### **Exhaustive Functional Test Cases: Payment Module**

**Objective:** To provide a final, exhaustive suite of functional test cases for the Payment Module, incorporating detailed general, parameterized, and transactional function requirements. This document is designed to ensure maximum test coverage by validating input boundaries, system configurations, and complex financial scenarios.

#### **1. General Test Cases**

This section covers fundamental validation, security, and system behavior across the application.

##### **1.1. Input Validation & Boundary Testing**

| **Test Case ID** | **Test Scenario** | **Test Steps** | **Test Data** | **Expected Result** |
| --- | --- | --- | --- | --- |
| **GEN-VAL-001** | **Min/Max Characters: billingId** | 1. Attempt to create an SO with a billingId of 1 character.<br>2. Attempt to create an SO with a billingId of 51 characters. | billingId: "A", "A...A" (51 chars) | 1. Success.<br>2. UI validation error: "Billing ID cannot exceed 50 characters." API returns 400 Bad Request. |
| **GEN-VAL-002** | **Blank Input: Mandatory Fields** | 1. On the "Add SO" form, enter only spaces into the billingId field.<br>2. Submit the form. | billingId: " " | UI validation error: "This field is required." The form should not submit. API should reject the request. |
| **GEN-VAL-003** | **Special Characters (Standard)** | 1. Create an SO with a billingId containing standard special characters. | billingId: Test-SO\_123.45 | The system accepts and displays the characters correctly. |
| **GEN-VAL-004** | **Special Characters (Extended)** | 1. Create a Product with a name containing extended special characters. | productName: !@#$%^&\*()\_+-=[]{}|;:'",<.>/? | The system should accept and display the characters correctly without errors or security issues (e.g., Cross-Site Scripting). |
| **GEN-VAL-005** | **Data Duplication: Standing Order** | 1. Create and approve a Standing Order for a specific productCode and billingId.<br>2. Attempt to create a second, identical Standing Order. | **Product:** "Indihome"<br>**Billing ID:** "12345" | A validation error should appear: "An active Standing Order already exists for this product and billing ID." |
| **GEN-VAL-006** | **File Size: Export Excel (Large)** | 1. Generate a list of 10,000+ Standing Orders.<br>2. Click the "Export" button. | N/A | The export process should complete successfully within a reasonable time. The generated file should not be corrupted. |
| **GEN-VAL-007** | **File Size: Export Excel (Empty)** | 1. Filter the SO list to a state where no records exist.<br>2. Click the "Export" button. | N/A | An Excel file is downloaded containing only the header row. |
| **GEN-VAL-008** | **Mandatory Fields (Rejection)** | 1. On the "Reject Confirmation Dialog", attempt to click "Reject" without entering notes. | **Notes:** (empty) | The "Reject" button is disabled. A validation message "Input Notes is mandatory" is clearly visible. |
| **GEN-VAL-009** | **Numeric Input Validation** | 1. Attempt to enter text into a numeric-only field (e.g., "Start Debit Date"). | **Start Debit Date:** "abc" | The UI should prevent text input, or display a validation error "Please enter a valid number." |
| **GEN-VAL-010** | **International Characters (Unicode)** | 1. Create a Product with a name containing Unicode characters.<br>2. Create an SO using that product. | productName: "Produk Pembayaran (你好)" | The name should be stored and displayed correctly throughout the UI and in any exported files without corruption. |
| **GEN-VAL-011** | **Concurrent Submission (Maker)** | 1. Two different Maker users attempt to create the exact same Standing Order at the exact same time. | **Product:** "PLN"<br>**Billing ID:** "54321" | Only one Standing Order should be successfully created. The second user should receive an error message: "This Standing Order has just been created by another user." |
| **GEN-VAL-012** | **Concurrent Approval (Checker)** | 1. An SO is pending approval.<br>2. Two different Checker users open the approval screen for the same SO simultaneously.<br>3. Checker A approves. Checker B then attempts to approve. | N/A | Checker A's approval succeeds. Checker B receives an error message: "This item has already been processed by another user." |

