**Documentation of Components, Data Flow, and Risk Assessment Logic**

**1. Components of the System**

**AWS Lambda Functions:**

* **processTransaction:** Handles incoming POST requests containing financial transaction data. Performs data anonymization, enrichment, risk assessment, encryption, and stores processed data securely.
* **retrieveTransaction:** Retrieves processed transaction data based on transaction ID.

**API Gateway:**

* Manages HTTP endpoints (/transaction for processing, /transaction/{id} for retrieval) that interface with Lambda functions.

**AWS S3:**

* Stores encrypted processed data securely (not implemented in local setup).

**Encryption and Anonymization:**

* Utilizes AES-256 encryption and SHA-256 hashing for data security and anonymization of sensitive personal identifiers.

**External Data Enrichment:**

* Integrates with an external API (mocked in local setup) to enrich transaction data with currency conversion rates.

**2. Flow of Data**

1. **Incoming Request (API Gateway):**
   * Receives POST request with financial transaction data in JSON format.
2. **Data Processing (processTransaction Lambda):**
   * **Anonymization:** Personal identifiers (userId, name, email, phone) are anonymized using SHA-256 hashing.
   * **External Data Enrichment:** Retrieves currency conversion rates and enriches transaction data (amountUSD).
   * **Risk Assessment:** Computes risk score based on transaction amount, frequency, geographical location, and past behavior.
   * **Encryption:** Encrypts enriched data (including risk score) using AES-256 and RSA encryption for key management.
3. **Storage (S3 - not implemented in local setup):**
   * Securely stores encrypted transaction data for future retrieval.
4. **Data Retrieval (retrieveTransaction Lambda):**
   * Handles GET request with transaction ID to retrieve encrypted transaction data.

**3. Logic Behind Risk Assessment Algorithms**

* **Amount Risk Assessment:**
  + Calculates risk based on transaction amount relative to a high-risk threshold.
* **Frequency Risk Assessment:**
  + Analyzes transaction frequency within a past month to detect potential anomalies.
* **Geographical Risk Assessment:**
  + Flags transactions originating from high-risk geographical locations.
* **Past Behavior Risk Assessment:**
  + Evaluates transaction consistency based on user and merchant category, identifying anomalies in transaction patterns.
* **Aggregate Risk Score Calculation:**
  + Combines individual risk assessments using weighted factors to compute an overall risk score for the transaction.

**Detailed Guide on Setting Up and Testing**

**Setup Instructions**

**Prerequisites:**

* Install Node.js and npm from [nodejs.org](https://nodejs.org/).
* Install Serverless Framework globally: npm install -g serverless.
* Configure AWS CLI with appropriate credentials.

**Steps:**

1. **Clone the Repository:**

bash

git clone <repository-url>

cd financial-app

1. **Install Dependencies:**

bash

npm install

1. **Configure AWS Credentials:**
   * Set up AWS credentials using AWS CLI: aws configure.
2. **Deploy the Application:**

bash

serverless deploy

This command deploys Lambda functions, API Gateway, and other configured AWS resources.

1. **Testing Locally:**
   * Start Serverless Offline for local testing:

bash

serverless offline start

* + Use tools like Postman or curl to send POST requests to http://localhost:3000/transaction with sample transaction data.

**Unit Testing**

1. **Unit Tests for Components:**
   * Write unit tests using frameworks like Jest or Mocha.
   * Focus on testing:
     + Encryption and decryption functions.
     + Anonymization functions.
     + Risk assessment algorithms.
2. **Integration Testing:**
   * Test the integration between Lambda functions and AWS services (S3, API Gateway).
   * Verify end-to-end functionality of data processing and retrieval.

**Documentation**

* **Component Documentation:**
  + Provide detailed descriptions of each Lambda function, API Gateway setup, and S3 integration.
  + Include code snippets and diagrams illustrating the flow of data.
* **Setup Guide Documentation:**
  + Step-by-step instructions on setting up the development environment, deploying the application, and configuring AWS services.
  + Include screenshots or command outputs where necessary.
* **Testing Documentation:**
  + Document test scenarios and expected outcomes.
  + Include instructions for running unit tests, integration tests, and local testing.

**Conclusion**

This comprehensive documentation and setup guide ensure clarity in understanding the serverless TypeScript application for financial data processing. Developers can follow these steps to deploy, test, and maintain the application effectively in AWS Lambda environment. Adjustments can be made based on specific project requirements and AWS configuration details.