|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ITU Logo | INTERNATIONAL TELECOMMUNICATION UNION  **TELECOMMUNICATION STANDARDIZATION SECTOR**  STUDY PERIOD 2017-2020 | | FG-AI4H-K-043 | |
| **ITU-T Focus Group on AI for Health** | |
| **Original: English** | |
| **WG(s):** | | Plenary | E-meeting, 27-29 January 2021 | |
| **DOCUMENT** | | | | |
| **Source:** | | Open Code Project – Management Team | | |
| **Title:** | | Open Code Project Status Update | | |
| **Purpose:** | | Discussion | | |
| **Contact:** | | Marc Lecoultre MLLab Switzerland | | Tel: +41 79 321 09 29 E-mail: [ml@mllab.ai](mailto:ml@mllab.ai) |
| **Contact:** | | Elora Schörverth Fraunhofer HHI Berlin, Germany | | E-mail: [elora.schoerverth@hhi.fraunhofer.de](mailto:elora.schoerverth@hhi.fraunhofer.de) |
| **Contact:** | | Pradeep Balachandran Technical Consultant (Digital Health) India | | E-mail: [pbn.tvm@gmail.com](mailto:pbn.tvm@gmail.com) |
| **Contact:** | | Dominik Schneider Merck KGaA Darmstadt, Germany | | E-mail: [dominik.a.schneider@merckgroup.com](mailto:dominik.a.schneider@merckgroup.com) |
| **Contact:** | | Joachim Krois Charité - Universitätsmedizin Berlin Berlin, Germany | | E-mail: joachim.krois@charite.de |
| **Contact:** | | Shobha Iyer Big data Consultant (Freelancer)  Frankfurt am Main, Germany | | E-mail: [shobhaiyer@thebigdataanalytics.com](mailto:shobhaiyer@thebigdataanalytics.com) |
| **Contact:** | | Luis Oala Fraunhofer HHI Berlin, Germany | | E-mail: [luis.oala@hhi.fraunhofer.de](mailto:luis.oala@hhi.fraunhofer.de) |
| **Contact:** | | Shruti Choudhary University of Oxford Oxford, UK | | E-mail: [shruti.choudhary@kellogg.ox.ac.uk](mailto:shruti.choudhary@kellogg.ox.ac.uk) |

|  |  |
| --- | --- |
| **Abstract:** | This Open Code Project aims to produce the digital building blocks (six software packages) that compose the FG-AI4H Assessment Platform. The assessment platform, which can be distinguished from AI "challenge" platforms through its consideration of regulatory guidelines and the needs of other AI for health stakeholders, supports the end-to-end assessment of AI for health algorithms. |

**TABLE OF CONTENTS**

[1 Introduction 5](#_Toc62568183)

[1.1 Overview 5](#_Toc62568184)

[1.2 Goals 5](#_Toc62568185)

[1.3 Functional scope 5](#_Toc62568186)

[1.4 Components 6](#_Toc62568187)

[1.5 System design 7](#_Toc62568188)

[1.6 System landscape 9](#_Toc62568189)

[1.7 System architecture 12](#_Toc62568190)

[1.8 Project timeline 14](#_Toc62568191)

[1.9 MVP scope 15](#_Toc62568192)

[2 Core Package 16](#_Toc62568193)

[2.1 Overview 16](#_Toc62568194)

[2.2 Design 16](#_Toc62568195)

[2.3 Implementation 16](#_Toc62568196)

[3 Data Acquisition and Storage Package 18](#_Toc62568197)

[3.1 Overview 18](#_Toc62568198)

[3.2 Design 18](#_Toc62568199)

[3.3 Implementation 20](#_Toc62568200)

[3.4 Metadata 21](#_Toc62568201)

[4 Evaluation Package 22](#_Toc62568202)

[4.1 Overview 22](#_Toc62568203)

[4.2 Evaluation process 22](#_Toc62568204)

[4.3 Architecture 23](#_Toc62568205)

[4.4 Implementation 24](#_Toc62568206)

[5 Reporting Package 24](#_Toc62568207)

[5.1 Reporting package-overview 24](#_Toc62568208)

[5.2 Reporting service-workflow description 25](#_Toc62568209)

[5.3 Reporting package-functionality matrix 25](#_Toc62568210)

[5.4 Reporting package -architectural design 26](#_Toc62568211)

[5.5 Reporting package - module design 26](#_Toc62568212)

[5.6 Reporting package -component-level design 27](#_Toc62568213)

[5.7 Reporting service – modalities 28](#_Toc62568214)

[5.8 Reporting service -parameter configuration criteria 30](#_Toc62568215)

[5.9 Reporting service - visualization formats 30](#_Toc62568216)

[5.10 Reporting package - design evaluation 30](#_Toc62568217)

[6 Project organization 32](#_Toc62568218)

[6.1 Overview 32](#_Toc62568219)

[6.2 Meetings 32](#_Toc62568220)

[7 Core Package 32](#_Toc62568221)

[7.1 Membership 32](#_Toc62568222)

[7.2 Meetings 32](#_Toc62568223)

[7.3 Documentation 32](#_Toc62568224)

[8 Acquisition Package 32](#_Toc62568225)

[8.1 Membership 32](#_Toc62568226)

[8.2 Meetings 33](#_Toc62568227)

[8.3 Documentation 33](#_Toc62568228)

[9 Data Storage Package 33](#_Toc62568229)

[10 Annotation Package 33](#_Toc62568230)

[10.1 Membership 33](#_Toc62568231)

[10.2 Meetings 33](#_Toc62568232)

[10.3 Documentation 33](#_Toc62568233)

[11 Evaluation Package 33](#_Toc62568234)

[11.1 Membership 33](#_Toc62568235)

[11.2 Meetings 33](#_Toc62568236)

[11.3 Documentation 34](#_Toc62568237)

[12 Reporting Package 34](#_Toc62568238)

[12.1 Membership 34](#_Toc62568239)

[12.2 Meetings 34](#_Toc62568240)

[12.3 Documentation 34](#_Toc62568241)

[13 Tooling 34](#_Toc62568242)

[13.1 Slack 34](#_Toc62568243)

[14 Resources 35](#_Toc62568244)

[14.1 General Meeting Notes 35](#_Toc62568245)

[14.2 Working Documents 35](#_Toc62568246)

[15 Communication 36](#_Toc62568247)

**List of Tables**

Table 1.1: Platform Components 6

Table 12.4: Slack channels 35

**List of Figures**

Figure 1.1: Data submission user journey 8

Figure 1.2: Data storage user journey 8

Figure 1.3: Data management user journey 8

Figure 1.4: AI solution evaluation user journey 9

Figure 1.5: Reporting user journey 9

Figure 1.6: Benchmark management user journey 10

Figure 1.7: Data management user journey 11

Figure 1.8: Microservices architecture 12

Figure 1.9: A microservices architecture example on Azure 12

Figure 1.10: Platform architecture 13

Figure 1.11: Project timeline 14

Figure 2.1: Component diagram for Core Services 16

Figure 2.2: AWS FHIR system architecture 17

Figure 3.1: Data acquisition service - System context 18

Figure 3.2: Data acquisition service - Container Diagram 19

Figure 3.3: Data acquisition service - API application - Component Diagram 19

Figure 3.4: Dataset management Webview (MVP, screenshot from 2020-01-23) 20

Figure 3.5: The addition of a new dataset (MVP, screenshot from 2020-01-23) 20

Figure 3.6: Example data specification sheet for the dataset used in the MVP (Meeting K) 21

Figure 4.1: An overview of the WG-DAISAM evaluation process for AI4H models. 22

Figure 4.2: EvalAi architecture 23

Figure 4.3: Evaluation service UML 24

Figure 5.1: Reporting service-activity diagram 25

Figure 5.2: Reporting package-architecture overview 26

Figure 5.3: Reporting package-component-level diagram 27

Figure 5.4: Reporting service app - home page screenshot 28

Figure 5.5: Reporting service-specimen model report 29

# Introduction

## Overview

Artificial intelligence (AI) can provide the basis for tools that improve global health care, bringing us closer to the realization of the third sustainable development goal: good health and well-being for all.

However, before AI-based tools are integrated in medical practice and applied to patients, it must be demonstrated that they serve the intended purpose without unintended effects. Of the AI-based tools that are currently used in medical practice, most were developed and regulated for a limited (e.g., national) public. Consequently, the adoption of AI-based tools is fragmented across the globe. As health is an issue that transcends borders, ITU/WHO FG-AI4H encourages a collective effort among stakeholders (including developers, regulators, healthcare practitioners, and public health institutes) from across the globe to ensure the safety and trustworthiness of AI-based tools and to permit their widespread implementation.

This Open Code Project aims to produce the digital building blocks (six software packages) that compose the FG-AI4H Assessment Platform. The assessment platform, which can be distinguished from AI "challenge" platforms through its consideration of regulatory guidelines and the needs of other AI for health stakeholders, supports the end-to-end assessment of AI for health algorithms. You may find out more information on the [Wiki site](https://dev.azure.com/mllabai/FG-AI4H%20Assessment%20Platform/_wiki/wikis/FG-AI4H-Assessment-Platform.wiki/1/AI4H-Assessment-Platform).

## Goals

The life cycle of AI for health has several steps. First, annotated health data are compiled. Then, an AI for health model is developed and carefully evaluated. At each step, medical, technological, and regulatory considerations play a critical role. Within this life cycle, we have identified two opportunities to advance the field of AI for health:

The Arbiter Problem:

1. Challenge: Companies do not want to share their data and solution; both are opaque to regulators.
2. Idea: The software platform can serve as a safe and neutral arbiter between parties.

Health AIs at Scale:

1. Challenge: Regulatory compliance of AI for health is a country-dependent process, which brings considerable costs.
2. Idea: Map country requirements to automated tests.

The Open Code Project capitalizes on these opportunities to make AI for health usable at scale.

Our product vision is the following:

*For health AI companies and regulators, who need to prove that a health AI product is fit for purpose, the AI4H Assessment Platform is a software platform that supports the end-to-end process for assessing health AI algorithms on a global level. Unlike e.g. EvalAI and other existing AI assessment platform products our platform focuses specifically on healthcare and covers all additional aspects, including ground truth annotation, data & metadata management and reporting for health AI regulators.*

## Functional scope

The Open Code Project produces software comprising the foundation of the FG-AI4H Assessment Platform and addressing the aforementioned—and other—challenges in the field of AI for health. These are: Data Acquisition Package (DAP), Data Storage Package (DP), an Annotation Package (AP), an Inference Package (IP), an Evaluation Package (EP), and a Reporting Package (RP). The following table highlights the purpose, functionalities, and target groups for each package.

## Components

Table 1.1: Platform Components

| Components and Drivers | Description and Purpose | Functionalities | Target Groups |
| --- | --- | --- | --- |
| Data Acquisition Package (DAP)  Joachim Krois | High-quality data are required to train and evaluate AI solutions. DAP coordinates the compilation of such data and ensures the availability of metadata and (where relevant) patient consent information. | Facilitates data compilation for many modalities; registers data and metadata; ingests data from DP; and manages patient consent information. | Manufacturers and medical experts |
| Data Storage Package (DP)  Andrea Romaoli Garcia | The basis for all subsequent packages is an orderly processing and storage of data and metadata. Medical data storage requires adherence to strict guidelines that preserve the privacy and safety of patients. DP provides data storage guidelines that consider these constraints. | Provides safe and secure storage of medical data; serves as an interface for other packages (AP, IP, and EP) to access this data; and offers data governance that complies with data protection laws. | Manufacturers, medical experts, and notified regulatory bodies |
| Annotation Package (AP)  Marc Lecoultre | High-quality annotated data provide the basis for supervised learning. Unfortunately, production is challenging and labour intensive. Certain features must be considered when evaluating an annotation: the quality of labels, the number of expert opinions, and the handling of non-unanimous decisions. AP brings together leading health experts across the globe to produce the highest-quality annotations at maximum efficiency. | Provides an annotation interface for many modalities; includes collaboration features; develops a network of annotation experts; and creates notifications of pending annotation tasks | Manufacturers and notified regulatory bodies |
| Inference Package (IP)  Luis Oala, Alixandro Werneck | For many health cases (e.g., the detection of breast cancer tissue or the classification of skin irritations), multiple parameters and AI models can provide viable solutions. IP allows various models to be evaluated in parallel for a benchmarking result. | Loads AI models; manages models undergoing prediction tasks (considers versioning, meta-data, etc.); and orchestrates computations | Manufacturers and notified regulatory bodies |
| Evaluation Package (EP)  Luis Oala, Elora Schörverth, Alixandro Werneck | Model performance is dependent on the choice of metric and possible parameters. Thus, it is of utmost importance to have a framework that allows for the comparison of the performance of different AI models. EP provides meaningful, state-of-the-art metrics that promise high expressibility. | Offers testing measures and methods for different quality dimensions including interpretation, bias, uncertainty, and robustness; questionnaires provide qualitative evaluation | Manufacturers and notified regulatory bodies |
| Reporting Package (RP)  Pradeep Balachandran | RP delivers a standardized format for communicating and reporting the properties, features, intended use, and limitations of AI for health to help connect different stakeholders. | A customizable reporting interface that presents the results of EP | Manufacturers, notified regulatory bodies, users of AI for health (doctors, patients), and vendors of AI for health |

## System design

Personas

We have worked on identifying "Personas", i.e. fictional characters, which we create based upon our research in order to represent the different user types that might use our service, product, site, or brand in a similar way.

