**Design Document: HealthData Insights Platform**

This document outlines the design and architecture of the **HealthData Insights** platform. Its purpose is to provide a comprehensive overview for conducting a thorough threat modeling analysis.

**1. System Overview 🩺**

**HealthData Insights** is a cloud-based Software-as-a-Service (SaaS) platform designed for medical researchers. It allows registered research institutions to upload anonymized patient datasets, process them using built-in analysis tools, and collaborate on findings. The system prioritizes data security, integrity, and compliance with regulations like GDPR and HIPAA.

**Core functionalities:**

* User and organization management.
* Secure, encrypted upload and storage of large datasets (e.g., genomic data, clinical trial results).
* A suite of data analysis tools (e.g., statistical analysis, machine learning model training).
* Collaboration features for sharing analysis results within a research organization.

**2. Key Use Cases**

* **UC-1: Researcher Registration & Authentication**: A new **Researcher** from a subscribed institution registers for an account. The system verifies their institutional email. Subsequent logins are handled via a multi-factor authentication (MFA) process.
* **UC-2: Data Upload**: An authenticated **Researcher** uploads a large, anonymized dataset in CSV or JSON format from their local machine via a web interface. The data is transferred over HTTPS to the platform's ingestion service.
* **UC-3: Data Processing**: A **Researcher** initiates a data analysis job (e.g., "Variant Calling") on an uploaded dataset. A processing worker retrieves the data, performs the computation, and stores the results.
* **UC-4: Results Viewing**: A **Researcher** views the results of a completed analysis job through the web interface. The results are rendered as visualizations and downloadable reports.
* **UC-5: Admin Role Management**: An **Organizational Admin** (a type of Researcher) can invite or remove other researchers from their institution's account.

**3. Data Flow Diagrams (DFDs)**

**DFD Level 0: Context Diagram**

A single process, **HealthData Insights Platform (1.0)**, interacts with three external entities:

1. **Researcher**: Sends login credentials, upload data, and analysis requests. Receives session tokens and analysis results.
2. **Organizational Admin**: Sends user management requests. Receives confirmation.
3. **External Identity Provider (IdP)**: Interacts with the platform for SAML-based Single Sign-On (SSO) for enterprise clients.

**DFD Level 1: High-Level Process Breakdown**

* **Process 1.1: User Authentication**:
  + **Input**: Credentials from **Researcher**.
  + **Output**: Session Token to **Researcher**.
  + **Data Store**: Interacts with UsersDB.
* **Process 1.2: Data Ingestion**:
  + **Input**: Anonymized Dataset from **Researcher**.
  + **Output**: Dataset write confirmation.
  + **Data Store**: Writes to RawDataBucket.
* **Process 1.3: Data Analysis**:
  + **Input**: Analysis Job Request from **Researcher**.
  + **Output**: Job status update.
  + **Data Stores**: Reads from RawDataBucket, writes to ProcessedDataBucket and ResultsDB.
* **Process 1.4: Results Presentation**:
  + **Input**: View Results Request from **Researcher**.
  + **Output**: Formatted Results to **Researcher**.
  + **Data Store**: Reads from ResultsDB.

**4. Architecture Views 🏗️**

**Logical View**

The system is composed of several microservices orchestrated via an API Gateway.

1. **Frontend Web App**: A Single Page Application (SPA) built with React that runs in the user's browser. It communicates with the backend via the API Gateway.
2. **API Gateway**: A central entry point for all client requests. It handles routing, rate limiting, and initial authentication token validation.
3. **Authentication Service**: Manages user registration, login, MFA, and session token generation (JWTs).
4. **Data Ingestion Service**: Handles large file uploads, validates file formats, and places data into a temporary storage queue.
5. **Analysis Orchestrator Service**: Manages the lifecycle of analysis jobs. It listens for new job requests, queues them, and triggers processing workers.
6. **Processing Workers**: A pool of containerized applications that perform the actual computation. They are ephemeral and scale based on demand. They fetch data from storage, process it, and write the results back.
7. **Results Service**: Provides endpoints for querying and retrieving processed results and metadata.

**Physical / Deployment View**

The entire platform is deployed on **Amazon Web Services (AWS)** within a single region (eu-central-1), distributed across multiple Availability Zones (AZs) for high availability.

* **VPC (Virtual Private Cloud)**: The core network boundary.
  + **Public Subnet**: Contains the Application Load Balancer (ALB), which acts as the public-facing entry point.
  + **Private Subnets**: Contain all backend services and databases. These services do not have public IP addresses and can only be accessed through the ALB or via internal networking.
* **Compute**:
  + Services like the API Gateway, Authentication, Orchestrator, and Results are deployed as containers on **AWS Fargate**.
  + **Processing Workers** are also run as Fargate tasks, scaled independently by an SQS queue length metric.
* **Data Stores**:
  + **UsersDB**: An **Amazon RDS** for PostgreSQL instance (Multi-AZ).
  + **RawDataBucket & ProcessedDataBucket**: **Amazon S3** buckets with encryption at rest (SSE-S3) enabled, versioning, and strict bucket policies. Access is granted via IAM roles.
  + **ResultsDB**: An **Amazon DynamoDB** table for storing analysis metadata and results.
* **Networking**:
  + All traffic from the internet enters through the **ALB**, which terminates TLS (HTTPS).
  + Communication between services inside the VPC is done over the internal network.
  + **AWS WAF** (Web Application Firewall) is configured on the ALB to protect against common web exploits.

**5. Technology Stack 💻**

| Component | Technology/Service |
| --- | --- |
| Frontend | React, TypeScript |
| Backend Services | Node.js with Express.js, Python for data processing |
| API Gateway | Amazon API Gateway |
| Compute | AWS Fargate (Docker containers) |
| Relational Database | Amazon RDS for PostgreSQL |
| NoSQL Database | Amazon DynamoDB |
| Object Storage | Amazon S3 |
| Message Queue | Amazon SQS (for analysis job queue) |
| CI/CD | GitHub Actions, AWS CodePipeline |
| IaC | AWS CDK (Cloud Development Kit) / Terraform |
| Authentication | JWTs, bcrypt for password hashing, SAML 2.0 integration |

Export to Sheets

**6. Trust Boundaries & External Dependencies**

**Trust Boundaries**

1. **External User Zone -> Application Frontend**: The primary trust boundary. All input from the **Researcher's** browser is untrusted.
2. **Internet -> VPC Public Subnet**: Traffic entering the ALB. Protected by AWS WAF.
3. **VPC Public Subnet -> VPC Private Subnet**: All traffic to backend services is routed from the ALB and is considered more trusted, but still requires service-to-service authentication (e.g., via IAM roles).
4. **Application -> Data Stores**: The boundary between compute services and persistent storage. Access is controlled via strict IAM roles and database credentials.

**External Dependencies**

* **Third-Party Identity Provider (IdP)**: For enterprise customers using SSO, the platform trusts the SAML assertions provided by the customer's IdP (e.g., Azure AD, Okta). The connection and assertion signatures must be secure.
* **Email Service (e.g., Amazon SES)**: Used for sending verification and notification emails. The platform trusts this service to deliver emails securely.
* **NPM / PyPI Registries**: The software supply chain for third-party libraries used in development is an external dependency.