### **Exhaustive API Test Cases: Payment Uploader - Payment Module (Enhanced)**

**Objective:** To provide an exhaustive and highly detailed suite of API test cases for the System Integration Testing (SIT) of the Payment Module. These test cases are designed to validate every aspect of the API's functionality, including data validation, business logic, status transitions, security, and integration with downstream systems.

**Note:** API endpoints are inferred based on RESTful principles and the service architecture described in the TDD.

#### **1. Standing Order Management (/api/standing-orders)**

This endpoint manages the entire lifecycle of standing order registrations.

##### **1.1. Create Standing Order (POST /api/standing-orders)**

| **Test Case ID** | **Test Scenario** | **Request Payload / Key Parameters** | **Expected HTTP Status** | **Expected Response Body / Outcome** |
| --- | --- | --- | --- | --- |
| **SO-POST-001 - 031** | **Happy Path: Create SO for All 31 Billers** | For each of the 31 billers in Detail biller SYB.csv:<br>{ "productCode": "...", "billingId": "...", "debitAccountNumber": "...", "chargeAccountNumber": "..." } | 201 Created | Returns the newly created Standing Order object with approvalStatus: "Submitted". |
| **SO-POST-032** | **Happy Path: chargeAccountNumber is auto-filled** | { "productCode": "...", "billingId": "...", "debitAccountNumber": "ACCT\_A", "chargeAccountNumber": null } | 201 Created | The created SO object in the DB has chargeAccountNumber set to "ACCT\_A". |
| **SO-POST-033** | **Validation: Missing billingId** | { "productCode": "PDAM-BTM", "debitAccountNumber": "..." } | 400 Bad Request | Error message: "billingId is a required field." |
| **SO-POST-034** | **Validation: Missing productCode** | { "billingId": "...", "debitAccountNumber": "..." } | 400 Bad Request | Error message: "productCode is a required field." |
| **SO-POST-035** | **Validation: Missing debitAccountNumber** | { "productCode": "PDAM-BTM", "billingId": "..." } | 400 Bad Request | Error message: "debitAccountNumber is a required field." |
| **SO-POST-036** | **Validation: Empty billingId** | { "billingId": "", ... } | 400 Bad Request | Error message: "billingId cannot be empty." |
| **SO-POST-037** | **Validation: Invalid Data Type for startDebitDate** | { "startDebitDate": "abc", ... } | 400 Bad Request | Error message: "startDebitDate must be a valid number." |
| **SO-POST-038** | **Validation: Out of Range startDebitDate** | { "startDebitDate": 32, ... } | 400 Bad Request | Error message: "startDebitDate must be between 1 and 31." |
| **SO-POST-039** | **Integration: Invalid productCode** | { "productCode": "INVALID-CODE", ... } | 404 Not Found | Error message: "Product with code 'INVALID-CODE' not found." |
| **SO-POST-040** | **Integration: Invalid billingId (Biller Rejects)** | { "billingId": "999999999", ... } | 400 Bad Request | Error message from Biller Aggregator: "Billing ID not found." |
| **SO-POST-041** | **Integration: Invalid debitAccountNumber (OVS-BAS)** | { "debitAccountNumber": "DORMANT-ACCT", ... } | 400 Bad Request | Error message: "Account number is invalid or dormant." |
| **SO-POST-042** | **Integration: Biller Aggregator Timeout** | (Configure simulator to not respond) | 504 Gateway Timeout | Error message: "The request to the biller aggregator timed out." |
| **SO-POST-043** | **Integration: OVS-BAS Timeout** | (Configure simulator to not respond) | 504 Gateway Timeout | Error message: "The request to the core banking system timed out." |
| **SO-POST-044** | **Security: Unauthorized (No Token)** | (No Authorization header) | 401 Unauthorized | Standard unauthorized error response. |
| **SO-POST-045** | **Security: Forbidden (Checker Role)** | Authorization: Checker's JWT | 403 Forbidden | Error message: "User does not have permission to create this resource." |