##### **1.2. Logging & Error Handling**

| **Test Case ID** | **Test Scenario** | **Test Steps** | **Expected Result** |
| --- | --- | --- | --- |
| **GEN-LOG-001** | **User Access Log (Success/Fail)** | 1. User Maker\_A logs in successfully.<br>2. User Checker\_A logs in with an invalid password. | 1. The Activity Log records a successful login event for Maker\_A with timestamp and IP address.<br>2. The system security log records a failed login attempt for Checker\_A. |
| **GEN-LOG-002** | **Transaction Log (Audit Trail)** | 1. Maker\_A creates an SO.<br>2. Approver\_A approves it.<br>3. Maker\_A amends it.<br>4. Approver\_A rejects the amendment. | The "Data History" for that SO should show four distinct entries, correctly timestamped and attributed to the correct user and activity ("Submitted", "Approved", "Submitted", "Rejected"). |
| **GEN-ERR-001** | **Warning Message (Biller API Down)** | 1. Simulate a scenario where the Biller Aggregator API is down.<br>2. A Maker attempts to create a new SO. | A user-friendly error message is displayed on the UI: "Unable to connect to the biller service. Please try again later." The system does not crash. |
| **GEN-ERR-002** | **Warning Message (OVS-BAS Down)** | 1. Simulate a scenario where the OVS-BAS/JHUB API is down.<br>2. A Maker attempts to validate an account number. | A user-friendly error message is displayed: "Unable to validate account information at this time. Please try again later." |
| **GEN-ERR-003** | **Database Connection Failure** | 1. In the test environment, temporarily disable the application's connection to the PostgreSQL database.<br>2. Attempt to log in or access any page. | The user is presented with a generic but user-friendly error page (e.g., "The service is temporarily unavailable. Please try again later.") instead of a technical stack trace. |

#### **2. Parameterized Test Cases**

This section covers the system's behavior based on configurable parameters.

| **Test Case ID** | **Test Scenario** | **Test Steps** | **Test Data** | **Expected Result** |
| --- | --- | --- | --- | --- |
| **PARAM-JOB-001** | **Job Schedule: Off-Hours** | 1. In "Master Job Scheduler", configure the payment worker to run only between 09:00 and 17:00.<br>2. Manually trigger the job at 18:00. | **Time:** 18:00 WIB | The job should either not run or run and immediately exit, logging a message "Outside of operational hours." No payments should be processed. |
| **PARAM-JOB-002** | **Job Schedule: Holiday** | 1. Ensure today's date is configured as a public holiday (retrieved from DWH).<br>2. Trigger the payment worker job. | **Date:** Public Holiday | The job should not process any payments, logging a message "Skipping execution due to public holiday." |
| **PARAM-JOB-003** | **Job Schedule: Interaction with Holiday** | 1. Configure the worker job to run at 10:00.<br>2. Configure today as a public holiday.<br>3. Configure tomorrow as a working day. | N/A | The job should not run today. The job should run successfully tomorrow and process any payments that were due today. |
| **PARAM-DATA-001** | **Data Retention (Live to Archive)** | 1. Create a transaction and manually age its timestamp in the database to be 7 months old.<br>2. Trigger the data retention/archiving job. | **Transaction Date:** 7 months ago | The transaction record is successfully moved from the live ts\_payment\_status table to the archive database. |
| **PARAM-DATA-002** | **Data Retention (Archive to Backup)** | 1. Create an archived transaction and manually age it to be 13 months old.<br>2. Trigger the archiving job. | **Archive Date:** 13 months ago | The transaction record is successfully moved from the archive database to the long-term backup server. |
| **PARAM-FEE-001** | **Charge Configuration** | 1. In "Master Charge", update the "Standing Order Creation" fee from 5000 to 7500 and approve the change.<br>2. Create and approve a new Standing Order. | **Fee:** 7500 | The customer's account is debited exactly 7500 for the registration fee. |

#### **3. Transactional Function Test Cases**

This section covers complex financial scenarios related to accounts, currency, and transaction timing.

