Tab 1

ATM Monitoring System - System Design

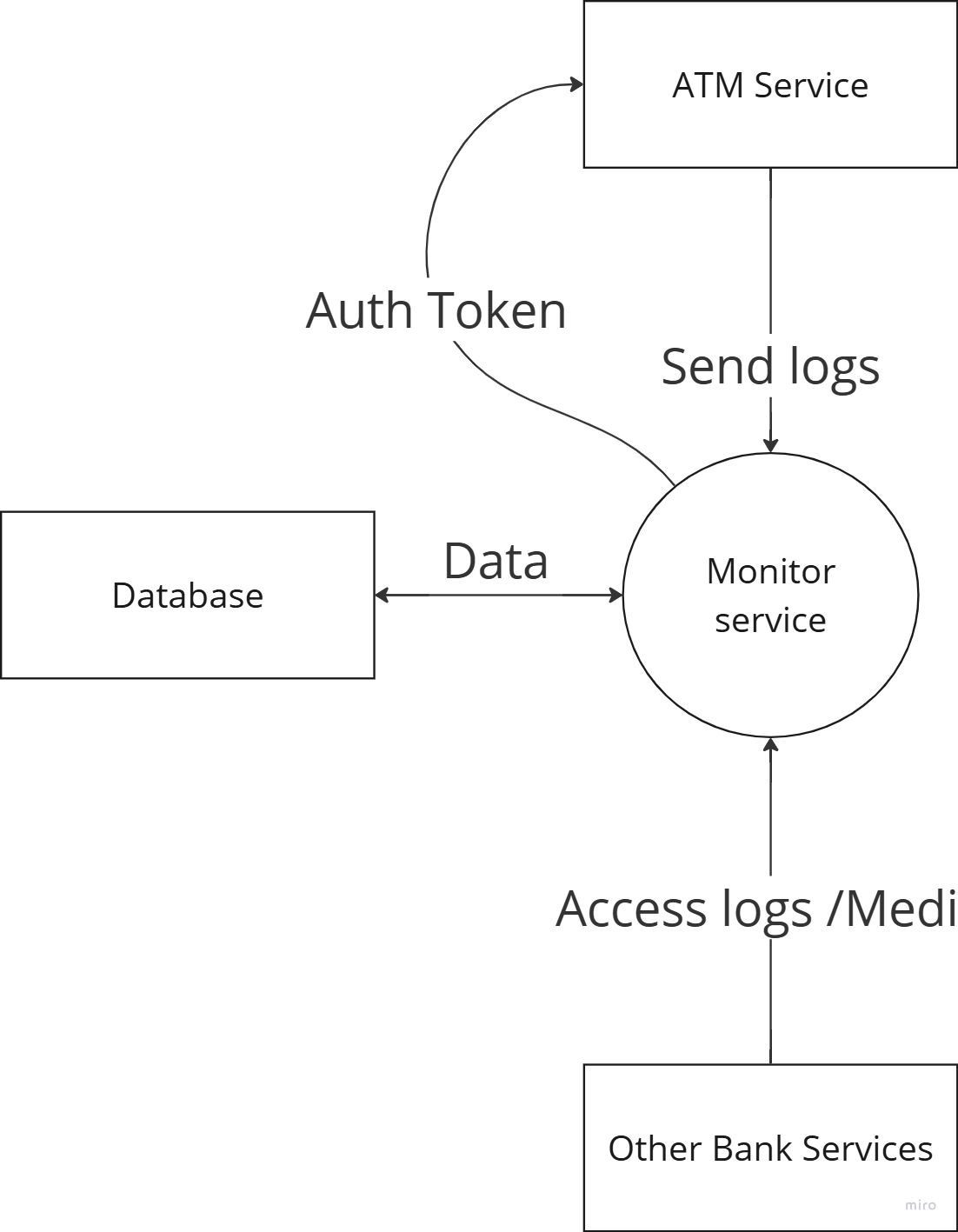
**1. Architecture Overview**

The system is a microservice-based application with RESTful APIs that provide real-time ATM monitoring. The application will:

Run on individual ATMs and report to the central bank system.

