**Bank Statement PoC - API Specification Document**

**Common Conventions**

* **Base URL**: /api/v1
* **Content-Type**: application/json unless noted (e.g., multipart/form-data for file uploads)
* **Authentication**: (Optional for PoC – can be added later)
* **File Types Supported**: .pdf, .jpg, .png, .xls, .xlsx, .csv, .msg

**1. Upload Service**

**Endpoint: POST /upload**

**Description**: Upload a document (bank statement, receipt, invoice). Store, validate, and extract text.

**Request Body Content-Type**: multipart/form-data

**Request Body:**

| **Field** | **Type** | **Required** | **Description** |
| --- | --- | --- | --- |
| file | File | Yes | Document to be uploaded |

**Flow:**

1. Save file locally.
2. Validate file (blur, brightness, corruption, etc.)
3. Extract text using OCR for image, PDF parser for PDF, etc.
4. Determine document type: bank, receipt, or invoice (basic classifier or rule-based for now).
5. Extract relevant fields (placeholders for now if extractor is WIP).

**Success Response: Bank Transaction**

{

"status": "success",

"document\_type": "bank\_transaction",

"extracted\_fields": {

"date": "2024-06-10",

"description": "POS PURCHASE AMAZON",

"debit": 1200.00,

"credit": null,

"balance": 35000.75

}

}

**Success Response: Receipt**

{

"status": "success",

"document\_type": "receipt",

"extracted\_fields": {

"date": "2024-06-09",

"store\_name": "Big Bazaar",

"amount": 850.00,

"description": "Grocery shopping",

"tax\_code": "GSTIN1234XYZ"

}

}

**Success Response: Invoice**

{

"status": "success",

"document\_type": "invoice",

"extracted\_fields": {

"date": "2024-06-08",

"supplier": "Flipkart Seller Pvt. Ltd.",

"amount": 12500.00,

"description": "Electronic purchase",

"tax\_code": "GSTIN5678ABC"

}

}

**Response (Failure - Image Validation):**

{

"status": "error",

"error": "Blurry image detected. Please reupload with better quality."

}

**2. Transaction Matching Service**

**Tables Involved:**

* **Transaction Table**: transaction\_id, date, supplier, amount
* **Receipt Table**: receipt\_id, date, supplier, amount, ocr\_text

**GET /transactions**

**Description**: Get all transaction records.  
**Response**:

[

{

"transaction\_id": "txn\_001",

"date": "2024-06-10",

"supplier": "Amazon",

"amount": 1200.00

},

{

"transaction\_id": "txn\_002",

"date": "2024-06-11",

"supplier": "Uber",

"amount": 350.00

}

]

**GET /receipts**

**Description**: Get all receipt records.  
**Response**:

[

{

"receipt\_id": "rcp\_001",

"date": "2024-01-05",

"supplier": "Amazon",

"amount": 5000,

"ocr\_text": "..."

}

]

**GET /transaction/{transaction\_id}**

**Description**: Get a transaction by ID.  
**Path Param**: transaction\_id

**Example**:  
GET /transaction/txn\_001

**Response**:

{

"transaction\_id": "txn\_001",

"date": "2024-01-05",

"supplier": "Amazon",

"amount": 5000

}

**GET /receipt/{receipt\_id}**

**Description**: Get a receipt by ID.  
**Path Param**: receipt\_id

**Example**:  
GET /receipt/rcp\_001  
**Response**:

{

"receipt\_id": "rcp\_001",

"date": "2024-01-05",

"supplier": "Amazon",

"amount": 5000,

"ocr\_text": "..."

