CAPSTONE TEAM 5 PART 2

Executive Summary (From Part 1)

CLIENT COMPANY

Harvard Health Center (HHC) is a federally qualified nonprofit health center and an FTCA-deemed provider of medical, dental, and behavioral health services. Founded in 1984, it currently has offices in four rural cities in North Georgia.

TECHNOLOGY SUPPLIER

Integrationals.com is a technology provider specializing in integrating healthcare data, applying predictive models to improve patient outcomes, and developing applications that present a unified view of patient medical histories for patients and healthcare providers. We also focus on enabling interoperability between healthcare applications through the FHIR API. Our target market includes small to medium-sized medical offices and hospital chains in rural America, helping them comply with the 21st Century Cures Act.

BUSINESS GOAL – PROBLEM STATEMENT

The HHC management aims to meet the federally mandated compliance deadlines of the Cures Act. Deadlines have been extended to 2022 and 2023 in some cases to account for the Covid-19 pandemic, but achieving compliance remains a complex task. They rely on an outdated Electronic Health Record (EHR) system that does not support FHIR, which prevents them from meeting the certification and information blocking requirements outlined in parts 170 and 171 of the Cures Act.

HHC hired Integrationals.com to assist in migrating HHC from its current EHR system to a fully FHIR-compliant platform. They also see an opportunity to fully leverage the interoperability benefits promoted by the CURES Act. With this in mind, Integrationals.com will create a plan to consolidate HHC's patient data into a centralized repository, enabling the application of ML models to predict health outcomes and identify at-risk patients more quickly. Additionally, they need to present the consolidated data to patients and healthcare providers through a user-friendly mobile app.

Information Blocking refers to the Cures Act, which requires medical organizations to provide easy access to patient medical data and prohibits them from restricting access to information. There is an official process for affected parties to report Information Blocking.

The Cures Act defines information blocking as business, technical, and organizational practices that prevent or significantly discourage access, exchange, or use of electronic health information (EHI). Information blocking is a clear violation of the Act when an actor knows, or (for some actors like EHR vendors) should know, that these practices are likely to interfere with access, exchange, or use of EHI.

The challenge is that a large amount of their clinical data exists in silos of unstructured patient notes. There is a need to extract key predictive features and turn them into a structured format suitable for ML processing.

HHC emphasized the need to stay compliant with HIPAA and HITECH requirements.

DIGITAL TECHNOLOGY SOLUTION STRATEGY

Integrations will help HHC select and implement a modern EHR system that already uses FHIR to speed up compliance with Cures. The project also involves converting and migrating data from the current system to the new EHR, along with training and support.

To help HHC achieve its second goal of enhancing patient outcomes and identifying at-risk patients, Integrationals will propose incorporating AWS HealthLake.

In this setup, HealthLake will leverage the Redox service to ingest patient healthcare data using the FHIR API, thereby enhancing capabilities for the Release of Information (ROI) workflow.

Click here for a diagram of the complex ROI: ROI Diagram

Once HealthLake processes the data, it will use its integrated Natural Language Processing (NLP) engine to extract key features from medical notes while building a comprehensive profile of each patient. HealthLake will then export the processed data to cloud storage, where Databricks will import and integrate it into its Delta Lake (Lakehouse) layer. The Databricks Feature Store technology will organize all available features into a format suitable for ML processing, allowing the Databricks engine, supplemented with data from insightsDB, to train and run predictive ML models. Integrationals will create a presentation layer that uses a REST API to combine the predicted information into the Patient and Provider App. Please refer to the Future System Flow Diagram for a visual overview.

The first phase, which is the focus of this project, will aim to provide easy access to healthcare providers. The second phase, part of a future project and currently out of scope, will allow direct access for patients.

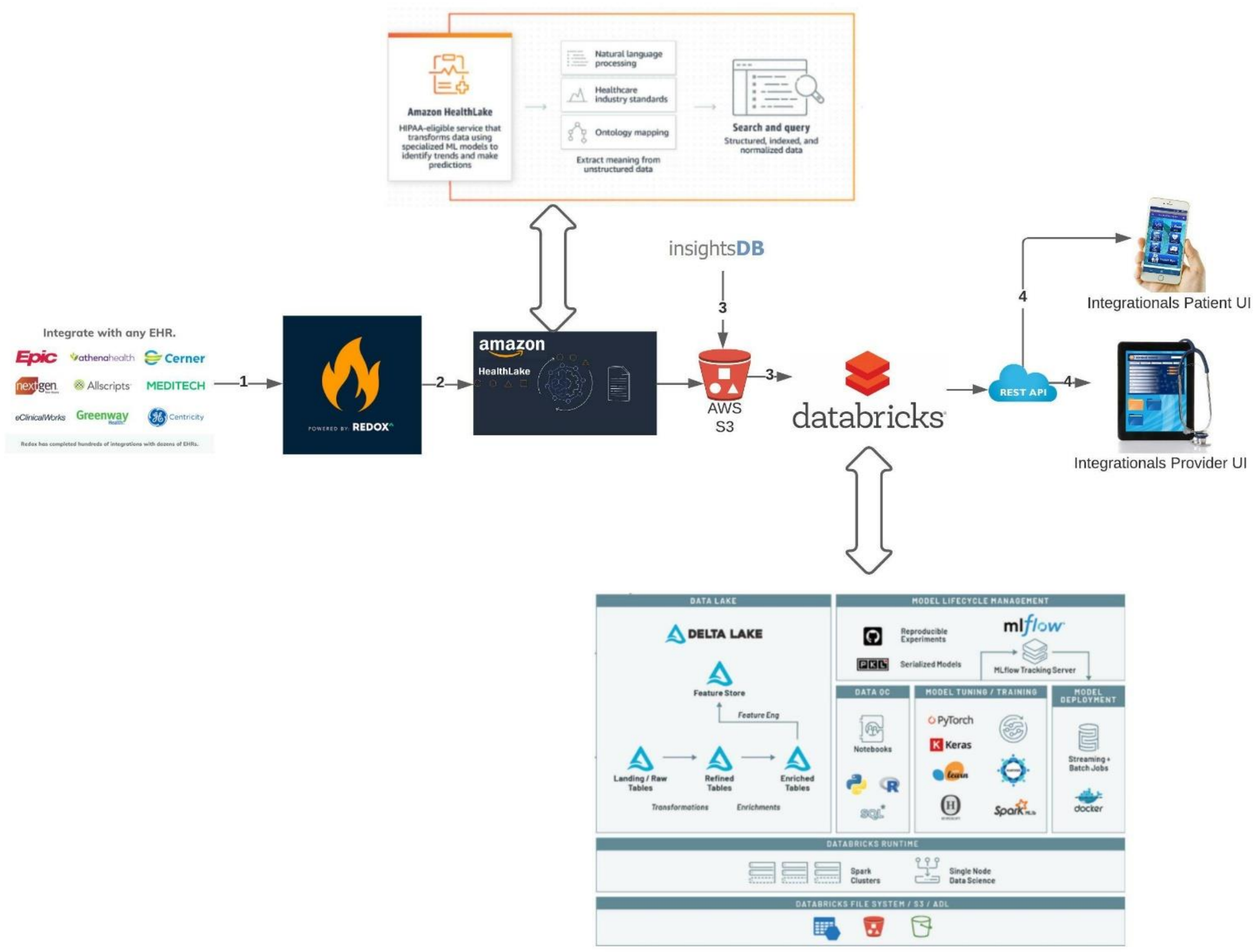
SUCCESS CRITERIA – EXPECTED BUSINESS OUTCOME

The success criteria for this engagement include enabling HHC to achieve interoperability with other healthcare systems through the FHIR standard API, eliminating information blocking, and thereby achieving Cares compliance.

HHC will also enhance patient outcomes by using ML predictions, taking corrective action proactively whenever possible, and utilizing a user-friendly app to access the consolidated information.

---- end of Executive Summary

I. Architectural Approach



This diagram shows how the main parts of the Integrationals Platform work together, starting on the left with a group of EHR systems that supply healthcare data, ending with a unified view of patient and medical condition prediction information on the right.

In step one, Redox offers a unified approach to ingest healthcare data from multiple EHR systems using the FHIR API, making data provisioning easier**.**

In step two, HealthLake retrieves data from Redox and utilizes its healthcare-specific Natural Language Processing capabilities to expand the patient record by extracting structured data, such as medical ontologies like ICD-10, from unstructured patient notes. HealthLake then exports its results to cloud storage.

In step three, Databricks imports data from cloud storage, coming from HealthLake and third-party insightDB. The Databricks component serves two roles. By implementing a Lakehouse architecture, Databricks functions as both the Data Warehouse that consolidates all patient information and a Machine Learning environment to train and deploy statistical models for predicting medical conditions, publishing its models via a REST API.

In step four, the Integrationals UI connects to the REST API endpoint and displays the information to providers and patients through its interface.

II. Software Solution

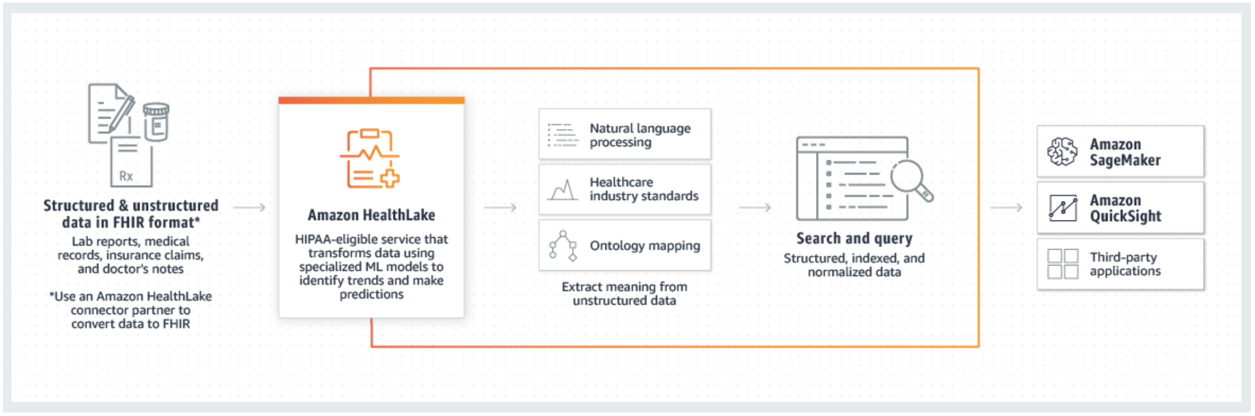
Redox Component

Please see section III. Integrations

HealthLake Component

Once REDOX processes the data, our system imports it into HealthLake, which stores the data in Amazon DynamoDB for CRUD operations. The data is then processed into an Amazon Elasticsearch Cluster to enable search functionalities. A key feature of AWS HealthLake is its integration with "Amazon Comprehend Medical," a natural language processing service designed specifically for medical use cases. Its main goal is to analyze all unstructured data to extract and index hundreds of medical data points needed for downstream analytics. After Comprehend Medical processes the data, it uses the newly extracted structured information to create a comprehensive and chronological view of patient health data, including prescriptions, procedures, and diagnoses. Finally, it stores the data in FHIR format, allowing AWS to search the text data seamlessly.

Once HealthLake processes the data, it moves it into object storage or S3 buckets, ready for batch import and export into machine learning models via Amazon SageMaker, Amazon QuickSight, or third-party applications available in the AWS Marketplace, such as Databricks.


Rx
Structured & unstructured data in FHIR format*
Lab reports, medical records, insurance claims, and doctor's notes
*Use an Amazon HealthLake connector partner to convert data to FHIR
Amazon HealthLake
HIPAA-eligible service that transforms data using specialized ML models to identify trends and make predictions
Natural language processing
Healthcare industry standards
Ontology mapping
Extract meaning from unstructured data
Search and query
Structured, indexed, and normalized data
Amazon SageMaker
Amazon QuickSight
Third-party applications

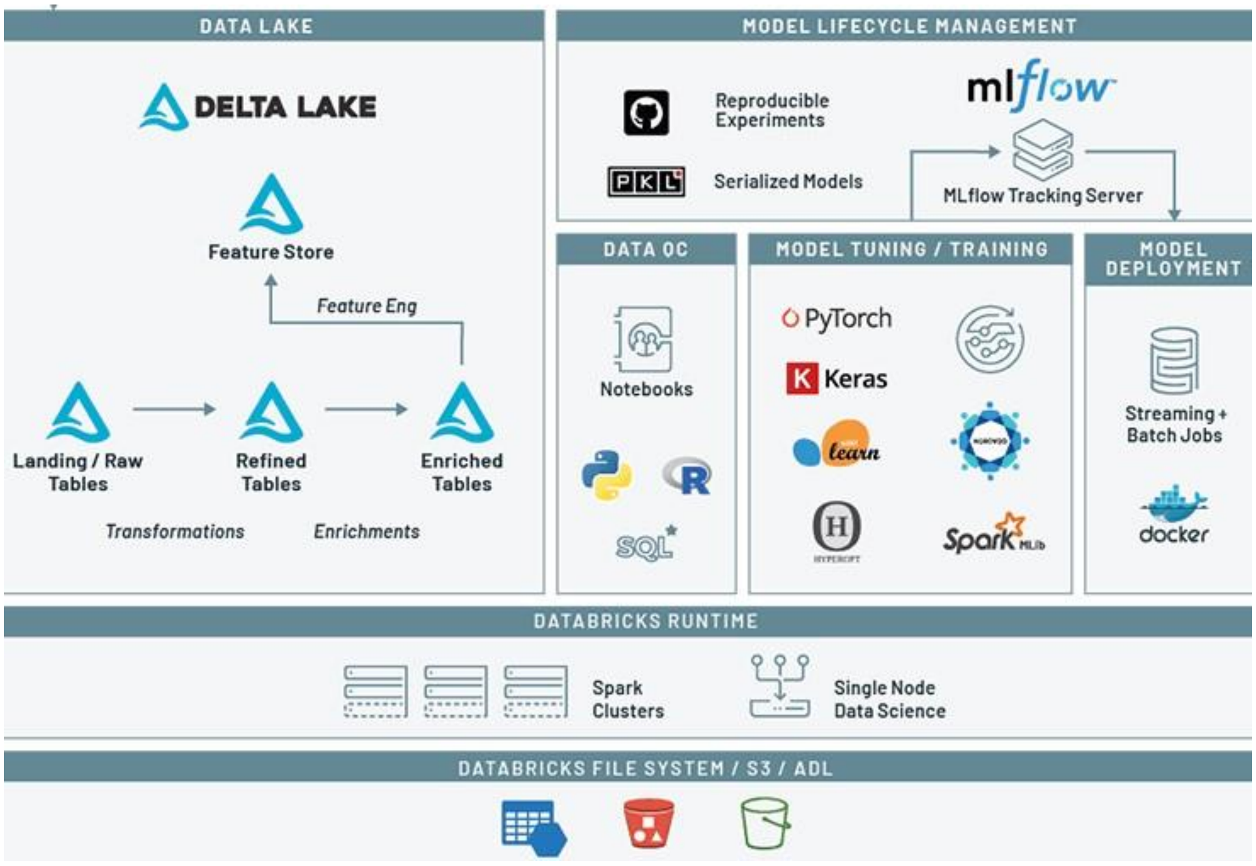
InsightsDB Component

Amazon provides a Data Exchange service as part of its AWS cloud platform, offering a wide range of datasets for research across various industries, including Healthcare and Life Sciences. The exchange serves as a centralized hub for accessing datasets that include Healthcare Claims, Electronic Health Records, Laboratory Results, Imaging, Genomic Sequencing, Patient-Reported Outcomes, and more.

One of the datasets available from the EHR section is insightsDB. This database contains de-identified patient demographic data, healthcare provider notes for radiology, physical exams, operations, medications, labs, drugs, and many other details. The dataset covers approximately six million unique patients and includes about eighteen million medical encounters over five years.

This information provides a training data foundation for machine learning models, supplementing our patient data from the EHR systems specific to our clients.

Databricks Component



Databricks is a Machine Learning (ML) platform composed of various components that work together to store data needed for business intelligence, training, testing, and deploying predictive models at scale. The following is a brief list of some primary Databricks components we will use in this project.

- **Spark:** The founders of Databricks created Spark. Spark is a data processing engine that automatically distributes data requests across many worker nodes and is one of the leading engines in the Big Data industry. In the Integrationals Platform, Databricks acts as the engine that manages requests to train predictive data models. Regarding scalability, Spark can handle petabyte-scale datasets, with some clusters comprising up to eight thousand nodes.

