Solution Design Document (SDD)

Bank Statement Reconciliation Automation

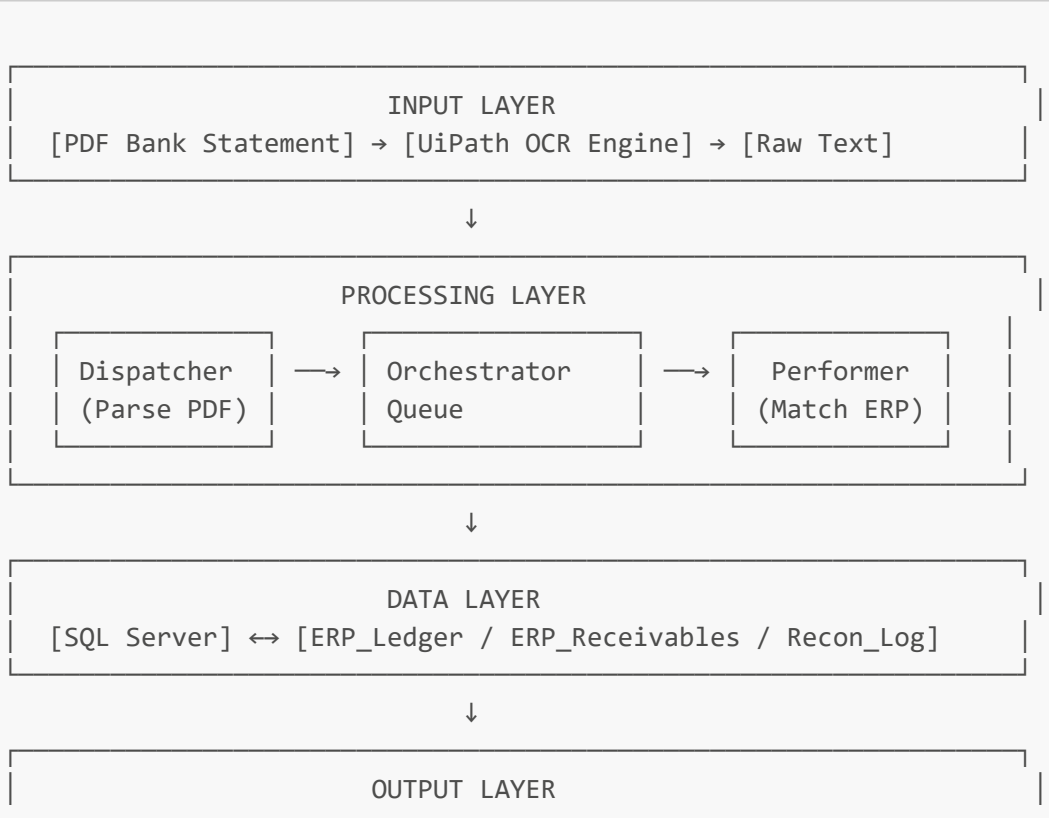
Document Information	
Document Title	Solution Design Document
Process Name	Bank Statement Reconciliation
Version	1.0
Author	RPA Developer_Riley Liao
Date	October 2025
Status	Approved

1. Executive Summary

This document provides the technical design for automating the Bank Statement Reconciliation process using UiPath RPA platform. The solution implements a **Dispatcher-Performer pattern** with Orchestrator Queue management, enabling scalable and fault-tolerant transaction processing.

2. Solution Architecture

2.1 High-Level Architecture



```
│  [Orchestrator Logs] + [Email Notifications] + [Reports]        │
└─────────────────────────────────────────────────────────────────┘
```

2.2 Technology Stack

Component	Technology	Version
RPA Platform	UiPath Studio	2024.10
Framework	REFramework	Standard
Orchestrator	UiPath Orchestrator	Cloud
Database	Microsoft SQL Server	2022
OCR Engine	UiPath.OCR.Activities	Latest
Programming	VB.NET	.NET 6
IDE	Visual Studio Code	Latest

2.3 Design Patterns

Pattern	Purpose
Dispatcher-Performer	Separate data extraction from processing
REFramework	Enterprise-grade transaction handling
Queue-Based Processing	Scalable, fault-tolerant architecture
Config-Driven	Externalized settings for maintainability

---

3. Component Design

3.1 Dispatcher Process

Project Name: `BankRecon_Dispatcher`

File	Purpose
Main.xaml	Entry point, orchestrates workflow
ReadBankStatement.xaml	PDF file reading
ExtractText.xaml	OCR text extraction
ParseTransactions.xaml	Regex pattern matching
AddToQueue.xaml	Orchestrator queue insertion

