### \*\*Chapter 6: Advanced Payment Features and Payouts\*\*

In this chapter, we will explore advanced payment features such as supporting multiple payment methods, handling partial payments, and integrating the Cashfree Payout API for disbursing funds. These features are essential for creating a comprehensive and flexible payment system in your Spring Boot application.

---

#### \*\*Topic 1: Supporting Multiple Payment Methods\*\*

Cashfree supports a variety of payment methods such as UPI, Net Banking, Debit/Credit Cards, and Wallets. Integrating multiple payment methods into your application will give customers more flexibility in how they want to pay.

---

##### \*\*Step 1: Understanding Payment Methods in Cashfree\*\*

Cashfree allows customers to choose from multiple payment options. When initiating a payment, you can specify the preferred payment method in the request or let Cashfree handle it through its hosted checkout page.

- Common payment modes:

- \*\*UPI\*\* – Unified Payments Interface

- \*\*CARD\*\* – Credit/Debit Card

- \*\*NB\*\* – Net Banking

- \*\*WALLET\*\* – Wallet Payments (Paytm, etc.)

---

##### \*\*Step 2: Setting Payment Modes via Hosted Checkout\*\*

When using Cashfree’s hosted payment page, Cashfree automatically shows all available payment options, and customers can choose their preferred method.

However, if you want to restrict the options shown to customers, you can specify allowed payment modes when generating the payment token.

---

##### \*\*Step 3: Sending Preferred Payment Modes in Payment Token Request\*\*

Modify the `PaymentRequest` to include preferred payment modes.

1. \*\*Model Update for PaymentRequest:\*\*

Add a field for preferred payment mode.

```java

public class PaymentRequest {

private String orderId;

private String orderAmount;

private String orderCurrency;

private String customerEmail;

private String customerPhone;

private String paymentMode; // Added field for payment mode

// Getters and setters

}

```

2. \*\*Service Method Update to Include Payment Modes:\*\*

In the `generatePaymentToken` method, include the preferred payment mode in the request to Cashfree.

```java

public String generatePaymentToken(PaymentRequest paymentRequest) throws Exception {

// Construct request payload

Map<String, String> requestBody = new HashMap<>();

requestBody.put("orderId", paymentRequest.getOrderId());

requestBody.put("orderAmount", paymentRequest.getOrderAmount());

requestBody.put("orderCurrency", paymentRequest.getOrderCurrency());

requestBody.put("customerEmail", paymentRequest.getCustomerEmail());

requestBody.put("customerPhone", paymentRequest.getCustomerPhone());

requestBody.put("paymentMode", paymentRequest.getPaymentMode()); // Adding payment mode

// Prepare headers

HttpHeaders headers = new HttpHeaders();

headers.setContentType(MediaType.APPLICATION\_JSON);

headers.set("x-client-id", cashfreeConfig.getAppId());

headers.set("x-client-secret", cashfreeConfig.getSecretKey());

HttpEntity<Map<String, String>> request = new HttpEntity<>(requestBody, headers);

// Send POST request to Cashfree API

String url = cashfreeConfig.getBaseUrl() + "/cftoken/order";

ResponseEntity<Map> response = restTemplate.postForEntity(url, request, Map.class);

if (response.getStatusCode() == HttpStatus.OK) {

// Extract token from response

Map<String, String> responseBody = response.getBody();

return responseBody.get("cftoken");

} else {

throw new Exception("Failed to create payment token");

}

}

```

3. \*\*Frontend Example to Include Payment Modes:\*\*

When initiating the payment from the frontend, specify the payment mode in the request.

```javascript

fetch('http://localhost:8080/api/payment/create', {

method: 'POST',

headers: {

'Content-Type': 'application/json',

},

body: JSON.stringify({

orderId: 'ORDER12345',

orderAmount: '500',

orderCurrency: 'INR',

customerEmail: 'customer@example.com',

customerPhone: '9876543210',

paymentMode: 'UPI' // Specifying UPI as the payment mode

})

})

.then(response => response.json())

.then(data => {

console.log('Payment Token:', data);

// Redirect to Cashfree payment page with token

})

.catch(error => {

console.error('Error:', error);

});

```

---

#### \*\*Topic 2: Handling Partial Payments\*\*

Sometimes, customers may wish to make partial payments. Cashfree supports partial payments, where the customer can pay a portion of the total amount and complete the rest at a later time.

---

##### \*\*Step 1: Enable Partial Payments in Cashfree\*\*

To enable partial payments in Cashfree, you will need to configure it in your dashboard or specify the allowed payment ranges when generating the order.

---

##### \*\*Step 2: Implementing Partial Payments in Spring Boot\*\*

You can allow partial payments by sending the \*\*minAmount\*\* and \*\*maxAmount\*\* fields when creating a payment order.