##### **1.2. Get Standing Orders (GET /api/standing-orders)**

| **Test Case ID** | **Test Scenario** | **Request Payload / Key Parameters** | **Expected HTTP Status** | **Expected Response Body / Outcome** |
| --- | --- | --- | --- | --- |
| **SO-GET-001** | **Happy Path: Get All Standing Orders** | (No parameters) | 200 OK | Returns a JSON array of all standing order objects. |
| **SO-GET-002** | **Filtering: Get by approvalStatus** | ?approvalStatus=Approved | 200 OK | Returns a JSON array of SOs only with "Approved" status. |
| **SO-GET-003** | **Filtering: Get by applicationStatus** | ?applicationStatus=Amendment | 200 OK | Returns a JSON array of SOs only with "Amendment" status. |
| **SO-GET-004** | **Filtering: Get by accountNumber** | ?accountNumber=... | 200 OK | Returns a JSON array of SOs matching the account number. |
| **SO-GET-005** | **Filtering: Get by productCode** | ?productCode=PDAM-BTM | 200 OK | Returns a JSON array of SOs for the specified product. |
| **SO-GET-006** | **Filtering: Get by addedBy** | ?addedBy=Reyhan | 200 OK | Returns a JSON array of SOs added by "Reyhan". |
| **SO-GET-007** | **Filtering: Get by Date Range** | ?lastUpdatedFrom=...&lastUpdatedTo=... | 200 OK | Returns SOs updated within the specified date range. |
| **SO-GET-008** | **Filtering: Combined Filters** | ?approvalStatus=Approved&productCode=PDAM-BTM | 200 OK | Returns a JSON array of SOs that are both "Approved" AND for "PDAM-BTM". |
| **SO-GET-009** | **Filtering: No Results** | ?billingId=NONEXISTENT | 200 OK | Returns an empty JSON array []. |
| **SO-GET-010** | **Pagination: Get First Page** | ?page=1&pageSize=10 | 200 OK | Returns the first 10 SOs. Response headers should contain pagination info (e.g., X-Total-Count). |
| **SO-GET-011** | **Pagination: Get Second Page** | ?page=2&pageSize=10 | 200 OK | Returns SOs 11 through 20. |
| **SO-GET-012** | **Pagination: Invalid Page Size** | ?pageSize=9999 | 400 Bad Request | Error message: "Page size exceeds maximum limit." |
| **SO-GET-013** | GET /api/standing-orders/:id | **Happy Path: Get SO by ID** | id: Valid SO ID | 200 OK |
| **SO-GET-014** | GET /api/standing-orders/:id | **Negative: Get Non-Existent SO** | id: 99999 | 404 Not Found |
| **SO-GET-015** | GET /api/standing-orders/:id | **Validation: Invalid ID format** | id: abc | 400 Bad Request |