We have identified five Roles:

* **Data Owner**, is a user that owns data and wants to store them on the platform. He can then share his datasets with the community
* **AI Solution Owner**, is a user that owns an AI solution to solve a specific task. He can request a benchmark for his solution or publish it as an Inference Service.
* **Benchmark Owner**, is an internal user that administers one or more benchmarks
* **Platform administrator**, is an internal user that administers the platform
* **Annotator**, is a user that participates in the annotation of datasets.

User journeys

We have defined several user journeys to help us draft the system requirements. Each user journey is experienced by one or more personas.

The first process corresponds to the description of the data submission. A Data Owner wants to upload a new dataset to the platform. If he is new, then he needs to sign-up otherwise he can directly sign-in. He initiates a data submission process and fills the forms with information about the dataset. He submits the data which is securely transmitted to the platform and automatically checked. If the check was not successful, the data owner can submit a modified version.

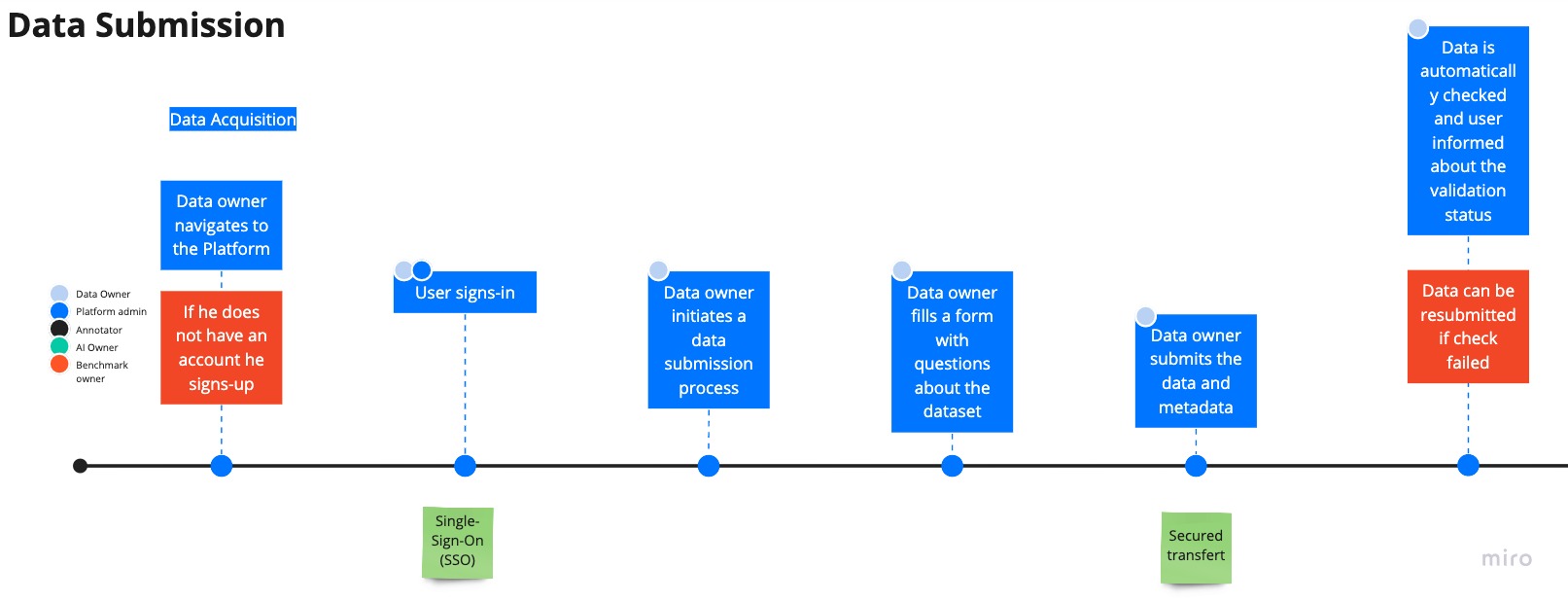


Figure 1.1: Data submission user journey

Once the data is stored on the platform, the Data Owner can grant access to the data.

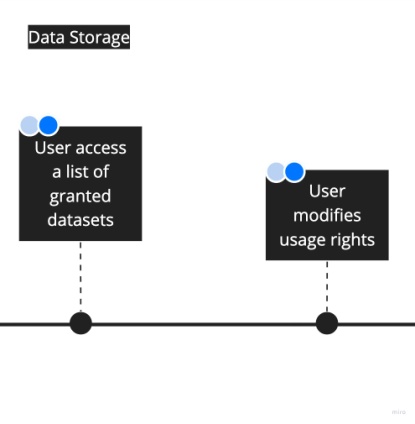


Figure 1.2: Data storage user journey

Each stored dataset can be managed by the Data Owner and other users that have been granted. The user can view a list of granted dataset, modify the access rights and open it to see the details (metadata, data elements). An audit trail is made available to trace all access and usage of the data.

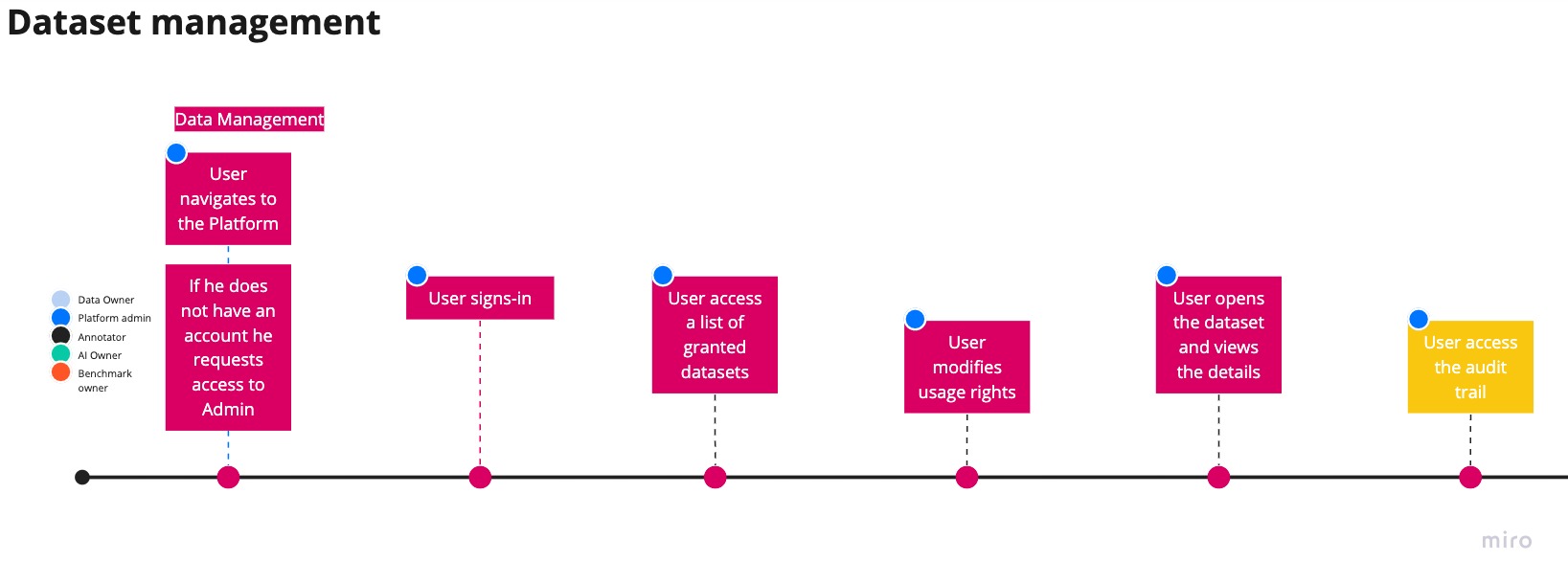


Figure 1.3: Data management user journey

An AI Solution Owner can request a benchmark of his solution. He needs to be a user of the platform and authenticated. If a benchmark for his solution exists, then he can initiate a request. He fills all required information and submits his request. The submitted data is evaluated by the platform.

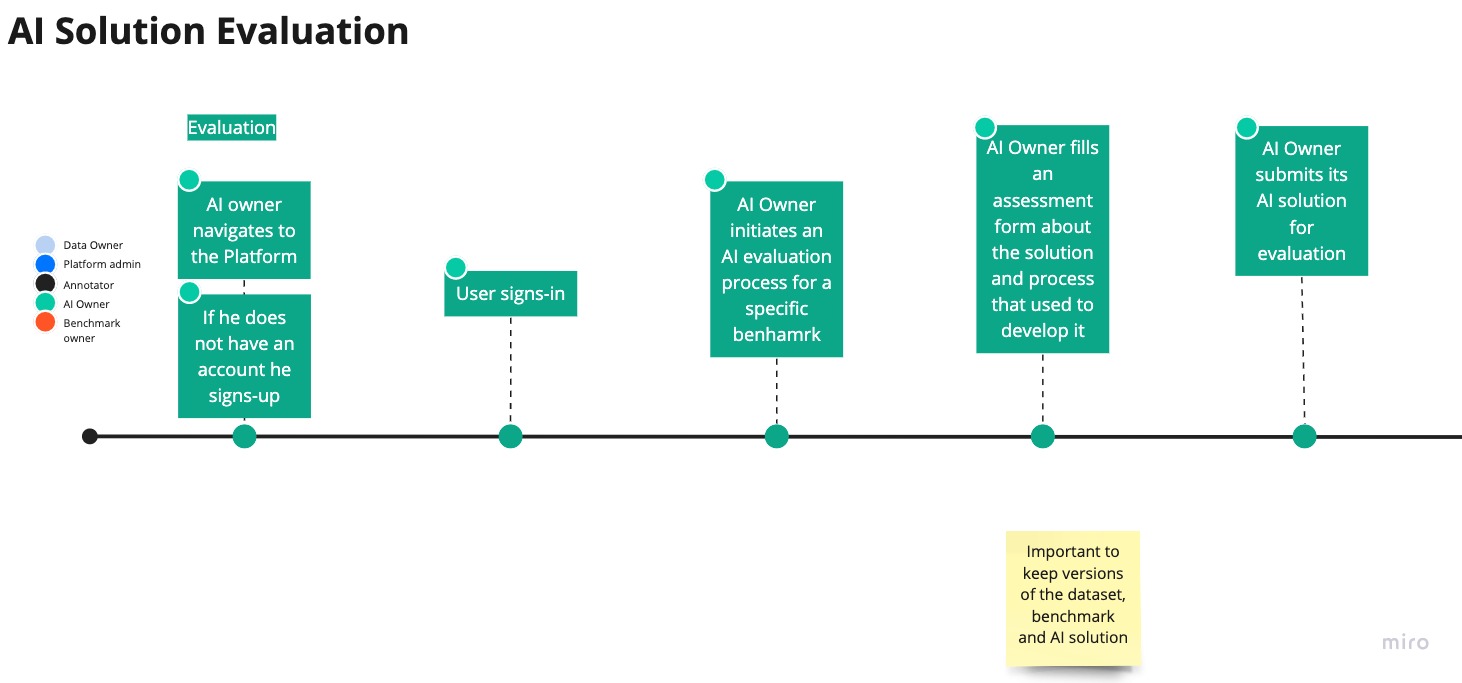


Figure 1.4: AI solution evaluation user journey

Once the benchmark process is completed, the AI Solution Owner can request a report. He can display a comprehensive report and request a signed PDF. He is also able to share the results publicly on the platform.

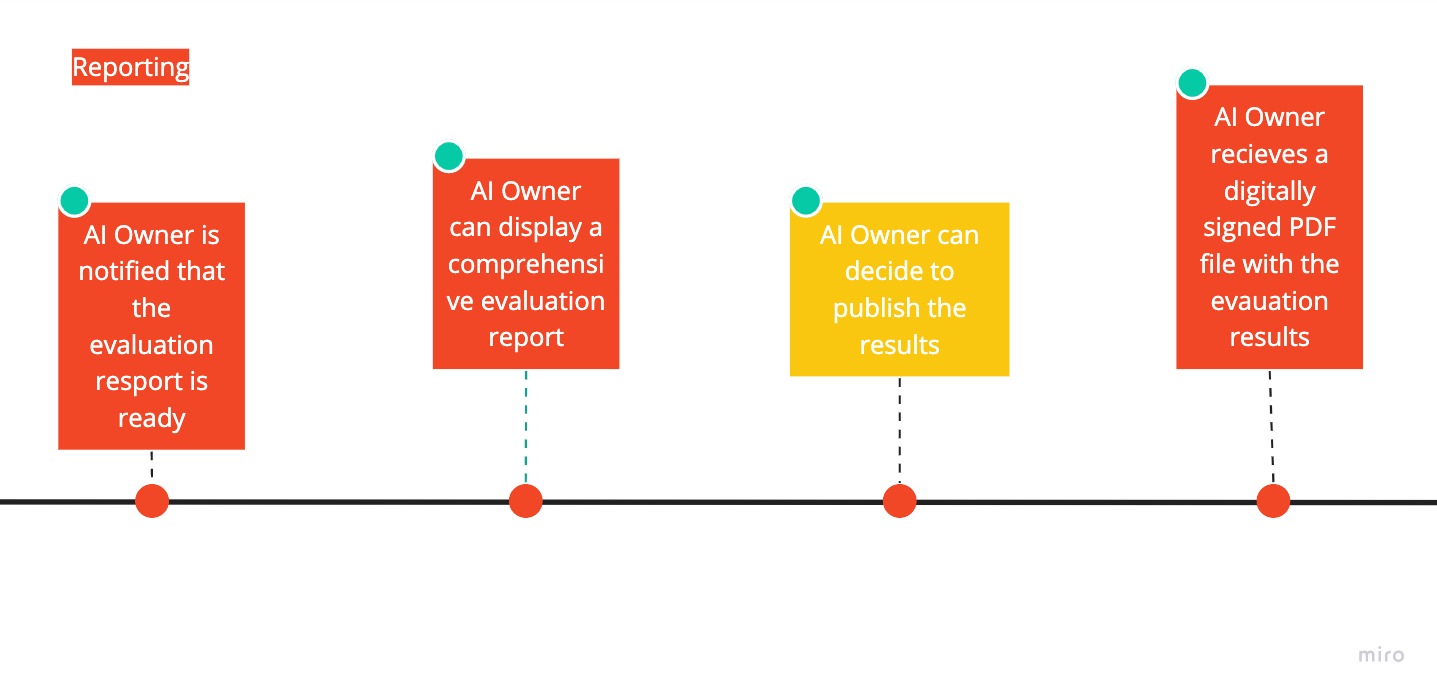


Figure 1.5: Reporting user journey

To provide AI solution benchmarking, benchmarks need to be set up for each AI task. A Benchmark Owner can access the platform and create a new benchmark process. He has to provide the evaluation description and related scripts. When validated, the benchmark can be published and used by AI Solution Owners. Refer to Figure 1.6.