Key Variables:

Variable	Type	Description

Variable	Type	Description
str_FilePath	String	PDF file location
str_OcrText	String	Extracted OCR text
dt_Transactions	DataTable	Parsed transaction data
int_ItemsAdded	Int32	Counter for queue items

3.1.1 OCR Strategy & Fallback Mechanism

PDF Type Detection:

PDF Type	Detection Method	Extraction Approach
Native Digital	Text layer present	Read PDF Text (no OCR needed)
Scanned Image	No text layer	UiPath Document OCR / Tesseract
Mixed	Partial text	Hybrid: Text + OCR for images

Extraction Fallback Logic:

```
Try: Read PDF Text (native)
├── Success → Proceed to Regex parsing
└── Fail (no text) →
    Try: UiPath Document OCR
    ├── Success → Proceed to Regex parsing
    └── Low confidence (<80%) → Flag for manual review
```

Regex Failure Handling:

Missing Field	Action	Queue Status
Invoice_No only	Add with Status="Review Required"	New
Amount missing	Skip, log to exception file	Not Added
Date missing	Use current date as fallback	New
All fields missing	Skip, log exception	Not Added

3.1.2 Date Format Standardization

Bank statements may use various date formats:

Source Format	Example	Converted
DD-MMM-YY	12-Oct-25	2025-10-12
MM/DD/YYYY	10/12/2025	2025-10-12

Source Format	Example	Converted
DD.MM.YYYY	12.10.2025	2025-10-12

Requirement: Standardize all extracted dates to `yyyy-MM-dd` format in `ParseTransactions.xaml` before adding to Queue.

```
' VB.NET Date Normalization
Dim standardizedDate As String = DateTime.Parse(rawDate).ToString("yyyy-MM-dd")
```

> ⚠ **Critical:** This prevents DateTime conversion failures when Performer writes to SQL Server.

3.2 Performer Process

Project Name: `BankRecon_Performer`

File	Purpose
Main.xaml	REFramework entry point
InitAllSettings.xaml	Load Config.xlsx
GetTransactionData.xaml	Get queue item
Process.xaml	Core matching logic
SetTransactionStatus.xaml	Update queue item status

Key Variables:

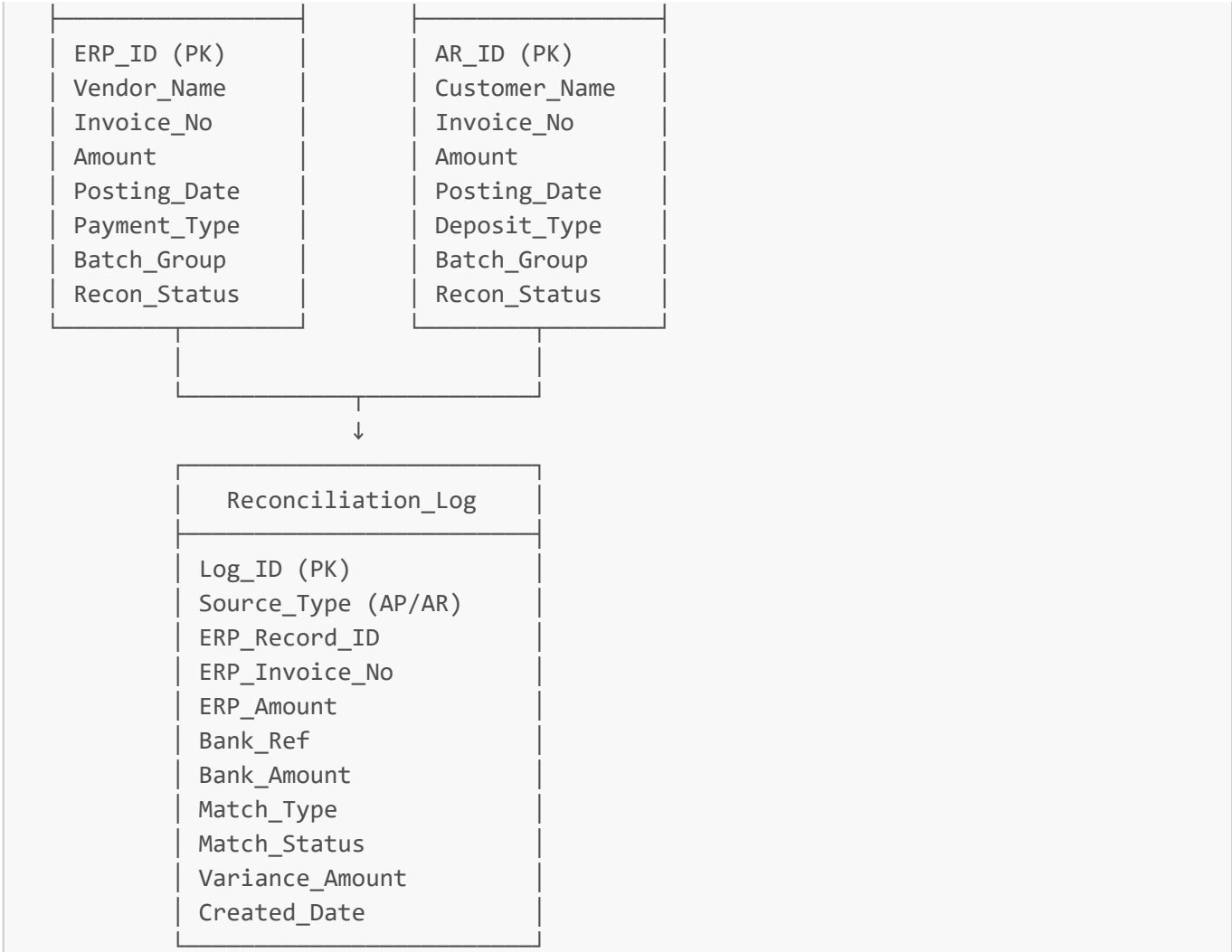
Variable	Type	Description
in_TransactionItem	QueueItem	Current transaction
in_Config	Dictionary	Configuration settings
str_InvoiceNo	String	Invoice number from queue
dec_Amount	Decimal	Transaction amount
str_MatchType	String	Match result type
dt_ExactMatch	DataTable	SQL query result