- **Delta Lake:** The Data Lake has served as the primary storage for Big Data for many years. Initially, because of the size and speed needed for Big Data processing, typical data warehouses couldn't meet these technical requirements, so the Data Lake fulfilled the Big Data needs. Over time, Data Lakes expanded without the control found in Data Warehouses and turned into Data Swamps, which became hard to manage. The next-generation storage model is called a Data Lakehouse, combining the best aspects of Data Lakes and Data Warehouses. Delta Lake is Databricks' implementation of the Data Lakehouse model. We will integrate and manage all healthcare data from various EHRs in Delta Lake. It is essential to understand that Delta Lake is not redundant with HealthLake, since HealthLake is specific to healthcare, while Delta Lake is more general.
- **Feature Store:** ML algorithms require input in a feature format, usually a numeric representation of raw data, although syntactic pattern recognition uses structural features such as strings and graphs. For an elementary example, raw data from a dataset of one million records of people's ages becomes a feature vector with ten age bins by decade, with one compartment for ages 0-9, another for 10-19, 20-29, and so on until all ages are grouped into bins. Feature engineering involves creating the most suitable features for a dataset and predictive model to maximize accuracy. A Feature Store centralizes all features across the enterprise, providing complete visibility and reuse across teams, avoiding code duplication, and simplifying the deployment and maintenance of predictive models in production. Databricks includes an integrated feature store in its platform, and we use it to store some of the features for our sample predictive model.
- **Notebooks:** Jupyter notebooks are essential tools for data scientists, enabling the development and testing of predictive models in multiple programming languages such as Python and R. The Databricks platform includes an integrated notebook development environment, and the sample predictive model in this paper originated from a Databricks Jupyter notebook.
- **ML Libraries:** The Databricks platform supports many ML libraries, such as TensorFlow, SparkMLib, H2O, Keras, Scikit-Learn, among others, some of which we use in our sample predictive model.
- **ML Flow:** Finding the most accurate model for a dataset involves many experiments with different feature sets, algorithms, hyper-parameter values, evaluations, and data split configurations. ML Flow helps data scientists manage their daily experimentation process by providing a central repository for all experiments, enabling collaboration, and making model preparation easier before deployment. In our example, the model training process saves the trained model in the ML Flow registry, from which it is loaded before deployment.
- **Model Serving:** One of the challenges in the machine learning workflow is deploying a model into production. A Data Scientist works with Jupyter Notebooks using Python to find the model with the best predictive accuracy. The Data Scientist then hands over the model to a Data Engineer, who often recreates it in another language, such as Java, and

integrates the Java code into an existing or new application. Databricks Model Serving, an advanced feature of ML Flow, enables the Data Scientist to deploy predictive models into production as REST Endpoints, allowing them to manage the entire machine learning workflow. Deploying Data Science functions as REST Endpoints makes it easier to build and use modular applications. We use this feature to make the predictive model available to the Integrationals Application UI.

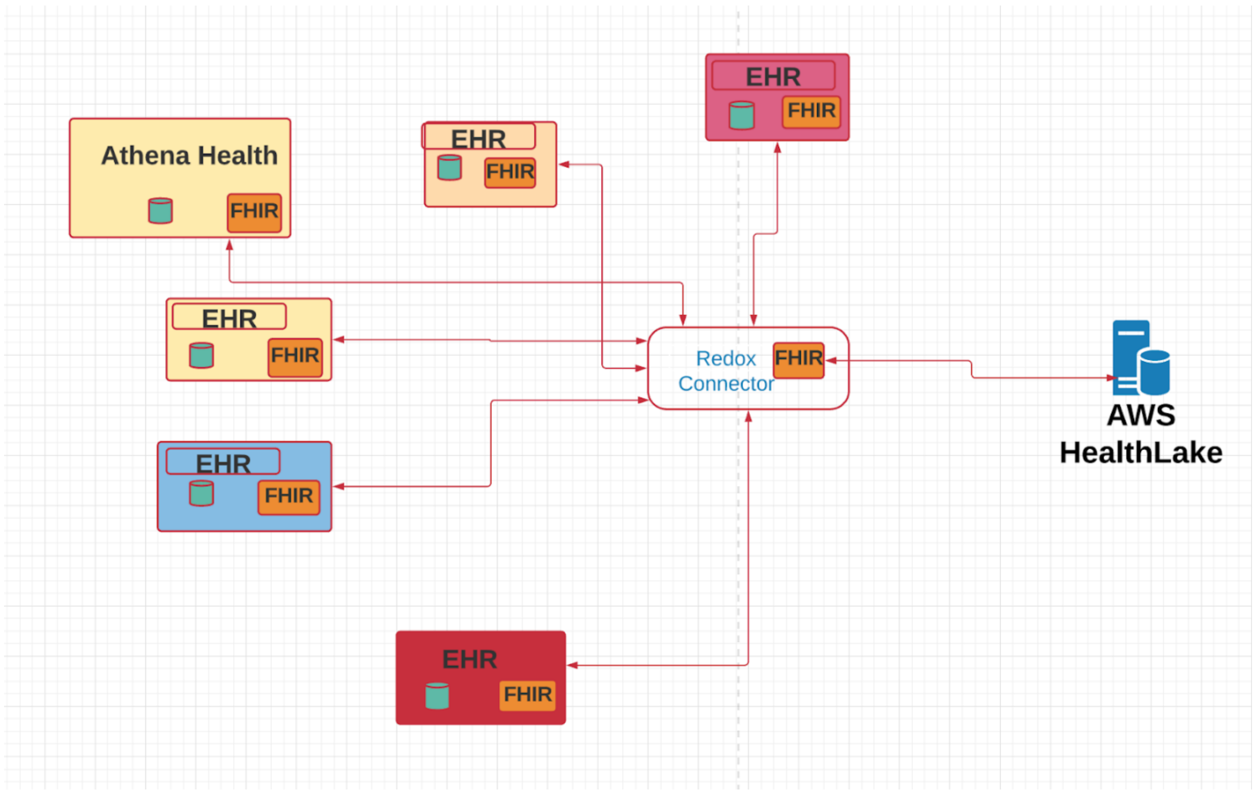
Integrationals Website Component

We define Medical Affiliates as external organizations that own medical data related to our patients, such as hospitals, medical specialists, previous primary care physicians, pharmacies, imaging and radiology centers, and medical laboratories. Each Medical Affiliate uses different software to store their patients' records.

Our system uses the FHIR standard to connect with other medical affiliates through APIs and retrieve HHC patients' medical information. Python is the programming language used to develop our system.

III. Integrations

As part of our solution, we help clients select a new scalable EHR system for their practice, and in this case, we recommend the EHR software from Athenahealth. To gather all patient records that might be scattered across different systems and institutions, our platform uses the Fast Healthcare Interoperability Resources (FHIR) standard to exchange data. Similarly, the Redox engine connects various EHR systems or data sources to our healthcare data lake (HealthLake). A part of the complete solution includes a set of EHR systems housing patient or healthcare data, connected to a central data warehouse through the Redox engine connector, as shown below.

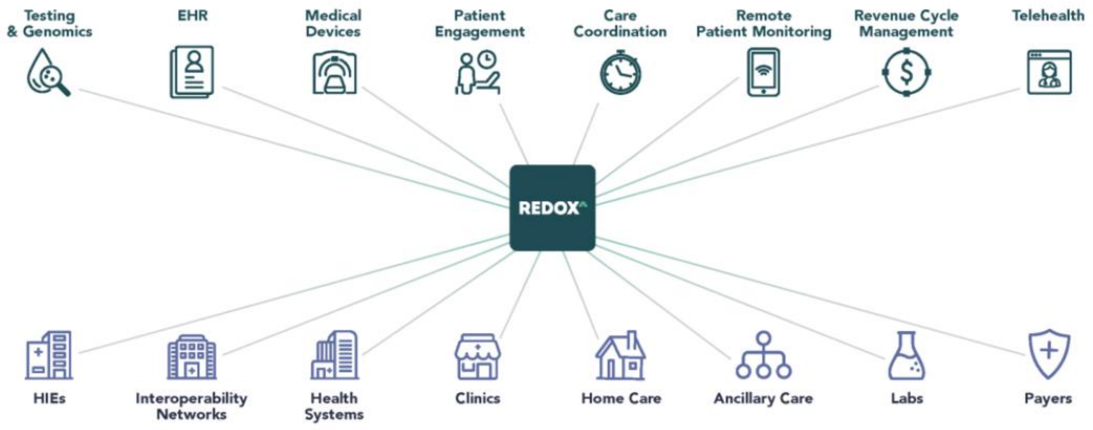


Athenahealth

Athenahealth is a cloud-based, scalable EHR system that offers web-based practice management software, including full patient access, claims processing, payer reports, and more. This EHR software is similar to most others and is internet-based, meaning it stores all medical records online, making remote access convenient. Athenahealth can connect to Redox using its built-in API that is compatible with FHIR standards.

Redox Engine

Redox is a platform that simplifies healthcare integration by providing developers with a modern, reliable JSON-based API that connects to any Healthcare System. It eliminates the complexity of outdated protocols and various EHR APIs to focus on innovation.



The Redox connector can import both structured and unstructured healthcare data from various source systems through its FHIR APIs. Redox automatically converts data to the FHIR standard before sending it downstream.

IV. Data Design and Management

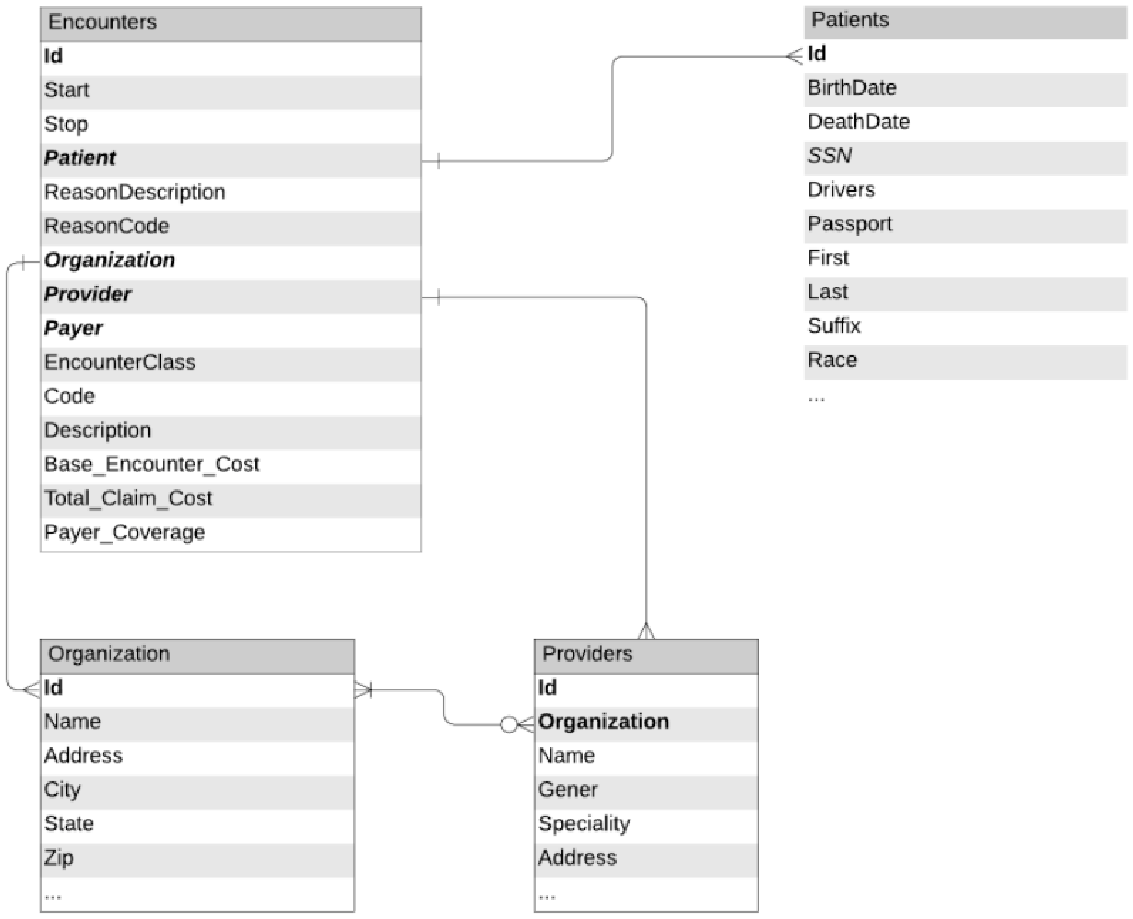
The U.S. Core Data for Interoperability (USCDI) is the standardized set of data classes and elements (or tables and columns) that the Office of the National Coordinator for Health Information Technology (ONC) has published. The purpose of USCDI is to guide the healthcare industry in implementing systems capable of exchanging data using the FHIR API. Here is a list of some primary data classes:

- Assessment and plan of treatment
- Care Team members
- Clinical Notes
- Goals
- Health concerns
- Immunizations
- Laboratory
- Medications
- Patient Demographics
- Problems
- Procedures
- Provenance
- Smoking Status
- Unique device identifier
- Vital signs
- Encounters (Office Visits)

The Integrationals Platform will ultimately support the complete USCDI data model. Below, we showcase some of the initial portions of the USCDI we implement in Phase 1.

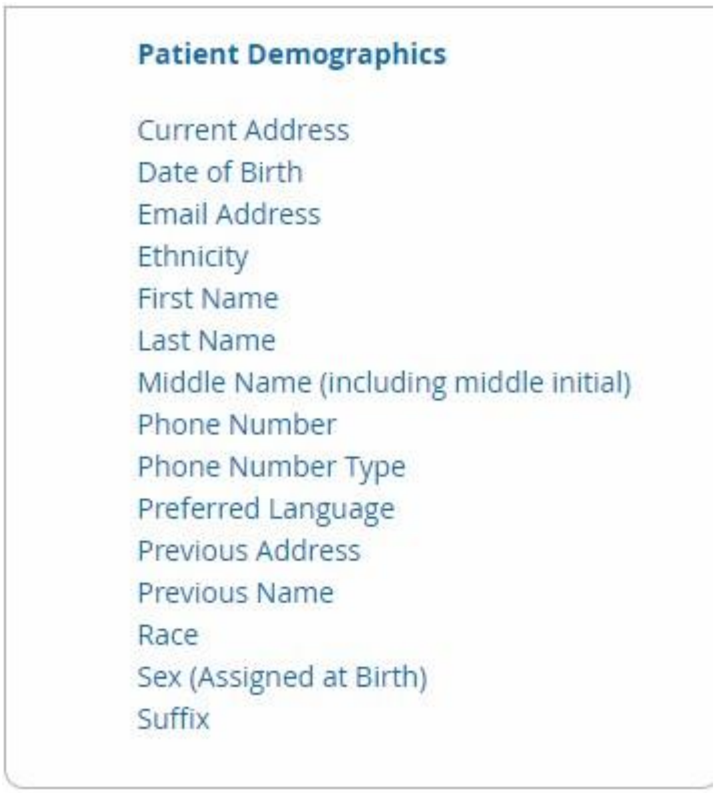
Predictive Model Data Classes and Elements

The data model below illustrates the entity relationships of the classes used by our statistical model as its prediction source.

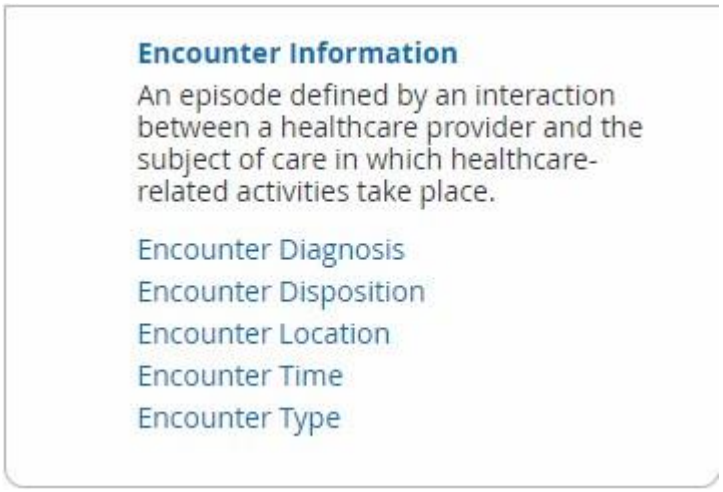


USCDI/FHIR matching data structure examples

Here, we present the attribute list of two classes directly from the USCDI.


Patient Demographics

Current Address
Date of Birth
Email Address
Ethnicity
First Name
Last Name
Middle Name (including middle initial)
Phone Number
Phone Number Type
Preferred Language
Previous Address
Previous Name
Race
Sex (Assigned at Birth)
Suffix


Encounter Information

An episode defined by an interaction between a healthcare provider and the subject of care in which healthcare-related activities take place.