## System landscape

This is a system landscape diagram of the platform showing dependencies and high-level relationships between components (Roles, Components). Refer to Figure 1.7.

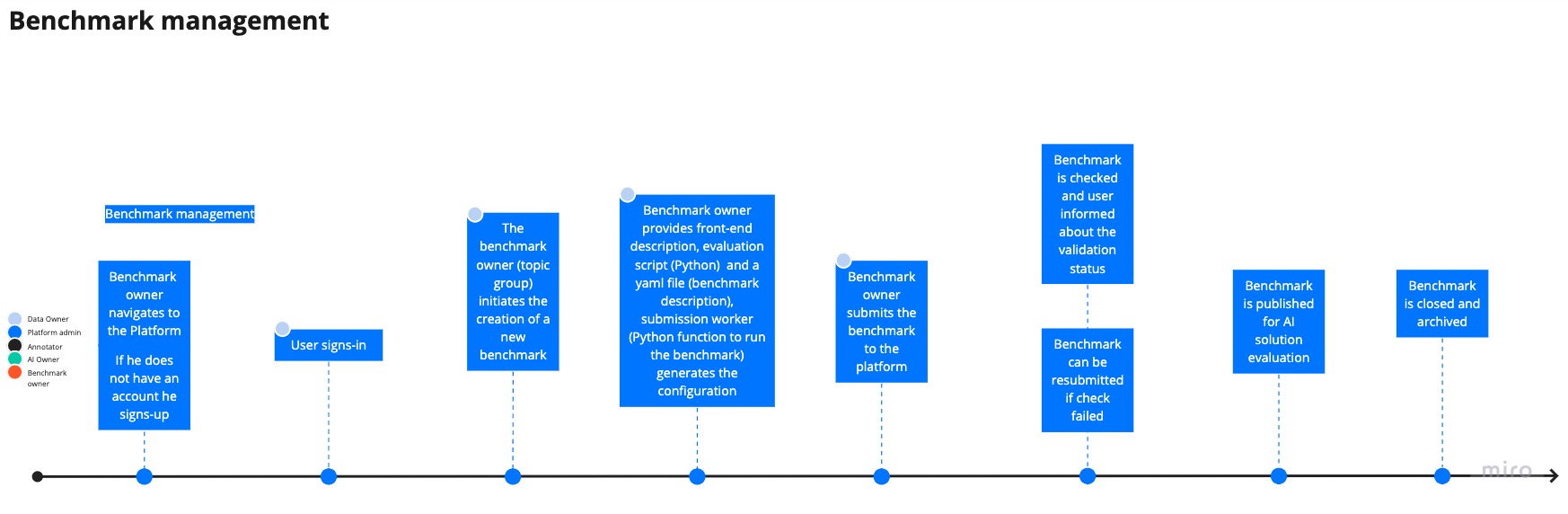


Figure 1.6: Benchmark management user journey

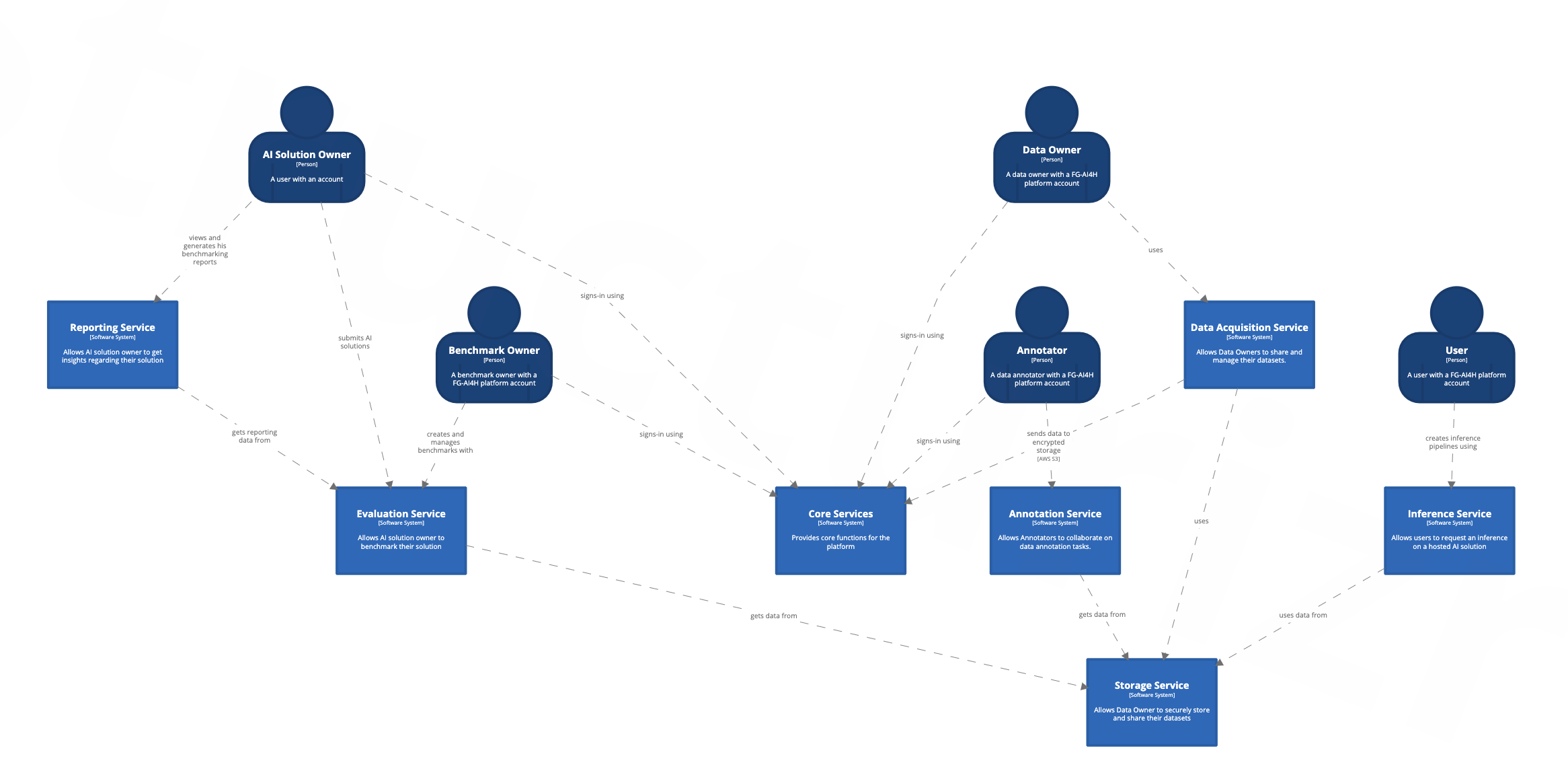


Figure 1.7: Data management user journey

## System architecture

For the implementation of the platform, we decided to use a modern and modular architecture. Microservices are a popular architectural style for building applications that are resilient, highly scalable, independently deployable, and able to evolve quickly.

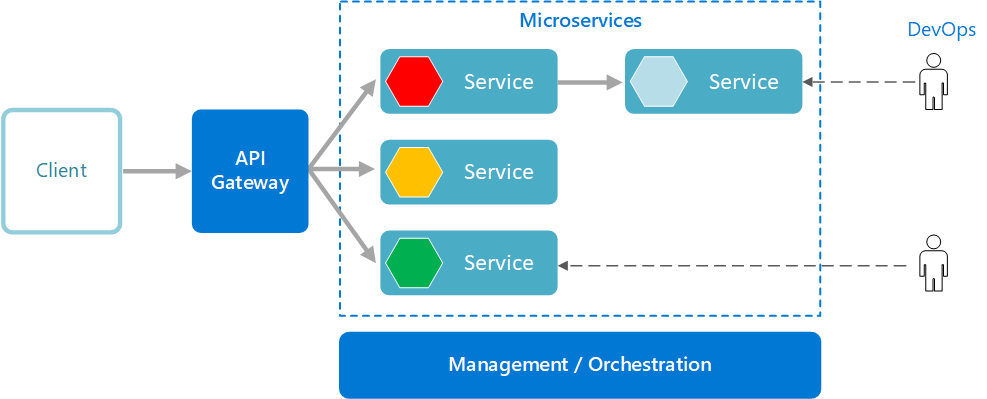


Figure 1.8: Microservices architecture

This type of architecture can be implemented on different cloud platforms. An Azure implementation would look like the architecture depicted in Figure 1.9.

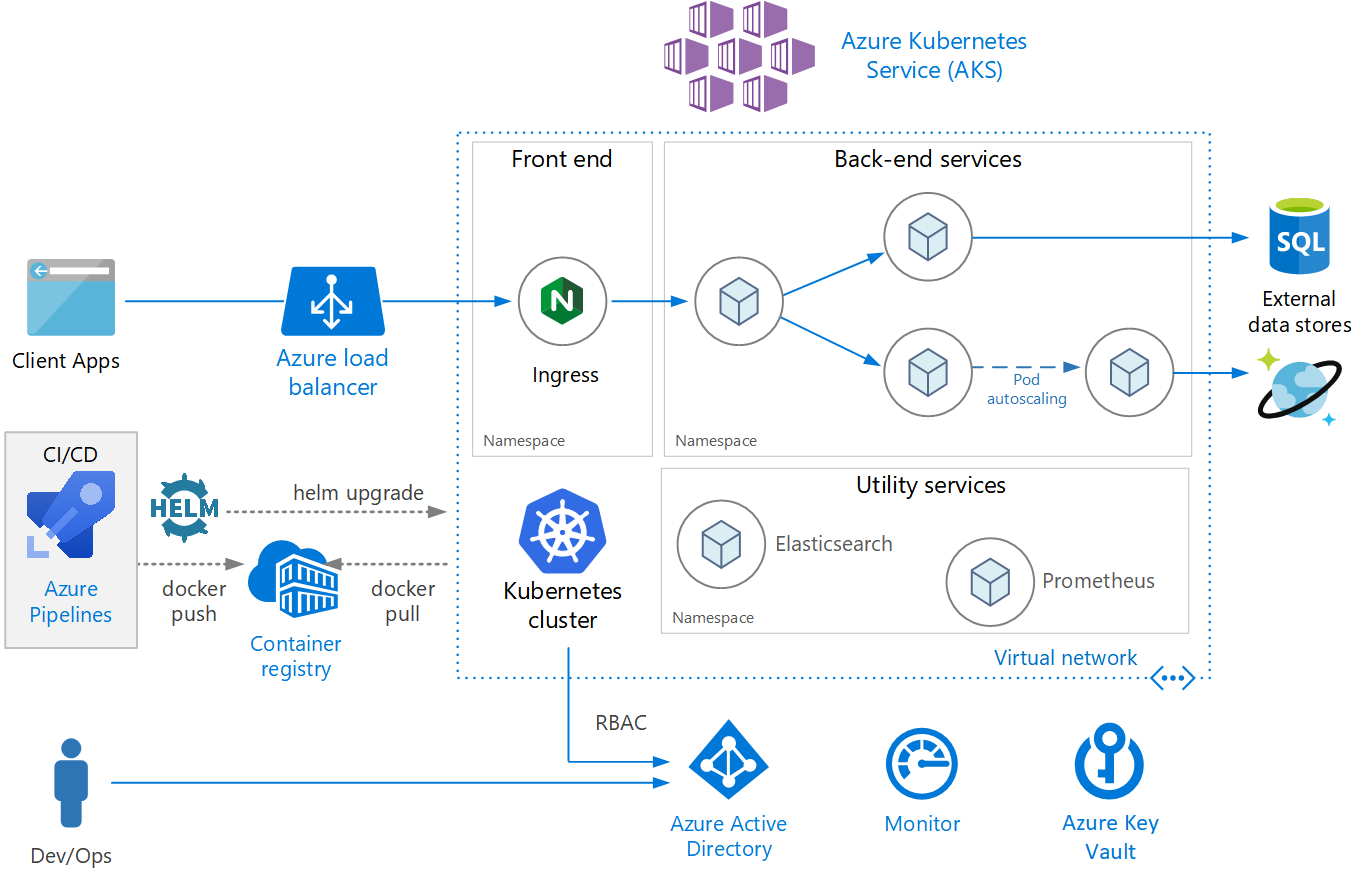


Figure 1.9: A microservices architecture example on Azure

It was decided to use AWS as our Cloud provider for the Open Code project. The motivation comes from the fact that the Evaluation Package is built on top of an open-source software called [EvalAI](https://github.com/Cloud-CV/EvalAI) which works on AWS. Later (after MVP), the selection of a Cloud provider will be reassessed, although our goal is to be independent in terms of proprietary solutions and services. Therefore, the microservices architecture was implemented using the components found in Figure 1.10.