---

4. Database Design

4.1 Entity Relationship Diagram

```
┌──────────────────┐       ┌──────────────────┐
│   ERP_Ledger     │       │ ERP_Receivables  │
│   (AP Table)     │       │   (AR Table)     │
```



4.2 Table Definitions

ERP_Ledger (Accounts Payable)

```
CREATE TABLE ERP_Ledger (
    ERP_ID NVARCHAR(20) PRIMARY KEY,
    Vendor_Name NVARCHAR(100),
    Invoice_No NVARCHAR(50) NOT NULL,
    Amount DECIMAL(18,2) NOT NULL,
    Posting_Date DATE,
    Payment_Type NVARCHAR(20) DEFAULT 'STANDARD',
    Batch_Group NVARCHAR(30),
    Status NVARCHAR(20) DEFAULT 'Paid',
    Reconciliation_Status NVARCHAR(50) DEFAULT 'Pending',
    Matched_Bank_Ref NVARCHAR(50),
    Matched_Date DATETIME,
    INDEX IX_Invoice (Invoice_No),
    INDEX IX_Status (Reconciliation_Status)
);
```

ERP_Receivables (Accounts Receivable)

```
CREATE TABLE ERP_Receivables (
    AR_ID NVARCHAR(20) PRIMARY KEY,
    Customer_Name NVARCHAR(100),
    Invoice_No NVARCHAR(50) NOT NULL,
    Amount DECIMAL(18,2) NOT NULL,
    Posting_Date DATE,
    Deposit_Type NVARCHAR(20) DEFAULT 'STANDARD',
    Batch_Group NVARCHAR(30),
    Reconciliation_Status NVARCHAR(50) DEFAULT 'Pending',
    Matched_Bank_Ref NVARCHAR(50),
    Matched_Date DATETIME,
    INDEX IX_Invoice (Invoice_No),
    INDEX IX_Status (Reconciliation_Status)
);
```