Encounter Diagnosis
Encounter Disposition
Encounter Location
Encounter Time
Encounter Type

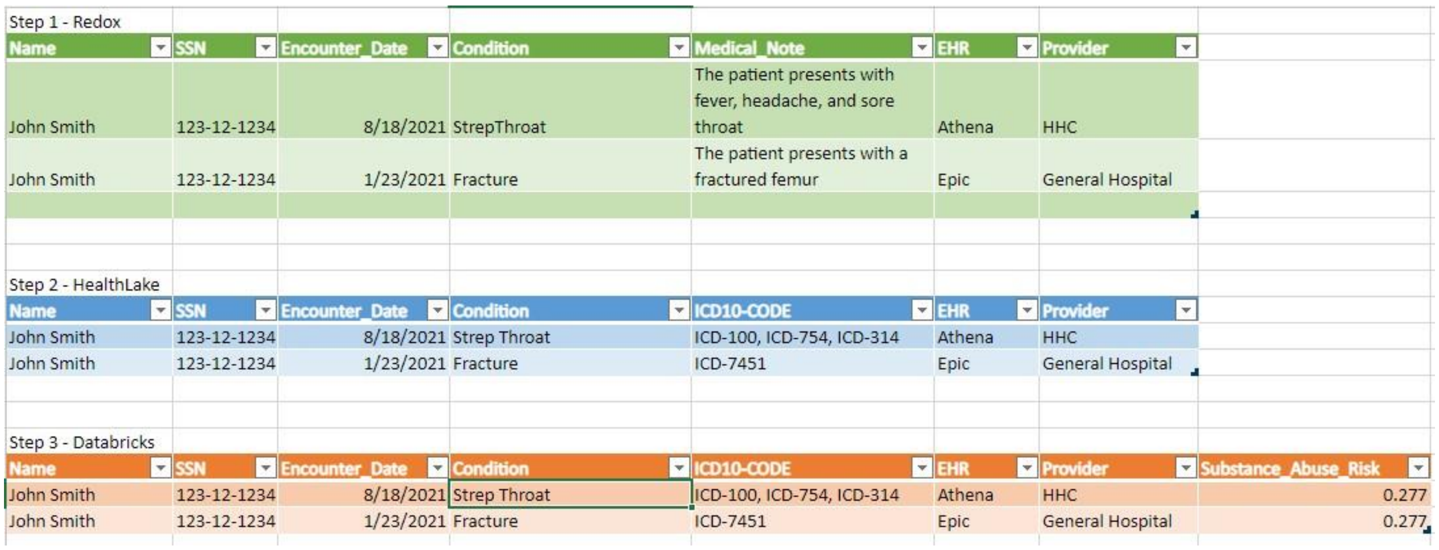
Data Flow Diagram

Here is a basic data flow across some of the platform components.

In step 1, we begin with data from two different EHRs from separate organizations. Redox retrieves the data from these source systems and feeds it into HealthLake.

In step 2, HealthLake processes the free-form text medical notes from the source systems and, with its NLP capabilities, transforms this unstructured data into structured information in the form of ICD-10 medical codes. HealthLake then exports this expanded patient record to cloud storage.

In Step 3, Databricks processes the data, consolidates it, and uses its predictive model to determine that John Smith has a 27% chance of developing a substance abuse condition.



Step 1 - Redox

Name	SSN	Encounter_Date	Condition	Medical_Note	EHR	Provider
John Smith	123-12-1234	8/18/2021	StrepThroat	The patient presents with fever, headache, and sore throat	Athena	HHC
John Smith	123-12-1234	1/23/2021	Fracture	The patient presents with a fractured femur	Epic	General Hospital

Step 2 - HealthLake

Name	SSN	Encounter_Date	Condition	ICD10-CODE	EHR	Provider
John Smith	123-12-1234	8/18/2021	Strep Throat	ICD-100, ICD-754, ICD-314	Athena	HHC
John Smith	123-12-1234	1/23/2021	Fracture	ICD-7451	Epic	General Hospital

Step 3 - Databricks

Name	SSN	Encounter_Date	Condition	ICD10-CODE	EHR	Provider	Substance_Abuse_Risk
John Smith	123-12-1234	8/18/2021	Strep Throat	ICD-100, ICD-754, ICD-314	Athena	HHC	0.277
John Smith	123-12-1234	1/23/2021	Fracture	ICD-7451	Epic	General Hospital	0.277

In Step 4, we will display these results to users through the Integrationals Application UI, where doctors might choose to implement substance abuse prevention care for John Smith.

Data Management Concepts

Data management involves many aspects that need careful thought and detailed explanation. Here are some key concepts that deserve particular focus.

- **Data Handling Ethics:** Data handling ethics focus on how to acquire, store, manage, interpret, analyze, and dispose of data in ways that adhere to ethical principles, including community responsibility. The company must develop an Ethical Data Handling Strategy that ensures consistency throughout the organization and compliance with regulation standards (GDPR, HIPPA) and regulatory bodies.
- **Data Governance:** Data Governance involves exercising authority, control, and shared decision-making over managing data assets. Some activities in this area include developing a Data Governance Strategy, conducting readiness assessments, and verifying business alignment. Part of the strategy is to build confidence and trust among our platform users by adopting and supporting an external audit process that ensures transparency of our controls and regulations.
- **Metadata Management:** involves the planning, implementation, and control activities that facilitate access to high-quality, integrated metadata. Some of these activities include understanding metadata requirements, defining a metadata architecture, creating and maintaining metadata, and generating metadata reports.
- **Data Security:** Data security involves defining, planning, developing, and executing security policies and procedures to ensure proper authentication, authorization, access, and auditing of data and information assets. Best practices are especially important for our platform due to the large volume of healthcare data. Two key technologies for scaling data security are Label Security and Data Vault. Label Security enables detailed data access control at a granular level, while Data Vault allows database administrators to

manage database instances with super-user privileges without exposing the data, which is a serious security risk. We will separate Personal Identifiable Information (PII) into a separate database and use Label Security and Data Vault to safeguard the anonymity of healthcare data throughout the platform. Outside the PII database, only surrogate keys will be used to represent PII data. Activities in this area include identifying data security requirements, creating security policies and standards, assessing current security risks, and implementing appropriate controls and procedures.

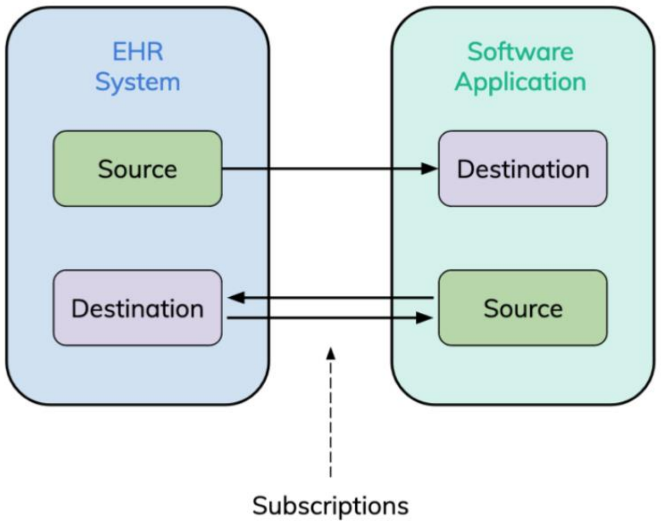
- **Data Quality Management:** involves planning, execution, and oversight activities that apply quality management techniques to data, ensuring it is suitable for use and meets the needs of data consumers. These activities include defining high-quality data, establishing a data quality strategy, performing data quality assessments, and identifying and prioritizing areas for improvement.

V. Solution Demonstration

Redox

Sources, Destinations, & Subscriptions

In Redox Engine terminology, Sources, Destinations, and Subscriptions collectively represent the data transfer flow across the Redox Platform.

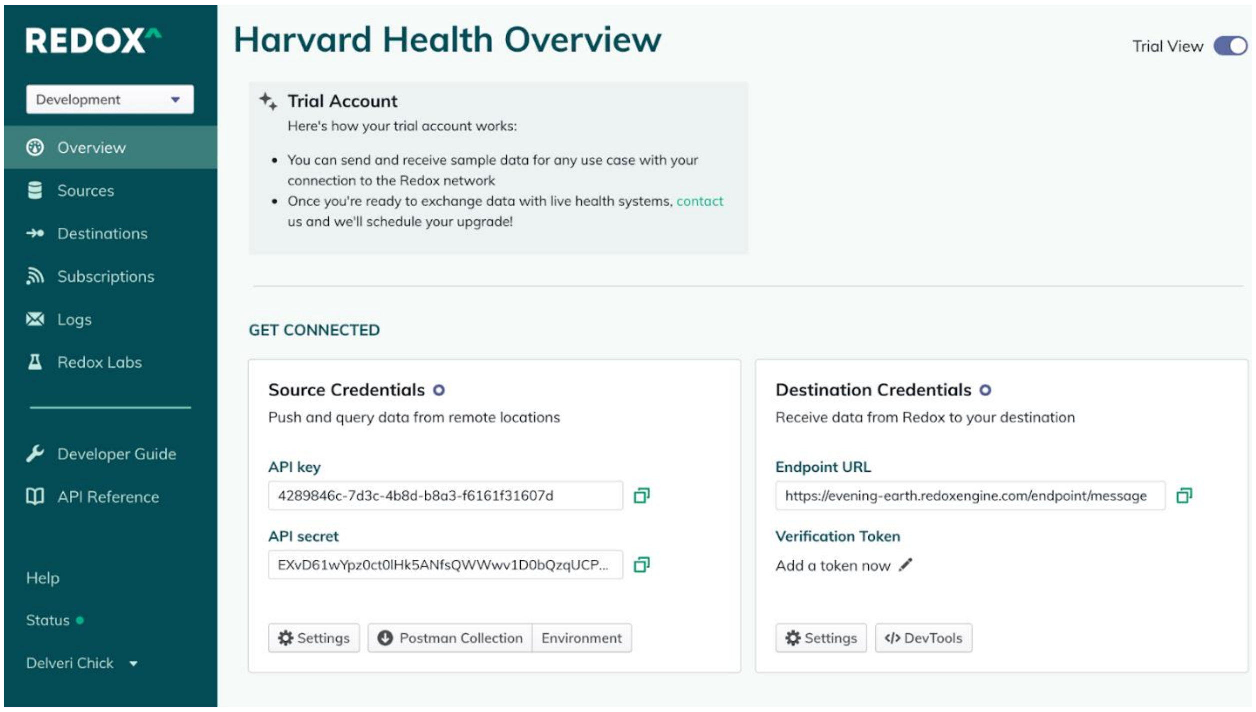


Sources and destinations are specific data system configurations that specify how an application or EHR system connects to Redox. For example, Athenahealth will be the source for our client, and Amazon HealthLake will serve as our destination.

Subscriptions are routing rules that enable data transfer. Specifically, each subscription specifies a Data Model for transferring data between a source (Athenahealth) and a destination (HealthLake). The system supports multiple subscription configurations for any given source or destination.

Redox engine is a fully online service that requires signing up to create an admin account. The account provides access to a Redox dashboard where an administrator can easily configure Sources, Destinations, and Subscriptions.

Redox Dashboard Overview



Harvard Health Overview

Trial Account

Here's how your trial account works:

- You can send and receive sample data for any use case with your connection to the Redox network
- Once you're ready to exchange data with live health systems, contact us and we'll schedule your upgrade!

GET CONNECTED

Source Credentials

Push and query data from remote locations

API key: 4289846c-7d3c-4b8d-b8a3-f6161f31607d

API secret: EXvD61wYpz0ct0lHk5ANfsQWWwv1D0bQzqUCP...

Settings | Postman Collection | Environment

Destination Credentials

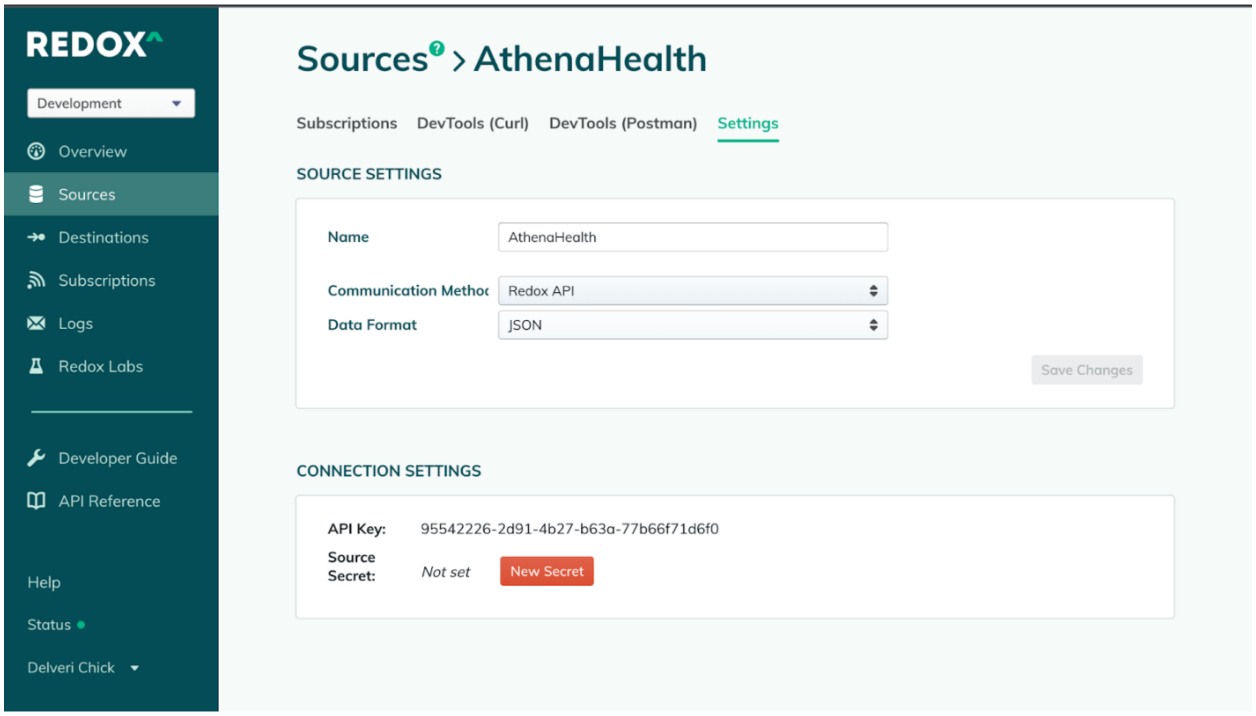
Receive data from Redox to your destination