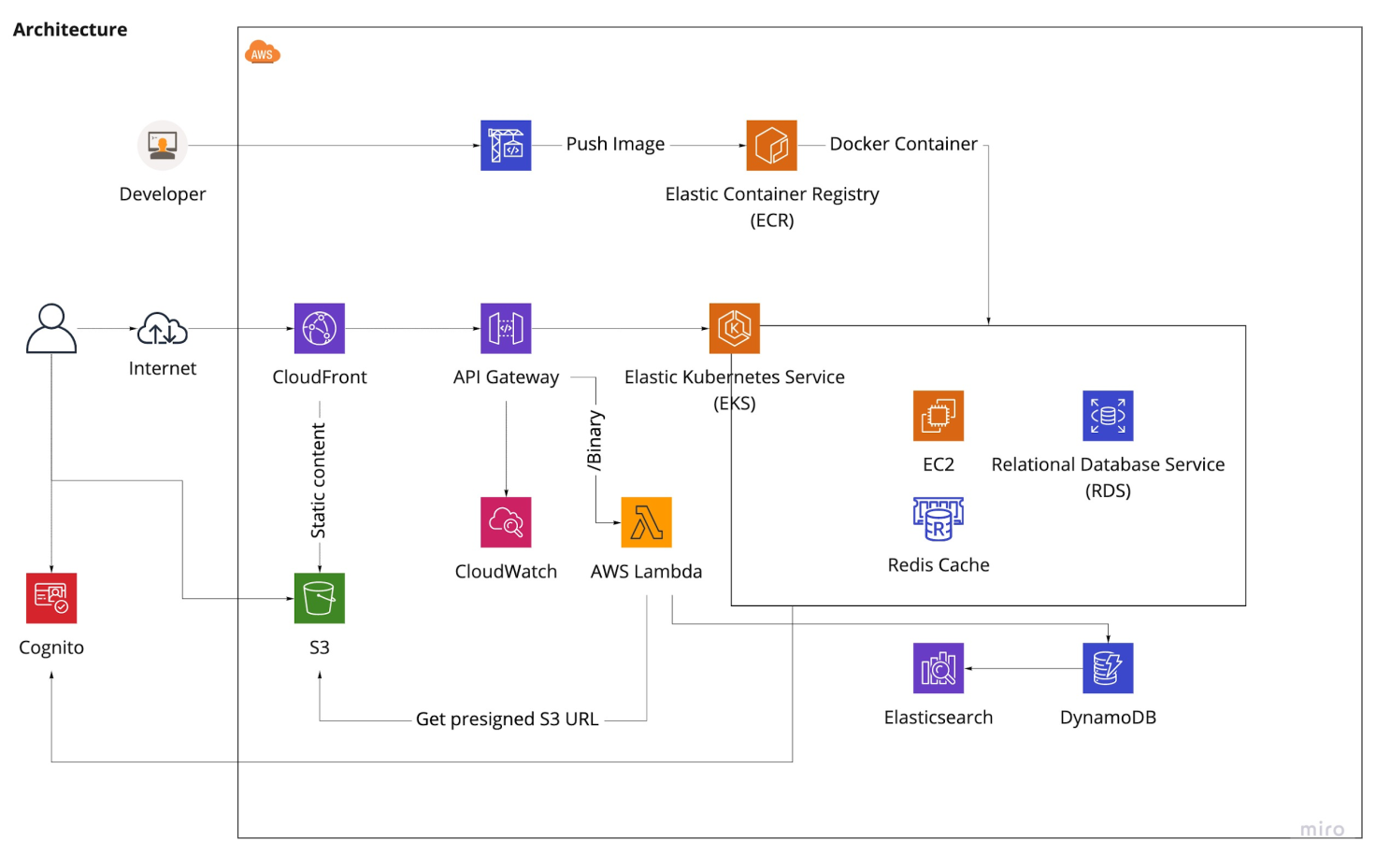


Figure 1.10: Platform architecture

Cognito

Users authenticate using AWS Cognito service. They get a JWT token they use to get access to the required services. To use Cognito User Pools within your apps please refer to the documentation: <https://docs.aws.amazon.com/cognito/latest/developerguide/cognito-integrate-apps.html>.

The endpoint is secured by API keys and Cognito for user-level authentication and user-group authorization.

S3 Bucket

To upload private datasets we use pre-signed S3 URL. This mechanism is managed by a Lambda function available in the AWS FHIR implementation.

EKS Cluster

The EKS cluster is running with internet-facing access of the K8S API endpoints now with the EC2 cluster nodes in private subnets.

Autoscaling of EC2 cluster nodes is also already enabled and tested.

To connect, make sure you have latest AWS CLI v2 and kubectl installed, then configure the following AWS user for the AWS CLI:

Those interested in having access should send a request to the Core Package Team.

Next, execute following on the command line:

|  |
| --- |
| CLUSTER\_NAME=ai4h-mvp-k8s-dev  aws eks --region eu-central-1 update-kubeconfig --name $CLUSTER\_NAME  code ~/.kube/config  # -> make sure that path to aws binary is set, e.g. command: /usr/local/bin/aws  kubectl config use-context arn:aws:eks:eu-central-1:[ACCOUNT\_ID]:cluster/$CLUSTER\_NAME  kubectl get nodes |

This should display the K8S EC2 cluster nodes. If it works you are connected and can deploy your code.

We recommend to use Lens for getting a quick overview of the K8S cluster state and find it very helpful: <https://k8slens.dev/>.

Cloud Front, API Gateway

The endpoint is hosted using API Gateway.

DynamoDB, Elastisearch

The database and storage layer consists of Amazon DynamoDB and S3, with Elasticsearch as the search index for the data written to DynamoDB.

## Project timeline

The initial steps of the Open Code project started after meeting I (7-8 May 2020). During the summer 2020 we prepared the scope and requirements and were ready to start just before meeting J (30 Sept. - 2 Oct. 2020).

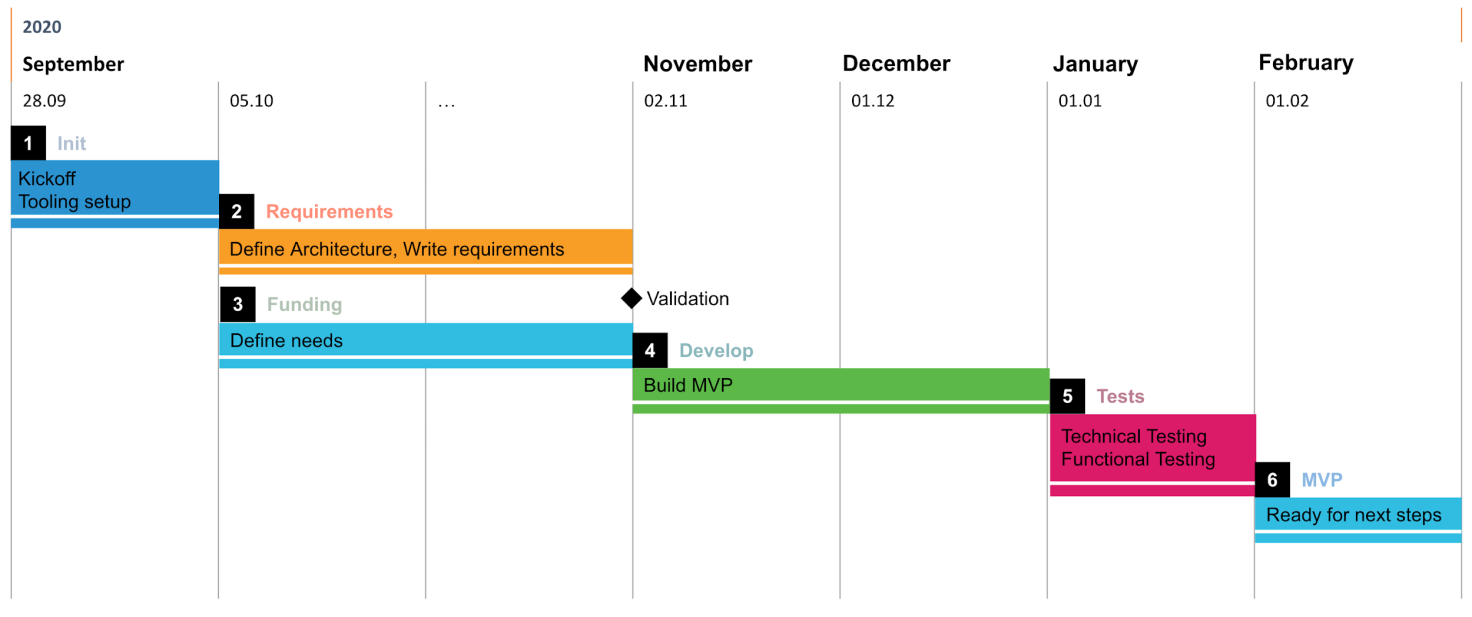


Figure 1.11: Project timeline

We set our first milestone at the end of January. The goal was to have a first MVP (Minimum Viable Product) available for meeting K (27-29 January 2021).

## MVP scope

During this first iteration we want to test the feasibility of the envisioned platform, define the project governance, associated risk management, tools and communication. The goal is to have a preview of all the components required to build the platform. We want these modules to be ready for a live demo.

This phase can be compared to a blueprint phase of the project. The features, system requirements and system architecture are defined in this iteration.

The work and deliverables done during this phase will be used in the next phase of the project. We expect to discuss the next steps and project organization during meeting K.

The features implemented in the MVP phase are described in each package section later in this document.

# Core Package

## Overview

The Core Package is responsible to deliver common functionalities across the different packages. It is in charge of delivering authentication and authorization. Provide, monitor and maintain the overall infrastructure of the platform.

## Design

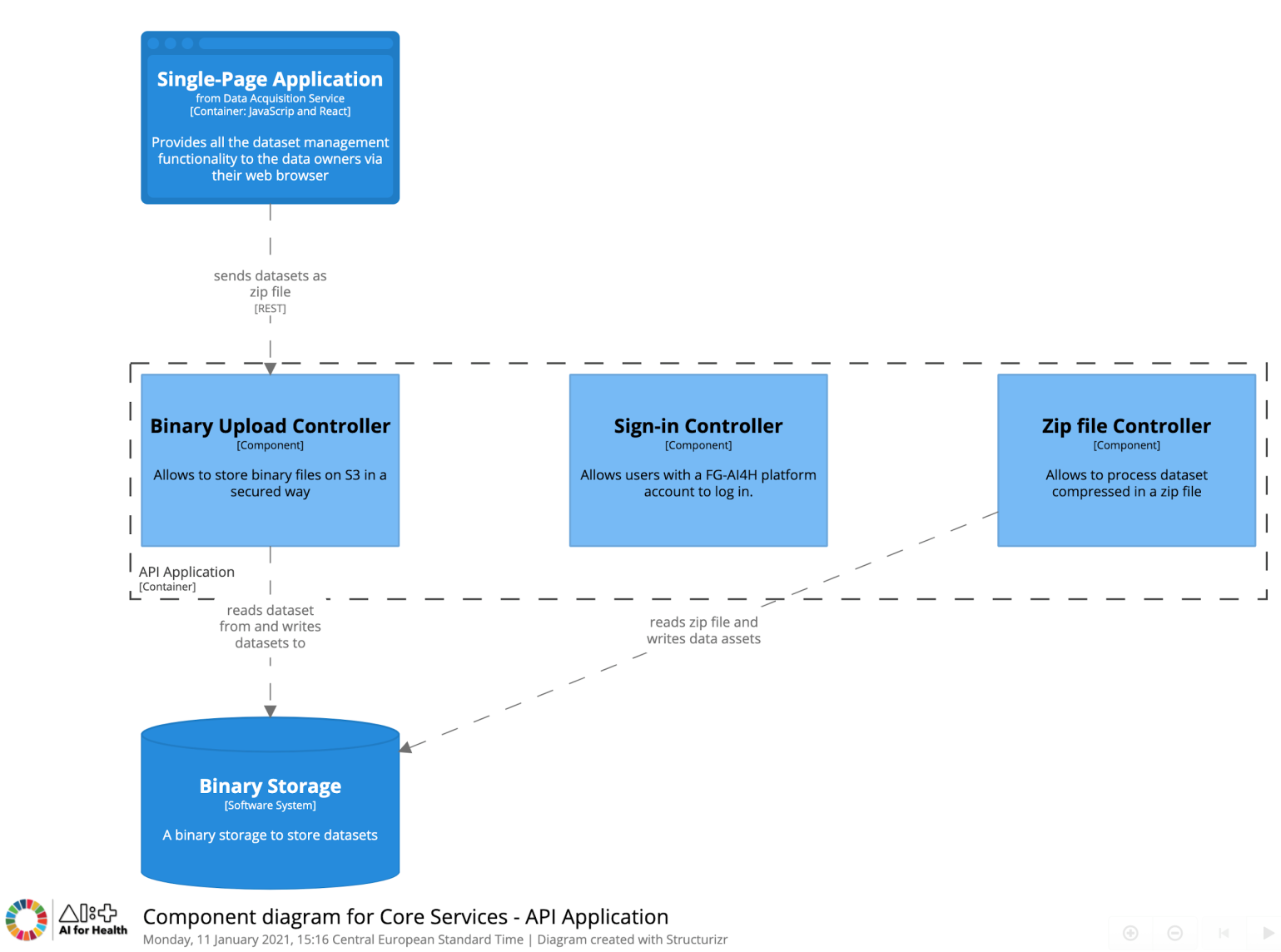


Figure 2.1: Component diagram for Core Services

## Implementation

For the MVP milestone, we have limited the features that we implemented. We decided to build our solution on top of the FHIR implementation of AWS.