##### **1.3. Update/Amend/Terminate/Delete Standing Order**

| **Test Case ID** | **API Endpoint & Method** | **Test Scenario** | **Request Payload / Key Parameters** | **Expected HTTP Status** | **Expected Response Body / Outcome** |
| --- | --- | --- | --- | --- | --- |
| **SO-PUT-001** | PUT /api/standing-orders/:id | **Happy Path: Amend an Approved SO** | id: Valid Approved SO ID<br>{ "debitAccountNumber": "NEW-ACCT", "applicationStatus": "Amendment" } | 200 OK | Returns the updated SO object. approvalStatus is now "Submitted". applicationStatus is "Amendment". |
| **SO-PUT-002** | PUT /api/standing-orders/:id | **Negative: Update a Non-Existent SO** | id: 99999 | 404 Not Found | Error message: "Standing Order with ID 99999 not found." |
| **SO-PUT-003** | PUT /api/standing-orders/:id | **Security: Attempt to Update a Submitted SO** | id: Valid Submitted SO ID | 403 Forbidden | Error message: "Cannot modify a Standing Order that is pending approval." |
| **SO-TERM-001** | POST /api/standing-orders/:id/terminate | **Happy Path: Terminate an Approved SO** | id: Valid Approved SO ID<br>{ "finalDebit": "2025-12" } | 200 OK | Returns the updated SO object. approvalStatus is now "Submitted". applicationStatus is "Terminate". |
| **SO-TERM-002** | POST /api/standing-orders/:id/terminate | **Validation: Missing finalDebit date** | id: Valid Approved SO ID<br>{} | 400 Bad Request | Error message: "finalDebit is a required field for termination." |
| **SO-DEL-001** | DELETE /api/standing-orders/:id | **Happy Path: Delete a Submitted SO** | id: Valid Submitted SO ID<br>{ "notes": "User request" } | 204 No Content | The SO is successfully deleted. |
| **SO-DEL-002** | DELETE /api/standing-orders/:id | **Validation: Delete without mandatory notes** | id: Valid Submitted SO ID<br>{} | 400 Bad Request | Error message: "notes field is mandatory for deletion." |
| **SO-DEL-003** | DELETE /api/standing-orders/:id | **Security: Attempt to Delete Approved SO** | id: Valid Approved SO ID | 403 Forbidden | Error message: "Approved Standing Orders cannot be deleted." |

##### **1.4. Approve/Reject Standing Order**

| **Test Case ID** | **API Endpoint & Method** | **Test Scenario** | **Request Payload / Key Parameters** | **Expected HTTP Status** | **Expected Response Body / Outcome** |
| --- | --- | --- | --- | --- | --- |
| **SO-APP-001** | POST /api/standing-orders/:id/approve | **Happy Path: Approve a Pending SO** | id: Valid Pending SO ID<br>{ "notes": "Verified" } (optional) | 200 OK | Returns the SO object with approvalStatus: "Approved". Triggers fee booking. |
| **SO-APP-002** | POST /api/standing-orders/:id/approve | **Integration: Fee Debit Fails** | id: Valid Pending SO ID (for an account with insufficient funds for the fee) | 400 Bad Request | Error message: "Approval failed: Insufficient funds for registration fee." |
| **SO-APP-003** | POST /api/standing-orders/:id/approve | **Status Transition: Approve an Already Approved SO** | id: Valid Approved SO ID | 409 Conflict | Error message: "Standing Order is already in 'Approved' state." |
| **SO-APP-004** | POST /api/standing-orders/:id/approve | **Status Transition: Approve a Rejected SO** | id: Valid Rejected SO ID | 409 Conflict | Error message: "Cannot approve a 'Rejected' Standing Order. It must be resubmitted." |
| **SO-APP-005** | POST /api/standing-orders/:id/reject | **Happy Path: Reject a Pending SO** | id: Valid Pending SO ID<br>{ "notes": "Incorrect account number provided." } | 200 OK | Returns the SO object with approvalStatus: "Rejected". |
| **SO-APP-006** | POST /api/standing-orders/:id/reject | **Validation: Reject without mandatory notes** | id: Valid Pending SO ID<br>{} | 400 Bad Request | Error message: "notes field is mandatory for rejection." |
| **SO-APP-007** | POST /api/standing-orders/:id/approve | **Security: Forbidden (Maker Role)** | Authorization: Maker's JWT | 403 Forbidden | Error message: "User does not have permission to approve." |