Reconciliation_Log

```
CREATE TABLE Reconciliation_Log (
    Log_ID INT IDENTITY(1,1) PRIMARY KEY,
    Source_Type NVARCHAR(10) NOT NULL,
    ERP_Record_ID NVARCHAR(20),
    ERP_Invoice_No NVARCHAR(50),
    ERP_Amount DECIMAL(18,2),
    Bank_Ref NVARCHAR(50),
    Bank_Amount DECIMAL(18,2),
    Match_Type NVARCHAR(50),
    Match_Status NVARCHAR(20),
    Matched_Amount DECIMAL(18,2),
    Variance_Amount DECIMAL(18,2),
    Batch_Group NVARCHAR(30),
    -- 🔗 Link back to Orchestrator for debugging
    Orchestrator_Reference_ID NVARCHAR(100),  -- Queue Item Key
    Orchestrator_Job_ID NVARCHAR(50),         -- Job execution ID
    Created_Date DATETIME DEFAULT GETDATE(),
    INDEX IX_MatchType (Match_Type),
    INDEX IX_Created (Created_Date),
    INDEX IX_Orchestrator (Orchestrator_Reference_ID)
);
```

---

5. Matching Algorithm

5.1 Decision Flow

```
START
  ↓
[Get Queue Item]
  ↓
```

```
[Identify Transaction Type]
  ├── RCPT-/DEP-/TFR- prefix → AR (ERP_Receivables)
  └── INV- prefix → AP (ERP_Ledger)
  ↓
[Query ERP by Invoice_No]
  ├── Not Found → [Secondary Matching Logic]
  │                 ├── Found by Vendor + Amount + Date → "Fuzzy Match
(Secondary)"
  │                 └── Still Not Found → BusinessRuleException
  └── Found → [Check Payment_Type]
                ├── BATCH → [Sum Batch Group]
                │             ├── Match → "Matched - Batch"
                │             └── No Match → "Mismatch"
                ├── SPLIT → [Sum Previous Payments]
                │             ├── Complete → "Matched - Split"
                │             └── Incomplete → "Partial"
                └── DEFAULT → [Calculate Variance]
                               ├── Variance = 0 → "Exact Match"
                               ├── Variance ≤ Tolerance → "Fuzzy Match"
                               └── Variance > Tolerance → "Mismatch"
  ↓
[INSERT to Reconciliation_Log]
  ↓
END
```

5.2 Secondary Matching Logic

When Invoice_No is not found or unclear in bank description:

```sql
-- Secondary matching: Vendor + Amount + Date Range
SELECT TOP 1 Invoice_No, Vendor_Name, Amount, Posting_Date
FROM ERP_Ledger
WHERE Reconciliation_Status = 'Pending'
  AND Amount BETWEEN @BankAmount - @Tolerance AND @BankAmount + @Tolerance
  AND Posting_Date BETWEEN DATEADD(day, -@ToleranceDays, @BankDate)
                       AND DATEADD(day, @ToleranceDays, @BankDate)
  AND Vendor_Name LIKE '%' + @VendorKeyword + '%'
ORDER BY ABS(Amount - @BankAmount) ASC
```

5.3 Batch/Split Concurrency Control

Problem: Multiple robots processing the same Batch_Group simultaneously could cause duplicate matching.

Solution: SQL Transaction Lock

```sql
BEGIN TRANSACTION

-- Lock the batch group records
SELECT * FROM ERP_Ledger WITH (UPDLOCK, ROWLOCK)
```

```
WHERE Batch_Group = @BatchGroup
  AND Reconciliation_Status = 'Pending'

-- Process matching logic here...

-- Update status atomically
UPDATE ERP_Ledger
SET Reconciliation_Status = 'Matched'
WHERE Batch_Group = @BatchGroup

COMMIT TRANSACTION
```

> 💡 **Best Practice:** Complex matching logic (Section 5.2 & 5.3) should ideally be encapsulated in **SQL Stored Procedures** (e.g., `sp_MatchTransactions`) rather than raw SQL in UiPath.
>
> **Benefits:**
>
> - DBA can optimize query performance independently
> - Logic changes (e.g., tolerance values) don't require UiPath package republication
> - Easier to unit test SQL logic in isolation

5.2 Variance Calculation

```
' VB.NET Expression
dec_Variance = Math.Abs(CDec(dt_ExactMatch.Rows(0)("Amount")) - dec_Amount)
```

5.3 Match Type Classification

Condition	Match Type	Match Status
Variance = 0	Exact Match	Matched
0 < Variance ≤ Tolerance	Fuzzy Match	Matched
Variance > Tolerance	Mismatch	Failed
Batch Sum = Bank Amount	Matched - Batch	Matched
Cumulative ≥ ERP Amount	Matched - Split	Matched
Cumulative < ERP Amount	Partial	Pending

6. Exception Handling

6.1 Exception Matrix

Exception Type	Trigger	Retry	Action
BusinessRuleException	No ERP record found	No	Email alert, log exception

Exception Type	Trigger	Retry	Action
System Exception	Database timeout	Yes (3x)	Retry, then escalate
Application Exception	Unexpected error	Yes (3x)	Screenshot, retry

6.2 Notification Strategy

Anti-Spam Design: Avoid sending individual emails for each Business Exception.