FHIR Works on AWS is a framework to deploy a FHIR server on AWS. This package is an example implementation of this framework. The power of this framework is being able to customize and add in additional FHIR functionality for our unique use-case.

The system architecture consists of multiple layers of AWS serverless services. The endpoint is hosted using API Gateway. The database and storage layer consists of Amazon DynamoDB and S3, with Elasticsearch as the search index for the data written to DynamoDB. The endpoint is secured by API keys and Cognito for user-level authentication and user-group authorization. The diagram below shows the FHIR server's system architecture components and how they are related.

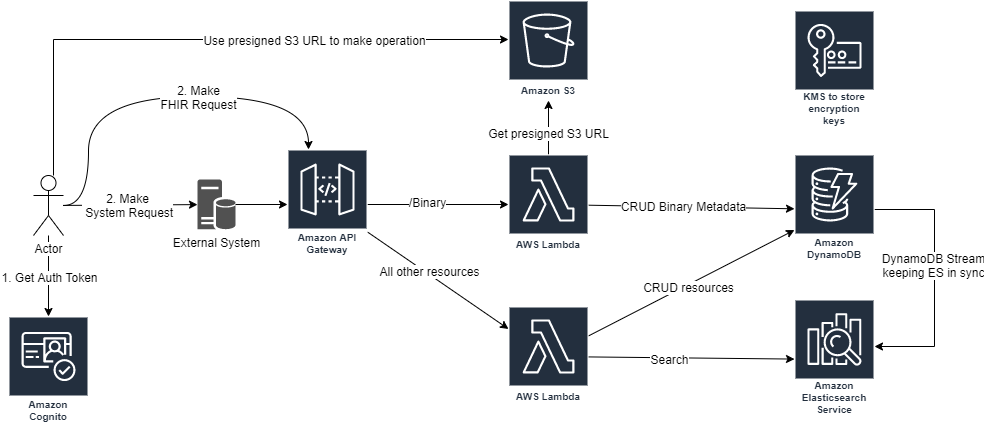


Figure 2.2: AWS FHIR system architecture

These components have been integrated into our system architecture (Figure 1.10) to serve multiple purposes.

An additional Lambda function to handle zip files has been deployed. This component is in charge of automatically unzipping the dataset archive submitted by Data Owners and pre-processing it. Please refer to section 3 for more details.

# Data Acquisition and Storage Package

*Note that during the execution and development process phase drivers and contributors decided to combine the Data Acquisition Package and the Data Storage Package into one package denoted as Data Acquisition and Storage (DAS) Package.*

## Overview

The main goal of the DAS package is to provide high-quality data for downstream usage. Therefore, the acquisition of data and the management of datasets are the two main working areas.

For the acquisition of data emphasis is put on the compilation of the raw data (data points and labels or outcomes), the availability of metadata (see section 3.4) as well as patient consent information. Consequently, the platform will allow for data ingestion, registration of data and metadata, management of patient consent information. In addition, standardized data and metadata requirements will be enforced and data quality checks will be performed.

The data storage functionality allows for processing and storing data and associated metadata. Consequently, the focus is on the safe and secure storage of raw data, labels and patient consent information, on data governance and data protection. The storage module provides an interface for other packages, such as the Evaluation Package (see section 4) and manages access to the data and metadata for downstream usage.

## Design

There are two main software services related to data acquisition and data storage. One is denoted "Data Acquisition Service" and the other "Data Storage Service". For the time being in the MVP phase we combine both functionality into a single service called "Data Acquisition and Storage Service" (Figure 3.1). The Data Acquisition Service allows data owners to share and manage their datasets. This service consists of a web-application that can be accessed by the data owner. The application contains a single-page application which provides dataset management functionalities to the data owner via their web browser (Figure 3.2). This single-page application is connected via an API (denoted as Dataset Controller) to the database and the Data Storage Service (Figure 3.3).

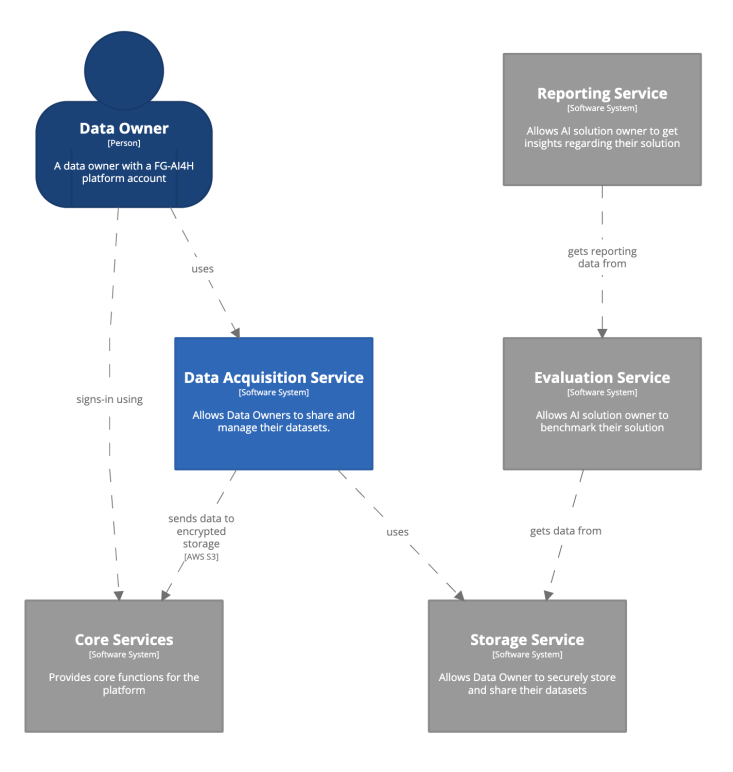


Figure 3.1: Data acquisition service - System context

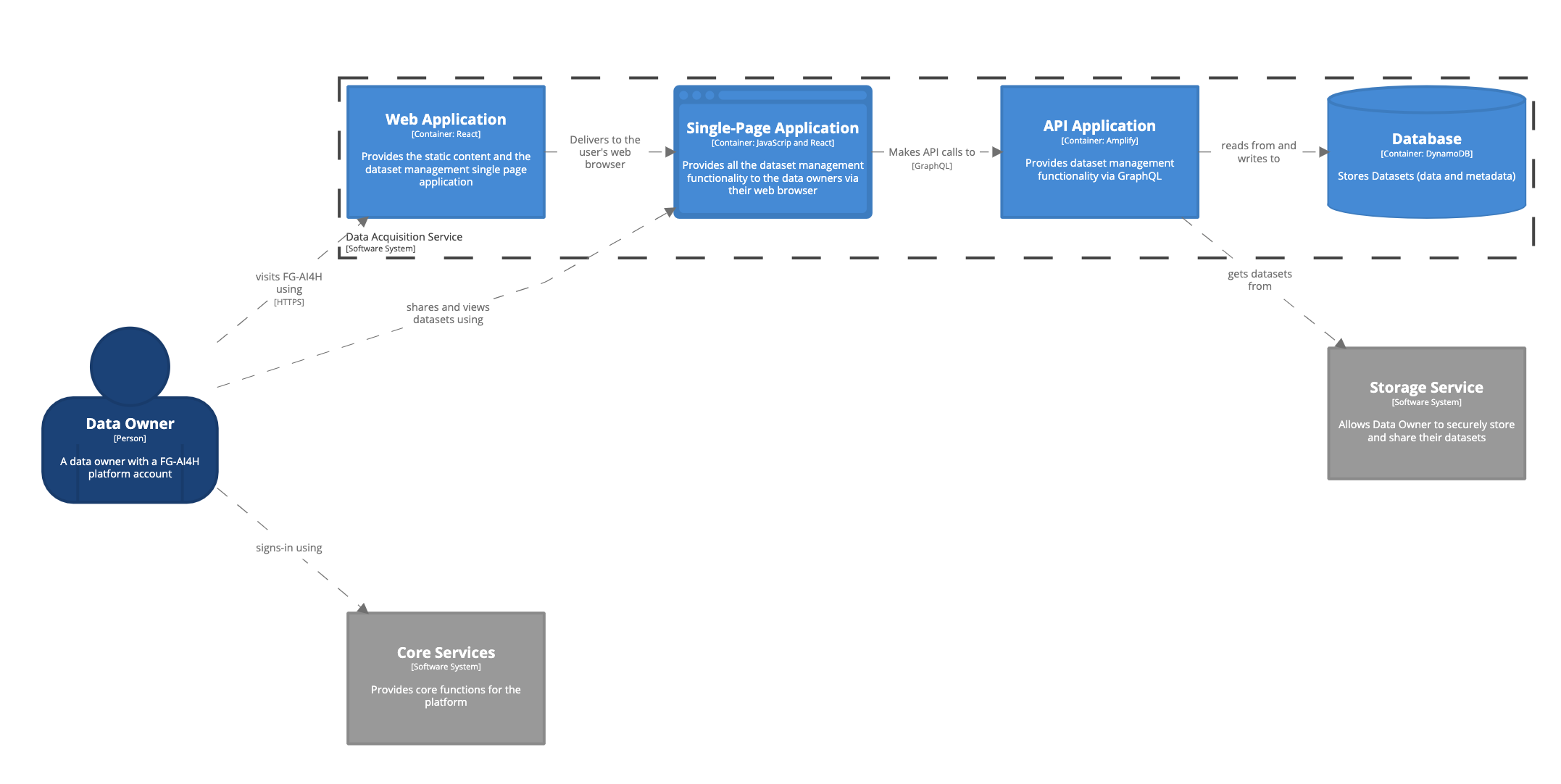


Figure 3.2: Data acquisition service - Container Diagram

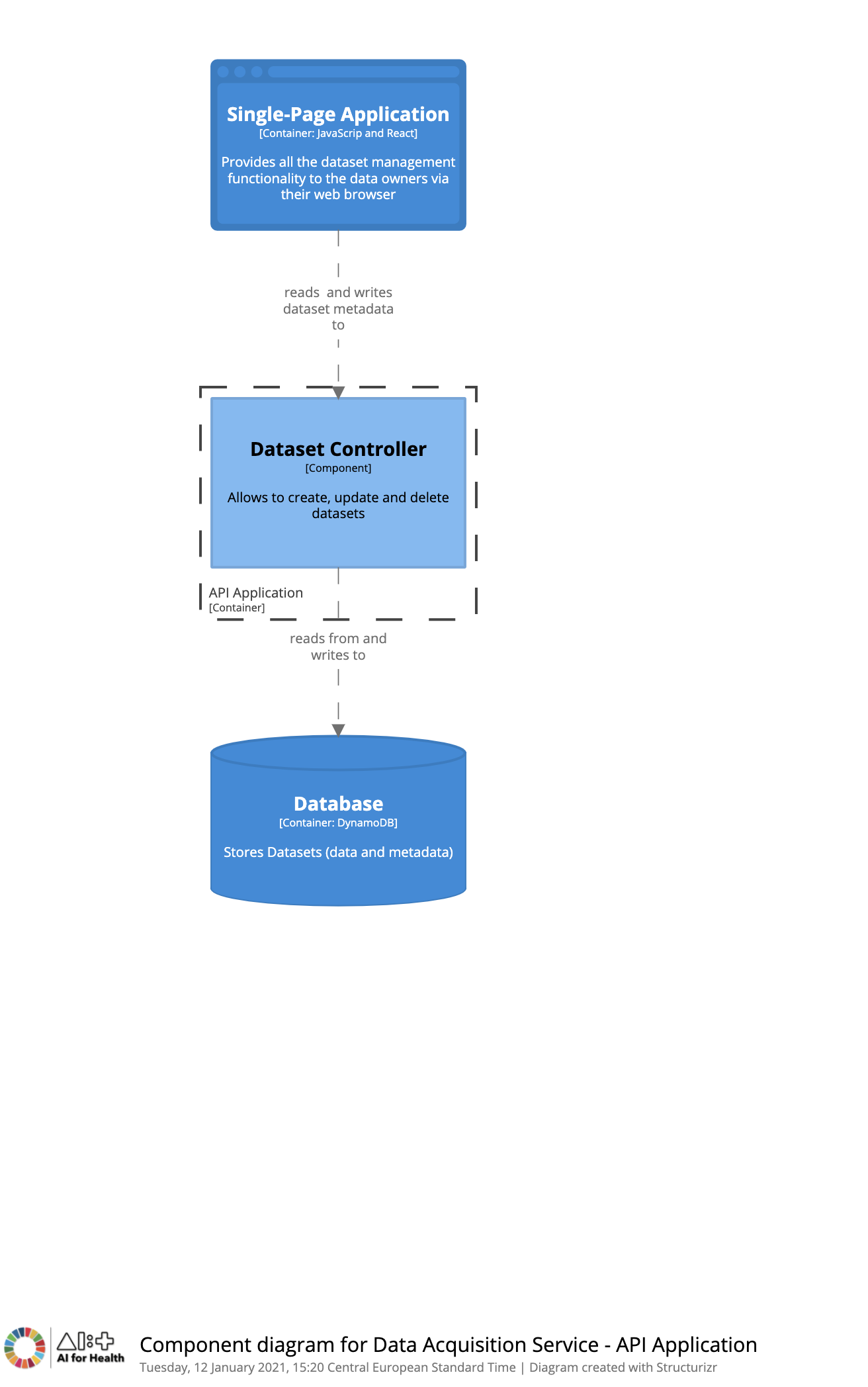


Figure 3.3: Data acquisition service - API application - Component Diagram

## Implementation

The main components are a:

1. Frontend web-interface;
2. Data Controller component to perform CRUD operations; and
3. Storage area to store datasets and metadata

The frontend web interface is built using React and Amplify which capture dataset and metadata over a secured wire. The Data Controller component is an auto-generated API from GraphQL which performs CRUD operations.