Endpoint URL: https://evening-earth.redoxengine.com/endpoint/message

Verification Token: Add a token now

Settings | DevTools

Creating a Source



Sources > AthenaHealth

Subscriptions DevTools (Curl) DevTools (Postman) Settings

SOURCE SETTINGS

Name	AthenaHealth
Communication Method	Redox API
Data Format	JSON

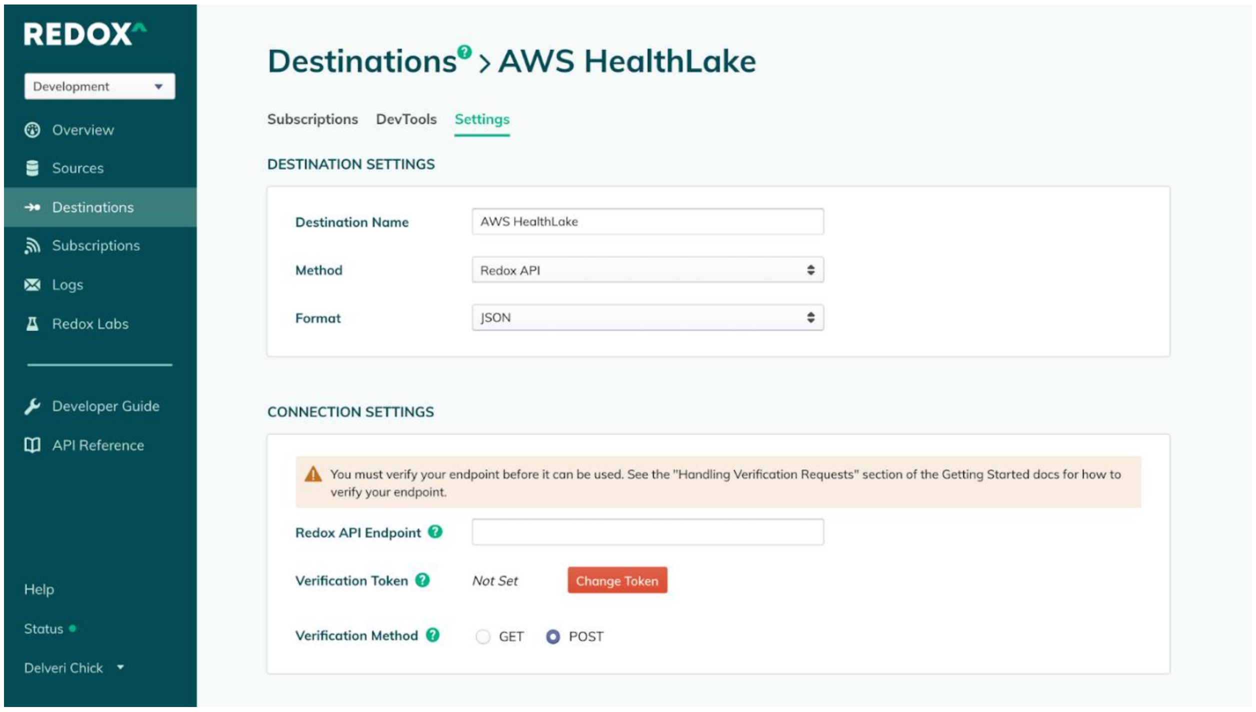
Save Changes

CONNECTION SETTINGS

API Key: 95542226-2d91-4b27-b63a-77b66f71d6f0

Source Secret: *Not set* New Secret

Configuring Destinations



Destinations > AWS HealthLake

Subscriptions DevTools Settings

DESTINATION SETTINGS

Destination Name	AWS HealthLake
Method	Redox API
Format	JSON

CONNECTION SETTINGS

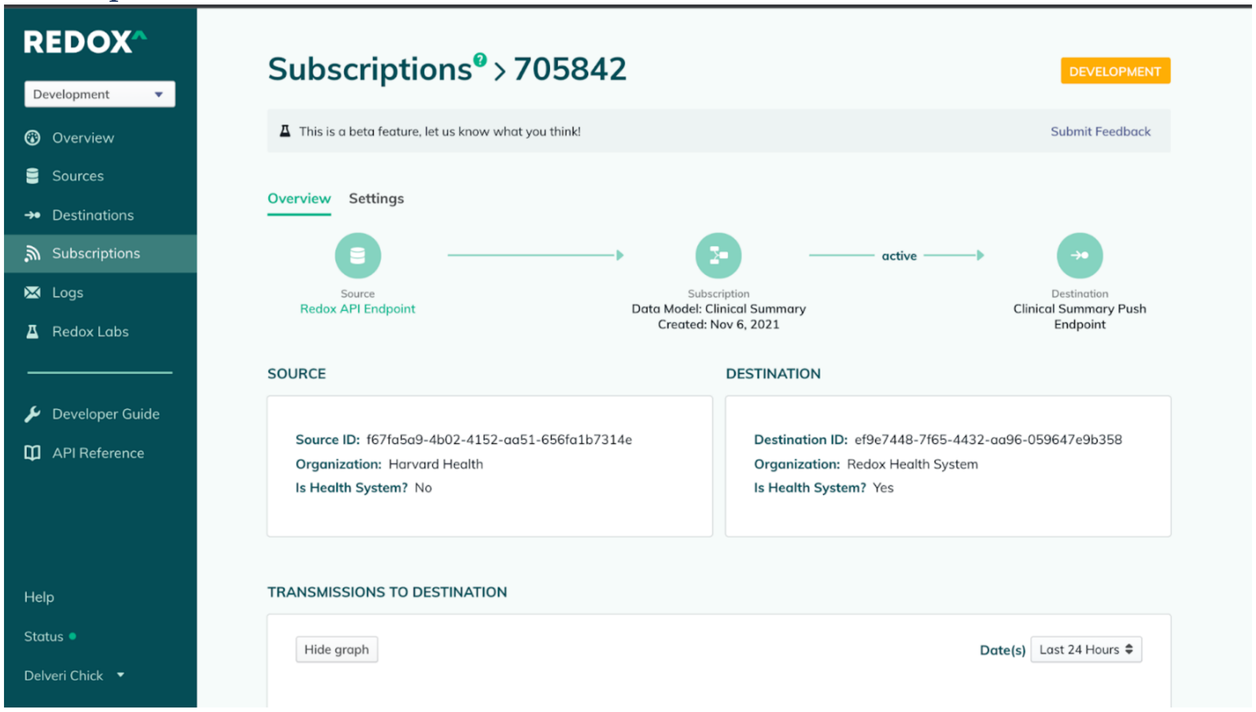
You must verify your endpoint before it can be used. See the "Handling Verification Requests" section of the Getting Started docs for how to verify your endpoint.

Redox API Endpoint

Verification Token *Not Set* Change Token

Verification Method GET [✓ POST]

Subscription



Subscriptions > 705842

DEVELOPMENT

This is a beta feature, let us know what you think! Submit Feedback

Overview Settings

Source
Redox API Endpoint

Subscription
Data Model: Clinical Summary
Created: Nov 6, 2021

active

Destination
Clinical Summary Push Endpoint

SOURCE

Source ID: f67fa5a9-4b02-4152-aa51-656fa1b7314e
Organization: Harvard Health
Is Health System? No

DESTINATION

Destination ID: ef9e7448-7f65-4432-aa96-059647e9b358
Organization: Redox Health System
Is Health System? Yes

TRANSMISSIONS TO DESTINATION

Hide graph

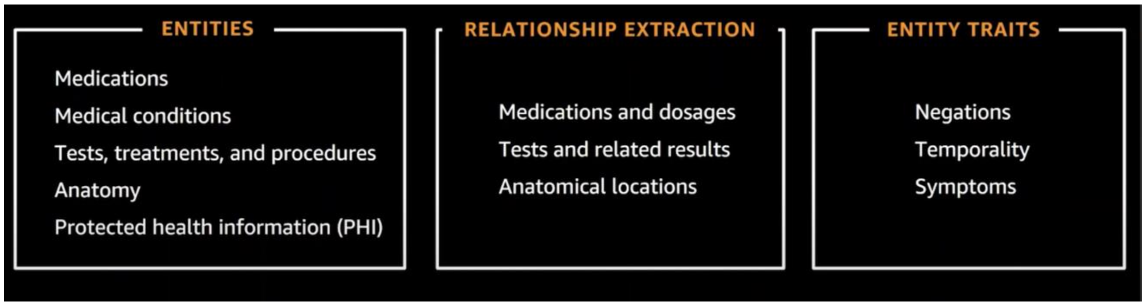
Date(s) Last 24 Hours

An Administrator must configure subscriptions in the Redox dashboard for each source data model to automate data transfer from Athenahealth or other EHR systems into HealthLake.

HealthLake

As mentioned earlier, Amazon HealthLake is a HIPAA-eligible service that helps healthcare practitioners, insurance companies, and pharmacies to convert, store, search, and analyze medical records in the cloud. Currently, healthcare providers can access a medical model that uses 30 data points or fewer; this is because of the significant manual effort needed to review and extract such information from individual medical records.

On the other hand, AWS HealthLake automatically extracts between 200 and 300,000 key clinical data points from clinical notes, medical imaging reports, lab reports, diagnoses, medications, a,nd procedures. Using Amazon Comprehend Medical, AWS's natural language processing service, information like that shown in the table below is extracted from unstructured text and transformed into structured data points. AWS HealthLake then tags each record to maintain an updated index of all records.



ENTITIES

Medications
Medical conditions
Tests, treatments, and procedures
Anatomy
Protected health information (PHI)

RELATIONSHIP EXTRACTION

Medications and dosages
Tests and related results
Anatomical locations

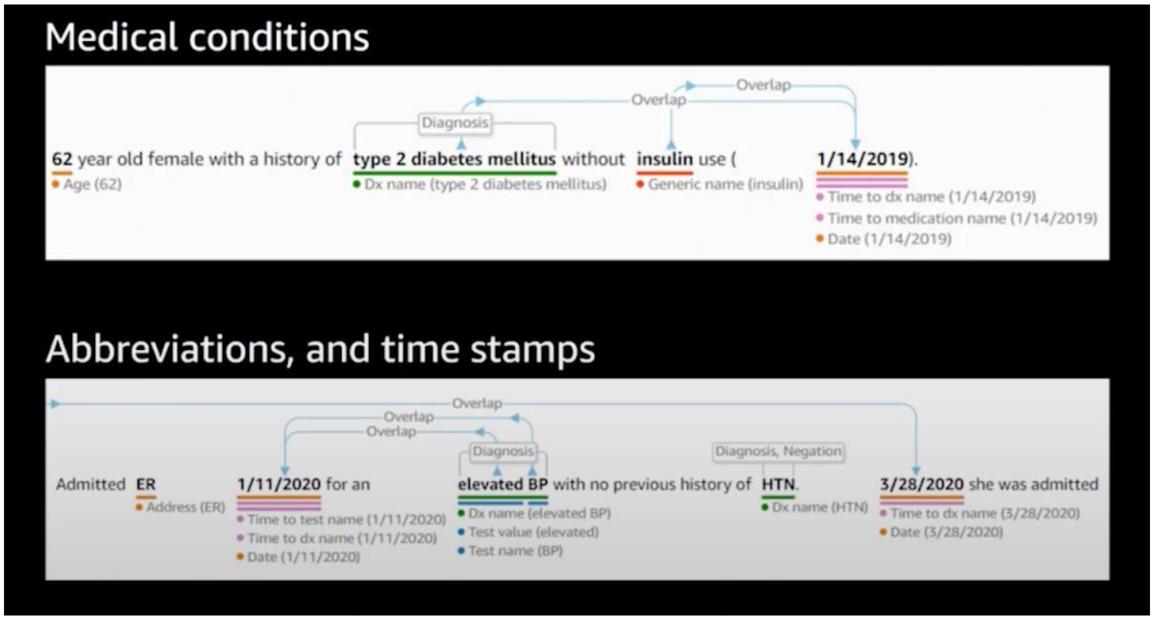
ENTITY TRAITS

Negations
Temporality
Symptoms

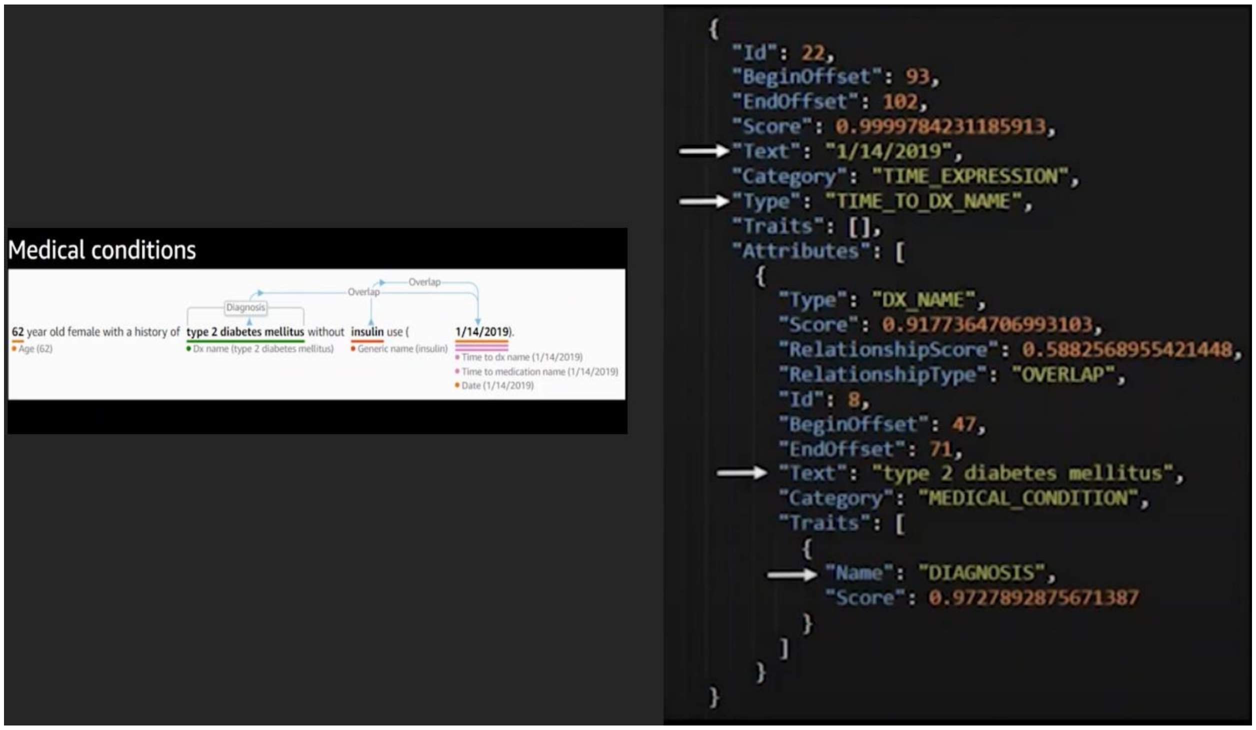
To illustrate the power a platform such as AWS HealthLake provides, we will review a four-sentence sample from a clinical medical note.

"The patient is a 62-year-old female with a history of Type 2 diabetes mellitus without insulin use (1/14/2019). Admitted ER 1/11/2020 for an elevated BP with no previous history of HTN. 3/28//2020 she was admitted for hypothyroidism and prescribed metformin (GLUCOPHAGE) 1000 mg take daily by mouth in evening. Follow up clinic visit (9/20/2020) with A1C results of …"

These four sentences contain important detailed medical information. As shown in the image below, Amazon HealthLake ML services automatically detect medical conditions like type 2 diabetes and hypertension. Additionally, it recognizes abbreviations such as BP, which stands for blood pressure, and blood pressure reading timestamps. It also detects negation cues, such as no previous signs of hypertension.


Medical conditions
Overlap
Overlap
Diagnosis
62 year old female with a history of type 2 diabetes mellitus without insulin use (1/14/2019).
Age (62)
Dx name (type 2 diabetes mellitus)
Generic name (insulin)
Time to dx name (1/14/2019)
Time to medication name (1/14/2019)
Date (1/14/2019)
Abbreviations, and time stamps
Overlap
Overlap
Overlap
Diagnosis
Diagnosis, Negation
Admitted ER 1/11/2020 for an elevated BP with no previous history of HTN. 3/28/2020 she was admitted
Address (ER)
Time to test name (1/11/2020)
Time to dx name (1/11/2020)
Date (1/11/2020)
Dx name (elevated BP)
Test value (elevated)
Test name (BP)
Dx name (HTN)
Time to dx name (3/28/2020)
Date (3/28/2020)

All the data points are extracted, structured, and added to a JSON object, which provides a detailed view of the information. On the far right of the image below, we can see an example of the JSON text generated from the medical notes.


Medical conditions

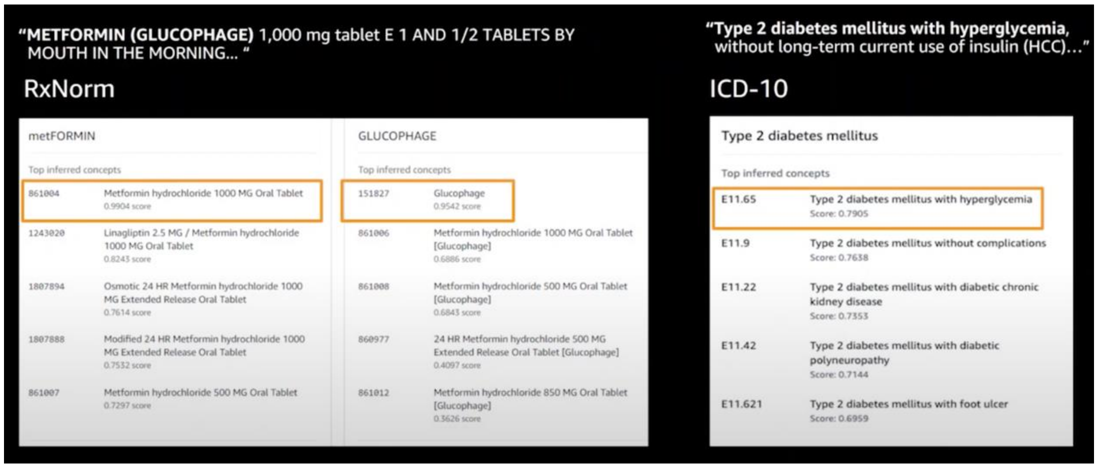
62 year old female with a history of **type 2 diabetes mellitus** without **insulin** use (**1/14/2019**).

