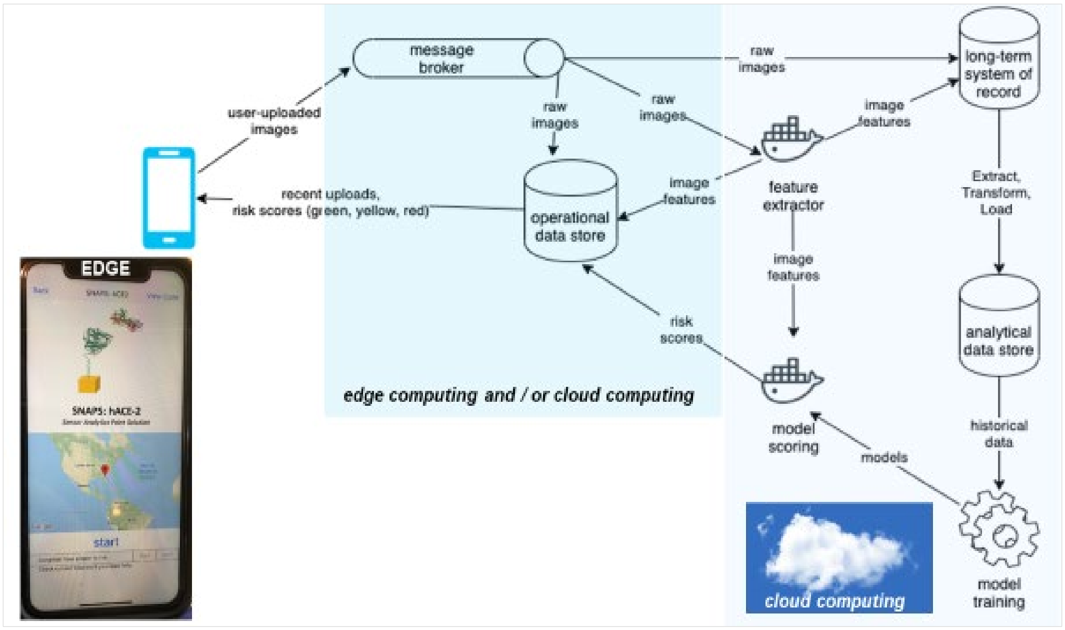
**APPENDIX – Figure 1: Brief Description of Major Components**[1]



**Message Broker**

When users upload images (the data after scanning with the HoloLens app or equivalent mobile tool), the mobile application on their phones writes messages with the image content and other metadata to a message-broker, which may be cloud-based message queuing (MQ[2]) protocol (open source software). The message broker allows devices to quickly offload data and confirm "sent!" to a user (if cloud based), thereby decoupling the user experience from the data store (even if it must use a temporary tinyDB on the device if the network is unavailable to access the cloud in real-time at the point of use). Messages can be queued in topics and the system might have autoscaling enabled so that as usage of the application increases, more users can be provisioned (process user uploads and get them stored). The uploads (data) are also sent by the message broker to the feature extractor and long-term storage database (may have to be batch upload when device is in the proximity of a gateway which can offer access to cloud services).

**Operational Data Store**

The message broker transfers uploads to ODS (Operational Data Store[3]), which may be a cloud-based managed service or part of the tinyDB on the device, if cloud is inaccessible at the point of use). ODS must be able to store image data (supports binary blog column type) alongside time-index numerical and character data. It is intended to only serve "hot" (nascent) data to the application. Older data may be evicted (batch uploaded to cloud managed facilities) to optimize on-device service and prevent data amplification. ODS is tuned for fast random reads and serves requests made by mobile app when users view recent uploads and additional metadata about those uploads, including "risk scores". ODS is optimized for fast writes and high efficiency time-series queries.

**Feature Extractor**

Extracts additional metadata from messages/images/data uploaded from the mobile app (also uploads it to the long-term system of record[4] which includes raw data uploaded from the application, similar to "master data" in ERP[5]). Feature Extractor may convert the uploaded image into a numeric matrix[6] or create hash table or other representation of a region[7] and correct for differences in resolution (for example, variation due to pixel density of cameras on different smartphones). Feature[8] vectors[9] may be maintained in the long-term system of record. It may be written to the operational data store to enable extraction/selection[10] of incoming data (uploads from message broker) relevant to these feature vectors.

**Long-term System of Record**

Mobile applications may never access data directly from this data store[11]. Interactive-speed queries to this data store may not be supported. When necessary, the objects stored in this "record" may be extracted and the data is loaded into an analytical data store. For object stores, this operation may be accomplished using query-over-files engines[12].

**Analytical Data Store**

Scientists and data experts will need historical data (from uploaded samples) and to train and evaluate machine learning (ML) models to assign risk scores to samples. Analytical data store (ADS database[13]) may be populated with data from the long-term system of record using scheduled batch data uploads.

**Model Training**

In model training[14], a statistical model is built from historical data. Models should be serializable[15] representations of the program generated by ML training. Serialization is essential for interoperability on different platforms. It is key to create composable models where models from different groups can be deconstructed to sub-elements which can be reconstructed to compose a new model (which may be greater than the sum of parts). Serialization enables the process of translating a data structure or object state into a format that can be stored or transmitted and reconstructed. Proprietary software vendors obfuscate or encrypt serialized data to prevent access but architectures such as CORBA[16] define the serialization formats in detail to enable open access.

**Model Scoring**

In model scoring, a model is called on input data, the model processes the input data and generates a prediction. The structure of this code depends on the design choices made in model training. For reliability of deployment, model scoring may run in a container[17] (an unit of software) which contains code (and all its dependencies) that uses a model to produce predictions on new input data. If model scoring runs in a container then the model can be arbitrary code in the developer's language[18] of choice. Model scoring requires features created previously by the feature extractor.