The storage area is a combination of S3 binary storage and DynamoDB. The dataset is KMS encrypted on S3.

The Data Acquisition and Storage Service provides a web user interface to upload datasets and transmit metadata for the dataset over a secured wired connection. On submit, the (potentially zipped) dataset file gets saved onto the AWS binary S3 bucket and the zip file is encrypted by KMS and thereafter unzipped using a lambda function. Another lambda service gets triggered to update DynamoDB with the URL of the dataset and the metadata for the dataset.

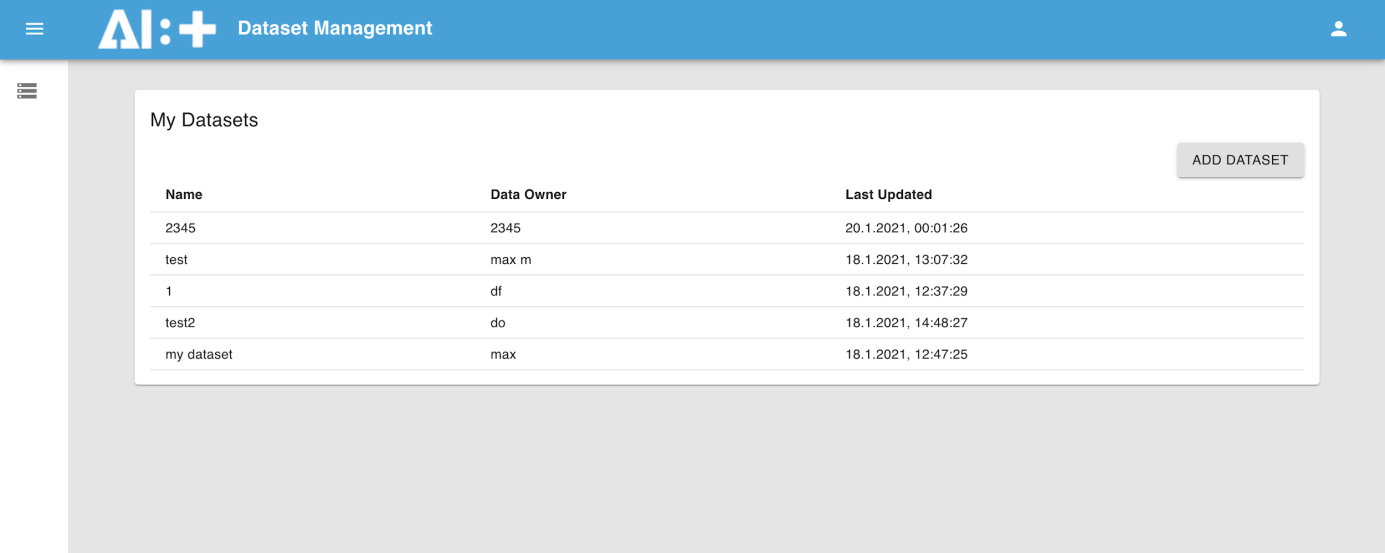


Figure 3.4: Dataset management Webview (MVP, screenshot from 2020-01-23)

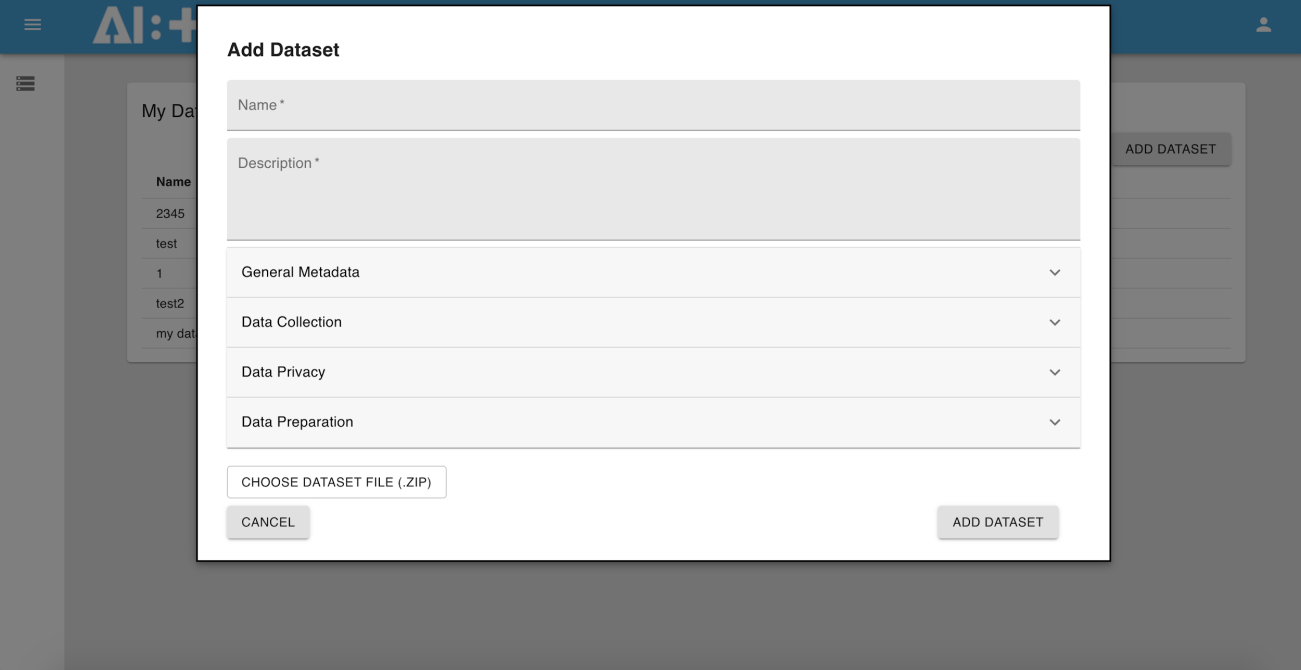


Figure 3.5: The addition of a new dataset (MVP, screenshot from 2020-01-23)

## Metadata

For the purpose of transparency, robustness and trustworthiness data and metadata requirements need to be specified and documented accordingly. These efforts are aligned with and informed by the deliverables DEL05"Data specification", DEL05\_1"Data requirements", and DEL05\_2"Data acquisition".

As an example the data specification sheet for the dataset used in the MVP, presented at Meeting K, 27-29 January 2021 is given in Figure 3.6 This sheet represents the metadata for the acquired dataset.

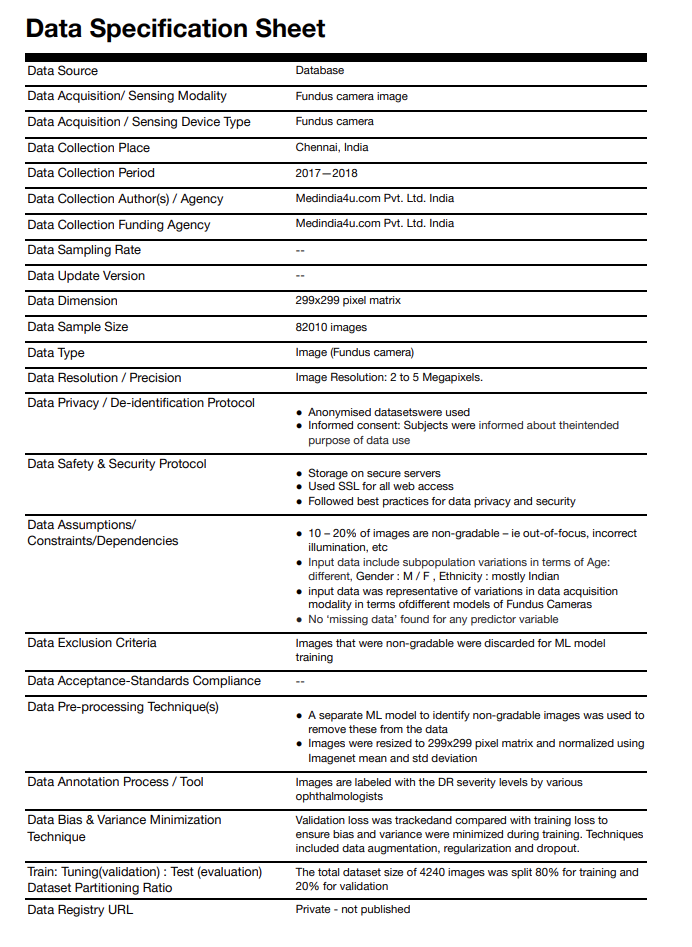


Figure 3.6: Example data specification sheet for the dataset used in the MVP (Meeting K)

Metadata acquired with the dataset can be used to tag datasets. Tagging of datasets enables searching of relevant datasets. Tags can play a significant role to filter and identify certain patterns in datasets to obtain data insights. Thus, metadata in effect can be a very powerful tool to tweak model data parameters in the Evaluation Package and refine algorithm. Metadata is directly correlated to clinical outcome and therefore is a very good predictor. Metadata helps in identifying the correct dataset for evaluation of a model. It helps match the data with the desired clinical outcome. For example, geographical differences can be seen for focal to patient distance for XRay machines, diabetic retinopathy model should identify early diagnosis of retinopathy to prevent blindness, so it should perform with higher accuracy in early grades of disease.

# Evaluation Package

## Overview

The evaluation package seeks to provide the users with a variety of benchmarks aimed to test the performance of their ML models. The current platform is built upon the open-source assessment platform EvalAi. EvalAi already provides essential functionalities, such as the creation and management of benchmarks and dockerized submission of ML models. The project is realized in Django, a Python-based web framework. Data and model storage, submission queueing and benchmark execution are all hosted on AWS. Currently, a demo, staging, and production version of the project are all deployed and continuously developed. In addition, we developed a qualitative questionnaire which the user needs to fill out before submitting a model. This is meant to complement the quantitative evaluation. The next step is to write an evaluation script for a diabetic retinopathy model and execute a full benchmarking cycle, involving questionnaire, submission, execution, and results.

## Evaluation process

The overall evaluation process is modelled according to the workflow proposed by WG-DAISAM: transmit, understand, audit and report. Each step takes reference to corresponding FG-AI4H documentation. An overview can be found in Figure 4.1. You can find more details on the process and example evaluations on real AI4H use cases in FGAI4H-J-049[[1]](#footnote-1)

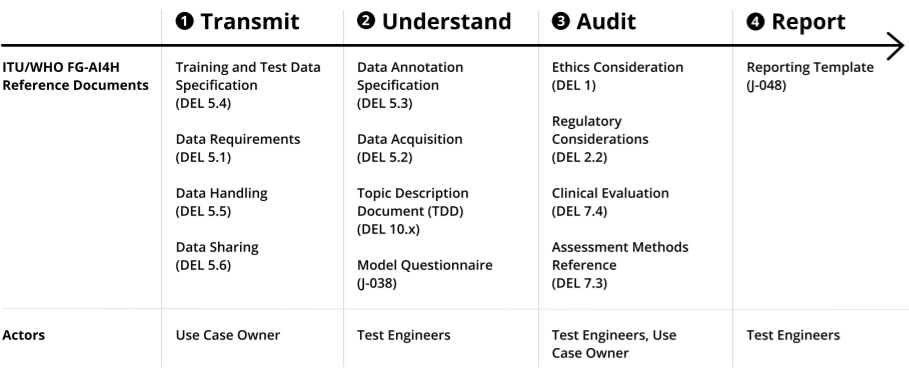
****

Figure 4.1: An overview of the WG-DAISAM evaluation process for AI4H models.

## Architecture

Figure 4.2 illustrates the EvalAi architecture.