Overlap
Overlap
Diagnosis
Age (62)
Dx name (type 2 diabetes mellitus)
Generic name (insulin)
Time to dx name (1/14/2019)
Time to medication name (1/14/2019)
Date (1/14/2019)

```
{
  "Id": 22,
  "BeginOffset": 93,
  "EndOffset": 102,
  "Score": 0.9999784231185913,
  "Text": "1/14/2019",
  "Category": "TIME_EXPRESSION",
  "Type": "TIME_TO_DX_NAME",
  "Traits": [],
  "Attributes": [
    {
      "Type": "DX_NAME",
      "Score": 0.9177364706993103,
      "RelationshipScore": 0.5882568955421448,
      "RelationshipType": "OVERLAP",
      "Id": 8,
      "BeginOffset": 47,
      "EndOffset": 71,
      "Text": "type 2 diabetes mellitus",
      "Category": "MEDICAL_CONDITION",
      "Traits": [
        {
          "Name": "DIAGNOSIS",
          "Score": 0.9727892875671387
        }
      ]
    }
}
```

We analyze the notes, data, extracted process, and the object extracted. For each medical entity, there will be a unique tag indicating its position within the text, marked by “BeginOffset” and “EndOffset.” A confidence score will also be provided to show the likelihood that the medical entity matches the text. As shown by the arrows, they first point to the date, 1/14/2019, which the system identified from the text. This method includes a concept of relationship type, linking the date to the diagnosis, which it categorizes as type 2 diabetes.

In addition to this, AWS HealthLake adds associated medical codes to the extracted text to identify concepts such as medications, brand names, or ICD-10 codes used for billing.



“**METFORMIN (GLUCOPHAGE)** 1,000 mg tablet E 1 AND 1/2 TABLETS BY MOUTH IN THE MORNING… “

RxNorm

metFORMIN

Top inferred concepts

861004	Metformin hydrochloride 1000 MG Oral Tablet 0.9904 score
1243020	Linagliptin 2.5 MG / Metformin hydrochloride 1000 MG Oral Tablet 0.8243 score
1807894	Osmotic 24 HR Metformin hydrochloride 1000 MG Extended Release Oral Tablet 0.7614 score
1807888	Modified 24 HR Metformin hydrochloride 1000 MG Extended Release Oral Tablet 0.7532 score
861007	Metformin hydrochloride 500 MG Oral Tablet 0.7297 score

GLUCOPHAGE

Top inferred concepts

151827	Glucophage 0.9542 score
861006	Metformin hydrochloride 1000 MG Oral Tablet [Glucophage] 0.6886 score
861008	Metformin hydrochloride 500 MG Oral Tablet [Glucophage] 0.6843 score
860977	24 HR Metformin hydrochloride 500 MG Extended Release Oral Tablet [Glucophage] 0.4097 score
861012	Metformin hydrochloride 850 MG Oral Tablet [Glucophage] 0.3626 score

“**Type 2 diabetes mellitus with hyperglycemia,** without long-term current use of insulin (HCC)…”

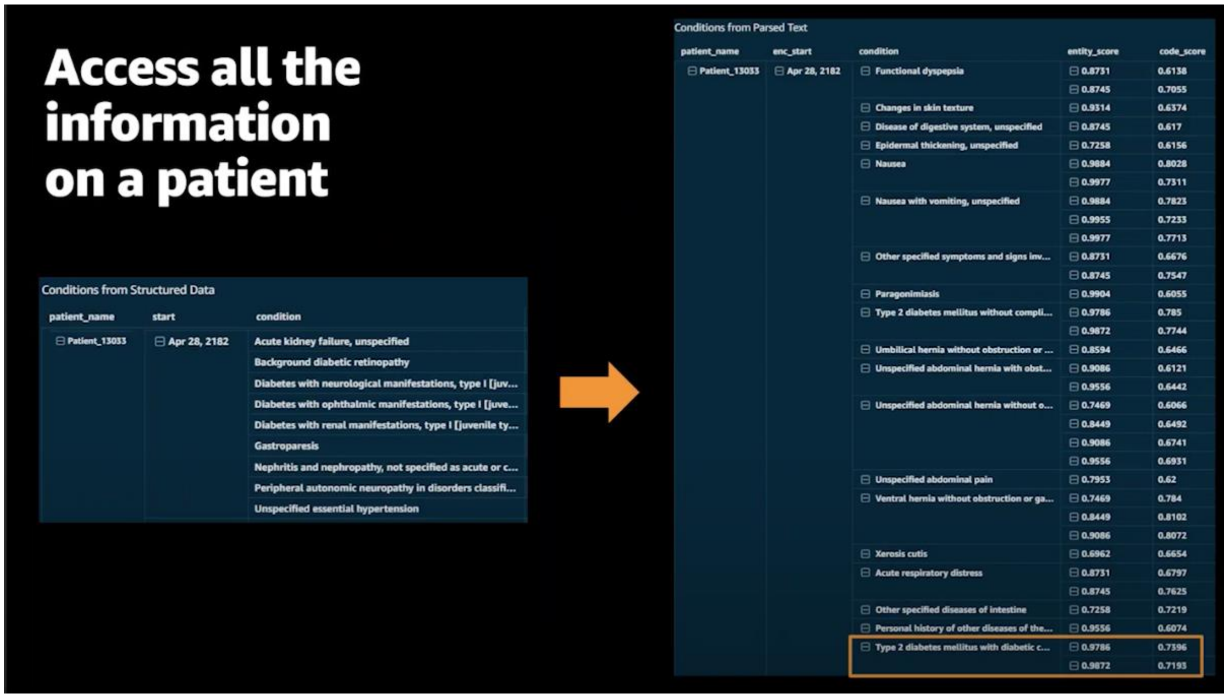
ICD-10

Type 2 diabetes mellitus

Top inferred concepts

E11.65	Type 2 diabetes mellitus with hyperglycemia Score: 0.7905
E11.9	Type 2 diabetes mellitus without complications Score: 0.7638
E11.22	Type 2 diabetes mellitus with diabetic chronic kidney disease Score: 0.7353
E11.42	Type 2 diabetes mellitus with diabetic polyneuropathy Score: 0.7144
E11.621	Type 2 diabetes mellitus with foot ulcer Score: 0.6959

Adding these associations is essential because it enables access to normalized data across all subtypes of clinical data during analysis and ML modeling. The system then appends all the unstructured data it extracted from text to the structured data, which gives access to all the information of an individual patient or group of patients.



Access all the information on a patient

Conditions from Structured Data

patient_name	start	condition
⊟ Patient_13033	⊟ Apr 28, 2182	Acute kidney failure, unspecified
		Background diabetic retinopathy
		Diabetes with neurological manifestations, type I [juv...
		Diabetes with ophthalmic manifestations, type I [juve...
		Diabetes with renal manifestations, type I [juvenile ty...
		Gastroparesis
		Nephritis and nephropathy, not specified as acute or c...
		Peripheral autonomic neuropathy in disorders classifi...
		Unspecified essential hypertension

Conditions from Parsed Text

patient_name	enc_start	condition	entity_score	code_score
⊟ Patient_13033	⊟ Apr 28, 2182	⊟ Functional dyspepsia	⊟ 0.8731	0.6138
			⊟ 0.8745	0.7055
		⊟ Changes in skin texture	⊟ 0.9314	0.6374
		⊟ Disease of digestive system, unspecified	⊟ 0.8745	0.617
		⊟ Epidermal thickening, unspecified	⊟ 0.7258	0.6156
		⊟ Nausea	⊟ 0.9884	0.8028
			⊟ 0.9977	0.7311
		⊟ Nausea with vomiting, unspecified	⊟ 0.9884	0.7823
			⊟ 0.9955	0.7233
			⊟ 0.9977	0.7713
		⊟ Other specified symptoms and signs inv...	⊟ 0.8731	0.6676
			⊟ 0.8745	0.7547
		⊟ Paragonimiasis	⊟ 0.9904	0.6055
		⊟ Type 2 diabetes mellitus without compli...	⊟ 0.9786	0.785
			⊟ 0.9872	0.7744
		⊟ Umbilical hernia without obstruction or ...	⊟ 0.8594	0.6466
		⊟ Unspecified abdominal hernia with obst...	⊟ 0.9086	0.6121
			⊟ 0.9556	0.6442
		⊟ Unspecified abdominal hernia without o...	⊟ 0.7469	0.6066
			⊟ 0.8449	0.6492
			⊟ 0.9086	0.6741
			⊟ 0.9556	0.6931
		⊟ Unspecified abdominal pain	⊟ 0.7953	0.62
		⊟ Ventral hernia without obstruction or ga...	⊟ 0.7469	0.784
			⊟ 0.8449	0.8102
			⊟ 0.9086	0.8072
		⊟ Xerosis cutis	⊟ 0.6962	0.6654
		⊟ Acute respiratory distress	⊟ 0.8731	0.6797
			⊟ 0.8745	0.7625
		⊟ Other specified diseases of intestine	⊟ 0.7258	0.7219
		⊟ Personal history of other diseases of the...	⊟ 0.9556	0.6074
		⊟ Type 2 diabetes mellitus with diabetic c...	⊟ 0.9786	0.7396
			⊟ 0.9872	0.7193

The combined data includes hundreds of medical data points to create comprehensive views of normalized data and export it to third-party applications for data analysis and running machine learning models.

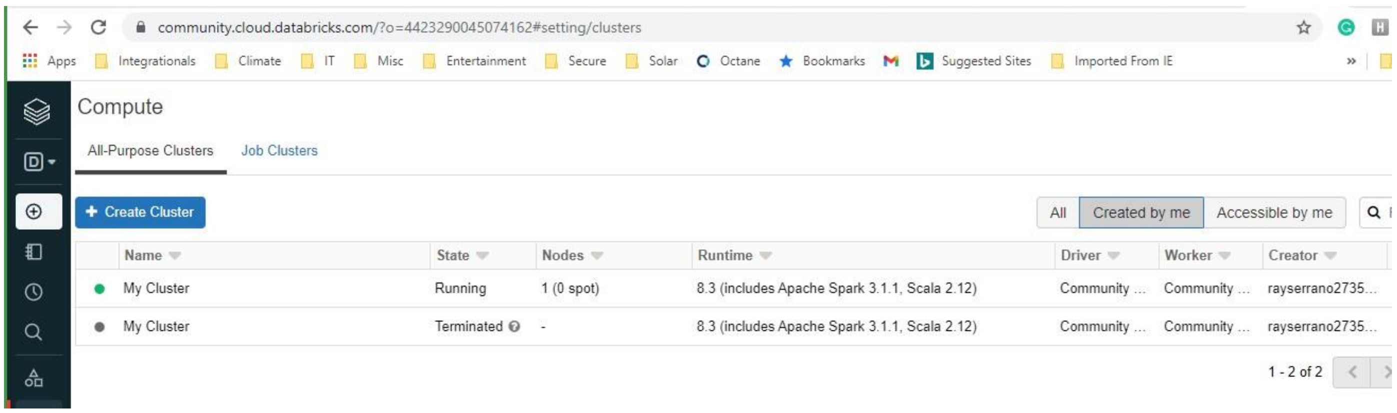
Additional Steps

Databricks

For the Databricks demonstration, we will show a simple series of steps that take us from loading healthcare data from cloud storage to predicting health conditions for new patient records. The full process involves many necessary steps we have left out here for brevity.

Cluster

A cluster is a group of machines that work together to provide an execution environment for our ML prediction workflow. Below, we see a basic setup called “My Cluster” in a “Running” state.

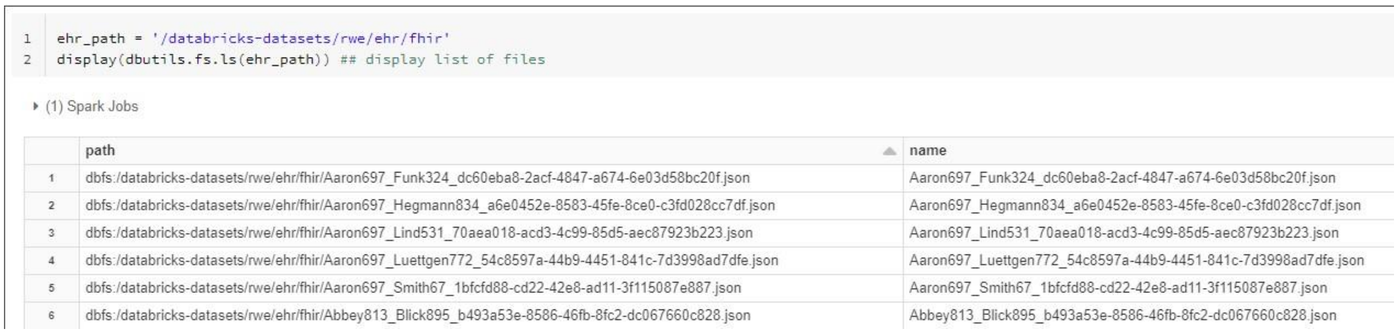


	Name	State	Nodes	Runtime	Driver	Worker	Creator
●	My Cluster	Running	1 (0 spot)	8.3 (includes Apache Spark 3.1.1, Scala 2.12)	Community ...	Community ...	rayserrano2735...
●	My Cluster	Terminated	-	8.3 (includes Apache Spark 3.1.1, Scala 2.12)	Community ...	Community ...	rayserrano2735...

1 - 2 of 2

List of FHIR files

Here we are presenting some of the data using the FHIR standard in JSON format.



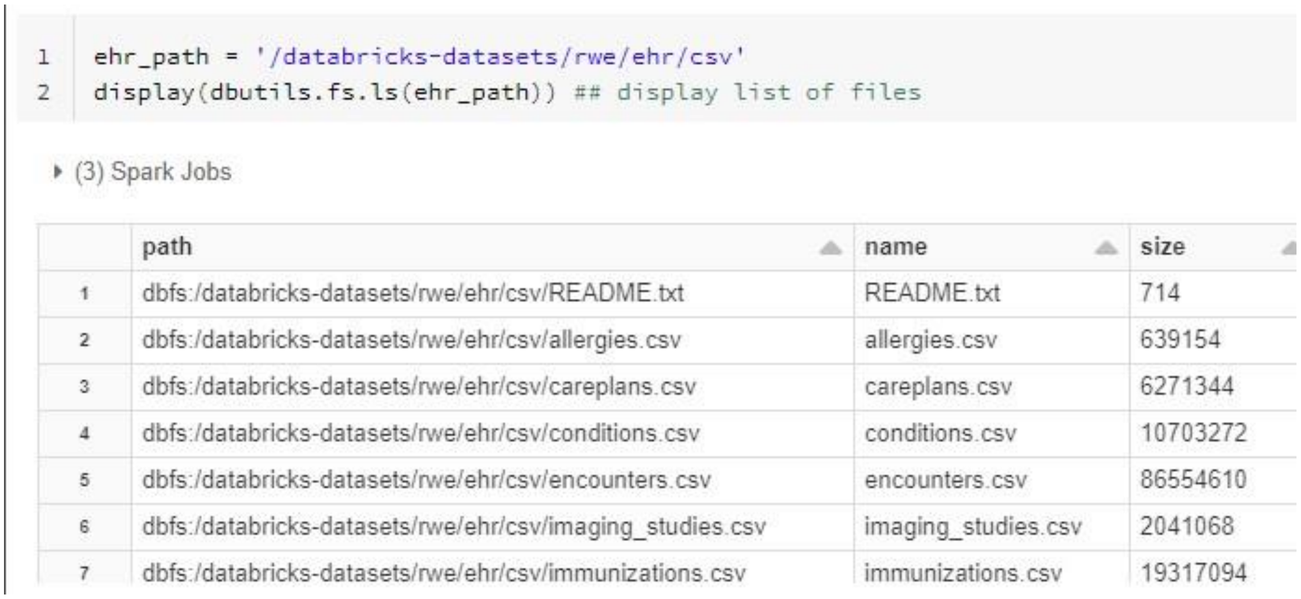
```
ehr_path = '/databricks-datasets/rwe/ehr/fhir'
display(dbutils.fs.ls(ehr_path)) ## display list of files
```

▸ (1) Spark Jobs

	path	name
1	dbfs:/databricks-datasets/rwe/ehr/fhir/Aaron697_Funk324_dc60eba8-2acf-4847-a674-6e03d58bc20f.json	Aaron697_Funk324_dc60eba8-2acf-4847-a674-6e03d58bc20f.json
2	dbfs:/databricks-datasets/rwe/ehr/fhir/Aaron697_Hegmann834_a6e0452e-8583-45fe-8ce0-c3fd028cc7df.json	Aaron697_Hegmann834_a6e0452e-8583-45fe-8ce0-c3fd028cc7df.json
3	dbfs:/databricks-datasets/rwe/ehr/fhir/Aaron697_Lind531_70aea018-acd3-4c99-85d5-aec87923b223.json	Aaron697_Lind531_70aea018-acd3-4c99-85d5-aec87923b223.json
4	dbfs:/databricks-datasets/rwe/ehr/fhir/Aaron697_Luettgen772_54c8597a-44b9-4451-841c-7d3998ad7dfe.json	Aaron697_Luettgen772_54c8597a-44b9-4451-841c-7d3998ad7dfe.json
5	dbfs:/databricks-datasets/rwe/ehr/fhir/Aaron697_Smith67_1bfcfd88-cd22-42e8-ad11-3f115087e887.json	Aaron697_Smith67_1bfcfd88-cd22-42e8-ad11-3f115087e887.json
6	dbfs:/databricks-datasets/rwe/ehr/fhir/Abbey813_Blick895_b493a53e-8586-46fb-8fc2-dc067660c828.json	Abbey813_Blick895_b493a53e-8586-46fb-8fc2-dc067660c828.json

List of CSV files

Here, we see the same data in CSV format. The FHIR standard is more detailed than the CSV files, providing a JSON FHIR file for each patient instead of the typical consolidated CSV approach.