**REFERENCES**

---

[1] Lamb, James (2020) ADD Appendix – Description of Major Components (*personal communication*) https://github.com/jameslamb

[2] Message Queueing using an open lightweight broker, such as, Message Queueing Telemetry Transport, MQTT (https://mqtt.org/) or RabbitMQ (https://www.rabbitmq.com/) or heavy-duty Apache Kafka (https://kafka.apache.org/). Self-managed or run behind a managed (IoT) service from cloud providers: AWS IoT Core (https://aws.amazon.com/iot-core/) or Azure IoT Hub (https://azure.microsoft.com/en-us/services/iot-hub/) or related services provided by other vendors (https://www.zdnet.com/article/the-top-cloud-providers-of-2020-aws-microsoft-azure-google-cloud-hybrid-saas/).

[3] Operational Data Store choices include InfluxDB (https://www.influxdata.com/), Apache Cassandra (https://cassandra.apache.org/) or Prometheus (https://prometheus.io/). Managed cloud database from Amazon https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html.

[4] Amazon S3 Glacier https://aws.amazon.com/glacier/

[5] Enterprise Resource Planning (ERP) – Master Data Management: Architecture and Technology https://www.element61.be/en/resource/master-data-management-mdm-architecture-technology

[6] Vesna Vučković (2008) *Image and its Matrix, Matrix and its Image.* Преглед НЦД 12 (2008) 17–31 http://elib.mi.sanu.ac.rs/files/journals/ncd/12/ncd12017.pdf

[7] H3: Uber's Hexagonal Hierarchical Spatial Index https://eng.uber.com/h3/

[8] Liu J, Ranka S, Kahveci T. *Classification and feature selection algorithms for multi-class CGH data.* Bioinformatics. 2008 July 1; 24(13):i86-95. doi: 10.1093/bioinformatics/btn145 https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2718623/pdf/btn145.pdf

[9] Bommert, Andrea, et al. "Benchmark for Filter Methods for Feature Selection in High-Dimensional Classification Data." *Computational Statistics & Data Analysis*, vol. 143, Mar. 2020, p. 106839. *DOI.org (Crossref)*, doi:10.1016/j.csda.2019.106839.

[10] Chen, R., Dewi, C., Huang, S. *et al.* Selecting critical features for data classification based on machine learning methods. *Journal of Big Data* **7**, 52 (2020). https://doi.org/10.1186/s40537-020-00327-4 https://journalofbigdata.springeropen.com/track/pdf/10.1186/s40537-020-00327-4

[11] Object stores: Amazon https://aws.amazon.com/s3/; Google Cloud https://cloud.google.com/storage; Microsoft Azure Blob https://azure.microsoft.com/en-us/services/storage/blobs/); Apache Cassandra https://medium.com/walmartglobaltech/building-object-store-storing-images-in-cassandra-walmart-scale-a6b9c02af593

[12]Query-Over-Files Engines: Presto (https://prestodb.io/), Apache Drill (https://drill.apache.org/) or Apache Spark SparkSQL (https://spark.apache.org/sql/). If using application-specific custom code that directly reads files, orchestrated with batch-scheduling engine:
Apache Airflow (https://airflow.apache.org/) or Prefect (https://www.prefect.io/)

[13] Analytical Data Store: Traditional relational database PostgreSQL (https://www.postgresql.org/) or hosted relational database products provided by cloud providers (https://aws.amazon.com/rds/). If intelligent caching of repeated queries is needed, data warehouse technologies (managed cloud service) include: Snowflake (https://www.snowflake.com/), Amazon Redshift (https://aws.amazon.com/redshift) or Google BigQuery (https://cloud.google.com/bigquery). If cloud is inaccessible, for on-premises option: Teradata (https://www.teradata.com/).

[14] Machine Learning (ML) model training tools: Apache Spark (https://spark.apache.org/), Dask (https://dask.org/) or Ray (https://rise.cs.berkeley.edu/projects/ray/). If application specificity does not require high degree of customization - use "autoML" tools - DataRobot (https://www.datarobot.com/), h2o (https://docs.h2o.ai/h2o/latest-stable/h2o-docs/automl.html), Amazon SageMaker Autopilot (https://aws.amazon.com/blogs/aws/amazon-sagemaker-autopilot-fully-managed-automatic-machine-learning/) or Google Cloud AutoML (https://cloud.google.com/automl).

[15] Tauro, C.J., Ganesan, N., Mishra, S.R., & Bhagwat, A. (2012). Object Serialization: A Study of Techniques of Implementing Binary Serialization in C++, Java and .NET. *International Journal of Computer Applications, 45*, 25-29. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.685.1077&rep=rep1&type=pdf

[16] Common Object Request Broker Architecture (CORBA) https://www.corba.org/

[17] Container: Docker https://www.docker.com/resources/what-container

[18] Developer's language of choice: Java jar (https://en.wikipedia.org/wiki/JAR_(file_format)), Python pickle file (https://docs.python.org/3/library/pickle.html), R rds file (https://stat.ethz.ch/R-manual/R-devel/library/base/html/readRDS.html) or a precompiled executable which can read in input data from "stdin" (standard input is a stream from which a program reads its input data) or from a file, created with C/C++ or language-agnostic description of a model: Predictive Model Markup Language (https://en.wikipedia.org/wiki/Predictive_Model_Markup_Language) or Portable Format for Analytics (http://dmg.org/pfa/).