}

**GET /transaction-matches/{transaction\_id}**

**Description**: Return a list of receipts/invoices that match the given transaction.

**Example**:  
GET /transaction-matches/txn\_001

**Response**:

{

"transaction\_id": "txn\_001",

"matched\_receipts": [

{

"receipt\_id": "rcp\_001",

"confidence\_score": 0.95

}

]

}

**POST /transaction-matches/by-data**

**Description**: Identify matches from transaction data

**Request Body**:

{

"date": "2024-06-10",

"supplier": "Amazon",

"amount": 1200.00

}

**Response**:

{

"matched\_receipts": [

{

"receipt\_id": "rcp\_001",

"confidence\_score": 0.93

}

]

}

**GET /transactions/unmatched**

**Description**: Get list of unmatched transactions.  
**Response**:

[

{

"transaction\_id": "txn\_005",

"date": "2024-01-09",

"supplier": "XYZ Store",

"amount": 1100

}

]

**3. Transaction Categorization Service**

**Table Involved:**

* **Transaction Table**: transaction\_id,date,supplier,amount,description,tax\_code,category

**GET /transactions**

**Response:**

[

{

"transaction\_id": "txn\_001",

"date": "2024-06-10",

"supplier": "Amazon",

"amount": 1200.00,

"category": "Shopping"

},

{

"transaction\_id": "txn\_002",

"date": "2024-06-11",

"supplier": "Uber",

"amount": 350.00,

"category": "Travel"

}

]

**GET /categories**

**Description**: Get all categories found in transactions.  
**Response**:

["Food", "Travel", "Utilities", "Shopping", "Healthcare"]

**GET /transaction-category/{transaction\_id}**

**Description**: Get predicted category for a transaction.

**Example**:  
GET /transaction-category/txn\_001  
**Response**:

{

"transaction\_id": "txn\_001",

"category": "Shopping",

"confidence": 0.87

}

**POST /transaction-category/by-data**

**Description**: Determine the category of a transaction based on user-provided details.

**Request Body**:

{

"date": "2024-06-10",

"supplier": "Amazon",

"amount": 1200.00

}

**Response**:

{

"category": "Shopping",

"confidence": 0.87

}

**4. Fraud/Anomaly and Duplicate Detection Service**

**Tables Involved:**

* **Duplicates Table**: transaction\_id,amount,supplier,timestamp,description
* **Anamoly Table**: transaction\_id,amount,supplier,timestamp
* **Fuzy Duplicates Table:** transaction\_id,ocr\_text

**GET /transactions**

**Response:**

[

{

"transaction\_id": "txn\_001",

"date": "2024-06-10",

"supplier": "Amazon",

"amount": 1200.00

}

]

**GET /transactions/duplicates**

**Description**: Get all duplicate transactions.  
**Response**:

[

{

"transaction\_id": "txn\_001",

"duplicate\_ids": ["txn\_010", "txn\_011"],

"match\_type": "exact"

},

{

"transaction\_id": "txn\_001",

"duplicate\_ids": ["txn\_010", "txn\_011"],

"match\_type": "fuzzy"

}

]

**GET /transactions/duplicates/{transaction\_id}**

**Description**: Get duplicates of a specific transaction.

**Example**:  
GET /transactions/duplicates/txn\_001  
**Response**:

[

{

"transaction\_id": "txn\_010",

"similarity\_score": 0.99

}

]

**POST /transactions/duplicates/by-data**

**Request Body**:

{

"date": "2024-06-10",

"supplier": "Amazon",

"amount": 1200.00

}

**Response**:

[

{

"transaction\_id": "txn\_010",

"similarity\_score": 0.98

}

]

**GET /transactions/anomalies**

**Description**: Get all anomalous or fraudulent transactions.  
**Response**:

[

{

"transaction\_id": "txn\_007",

"confidence": 0.92

}

]

**GET /transactions/check-anomaly/{transaction\_id}**

**Description**: Check if a specific transaction is anomalous or fraudulent.

**Example**:  
GET /transactions/check-anomaly/txn\_007  
**Response**:

{

"transaction\_id": "txn\_007",

"is\_anomaly": true,

"confidence": 0.88

}

**POST /transactions/check-anomaly/by-data**

**Request Body**:

{

"date": "2024-06-10",

"supplier": "Some Store",

"amount": 45000.00

}

**Response**:

{

"is\_anomaly": true,

"confidence": 0.93

}