#### **2. Payment Execution & Transaction Management (/api/transactions)**

| **Test Case ID** | **API Endpoint & Method** | **Test Scenario** | **Request Payload / Key Parameters** | **Expected HTTP Status** | **Expected Response Body / Outcome** |
| --- | --- | --- | --- | --- | --- |
| **PAY-EXEC-001** | POST /api/payments/execute-job | **Happy Path: Trigger Worker Job** | { "executionDate": "2025-08-04" } | 202 Accepted | Acknowledges that the batch job has been triggered for the given date. |
| **PAY-GET-001** | GET /api/transactions | **Happy Path: Get All Transactions** | (No parameters) | 200 OK | Returns a JSON array of all transaction objects. |
| **PAY-GET-002 - 012** | GET /api/transactions | **Filtering: Get by each of the 11 status types** | ?status=Paid, ?status=Balance Short, ?status=Overdue - Failed, etc. | 200 OK | Returns a JSON array of transactions only with the specified status. |
| **PAY-GET-013** | GET /api/transactions | **Filtering: Get by UI Status Group "Need Repair"** | ?statusGroup=NeedRepair | 200 OK | Returns transactions with statuses "Overdue - Balance Short" and "Overdue - Failed". |
| **PAY-GET-014** | GET /api/transactions/:id | **Happy Path: Get Transaction by ID** | id: Valid Transaction ID | 200 OK | Returns the JSON object for the specified transaction, including its status history. |
| **PAY-GET-015** | GET /api/transactions/:id | **Negative: Get Non-Existent Transaction** | id: 99999 | 404 Not Found | Error message: "Transaction with ID 99999 not found." |
| **PAY-REPAIR-001** | POST /api/transactions/:id/repair | **Happy Path: Repair a "Balance Short" Transaction** | id: Valid "Balance Short" Transaction ID<br>{ "notes": "Customer has topped up account" } | 202 Accepted | The transaction is submitted for approval. approvalStatus becomes "Submitted". |
| **PAY-REPAIR-002** | POST /api/transactions/:id/repair | **Status Transition: Attempt to Repair a "Paid" Transaction** | id: Valid "Paid" Transaction ID | 409 Conflict | Error message: "Cannot repair a transaction that is already 'Paid'." |
| **PAY-REPAIR-003** | POST /api/transactions/bulk-repair | **Happy Path: Bulk Repair** | { "transactionIds": [1, 2, 3], "notes": "Bulk repair initiated" } | 202 Accepted | Triggers the repair submission process for multiple transactions. |
| **PAY-UPDATE-001** | PUT /api/transactions/:id | **Happy Path: Update an Outstanding Transaction** | id: Valid Outstanding Transaction ID<br>{ "startDebitDate": "15" } | 202 Accepted | The transaction is submitted for approval. approvalStatus becomes "Submitted". |
| **PAY-UPDATE-002** | PUT /api/transactions/:id | **Status Transition: Attempt to Update a "Paid" Transaction** | id: Valid "Paid" Transaction ID | 403 Forbidden | Error message: "Cannot update a transaction that has already been processed." |
| **PAY-APP-001** | POST /api/transactions/:id/approve | **Happy Path: Approve a Repair Request** | id: Valid Transaction ID pending repair approval | 200 OK | The transaction approvalStatus becomes "Approved" and the re-execution job is triggered. |
| **PAY-APP-002** | POST /api/transactions/:id/reject | **Happy Path: Reject an Update Request** | id: Valid Transaction ID pending update approval<br>{ "notes": "..." } | 200 OK | The transaction approvalStatus becomes "Rejected". The original data is retained. |