Securely handle sensitive customer data.

Use authentication tokens for API requests.



**2. High-Level Components**

ATM Monitoring Service :

Deployed on each ATM.

Collects transaction data and device status and logs errors.

Uploads video/camera data to a centralised storage.

Central Monitoring System (API Gateway):

Acts as an entry point for API requests.

Handles authentication and authorisation using verifiable tokens.

Data Storage and Processing:

Relational Database (PostgreSQL): Stores transaction logs, failures, and customer data.

Object Storage (e.g., S3, Blob Storage): Stores video and image data.

Authentication and Security:

Use OAuth 2.0 or JWT (JSON Web Tokens) for secure API access.

**3. API Design**

Here’s a breakdown of the APIs needed:

**A. Authorization API**

**RoleAccess - Admin Token**

Endpoint: POST /api/v0.1/tokens/user

Description: Get an admin token for an ATM

Request:

{

"userName": "user",

“password”: “password”

}

Response:

{

"token": "eyJhbGciOiJSUzI1NiJ9.eyJpc3MiOiJzZWxmIiwi.."

}

**RoleAccess - ATM Token**

Endpoint: POST /api/v0.1/tokens/machine

.addHeader("Authorization", "Bearer eyJhbGciOiJSUzI1NiJ9…”)

Description: Get an admin token for an ATM

Request:

{ "bank": "icici", "deviceId": "10001", "expiry": 24 }

Response:

{

"token": "eyJhbGciOiJSUzI1NiJ9.eyJpc3MiOiJzZWxmIiwi.."

}

**B. Total Number of Customers in the Last 24 Hours**

Endpoint: GET /api/customers/last24hours

Description: Returns the number of unique customers who used the ATM in the last 24 hours.

Response:

{

"totalCustomers": 120

}

**C. Breakdown of Transactions by Type**

Endpoint: GET /api/v0.1/atm/banks/transaction-breakdown

Description: Returns the count of different transaction types (Deposit, Withdrawal, Balance Inquiry) in the last 24 hours.

Response:

{

"DEPOSIT": 2,

"WITHDRAW": 1,

"BALANCE\_ENQ": 3

}

**D. List of Failures Due to System or Device Issues**

Endpoint: GET /api/v0.1/atm/banks/failures

Description: Retrieves the list of failures, including downtime incidents and transaction context.

Response:

[

{

"id": 1,

"bank": "DemoBank",

"deviceId": "B223",

"transactionType": "WITHDRAW",

"failureType": "DOWN\_TIME",

"stamp": "2024-11-28T16:04:13.209+00:00",

"customerId": "cust-1121"

},

]

**E. Download Camera Images/Video by Time Range**

Endpoint: GET /api/v0.1/atm/banks/download-video/2024-11-28 02:44:09.999/2024-11-28 02:44:10.015

Description: Provides a download link for video or images within a specified time range.

Response:

{

"download\_link": "https://storage.example.com/videos/ATM1234\_20241126.mp4"

}

**4. Database Design**

Tables:

Transactions:

id (PK)

stamp (TIMESTAMP)

bank(TEXT)

customer\_id(TEXT)

device\_id(TEXT)

transaction\_type (TEXT)

**failures:**

id (PK)

stamp (TIMESTAMP)

bank(TEXT)

customer\_id(TEXT)

device\_id(TEXT)

failure\_type (TEXT)

transaction\_type (TEXT)

**Videolog:**

id (PK)

stamp (TIMESTAMP)

device\_id(TEXT)

path(TEXT)

**5. Security Measures**

Authentication & Authorization: Use JWT to validate API access.

Encrypted Communication: Ensure HTTPS is enforced for all API communication.

Data Masking: Mask sensitive data like account numbers in logs and responses.

Rate Limiting: Protect APIs from abuse by rate-limiting requests.

**6. Technologies**

Backend: Spring Boot (Kotlin)

Database: PostgreSQL

Storage: local file system for video storage.

Authentication: JWT,oAuth2.0