##### **3.1. Account Status & Type Validation**

| **Test Case ID** | **Test Scenario** | **Test Steps** | **Test Data** | **Expected Result** |
| --- | --- | --- | --- | --- |
| **TRX-ACCT-001** | **Sleeping Account** | 1. Configure a customer account in the OVS-BAS simulator with "Sleeping" status.<br>2. Attempt to approve a new SO for this account. | **Account Status:** Sleeping | The fee debit fails. The approval fails with an error: "Account is in sleeping status." |
| **TRX-ACCT-002** | **Inactive Account** | 1. Configure a customer account in the OVS-BAS simulator with "Inactive" status.<br>2. Trigger the payment worker for an SO linked to this account. | **Account Status:** Inactive | The Hold Amount request fails. The transaction status in PU becomes "Payment Failed" with a reason "Inactive Account". |
| **TRX-ACCT-003** | **Dormant Account** | 1. Configure a customer account in the OVS-BAS simulator with "Dormant" status.<br>2. Trigger the payment worker for an SO linked to this account. | **Account Status:** Dormant | The Hold Amount request fails. The transaction status in PU becomes "Payment Failed" with a reason "Dormant Account". |
| **TRX-ACCT-004** | **Account with Overdraft Facility** | 1. Configure an account in OVS-BAS with a balance of 100,000 and an overdraft limit of 500,000.<br>2. Trigger a payment for a bill of 250,000. | **Bill Amount:** 250,000 | The Hold Amount and Debit are successful. The payment is processed correctly. |
| **TRX-ACCT-005** | **Balance Short Transaction** | 1. Configure an account in OVS-BAS with a balance of 100,000.<br>2. Trigger a payment for a bill of 250,000. | **Bill Amount:** 250,000 | The Hold Amount fails. The transaction status in PU becomes "Balance Short". |
| **TRX-ACCT-006** | **Hold Account (Stop Payment)** | 1. Configure an account in OVS-BAS with a "Hold" or "Stop Payment" flag.<br>2. Trigger the payment worker for an SO linked to this account. | **Account Status:** Hold | The Hold Amount request fails. The transaction status in PU becomes "Payment Failed" with a reason "Account has a hold". |
| **TRX-ACCT-007** | **Pooling Account** | 1. An SO is configured with a pooling/header account.<br>2. Trigger a payment. | **Account Type:** Pooling | The system should correctly identify the account and process the debit as per the bank's rules for pooling accounts. |
| **TRX-ACCT-008** | **Zero Balance Account** | 1. Configure an account with a balance of 0.<br>2. Attempt to approve an SO with a registration fee. | **Balance:** 0 | The approval fails with an "Insufficient funds" error. |

##### **3.2. Branch, Currency, and Value Date**

| **Test Case ID** | **Test Scenario** | **Test Steps** | **Test Data** | **Expected Result** |
| --- | --- | --- | --- | --- |
| **TRX-BRANCH-001** | **Branch Code Variant: 3665** | 1. Create and process a payment using a debit account from branch 3665. | **Branch Code:** 3665 | The transaction is processed successfully. |
| **TRX-BRANCH-002** | **Branch Code Variant: 3667** | 1. Create and process a payment using a debit account from branch 3667. | **Branch Code:** 3667 | The transaction is processed successfully. |
| **TRX-CURR-001** | **Multi-Currency: USD Debit for IDR Bill** | 1. Create an SO for an IDR bill, but specify a valid USD account for the debit.<br>2. Trigger the payment worker. | **Debit Acct:** USD<br>**Bill:** IDR | The system correctly calculates the exchange rate, debits the equivalent amount in USD, and settles the IDR amount. The transaction details should show the exchange rate used. |
| **TRX-CURR-002** | **Exchange Rate Fluctuation** | 1. Process a USD->IDR payment in the morning.<br>2. Update the exchange rate in the system.<br>3. Process another USD->IDR payment in the afternoon. | **Rate 1:** 15,000<br>**Rate 2:** 15,100 | The two transactions should be processed using the correct exchange rate that was active at the time of their execution. |
| **TRX-VD-001** | **Future Value Date Transaction** | 1. Create a payment instruction with a value date of T+2.<br>2. Trigger the payment worker on day T. | **Value Date:** T+2 | The worker should ignore the transaction on day T and T+1. The transaction should only be picked up and processed on day T+2. |
| **TRX-VD-002** | **Back Value Date Transaction** | 1. A payment failed on T-1. It is repaired and approved on day T.<br>2. The system re-processes the payment. | **Original Date:** T-1 | The transaction should be processed with a booking date of T, but the value date should be correctly recorded as T-1 for accounting purposes. |
| **TRX-VD-003** | **Back Value Date on a Holiday** | 1. A payment fails on Friday (T-1). Monday (T+1) is a public holiday. The payment is repaired and approved on Tuesday (T+2). | **Original Date:** Friday (T-1) | The transaction should be processed with a booking date of Tuesday (T+2), but the value date should be recorded as Friday (T-1). |