1. \*\*Model Update for Partial Payment:\*\*

Add fields to allow partial payments.

```java

public class PaymentRequest {

private String orderId;

private String orderAmount;

private String orderCurrency;

private String customerEmail;

private String customerPhone;

private String paymentMode;

private String minAmount; // Added for partial payment

private String maxAmount; // Added for partial payment

// Getters and setters

}

```

2. \*\*Service Method to Handle Partial Payments:\*\*

Include the partial payment fields in the payment token request.

```java

public String generatePaymentToken(PaymentRequest paymentRequest) throws Exception {

// Construct request payload

Map<String, String> requestBody = new HashMap<>();

requestBody.put("orderId", paymentRequest.getOrderId());

requestBody.put("orderAmount", paymentRequest.getOrderAmount());

requestBody.put("orderCurrency", paymentRequest.getOrderCurrency());

requestBody.put("customerEmail", paymentRequest.getCustomerEmail());

requestBody.put("customerPhone", paymentRequest.getCustomerPhone());

requestBody.put("minAmount", paymentRequest.getMinAmount()); // Set minimum partial payment amount

requestBody.put("maxAmount", paymentRequest.getMaxAmount()); // Set maximum payment amount

// Rest of the code remains the same

}

```

---

#### \*\*Topic 3: Cashfree Payout API Integration\*\*

The \*\*Cashfree Payout API\*\* allows you to disburse funds to vendors, customers, or employees. This can be useful in cases where you need to issue payouts for services, refunds, or other payments.

---

##### \*\*Step 1: Overview of Payout API\*\*

The Payout API has several key features:

- \*\*Add Beneficiary\*\* – Add a recipient to whom payments will be sent.

- \*\*Transfer Funds\*\* – Transfer money to a beneficiary.

- \*\*Check Transfer Status\*\* – Verify if a payout was successful.

---

##### \*\*Step 2: Adding a Beneficiary\*\*

Before making a payout, you need to add a beneficiary (the person or entity to whom you're sending money).

- \*\*API Endpoint:\*\*

```

/payout/v1/addBeneficiary

```

- \*\*Method:\*\* `POST`

- \*\*Required Parameters:\*\*

- `beneId` – Unique ID for the beneficiary.

- `name` – Name of the beneficiary.

- `email` – Beneficiary's email.

- `phone` – Beneficiary's phone number.

- `bankAccount` – Beneficiary’s bank account number.

- `ifsc` – Bank IFSC code for the beneficiary.

---

##### \*\*Step 3: Implementing Add Beneficiary in Spring Boot\*\*

1. \*\*Model for Beneficiary Request:\*\*

Create a model to represent the beneficiary details.

```java

public class BeneficiaryRequest {

private String beneId;

private String name;

private String email;

private String phone;

private String bankAccount;

private String ifsc;

// Getters and setters

}

```

2. \*\*Service Method for Adding Beneficiary:\*\*

Implement the logic to add a beneficiary via Cashfree Payout API.

```java

public String addBeneficiary(BeneficiaryRequest beneficiaryRequest) throws Exception {

// Prepare request payload

Map<String, String> requestBody = new HashMap<>();

requestBody.put("beneId", beneficiaryRequest.getBeneId());

requestBody.put("name", beneficiaryRequest.getName());

requestBody.put("email", beneficiaryRequest.getEmail());

requestBody.put("phone", beneficiaryRequest.getPhone());

requestBody.put("bankAccount", beneficiaryRequest.getBankAccount());

requestBody.put("ifsc", beneficiaryRequest.getIfsc());

// Prepare headers

HttpHeaders headers = new HttpHeaders();

headers.setContentType(MediaType.APPLICATION\_JSON);

headers.set("Authorization", "Bearer " + payoutConfig.getPayoutAuthToken());

HttpEntity<Map<String, String>> request = new HttpEntity<>(requestBody, headers);

// Send POST request to Cashfree Payout API

String url = payoutConfig.getBaseUrl() + "/payout/v1/addBeneficiary";

ResponseEntity<Map> response = restTemplate.postForEntity(url, request, Map.class);

if (response.getStatusCode() == HttpStatus.OK) {

// Extract response

return response.getBody().get("message");

} else {

throw new Exception("Failed to add beneficiary");

}

}

```

---

#### \*\*What We’ve Covered So Far:\*\*

1. \*\*Multiple

Payment Methods:\*\* How to support multiple payment modes like UPI, Cards, and Net Banking.

2. \*\*Partial Payments:\*\* Implementing partial payments for orders.

3. \*\*Payout API:\*\* Adding a beneficiary using Cashfree Payout API.

---