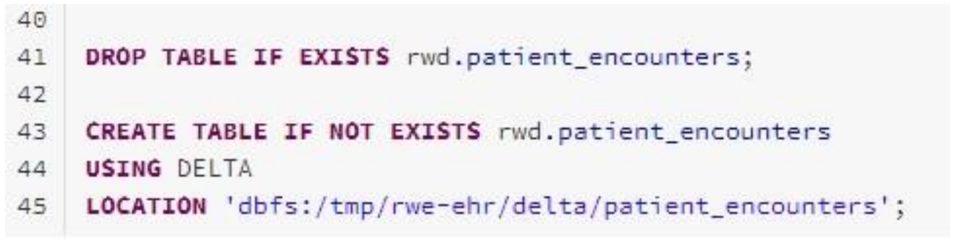
```
ehr_path = '/databricks-datasets/rwe/ehr/csv'
display(dbutils.fs.ls(ehr_path)) ## display list of files
```

▸ (3) Spark Jobs

	path	name	size
1	dbfs:/databricks-datasets/rwe/ehr/csv/README.txt	README.txt	714
2	dbfs:/databricks-datasets/rwe/ehr/csv/allergies.csv	allergies.csv	639154
3	dbfs:/databricks-datasets/rwe/ehr/csv/careplans.csv	careplans.csv	6271344
4	dbfs:/databricks-datasets/rwe/ehr/csv/conditions.csv	conditions.csv	10703272
5	dbfs:/databricks-datasets/rwe/ehr/csv/encounters.csv	encounters.csv	86554610
6	dbfs:/databricks-datasets/rwe/ehr/csv/imaging_studies.csv	imaging_studies.csv	2041068
7	dbfs:/databricks-datasets/rwe/ehr/csv/immunizations.csv	immunizations.csv	19317094

Patient Encounter Table Creation

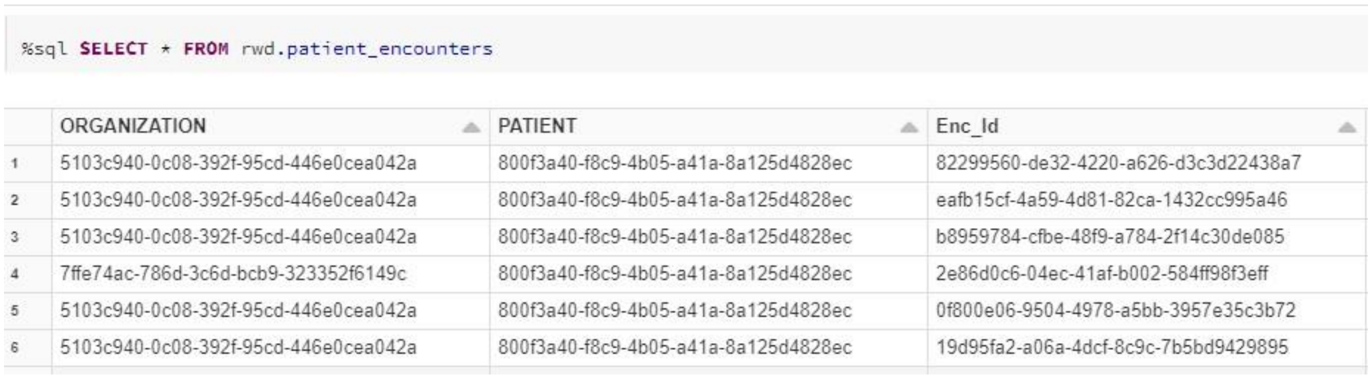
Here is an example of creating a table from one of the source files. Some pre-processing steps we applied to the source file have been skipped for simplicity.



```

DROP TABLE IF EXISTS rwd.patient_encounters;

CREATE TABLE IF NOT EXISTS rwd.patient_encounters
USING DELTA
LOCATION 'dbfs:/tmp/rwe-ehr/delta/patient_encounters';
```

Patient Encounter List in SQL

Here, we see an example of querying the source data using standard SQL. Please note that we have applied data masking to PII data.



```
%sql SELECT * FROM rwd.patient_encounters
```

	ORGANIZATION	PATIENT	Enc_Id
1	5103c940-0c08-392f-95cd-446e0cea042a	800f3a40-f8c9-4b05-a41a-8a125d4828ec	82299560-de32-4220-a626-d3c3d22438a7
2	5103c940-0c08-392f-95cd-446e0cea042a	800f3a40-f8c9-4b05-a41a-8a125d4828ec	eafb15cf-4a59-4d81-82ca-1432cc995a46
3	5103c940-0c08-392f-95cd-446e0cea042a	800f3a40-f8c9-4b05-a41a-8a125d4828ec	b8959784-cfbe-48f9-a784-2f14c30de085
4	7ffe74ac-786d-3c6d-bcb9-323352f6149c	800f3a40-f8c9-4b05-a41a-8a125d4828ec	2e86d0c6-04ec-41af-b002-584ff98f3eff
5	5103c940-0c08-392f-95cd-446e0cea042a	800f3a40-f8c9-4b05-a41a-8a125d4828ec	0f800e06-9504-4978-a5bb-3957e35c3b72
6	5103c940-0c08-392f-95cd-446e0cea042a	800f3a40-f8c9-4b05-a41a-8a125d4828ec	19d95fa2-a06a-4dcf-8c9c-7b5bd9429895

Condition

Our sample predictive model can calculate risk for several chronic health conditions. For this demonstration, our model defaults to Drug Overdose.

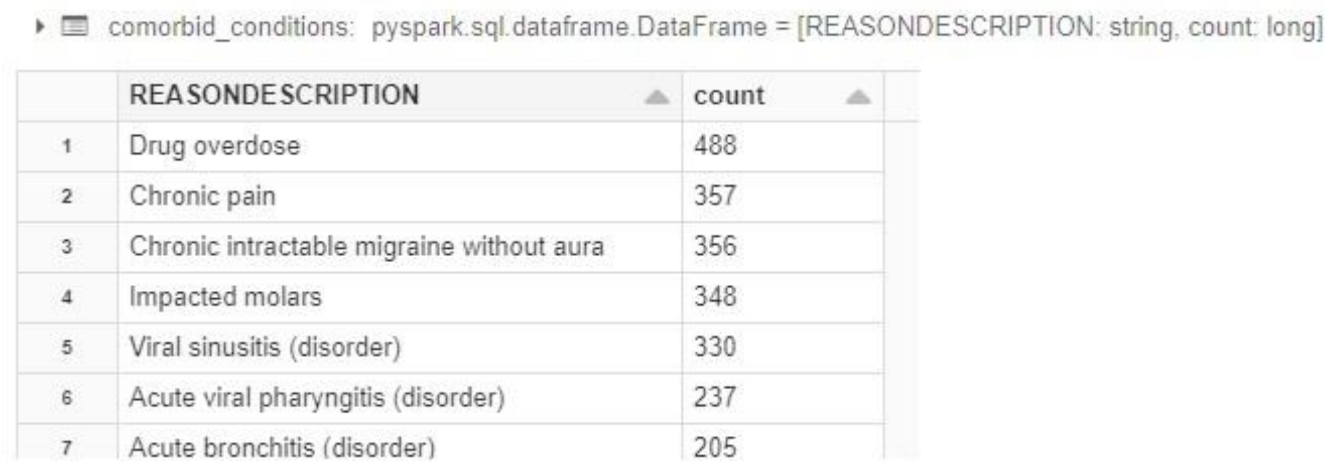
```
dbutils.widgets.text('condition', 'drug overdose', 'Condition to model')
dbutils.widgets.text('num_conditions', '10', '# of comorbidities to include')
dbutils.widgets.text('num_days', '90', '# of days to use')
dbutils.widgets.text('num_days_future', '10', '# of days in future for forecasting')
```

```
condition=dbutils.widgets.get('condition')
```

Comorbidities

Comorbidity, for example, refers to a specific mental health disorder and a substance use disorder that often coexist. The predictive model we are using for this example relies partly on comorbidities, and we can see a sample of conditions we will use to predict substance abuse. Some conditions, such as Drug Overdose, are prominent predictors, but others are less obvious.

```
display(comorbid_conditions)
```



```
comorbid_conditions: pyspark.sql.dataframe.DataFrame = [REASONDESCRIPTION: string, count: long]
```

	REASONDESCRIPTION	count
1	Drug overdose	488
2	Chronic pain	357
3	Chronic intractable migraine without aura	356
4	Impacted molars	348
5	Viral sinusitis (disorder)	330
6	Acute viral pharyngitis (disorder)	237
7	Acute bronchitis (disorder)	205

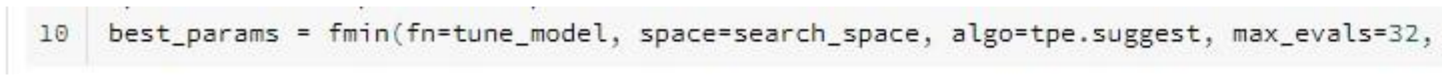
Selected Data Elements

Below, we see the predictors our model focuses on for this example. We categorize them into three groups: encounter, patient, and comorbidity columns.

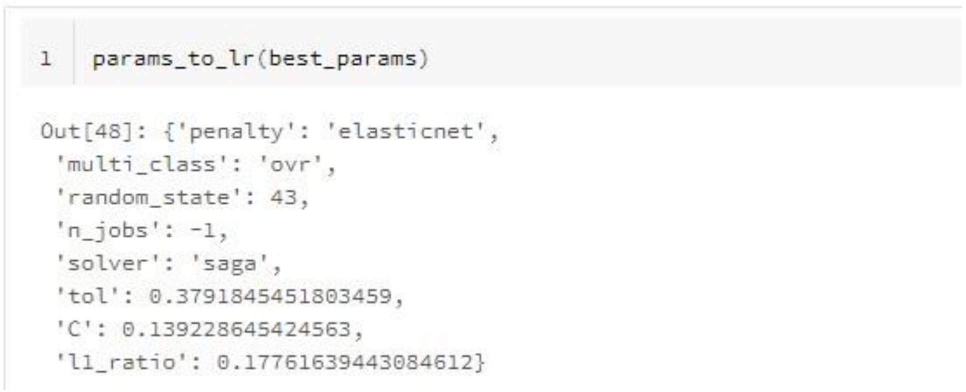
```
encounter_cols=['ENCOUNTERCLASS','COST','PROVIDER_ZIP','START_YEAR']
patient_cols = ['RACE','GENDER','ZIP','patient_age']
comorbidty_cols = ['recent_%s'%idx for idx in range(0,num_conditions)] + ['comorbidity_%s'%idx for idx in range(0,num_conditions)]
selected_features = encounter_cols+patient_cols+comorbidty_cols
```

Hyper-Parameter Tuning

ML algorithms have several parameters that influence their performance, and Hyper-Parameter Tuning is the process of finding the optimal parameter values to improve the model's accuracy. In the following line of code, we use the “fmin” function, which is part of the Python “hyperopt” module focused on parameter tuning. We then display the best parameter values, which we will use to train the predictive model.



```
best_params = fmin(fn=tune_model, space=search_space, algo=tpe.suggest, max_evals=32,
```

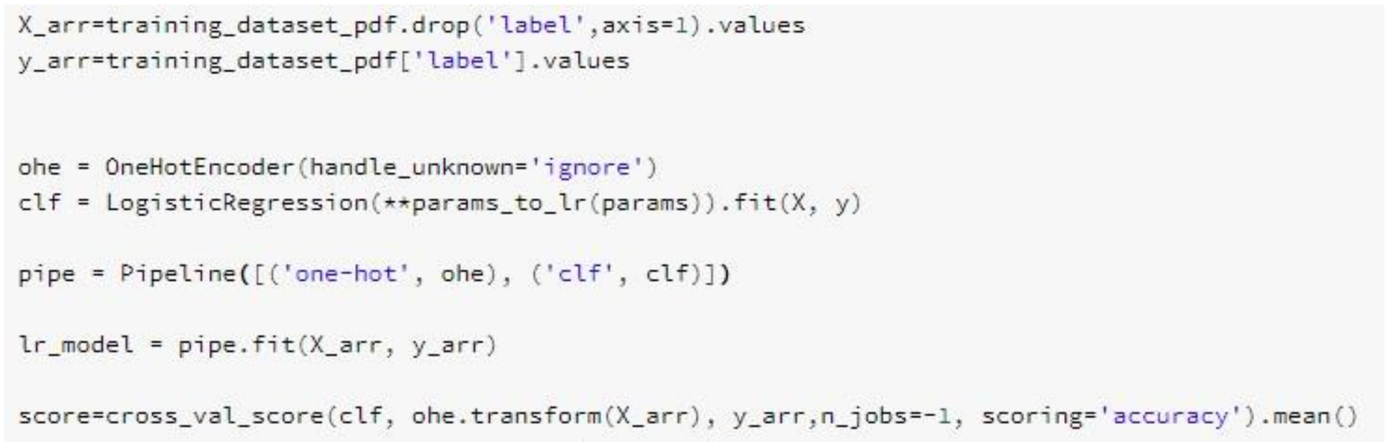


```
params_to_lr(best_params)
```

```
Out[48]: {'penalty': 'elasticnet',
 'multi_class': 'ovr',
 'random_state': 43,
 'n_jobs': -1,
 'solver': 'saga',
 'tol': 0.3791845451803459,
 'C': 0.139228645424563,
 'l1_ratio': 0.1776163944308461 2}
```

Train Predictive Model

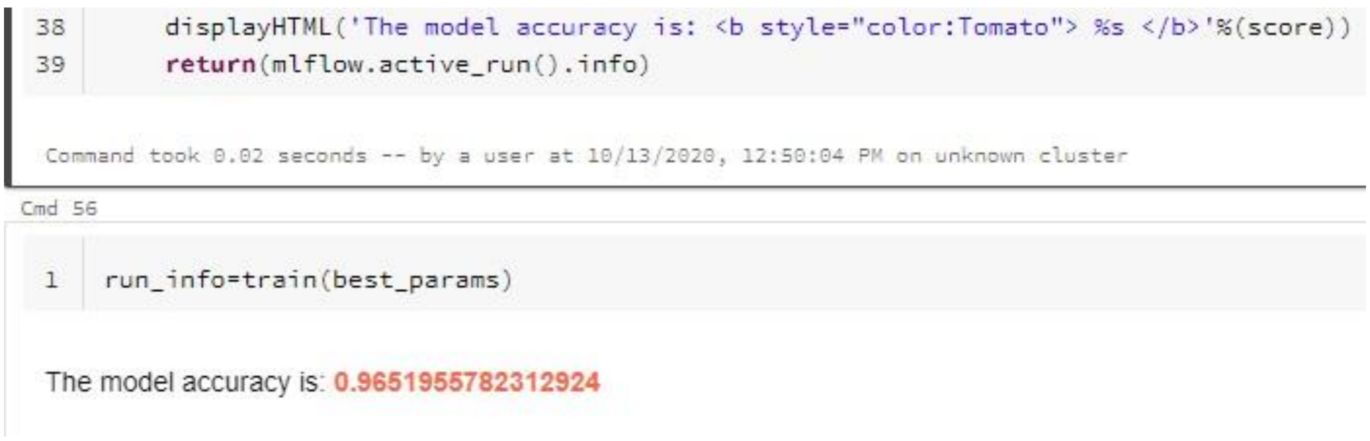
Here is a brief overview of the steps we follow to prepare our predictive model. We have completed many preliminary tasks to reach this stage. The "X" represents the data elements used for training the model, while the "y" indicates whether a patient has a substance abuse issue. Training a model involves several steps, which we group together using the Pipeline concept into a single operation. In this case, we combine two steps: one is a pre-processing step called One-Hot Encoding that handles categorical data, and the other is fitting the model, where we adjust the model coefficients to improve accuracy. We also performed a key preliminary step called hyper-parameter tuning to identify the best algorithm parameter values, which are stored in the params variable. Finally, we evaluate the model's accuracy using cross-validation and store the results in the score variable.



```
X_arr=training_dataset_pdf.drop('label',axis=1).values
y_arr=training_dataset_pdf['label'].values


ohe = OneHotEncoder(handle_unknown='ignore')
clf = LogisticRegression(**params_to_lr(params)).fit(X, y)

pipe = Pipeline([('one-hot', ohe), ('clf', clf)])

lr_model = pipe.fit(X_arr, y_arr)

score=cross_val_score(clf, ohe.transform(X_arr), y_arr,n_jobs=-1, scoring='accuracy').mean()
```

Show Model Accuracy

Here we see that our model is performing well at 97% accuracy. Our demo model uses sample data in this run, and the accuracy may vary, as with any other model.