Exception Type	Notification	Timing
System Exception	Immediate email alert	Real-time
Business Exception	Aggregated in daily report	End of Process

Daily Summary Report (End Process):

```
Subject: [Bank Recon] Daily Summary Report - {Date}

Processed: {TotalCount} transactions
Matched: {MatchedCount} ({MatchedPercent}%)
Failed: {FailedCount} items require manual review

[Attached: BankRecon_Exceptions_{Date}.xlsx]
```

Exception Excel Template:

Bank_Ref	Invoice_No	Amount	Error_Reason	Suggested_Action
REF001102	INV-NOTFOUND	$8,888	No ERP record	Verify in SAP
REF001103	INV-MISMATCH	$5,200	Variance $150 > $30	Check payment terms

7. Configuration Management

7.1 Config.xlsx Structure

Settings Sheet:

Name	Value	Description
OrchestratorQueueName	Bank_Recon_Queue	Queue name
logF_BusinessProcessName	BankRecon	Log folder prefix
MaxRetryNumber	3	Retry count

Constants Sheet:

Name	Value	Description

Name	Value	Description
Tolerance_Amount	30	Amount tolerance ($)
Tolerance_Days	3	Date tolerance (days)
Report_Email_To	finance@company.com	Notification email

Assets Sheet:

Name	Type	Description
DB_ConnectionString	Credential	SQL Server connection
SMTP_Credential	Credential	Email server login

---

8. Security Considerations

Area	Implementation
Database Access	SQL Server Windows Authentication
Credentials	Stored in Orchestrator Assets
Connection Strings	Encrypted in Config
Audit Trail	All transactions logged
Data Retention	90 days in Reconciliation_Log

---

9. Performance Requirements

Metric	Requirement
Transaction processing time	< 30 seconds per item
Queue throughput	100 items/minute
Database query response	< 2 seconds
System availability	99.5% uptime
Concurrent robots	Up to 5

---

10. Testing Strategy

10.1 Test Scenarios

Test Case	Input	Expected Output
TC-001	Exact match invoice	Status: Matched
TC-002	Fuzzy match (amount)	Status: Fuzzy Match

Test Case	Input	Expected Output
TC-003	Fuzzy match (date)	Status: Fuzzy Match
TC-004	BATCH payment	Status: Matched - Batch
TC-005	SPLIT payment	Status: Matched - Split
TC-006	No ERP record	BusinessRuleException
TC-007	Database timeout	System Exception + Retry

10.2 Test Data

- 9 Exact Match transactions
- 2 Fuzzy Match transactions
- 1 BATCH payment (3 invoices)
- 1 SPLIT payment (2 installments)
- 1 Unmatched transaction

---

11. Deployment

11.1 Package Structure

```
BankRecon_Dispatcher/
├── project.json
├── Main.xaml
├── Data/
│   └── Config.xlsx
└── .local/

BankRecon_Performer/
├── project.json
├── Main.xaml
├── Framework/
│   ├── InitAllSettings.xaml
│   ├── GetTransactionData.xaml
│   ├── Process.xaml
│   └── SetTransactionStatus.xaml
├── Data/
│   └── Config.xlsx
└── .local/
```

11.2 Orchestrator Setup

Component	Configuration
Queue	Bank_Recon_Queue
Process (Dispatcher)	BankRecon_Dispatcher

Component	Configuration
Process (Performer)	BankRecon_Performer
Trigger (Dispatcher)	Daily 6:00 AM
Trigger (Performer)	Queue-based (continuous)
Robot	Unattended Robot

12. Maintenance

12.1 Monitoring

Metric	Tool	Threshold
Queue length	Orchestrator Dashboard	Alert if > 100
Failed transactions	Orchestrator Monitoring	Alert if > 5%
Processing time	Logs	Alert if > 60 sec

12.2 Support Procedures

Issue	Resolution
Bot unresponsive	Restart robot from Orchestrator
Database connection failed	Check SQL Server status, VPN
OCR quality issues	Review PDF quality, adjust patterns

13. Approval

Role	Name	Signature	Date
Solution Architect			
RPA Lead			
Database Administrator			
Security Officer			

Document End