##### **3.3. Sanctions & Transaction Flags**

| **Test Case ID** | **Test Scenario** | **Test Steps** | **Test Data** | **Expected Result** |
| --- | --- | --- | --- | --- |
| **TRX-FLAG-001** | **Transaction Flag / Status Transitions** | 1. Process a single transaction through its entire lifecycle: create, fail for balance short, repair, approve, succeed. | N/A | The ts\_payment\_status\_history table should accurately reflect every status change: Outstanding -> Balance Short -> Submitted for Repair -> Approved for Repair -> Paid. |
| **TRX-SANC-001** | **Sanction Screening** | 1. Create an SO where the retrieved Customer Name from the biller matches an entry on a sanction list simulator.<br>2. Trigger the payment. | **Customer Name:** "Sanctioned Entity" | The payment process should be halted before execution. The transaction should be flagged and moved to a "Pending Sanction Review" status for compliance to investigate. |

#### **4. Website Functionality (UI/UX) Test Cases**

This section covers the user-facing functionality of the application, ensuring it is intuitive, error-free, and behaves as specified in the Payment Interface design.docx.

##### **4.1. General UI & Navigation**

| **Test Case ID** | **User Role** | **Test Scenario** | **Test Steps** | **Expected Result** |
| --- | --- | --- | --- | --- |
| **UI-GEN-001** | Any | **Happy Path: Successful Login** | 1. Navigate to the PU login page.<br>2. Enter valid Maker credentials.<br>3. Click "Login". | User is redirected to the PU dashboard. The user's name and the current date/time are displayed in the top-right corner. |
| **UI-GEN-002** | Any | **Navigation: Breadcrumb Trail** | 1. Log in and navigate to Payment > Standing Order.<br>2. Click on "Add Standing Order". | The breadcrumb at the top correctly displays "Payment / Standing Order / Add Standing Order". Each part is a clickable link to navigate back. |
| **UI-GEN-003** | Any | **UI: Refresh Button** | 1. Navigate to the Standing Order list page.<br>2. Click the "Refresh" button. | The data in the table reloads from the server without a full page reload. |
| **UI-GEN-004** | Any | **Browser Navigation** | 1. Navigate from the SO List to the Add SO page.<br>2. Click the browser's "Back" button. | The user is returned to the SO List page, and the previous state (e.g., filters, page number) is preserved. |

##### **4.2. Standing Order (SO) List Page**

| **Test Case ID** | **User Role** | **Test Scenario** | **Test Steps** | **Expected Result** |
| --- | --- | --- | --- | --- |
| **UI-SO-001** | Maker | **UI: Verify Columns (Maker View)** | 1. Navigate to the "Standing Order" page. | The table displays all specified columns: ID, Account Number, Account Name, Category, Product, Start Debit Date, Due Date, Application Status, Approval Status, Added By, Last Updated, Action. |
| **UI-SO-002** | Checker | **UI: Verify Columns (Checker View)** | 1. Log in as a Checker.<br>2. Navigate to the "Standing Order" page. | The "Action" column correctly shows "View" and "Approve/Reject" links for submitted items. |
| **UI-SO-003** | Any | **Function: Apply Combined Filters** | 1. Open the Filter panel.<br>2. Select Category: "PDAM".<br>3. Select Approval Status: "Approved".<br>4. Click "Apply". | The table refreshes to show only Standing Orders that match both filter criteria. |
| **UI-SO-004** | Any | **Function: Export to Excel** | 1. Apply a filter (e.g., Status = "Approved").<br>2. Click the "Export" button. | An Excel file is downloaded. The contents of the file must match the filtered data currently displayed in the table. |
| **UI-SO-005** | Maker | **Action Buttons Visibility** | 1. Find an SO with "Submitted" status.<br>2. Find an SO with "Approved" status. | 1. "Action" column shows "View" and "Delete".<br>2. "Action" column shows "View", "Update", and "Terminate". |