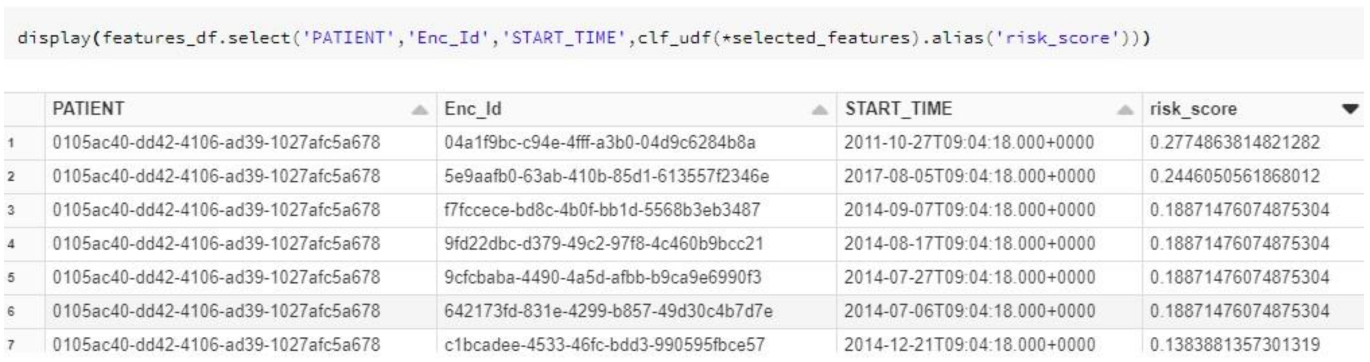


```
    displayHTML('The model accuracy is: <b style="color:Tomato"> %s </b>'%(score))
    return(mlflow.active_run().info)

Command took 0.02 seconds -- by a user at 10/13/2020, 12:50:04 PM on unknown cluster
```

```
Cmd 56

run_info=train(best_params)

The model accuracy is: 0.9651955782312924
```

Predictions

Here, our model predicts a 27% likelihood that Patient-1 (John Smith, obfuscated) will develop a substance abuse condition and a 13% chance for Patient-7.



```
display(features_df.select('PATIENT','Enc_Id','START_TIME',clf_udf(*selected_features).alias('risk_score')))
```

	PATIENT	Enc_Id	START_TIME	risk_score
1	0105ac40-dd42-4106-ad39-1027afc5a678	04a1f9bc-c94e-4fff-a3b0-04d9c6284b8a	2011-10-27T09:04:18.000+0000	0.2774863814821282
2	0105ac40-dd42-4106-ad39-1027afc5a678	5e9aafb0-63ab-410b-85d1-613557f2346e	2017-08-05T09:04:18.000+0000	0.2446050561868012
3	0105ac40-dd42-4106-ad39-1027afc5a678	f7fccece-bd8c-4b0f-bb1d-5568b3eb3487	2014-09-07T09:04:18.000+0000	0.18871476074875304
4	0105ac40-dd42-4106-ad39-1027afc5a678	9fd22dbc-d379-49c2-97f8-4c460b9bcc21	2014-08-17T09:04:18.000+0000	0.18871476074875304
5	0105ac40-dd42-4106-ad39-1027afc5a678	9cfcbaba-4490-4a5d-afbb-b9ca9e6990f3	2014-07-27T09:04:18.000+0000	0.18871476074875304
6	0105ac40-dd42-4106-ad39-1027afc5a678	642173fd-831e-4299-b857-49d30c4b7d7e	2014-07-06T09:04:18.000+0000	0.18871476074875304
7	0105ac40-dd42-4106-ad39-1027afc5a678	c1bcadee-4533-46fc-bdd3-990595fbce57	2014-12-21T09:04:18.000+0000	0.1383881357301319

Integrationals Application

The current HHC software used to manage electronic patient health records cannot retrieve other medical histories from healthcare affiliates outside of HHC. Integrations will add the necessary functions to connect via API requests to other systems and create a comprehensive patient record.

Our UI for the HHC office closely resembles the existing EHR software to minimize additional training for the HHC staff. They are already familiar with the current software; we just need to emphasize the new features.

The system displays patient information across multiple screens, accessible by clicking the respective button. Each screen shows specific data for the patient.

Patient Info: This tab displays the patient's demographics, insurance details, and personal information.

Medical History: This tab displays past medical visits, along with diagnoses and treatments.

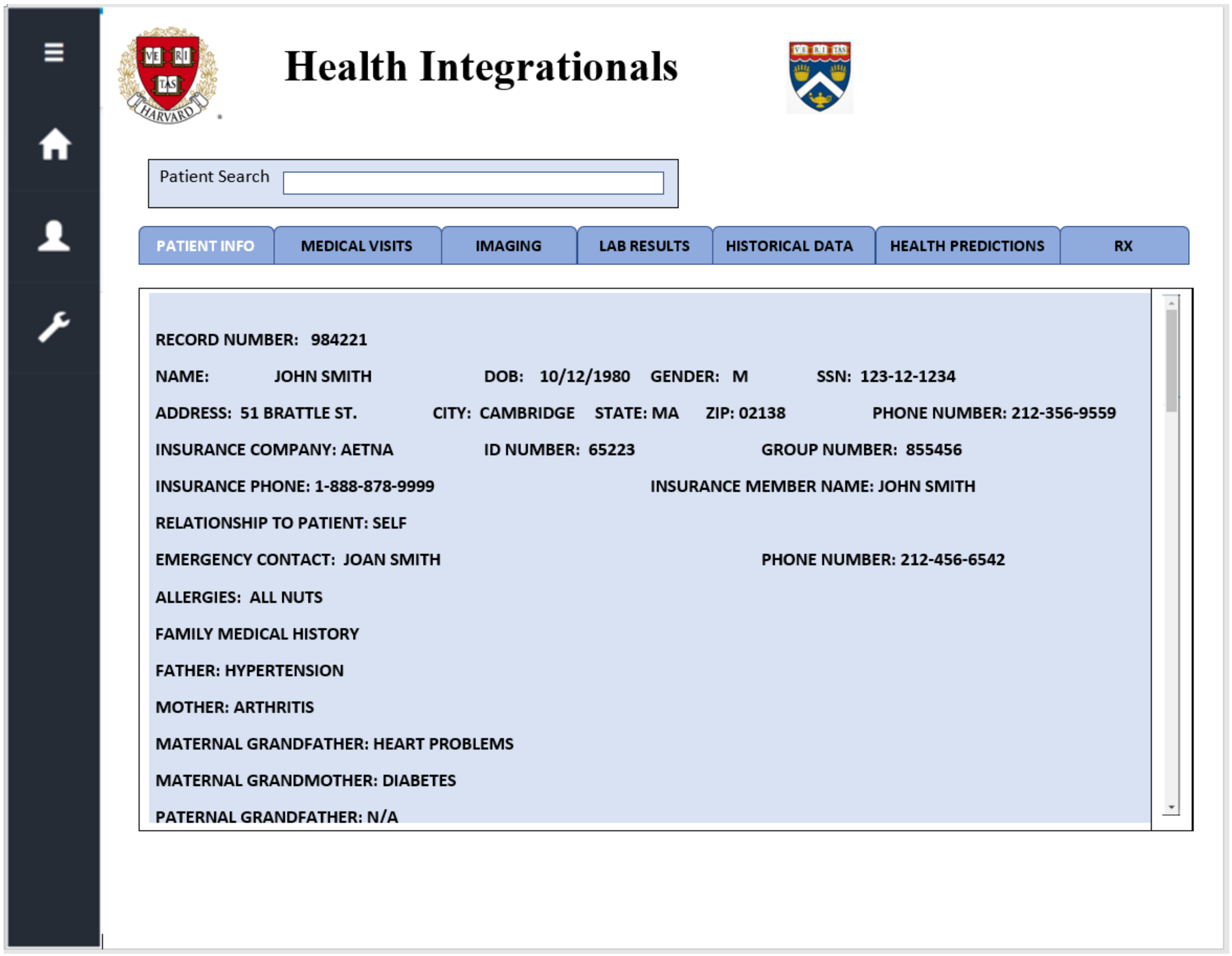
Imaging: Includes all X-rays, MRIs, CT scans, mammography, and others.

Lab Results: This tab displays blood tests, lipid panels, lab cultures, skin grafts, and other results.

Historical Data: Patient information from recent medical visits (HHC and other medical affiliates) organized by date. Include blood pressure, glucose levels, immunization records, weight, and growth chart.

Health Prediction: This is a new feature. With the complete medical history, the system predicts which chronic disease could develop in the patient and suggests steps to prevent it. In Phase 2, this feature can be accessed directly from the patient's phone from anywhere with an internet connection.

Rx: Medications prescribed to the patient, including current and past prescriptions.



RECORD NUMBER: 984221

NAME: JOHN SMITH DOB: 10/12/1980 GENDER: M SSN: 123-12-1234

ADDRESS: 51 BRATTLE ST. CITY: CAMBRIDGE STATE: MA ZIP: 02138 PHONE NUMBER: 212-356-9559

INSURANCE COMPANY: AETNA ID NUMBER: 65223 GROUP NUMBER: 855456

INSURANCE PHONE: 1-888-878-9999 INSURANCE MEMBER NAME: JOHN SMITH

RELATIONSHIP TO PATIENT: SELF

EMERGENCY CONTACT: JOAN SMITH PHONE NUMBER: 212-456-6542

ALLERGIES: ALL NUTS

FAMILY MEDICAL HISTORY

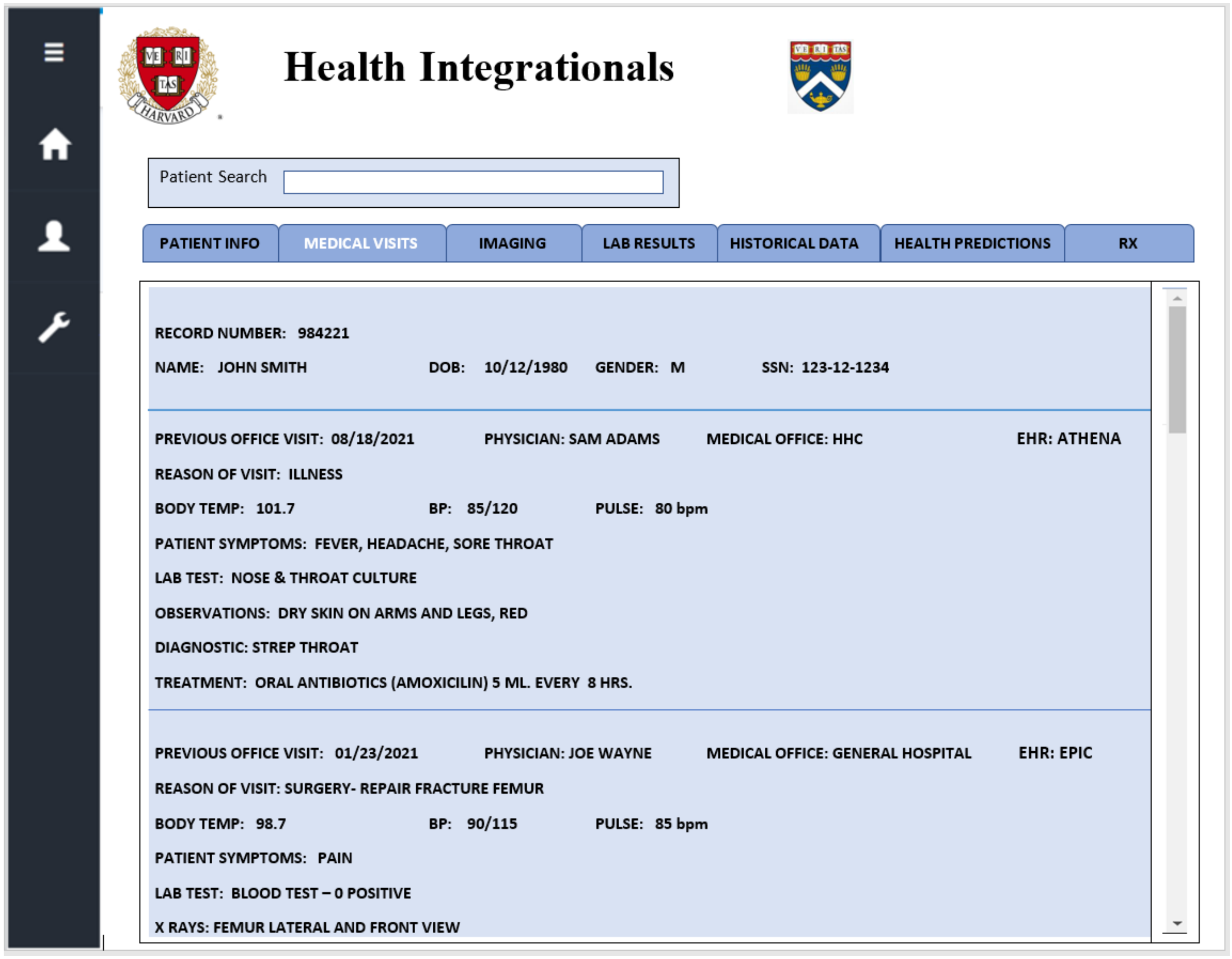
FATHER: HYPERTENSION

MOTHER: ARTHRITIS

MATERNAL GRANDFATHER: HEART PROBLEMS

MATERNAL GRANDMOTHER: DIABETES

PATERNAL GRANDFATHER: N/A



Health Integrationals

Patient Search

RECORD NUMBER: 984221

NAME: JOHN SMITH DOB: 10/12/1980 GENDER: M SSN: 123-12-1234

PREVIOUS OFFICE VISIT: 08/18/2021 PHYSICIAN: SAM ADAMS MEDICAL OFFICE: HHC EHR: ATHENA

REASON OF VISIT: ILLNESS

BODY TEMP: 101.7 BP: 85/120 PULSE: 80 bpm

PATIENT SYMPTOMS: FEVER, HEADACHE, SORE THROAT

LAB TEST: NOSE & THROAT CULTURE

OBSERVATIONS: DRY SKIN ON ARMS AND LEGS, RED

DIAGNOSTIC: STREP THROAT

TREATMENT: ORAL ANTIBIOTICS (AMOXICILIN) 5 ML. EVERY 8 HRS.

