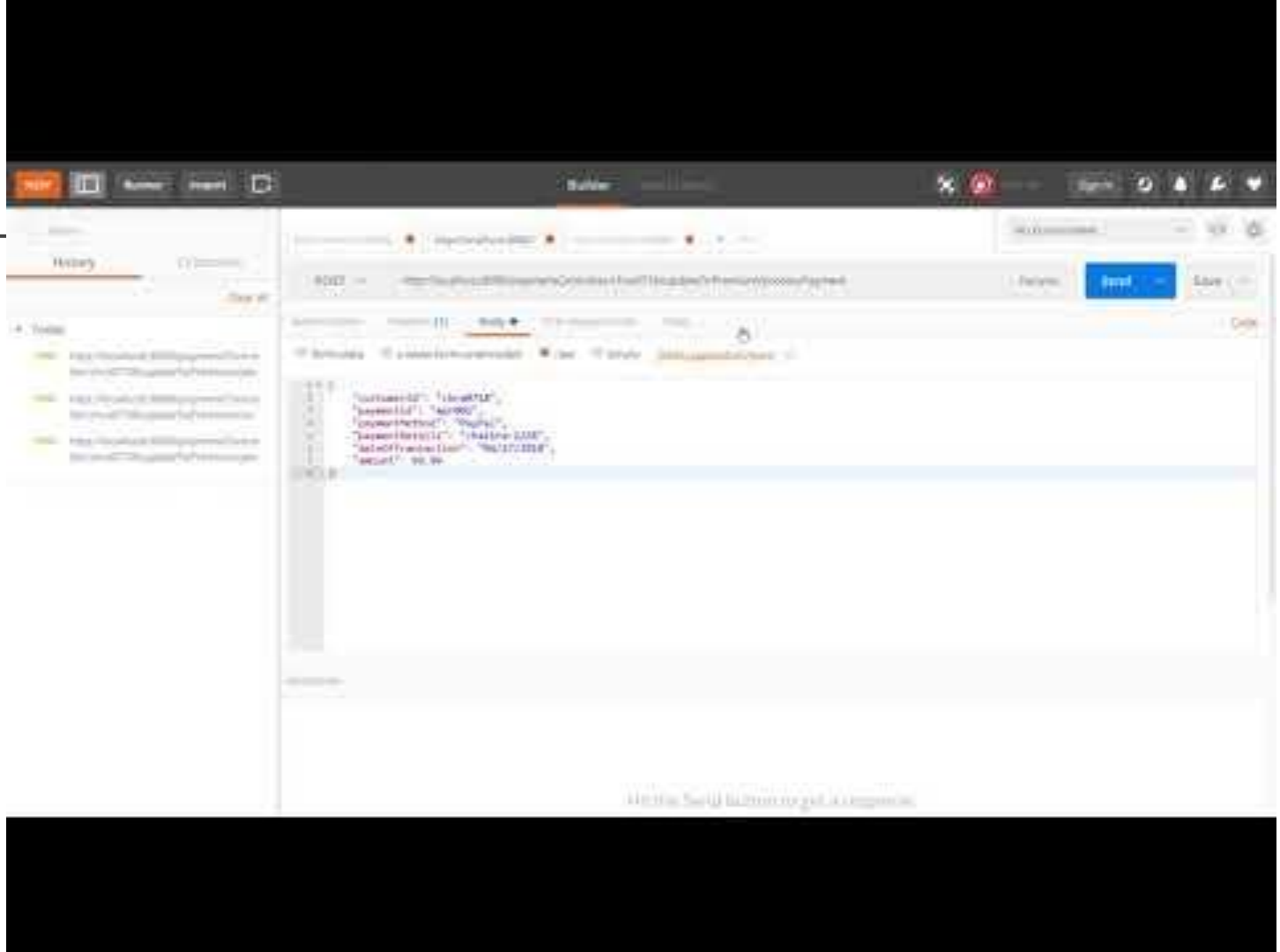
```java
public class PaymentServicePaypal extends PaymentService {
    protected boolean processPayment(Payment pay) {
        if(isPaymentValid(pay))
            return true;
        else
            return false;
    }

    protected boolean isPaymentValid(Payment pay) {
        String paymentDetails[] = pay.getPaymentDetails().split("-");
        String username = paymentDetails[0];
        String password = paymentDetails[1];
        double amount = pay.getAmount();

        //test values
        String testUN = "chaitra";
        String testPwd = "1234";
        double testAmount = 99.99;

        //check for validity
        if(username.equals(testUN) && password.equals(testPwd) && amount == testAmount) {
            return true;
        }
        else
            return false;
    }
```

```java
public class PaymentServiceCreditCard extends PaymentService{
    protected boolean processPayment(Payment pay) {
        if(isPaymentValid(pay))
            return true;
        else
            return false;
    }

    protected boolean isPaymentValid(Payment pay) {
        String paymentDetails[] = pay.getPaymentDetails().split("-");
        String creditCardType = paymentDetails[0];
        String cardHolderName = paymentDetails[1];
        long cardNumber = Long.valueOf(paymentDetails[2]);
        int cvv = Integer.valueOf(paymentDetails[3]);
        int cardMonth = Integer.valueOf(paymentDetails[4]);
        int cardYear = Integer.valueOf(paymentDetails[5]);
        double amount = pay.getAmount();

        //test values
        String testCardType = "Discover";
        String testName = "Chaitra Ramachandra";
        long testCardNumber = 6000700080009000L;
        int testCvv = 123;
        int testCardMonth = 12;
        int testCardYear = 2021;
        double testAmount = 99.99;

        //check for validity
        if(creditCardType.equals(testCardType) && cardHolderName.equals(testName) && cardNumber == testCardNumber
                && cvv == testCvv && cardMonth == testCardMonth && cardYear == testCardYear && amount == testAmount) {
            return true;
        }
        else
            return false;
    }
```

Use Cases Demo

- UR-22: As a customer I should be able to reject/accept the payment agreement
- UR-23: As a customer I should be able to choose my payment method (PayPal)
- UR-24: As a customer I should be able to choose my payment method (Credit Card)

Link to Video: https://youtu.be/sb6zSMF30pQ



Thank you.