#### **3. Product & Charges Management (/api/products, /api/charges)**

| **Test Case ID** | **API Endpoint & Method** | **Test Scenario** | **Request Payload / Key Parameters** | **Expected HTTP Status** | **Expected Response Body / Outcome** |
| --- | --- | --- | --- | --- | --- |
| **PROD-001** | POST /api/products | **Happy Path: Create New Product** | { "productCode": "NEW-PROD", "productName": "...", "category": "...", "dueDateType": "Non Personal", ... } | 201 Created | Returns the newly created product object with approvalStatus: "Submitted". |
| **PROD-002** | POST /api/products | **Validation: Duplicate productCode** | { "productCode": "PDAM-BTM", ... } | 409 Conflict | Error message: "Product with code 'PDAM-BTM' already exists." |
| **PROD-003** | GET /api/products | **Happy Path: Get All Products** | (No parameters) | 200 OK | Returns a JSON array of all product objects. |
| **PROD-004** | PUT /api/products/:id | **Happy Path: Update a Product** | id: Valid Product ID<br>{ "productName": "New Name" } | 200 OK | Returns the updated product object with approvalStatus: "Submitted". |
| **PROD-005** | DELETE /api/products/:id | **Happy Path: Delete a Product** | id: Valid Product ID<br>{ "notes": "..." } | 200 OK | Returns the product object with approvalStatus: "Deleted". (Soft delete) |
| **PROD-006** | POST /api/products/:id/approve | **Happy Path: Approve New Product Creation** | id: Valid "Submitted" Product ID | 200 OK | Product approvalStatus becomes "Approved". |
| **CHG-001** | POST /api/charges | **Happy Path: Create New Charge** | { "chargeKey": "Standing Order Creation", "value": 5000, "group": "Standing Order" } | 201 Created | Returns the newly created charge object with approvalStatus: "Submitted". |
| **CHG-002** | GET /api/charges | **Happy Path: Get All Charges** | (No parameters) | 200 OK | Returns a JSON array of all charge objects. |
| **CHG-003** | PUT /api/charges/:id | **Happy Path: Update a Charge** | id: Valid Charge ID<br>{ "value": 6000 } | 200 OK | Returns the updated charge object with approvalStatus: "Submitted". |
| **CHG-004** | DELETE /api/charges/:id | **Happy Path: Delete a Charge** | id: Valid Charge ID<br>{ "notes": "..." } | 200 OK | Returns the charge object with approvalStatus: "Deleted". |
| **CHG-005** | POST /api/charges/:id/approve | **Happy Path: Approve Charge Update** | id: Valid "Submitted" Charge ID | 200 OK | Charge approvalStatus becomes "Approved". |

#### **4. Reconciliation (/api/reconciliation)**

| **Test Case ID** | **API Endpoint & Method** | **Test Scenario** | **Request Payload / Key Parameters** | **Expected HTTP Status** | **Expected Response Body / Outcome** |
| --- | --- | --- | --- | --- | --- |
| **RECON-001** | POST /api/reconciliation/run | **Happy Path: Trigger Reconciliation** | { "date": "2025-08-04", "product": "All Biller except BPJS" } | 202 Accepted | Acknowledges that the reconciliation job has started. |
| **RECON-002** | GET /api/reconciliation/summary | **Happy Path: Get Reconciliation Summary** | ?date=... | 200 OK | Returns a summary object: { "totalPU": ..., "totalBiller": ..., "matched": ..., "mismatched": ... } |
| **RECON-003** | GET /api/reconciliation/details | **Happy Path: Get Reconciliation Details** | ?date=... | 200 OK | Returns a JSON array of detailed reconciliation records. |
| **RECON-004** | GET /api/reconciliation/details | **Filtering: Get Mismatched Details** | ?date=...&status=Mismatch | 200 OK | Returns a JSON array of only the mismatched reconciliation records. |
| **RECON-005** | POST /api/reconciliation/details/:id/remarks | **Happy Path: Add Remark** | id: Valid Detail ID<br>{ "notes": "Investigating amount difference." } | 201 Created | Returns the newly created remark object. |
| **RECON-006** | GET /api/reconciliation/details/:id/remarks | **Happy Path: View Remark History** | id: Valid Detail ID | 200 OK | Returns a JSON array of all remarks for that detail line. |
| **RECON-007** | POST /api/reconciliation/details/:id/match | **Happy Path: Manually Match Data** | id: Valid Mismatched Detail ID | 200 OK | The status of the detail line is updated to "Matched". |
| **RECON-008** | POST /api/reconciliation/details/:id/match | **Negative: Match an already Matched record** | id: Valid Matched Detail ID | 409 Conflict | Error message: "Record is already matched." |