**Use Case 1: Migrate Existing i-Scanned-Images from Oracle to AWS S3**

1. **Fetch Images from Oracle**
   * Query Oracle for records where status = 'PENDING'.
   * Retrieve **ISI Barcode**, image BLOB, and metadata (file type, size, creation date, etc.).
2. **Process and Prepare Image Data**
   * Convert the BLOB to an appropriate image format (JPEG, PNG, etc.).
   * Compress the image if isiFileCompressed = true.
3. **Upload Image to AWS S3 via IBM API Connect**
   * Call the **IBM APIC API** (acts as a gateway) to upload the image to S3.
   * S3 returns a **file URL** after a successful upload.
4. **Update Oracle with Migration Status**
   * Update the Oracle table with the S3 file URL and set status = 'COMPLETED'.
   * If the upload fails, set status = 'FAILED' and store the error message.
5. **Scheduled Batch Execution**
   * Implement a **Spring Scheduler** (@Scheduled) to periodically process pending images.
   * Ensure retry mechanisms for failed uploads.

**Use Case 2: Expose Spring Boot Microservice for CRUD Operations on Images**

1. **Spring Boot REST API** to handle CRUD operations:
   * POST /image/upload → Upload image to AWS S3 and store metadata in Oracle.
   * GET /image/{isibarcode} → Retrieve image metadata from Oracle and fetch image from S3.
   * PUT /image/{isibarcode} → Update existing image in S3 and update metadata in Oracle.
   * DELETE /image/{isibarcode} → Delete image from S3 and update status in Oracle.
2. **Process Incoming JSON Payload**
   * Expect fields: isibarcode, isifileType, isiFileSizeKB, isidatecreated, isifile (Base64).
   * Validate input before processing the request.
3. **Store Image Metadata in Oracle**
   * Save ISI Barcode, file type, size, creation date, and S3 file URL.
4. **Upload Image to AWS S3**
   * Call IBM APIC to securely upload the file to AWS S3.
   * Receive a **S3 URL** and store it in Oracle.
5. **Update Status in Oracle**
   * After successful operations, update status = 'COMPLETED'.
   * If an error occurs, update status = 'FAILED' with an error log.
6. **Authentication & Security**
   * Secure API endpoints with OAuth2, JWT, or API keys via IBM API Connect.
   * Implement access control to ensure only authorized users can perform CRUD operations.
7. **Logging & Monitoring**
   * Log every operation for auditing purposes.
   * Implement monitoring with **AWS CloudWatch** and database triggers.

**Outcome**

✅ Efficient migration of existing scanned images from Oracle to AWS S3.  
✅ A scalable Spring Boot microservice for real-time CRUD operations on images.  
✅ Secure image storage and retrieval via IBM APIC.

|  |  |  |
| --- | --- | --- |
| **Step** | **Exception Description** | **Handling Scenario** |
| Fetch Images from Oracle | No records found with status = 'PENDING' | Log and continue. No action needed. |
| Fetch Images from Oracle | Database connection failure | Retry mechanism and alerting system to notify admin. |
| Process and Prepare Image Data | Unsupported file format | Log and mark status as FAILED in Oracle. Notify admin. |
| Upload Image to AWS S3 | API request timeout | Retry with exponential backoff, log failure if retries exceed limit. |
| Upload Image to AWS S3 | Invalid credentials/API key expired | Log the issue, notify admin, and retry after fetching a new token. |
| Update Oracle with Migration Status | Database connection failure | Retry logic; if persistent, store in a retry queue and process later. |
| Spring Boot API - Upload | Invalid JSON payload | Return 400 Bad Request with error details. |
| Spring Boot API - Upload | S3 upload failure | Return 500 Internal Server Error, log the issue, and retry. |
| Spring Boot API - Retrieve | Image not found in Oracle | Return 404 Not Found with an appropriate message. |
| Spring Boot API - Delete | Deletion failure in S3 | Log the issue, retry deletion, and update Oracle accordingly. |
| Authentication & Security | Unauthorized API access | Return 401 Unauthorized with a meaningful error message. |

**Outcome**

✅ Efficient migration of existing scanned images from Oracle to AWS S3.  
✅ A scalable Spring Boot microservice for real-time CRUD operations on images.  
✅ Secure image storage and retrieval via IBM APIC.  
✅ Robust exception handling with retry mechanisms and logging.