##### **4.3. Add/Edit/Terminate SO Forms**

| **Test Case ID** | **User Role** | **Test Scenario** | **Test Steps** | **Expected Result** |
| --- | --- | --- | --- | --- |
| **UI-FORM-001** | Maker | **Add SO: Dynamic Product Dropdown** | 1. On the "Add SO" form, select "Multifinance" from the "Category" dropdown. | The "Product" dropdown is populated only with Multifinance products (e.g., Mandala Finance). |
| **UI-FORM-002** | Maker | **Add SO: Real-time Billing ID Validation** | 1. Select "Indihome" as the product.<br>2. Enter a valid Billing ID ("No Pelanggan"). | A loading indicator appears briefly, then the "Customer Name" and "Due Date" fields are auto-populated. |
| **UI-FORM-003** | Maker | **Add SO: Real-time Account Number Validation** | 1. Enter a valid 17-character "Account Number (Debit)". | A loading indicator appears briefly, then the "Account Name" field is auto-populated. |
| **UI-FORM-004** | Maker | **Add SO: Auto-fill Charge Account** | 1. Enter a valid "Account Number (Debit)".<br>2. Leave "Account Number (Charge)" blank.<br>3. Click "Submit". | The confirmation dialog "Debit account number will be used as the charge account number..." appears. |
| **UI-FORM-005** | Maker | **Edit SO: Form Population & Read-only Fields** | 1. Click "Update" on an existing Approved SO. | The "Edit Standing Order" form loads, pre-populated with all existing data. "Category", "Product", and "Billing ID" fields are disabled and cannot be edited. |
| **UI-FORM-006** | Maker | **Terminate SO: Form Layout** | 1. Click "Terminate" on an existing Approved SO. | The "Terminate Standing Order" form loads. Most fields are read-only. A mandatory "Final Debit" date picker is displayed. |

##### **4.4. Approval & Dialogs (Checker Role)**

| **Test Case ID** | **User Role** | **Test Scenario** | **Test Steps** | **Expected Result** |
| --- | --- | --- | --- | --- |
| **UI-APP-001** | Checker | **Approval Form: View SO Details** | 1. From the Task List, click to approve a new SO. | The "Standing Order Approval" page displays all details of the SO in read-only fields. "Approve" and "Reject" buttons are visible at the top. |
| **UI-APP-002** | Checker | **Approval Dialog: Reject Confirmation** | 1. On the approval page, click "Reject".<br>2. Attempt to confirm without entering notes. | The "Reject" button in the dialog is disabled. A validation message "Rejection reason is mandatory" is displayed. |
| **UI-APP-003** | Checker | **Approval Failure: Pop-up on Fee Debit Fail** | 1. Attempt to approve an SO where the fee debit will fail. | The "Approval Failed" pop-up appears with the message "Approval failed because the charge fee payment was unsuccessful." |

##### **4.5. Transaction Monitoring & Reconciliation**

| **Test Case ID** | **User Role** | **Test Scenario** | **Test Steps** | **Expected Result** |
| --- | --- | --- | --- | --- |
| **UI-TRN-001** | Any | **Transaction List: Default View & Tab Navigation** | 1. Navigate to the "Transaction" page.<br>2. Click the "All" tab. | The list defaults to the "Need Repair" tab. Clicking "All" correctly displays all transactions. |
| **UI-TRN-002** | Maker | **Transaction List: Bulk Repair** | 1. On the "Need Repair" tab, select the checkboxes for two overdue transactions.<br>2. Click the main "Repair" button at the top of the list. | A confirmation dialog appears. Upon confirmation, both selected transactions are submitted for repair approval. |
| **UI-REC-001** | Maker | **Reconciliation: View NOK/OK Result** | 1. Run a reconciliation that results in a mismatch.<br>2. Run a reconciliation that results in a perfect match. | 1. The summary status bar turns red and displays "Reconciliation NOK".<br>2. The summary status bar turns green and displays "Reconciliation OK". |
| **UI-REC-002** | Maker | **Reconciliation: Match Data Dialog** | 1. On a NOK result, find a mismatched transaction and click "Match Data". | The "Are you sure want to match this data?" dialog appears, correctly showing the "Total Amount by Biller" and "Total Amount in PU". |