PREVIOUS OFFICE VISIT: 01/23/2021 PHYSICIAN: JOE WAYNE MEDICAL OFFICE: GENERAL HOSPITAL EHR: EPIC

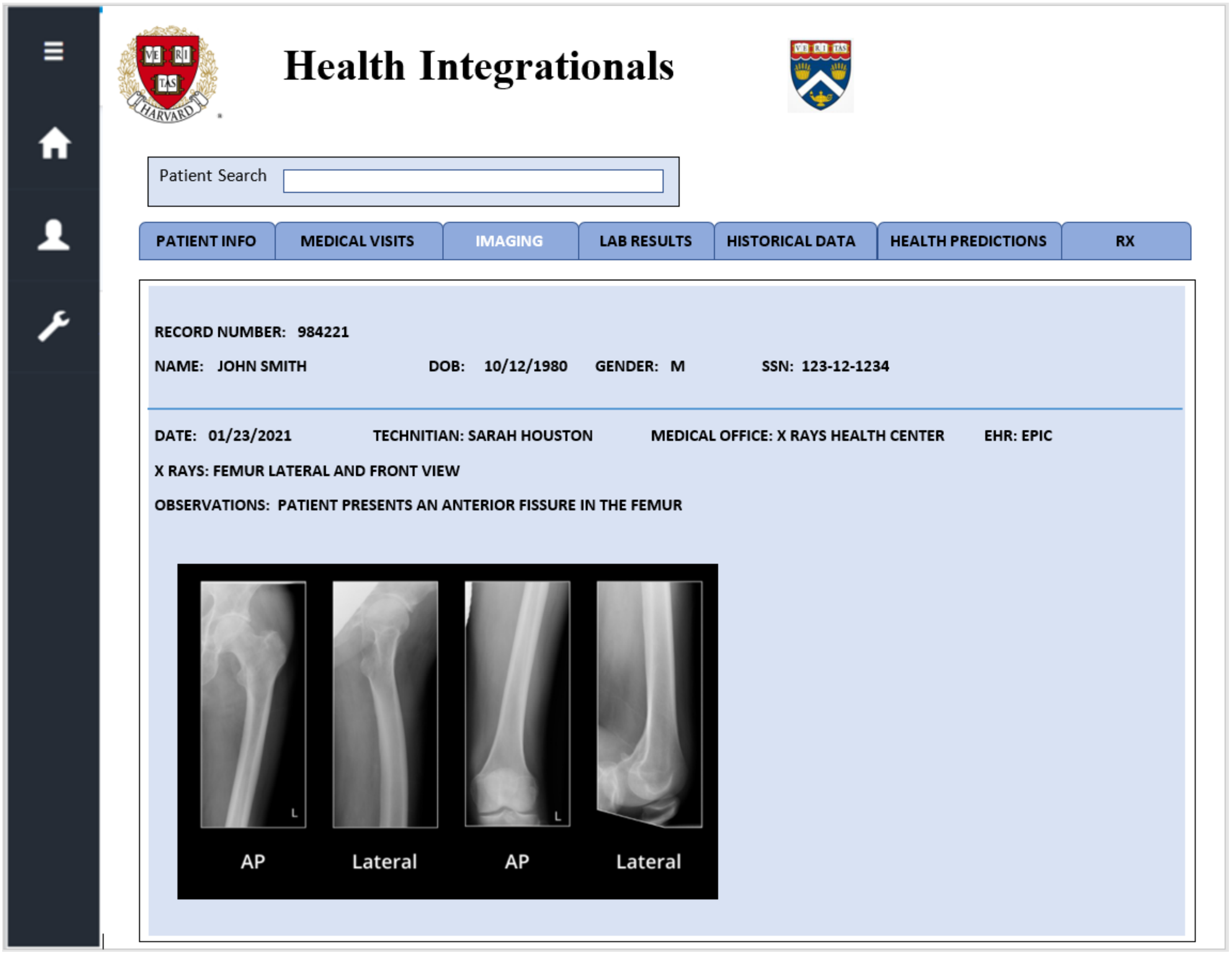
REASON OF VISIT: SURGERY- REPAIR FRACTURE FEMUR

BODY TEMP: 98.7 BP: 90/115 PULSE: 85 bpm

PATIENT SYMPTOMS: PAIN

LAB TEST: BLOOD TEST – 0 POSITIVE

X RAYS: FEMUR LATERAL AND FRONT VIEW



Health Integrationals

Patient Search

PATIENT INFO | MEDICAL VISITS | IMAGING | LAB RESULTS | HISTORICAL DATA | HEALTH PREDICTIONS | RX

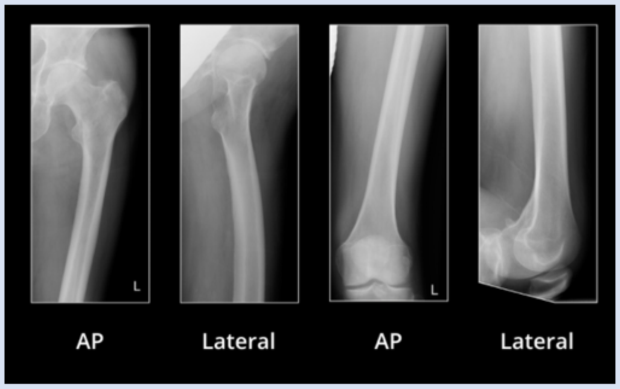
RECORD NUMBER: 984221

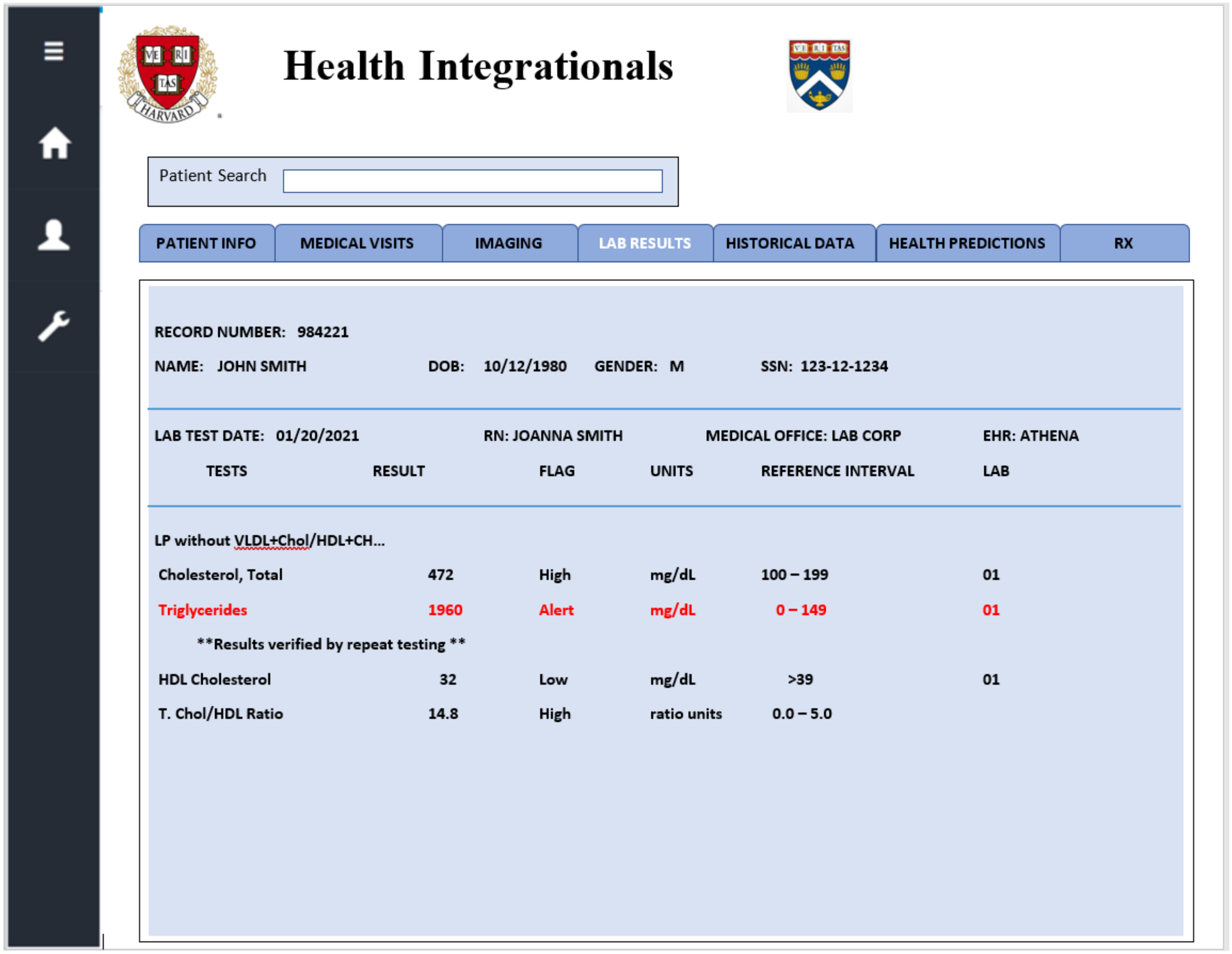
NAME: JOHN SMITH DOB: 10/12/1980 GENDER: M SSN: 123-12-1234

DATE: 01/23/2021 TECHNITIAN: SARAH HOUSTON MEDICAL OFFICE: X RAYS HEALTH CENTER EHR: EPIC

X RAYS: FEMUR LATERAL AND FRONT VIEW

OBSERVATIONS: PATIENT PRESENTS AN ANTERIOR FISSURE IN THE FEMUR





Health Integrationals

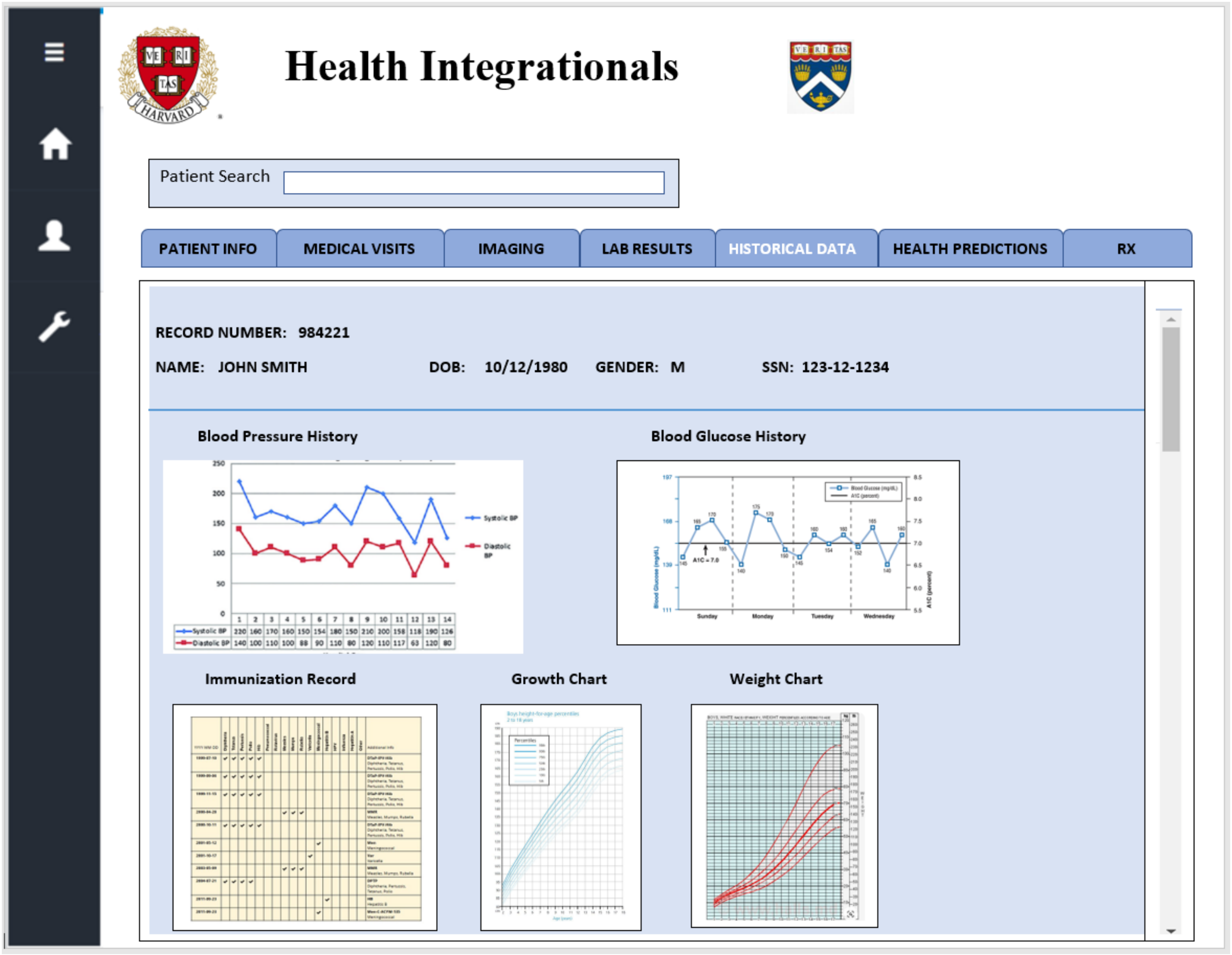
Patient Search

RECORD NUMBER: 984221

NAME: JOHN SMITH **DOB:** 10/12/1980 **GENDER:** M **SSN:** 123-12-1234

LAB TEST DATE: 01/20/2021 **RN:** JOANNA SMITH **MEDICAL OFFICE:** LAB CORP **EHR:** ATHENA

TESTS	RESULT	FLAG	UNITS	REFERENCE INTERVAL	LAB
LP without VLDL+Chol/HDL+CH...					
Cholesterol, Total	472	High	mg/dL	100 – 199	01
Triglycerides	1960	Alert	mg/dL	0 – 149	01
**Results verified by repeat testing **					
HDL Cholesterol	32	Low	mg/dL	>39	01
T. Chol/HDL Ratio	14.8	High	ratio units	0.0 – 5.0	



RECORD NUMBER: 984221

NAME: JOHN SMITH DOB: 10/12/1980 GENDER: M SSN: 123-12-1234

Blood Pressure History

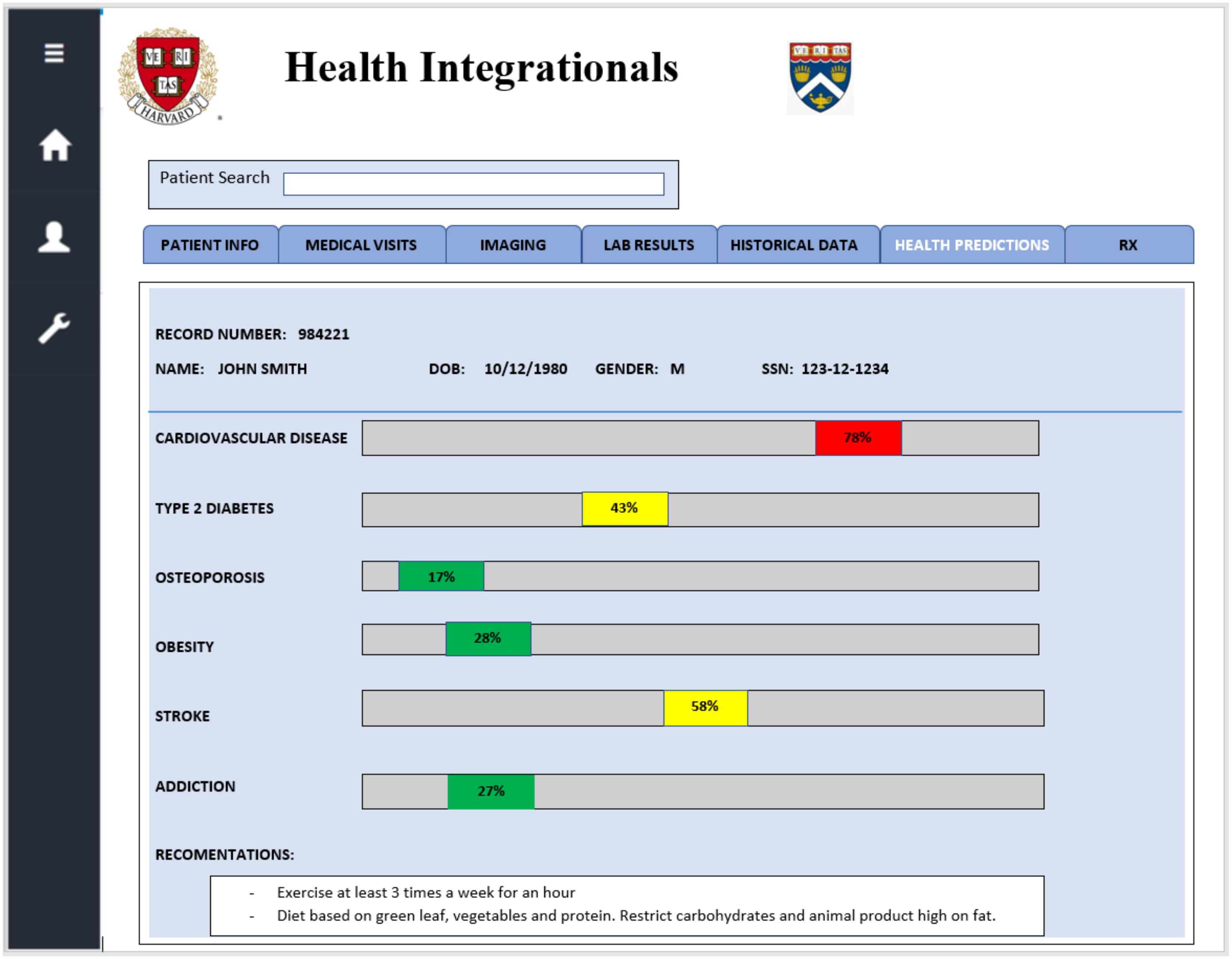
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Systolic BP	220	160	170	160	150	154	180	150	210	200	158	118	190	126
Diastolic BP	140	100	110	100	88	90	110	80	120	110	117	63	120	80

Blood Glucose History

Immunization Record

Growth Chart

Weight Chart



Health Integrationals

Patient Search

RECORD NUMBER: 984221

NAME: JOHN SMITH DOB: 10/12/1980 GENDER: M SSN: 123-12-1234

CARDIOVASCULAR DISEASE	78%
TYPE 2 DIABETES	43%
OSTEOPOROSIS	17%
OBESITY	28%
STROKE	58%
ADDICTION	27%

RECOMENTATIONS:

- Exercise at least 3 times a week for an hour
- Diet based on green leaf, vegetables and protein. Restrict carbohydrates and animal product high on fat.

This concludes the system architecture documentation.