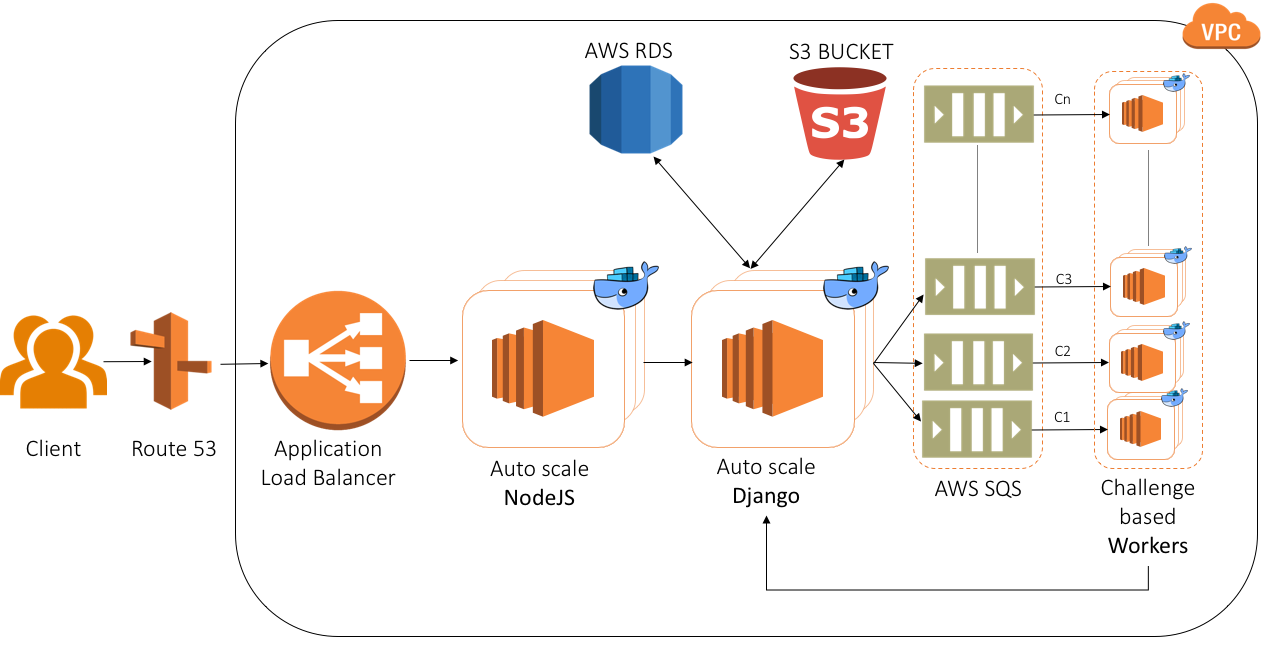


Figure 4.2: EvalAi architecture

Source: <https://evalai.readthedocs.io/en/latest/architecture.html>

EvalAi is structured into separate docker images, meaning frontend, backend, and workers all have their separate environments. Hosted on AWS, the containers are run on an EC2 instance, which is connected to an RDS database and an S3 Bucket for storing static data. Submissions by a user are currently made as dockerized ML models through a command-line tool. At successful submission, the docker image is also stored in AWS, where it will be tested against a benchmarking data set from the data acquisition package. To avoid a bottleneck, the submissions are also queued by AWS SQS. Once the submission has been evaluated, the results are returned to the user.

The current structure will be mostly kept in place with some changes such as the additional questionnaire, integration of dataset acquisition, a model sharing functionality, and the integration of the command line tool into the UI.

## Implementation

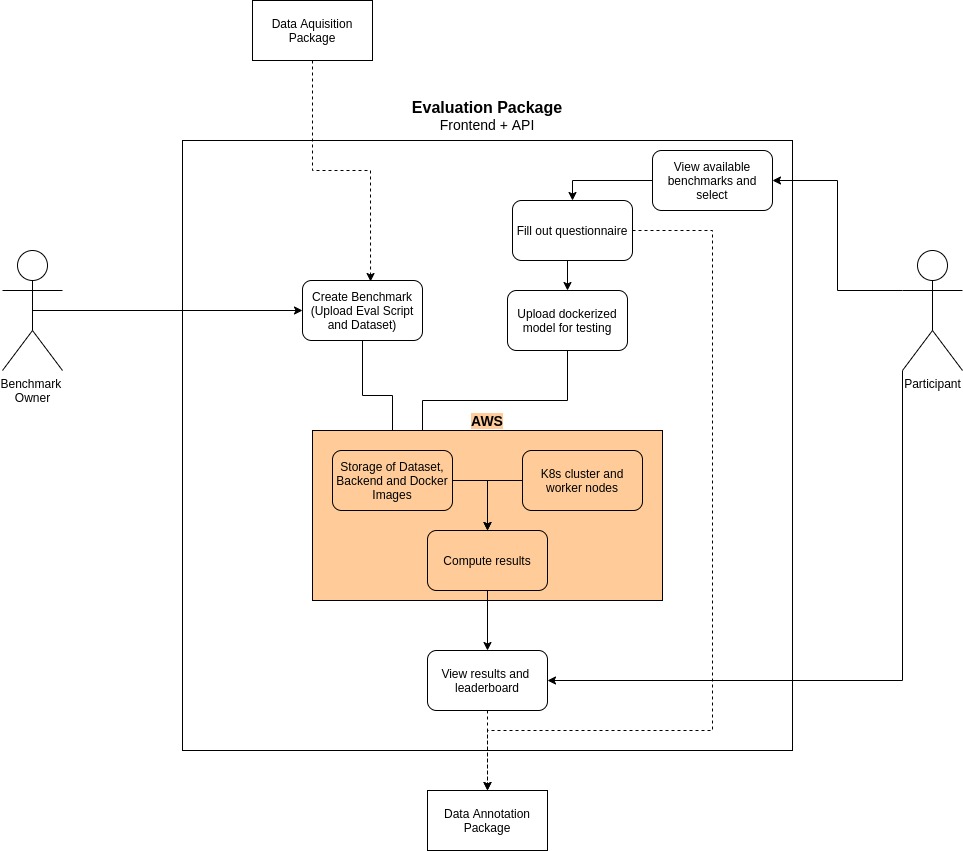


Figure 4.3: Evaluation service UML

Since the evaluation package makes use of EvalAi, essential functionalities (Figure 4.2) of the package have already been implemented. Most of the backend is written in Python and the frontend is mostly Angular-based. Essential steps of the implementation are configuring the project to run on our AWS instances and writing our own functioning benchmarks. New benchmarks are submitted to the platform as of a .yml file, detailing the benchmarking settings, alongside a Python evaluation script and the dataset.

# Reporting Package

## Reporting package-overview

* Reporting Package (RP) primarily deals with the preparation and presentation of AI4H model evaluation results generated by the Evaluation Package (EP) of the AI4H assessment platform.
* Reporting Package provides a unified structure to communicate and report the performance scores, metrics and their measures, properties, features and limitations of the AI4H model evaluation process as well as information related to the allied computational platform performance measures
* Reporting Package provides a customizable reporting interface to support ease of comparison, classification and reproducibility of different AI4H model evaluation results

## Reporting service-workflow description

Figure 5.1 shows the user journey for the reporting service

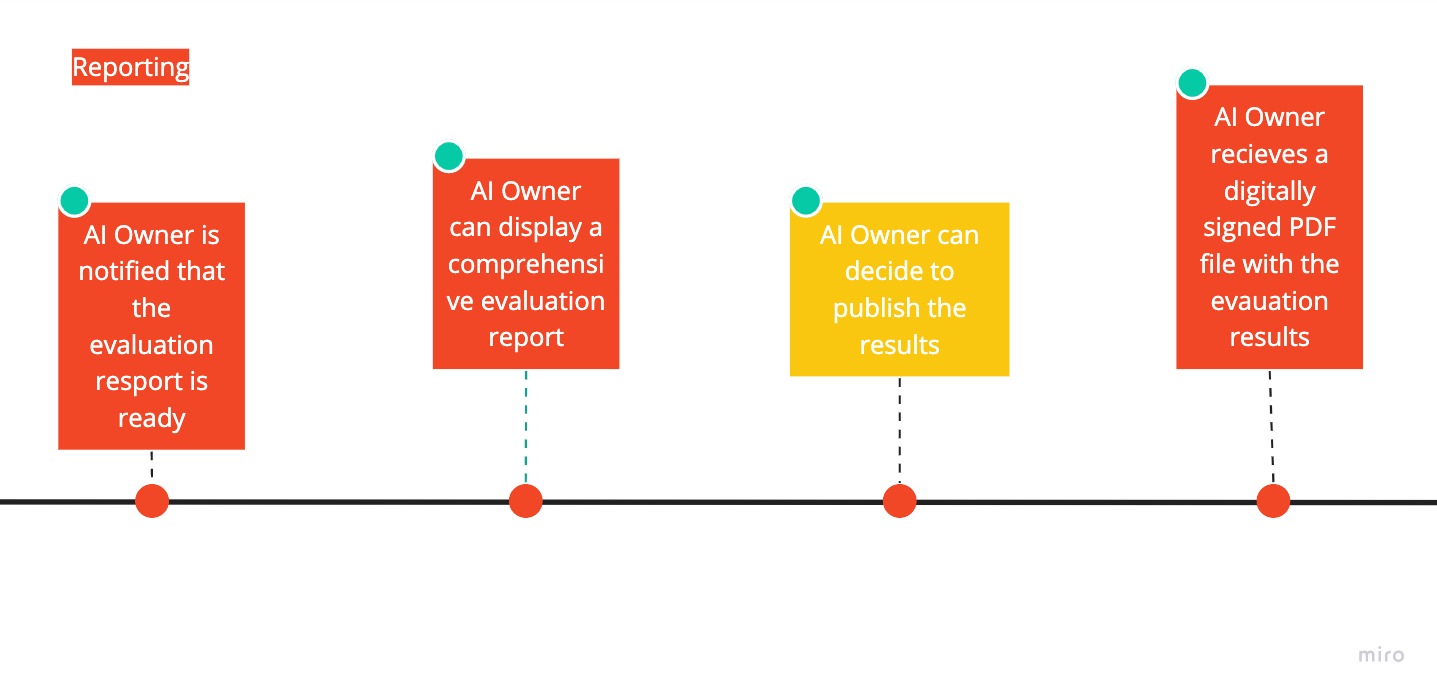


Figure 5.1: Reporting service-activity diagram

## Reporting package-functionality matrix

The functionality matrix for the reporting package is shown in Table 5.1.

Table 5.1: Functionality matrix

|  |  |
| --- | --- |
| **Functional Requirements** | |
| **User Interface Design** | * Graphical Screen/ Display and aesthetics (colour, shape, layout, etc) |
| * User roles, access privilege / rights |
| * Intended user goals and objectives |
| * Intended user tasks & sub-tasks |
| * User task sequence and user task hierarchy |
| * Interface objects (graphical screen / display, menus, widgets) |
| * Mapping of user tasks to interface actions |
| * Navigation nodes and links for each interface action |
| * Internationalization and accessibility standards |
|  | |
| **Inter-Package Communication** | * Data model / object schema validation |
| * Data access interface with Evaluation Package |
| * Data query set definition for data retrieval |
| * URL mapping for data retrieval-web API consumption |
|  | |
| **Report Management** | * Reporting parameter configuration |
| * Publish report (PDF, CSV, etc) |
| * Digitally signed report |
| * Report version control and storage |
| * Report retrieval |

## Reporting package -architectural design

* Reporting package is designed as a web application
* A Model-View-Template (MVT) architecture pattern is proposed for the design of the reporting package. An illustration of the architecture is shown in Figure 5.2

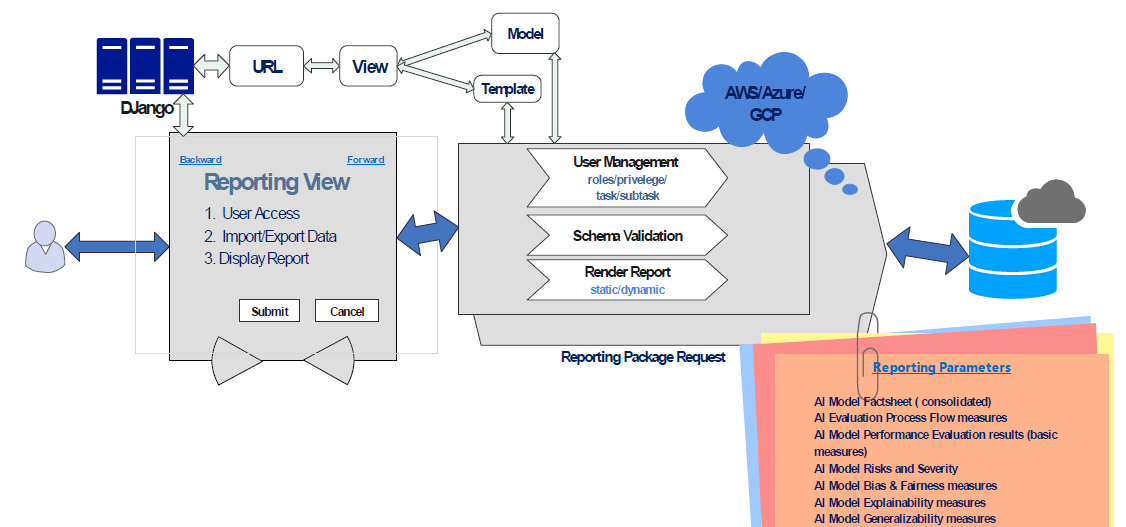


Figure 5.2: Reporting package-architecture overview

## Reporting package - module design

Table 5.2 shows the main software modules, their core functions and technologies used for their implementation.

Table 5.2: Reporting package-module design

|  |  |  |
| --- | --- | --- |
| Module | Core Function | Implementation Technology |
| Backend module | Defines the Web API/‌Service for retrieval of AI4H model evaluation result data in JSON format | Django-VIEW (Python) |
| Database module | Defines the schema for AI4H model evaluation result data | Django-MODEL (SQLIte DB) |
| Frontend module | Renders the report view of AI4H model evaluation result data | Django-TEMPLATE file (HTML, CSS) |

## Reporting package -component-level design

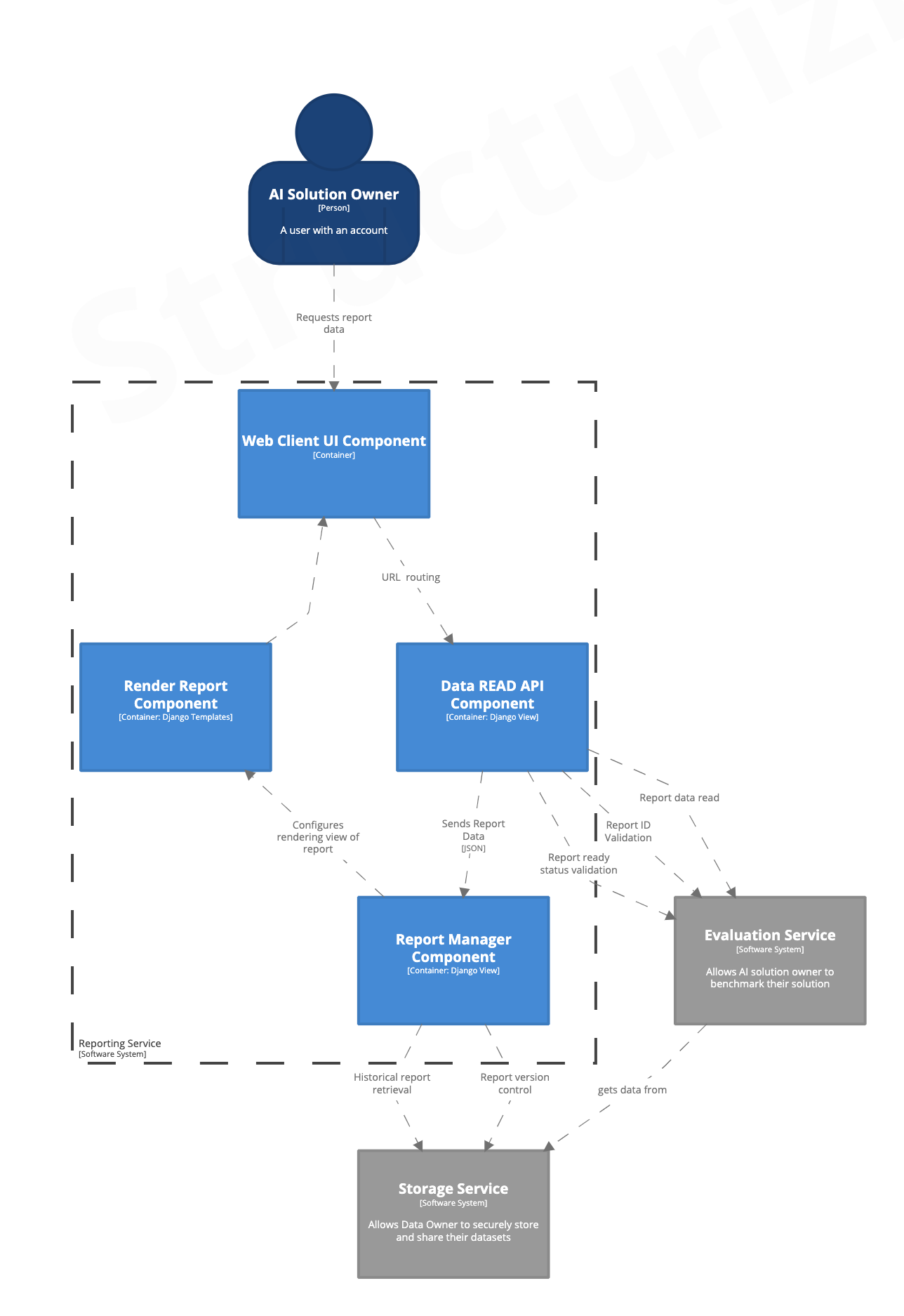


Figure 5.3: Reporting package-component-level diagram

Figure 5.3 shows the component –level description of the reporting service. The various components and their communication processes are explained as follows:

1. User sends data request via web client / browser (HTTP request)
2. Predefined URL pattern routes the HTTP request the intended VIEW function
3. URL mapping function matches the particular report ID pattern (string or digit) that is appended in the URL and passes this ID to the VIEW function
4. Data Read API - VIEW function processes the HTTP request, queries the database via MODEL objects and fetches the particular data record in JSON format
5. Report Manager –VIEW function formats and renders the data via TEMPLATE files (HTML pages) and returns the HTTP response to the user via web client / browser

## Reporting service – modalities

Figure 5.4 shows the screenshot of the user interface for the Reporting Service web application

* **Basic Report**: Basic reporting mode allows the user to generate a pre-configured report with basic minimum parameters for the AI4H model evaluation results. A specimen of the basic report generated for the Diabetic Retinopathy use case is shown in Figure 5.5
* **Custom Report:** Custom reporting mode allows the user to customize the report based on a set of configurable parameters for the AI4H model evaluation results



Figure 5.4: Reporting service app - home page screenshot

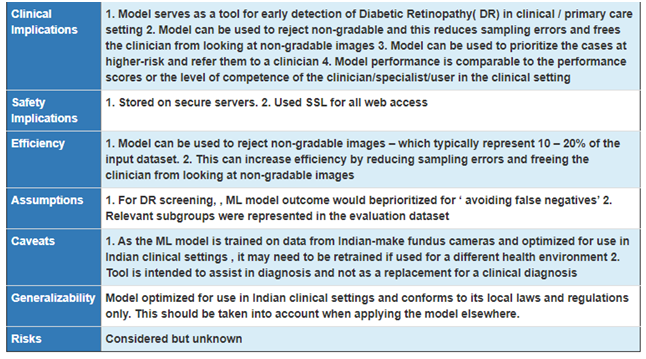
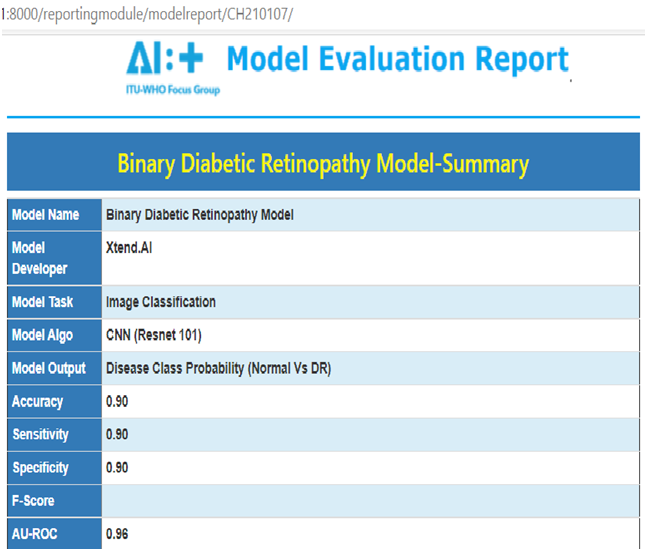


Figure 5.5: Reporting service-specimen model report

## Reporting service -parameter configuration criteria

* AI Model Factsheet (consolidated)
* AI Evaluation Process Flow measures
* AI Model Performance Evaluation results (basic measures)
* AI Model Risks and Severity
* AI Model Bias & Fairness measures
* AI Model Explainability measures
* AI Model Generalizability measures
* AI Model Interpretability measures
* AI Model Robustness measures
* AI Model Uncertainty measures
* AI Assessment platform -computational infrastructure measures
* Custom/ Domain specific / Clinical effectiveness -scores & measures
* Intended Use
* Patient Safety
* Device Security & Privacy
* Other

## Reporting service - visualization formats

* Text
* Dashboard
* Tables
* Graphical plots
* Statistical plots
* Comparison charts
* other

## Reporting package - design evaluation

* Evaluation criteria
  + Qualitative criteria
    - questionnaires
    - subjective scales
    - Other
  + Quantitative criteria
    - number of tasks completed in unit time
    - frequency of action
    - sequence of actions
    - number and types of errors encountered
    - error recovery time
    - time spent for online help
    - number of help references per unit time
    - other
* Prototype building
* Test plan and procedure creation

# Project organization

## Overview

The project is organized in six different work streams:

1. The Core Package (CP)
2. The Data Acquisition Package (DAP)
3. The Data Storage Package (DP)
4. The Data Annotation Package (AP)
5. The Evaluation Package (merged with Prediction Package) (EP)
6. The Reporting Package (RP)

Each package has at least one driver in charge of reporting activities to the project management team. Drivers organize the work in their streams and coordinate with contributor members.

## Meetings

The management team meets on a bi-weekly call on Thursdays (odd weeks) from 1230 to 1330 hours (Geneva time). The meeting agenda is publicly available in the following Google calendar: <https://calendar.google.com/calendar/u/2?cid=Y19obGJrOTBzNmFnczZpMjRxbnMzdXNyMml1Z0Bncm91cC5jYWxlbmRhci5nb29nbGUuY29t>.

# Core Package

## Membership

| Driver |
| --- |
| Marc Lecoultre |
| Members |
| Dominik Schneider |
| Shobha Iyer |

## Meetings

Bi-weekly calls on Thursdays (even weeks) from 1:30 to 3:00 pm CET. The meeting agenda is publicly available [here](https://calendar.google.com/calendar/u/2?cid=Y18yczg0M2U0ZDNiYWFhMjA4NzVjYmFna2N1NEBncm91cC5jYWxlbmRhci5nb29nbGUuY29t).

## Documentation

1. Wiki page

# Acquisition Package

## Membership

|  |
| --- |
| Driver |
| Joachim Krois |
| Members |
| Dominik Schneider |
| Shobha Iyer |
| Shruti Choudhary |
| Pradeep Balachandran |

## Meetings

## Documentation

# Data Storage Package

This package has been temporarily merged with the [acquisition package](#_The_acquisition_package).

# Annotation Package

## Membership

| Driver |
| --- |
| Marc Lecoultre |
| Members |
| Henry Hoffmann |

## Meetings

Not yet defined.

## Documentation

# Evaluation Package

## Membership

|  |
| --- |
| Driver |
| Elora Schörverth |
| Members |
| Luis Oala |
| Alixandro Werneck |
| Steffen Vogler, Bayer AG |
| Pradeep Balachandran |
| Danny Xie Li, Tecnológico de Costa Rica |
| Kamran Ali, Hasso Plattner Institute |

## Meetings

Not yet defined.

## Documentation

# Reporting Package

## Membership

| Driver |
| --- |
| Pradeep Balachandran |
| Members |
| Alixandro Werneck |
| Andrea Garcia |
| Danny Xie Li |
| Dominik Schneider |
| Elora Schörverth |
| Joachim Krois |
| Kamran Ali |
| Luis Oala |
| Marc Lecoultre |
| Shobha Iyer |
| Shruti Choudhary |
| Steffen Vogler |

## Meetings

Not yet defined.

## Documentation

# Tooling

The group uses the following toolset for communication, documentation and development purposes:

## Slack

As Slack workspace has been created to communicate on a daily basis between team members: assessmentpla-m174974.slack.com.

Several channels have been put in place, see Table 12.4.

Table 12.4: Slack channels

| Channel | Description |
| --- | --- |
| #general | General channel available to all members and linked to MS Teams workspace |
| #devops | Reflects all the updates done on Azure DevOps workspace |
| #module\_annotation\_package\_ap | Specific channel for the Annotation Package. It reflects all the updates done on the Annotation Package work items in Azure DevOps |
| #module\_data\_acquisition\_package\_dap | Specific channel for the Acquisition Package. It reflects all the updates done on the Annotation Package work items in Azure DevOps |
| #module\_datastorage\_package\_dp | Specific channel for the Data Storage Package. It reflects all the updates done on the Annotation Package work items in Azure DevOps |
| #module\_evaluation\_package\_ep | Specific channel for the Evaluation Package. It reflects all the updates done on the Annotation Package work items in Azure DevOps |
| #module\_reporting\_package\_rp | Specific channel for the Reporting Package. It reflects all the updates done on the Annotation Package work items in Azure DevOps |

# Resources

## General Meeting Notes

General meeting notes are grouped into the following Google Drive directory: <https://drive.google.com/drive/folders/1qF6Z_r373sDSKBy-YXbgLHGu-wbAlvdm?usp=sharing>

## Working Documents

* **Living Document**FGAI4H-OpenSourceSoftware: <https://docs.google.com/document/d/1eksm8dm7MYuNjtThRp-zmwlxvFXUrjSjDnkpuMwZNJ4/>
* **Azure DevOps**https://dev.azure.com/mllabai/FG-AI4H%20Assessment%20Platform
* **Functionality matrix**[FG-AI4H-OpenSourceSoftware-FunctionalityMatrix](https://docs.google.com/spreadsheets/d/1t3uwPQIKFKJejtjINWOkHR-h7nlZsuYy4pqhIdCSiEo/edit?usp=sharing)
* **Miro board**<https://miro.com/app/board/o9J_lfV5OMk=/>
* **bit.dev**[FGAI4H·Bit](https://bit.dev/fgai4h) – <https://bit.dev/fgai4h>
* **Figma**<https://www.figma.com/file/mbBGuSEvGxOFl4unb5NYj9/>
* **Structurizr**<https://structurizr.com/workspace/57327/>

# Communication

* **Slack**[https://assessmentpla-m174974.slack.com](https://assessmentpla-m174974.slack.com/)
* **Management meeting agenda**<https://calendar.google.com/calendar/u/2?cid=Y19obGJrOTBzNmFnczZpMjRxbnMzdXNyMml1Z0Bncm91cC5jYWxlbmRhci5nb29nbGUuY29t>
* **Core Package meeting agenda**<https://calendar.google.com/calendar/u/2?cid=Y18yczg0M2U0ZDNiYWFhMjA4NzVjYmFna2N1NEBncm91cC5jYWxlbmRhci5nb29nbGUuY29t>

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Oala, Luis, Jana Fehr, Luca Gilli, Pradeep Balachandran, Alixandro Werneck Leite, Saul Calderon-Ramirez, Danny Xie Li et al. "ML4H Auditing: From Paper to Practice." In *Machine Learning for Health*, pp. 280-317. PMLR, 2020. [↑](#footnote-ref-1)