A Methodology for a Scalable, Collaborative, and Resource-Efficient Platform, MERLIN, to Facilitate Healthcare AI Research

Raphael Y. Cohen and Vesela P. Kovacheva, Member, IEEE

Abstract—Healthcare artificial intelligence (AI) holds the potential to increase patient safety, augment efficiency and improve patient outcomes, yet research is often limited by data access, cohort curation, and tools for analysis. Collection and translation of electronic health record data, live data, and real-time high-resolution device data can be challenging and time-consuming. The development of clinically relevant AI tools requires overcoming challenges in data acquisition, scarce hospital resources, and requirements for data governance. These bottlenecks may result in resource-heavy needs and long delays in research and development of AI systems. We present a system and methodology to accelerate data acquisition, dataset development and analysis, and AI model development. We created an interactive platform that relies on a scalable microservice architecture. This system can ingest 15,000 patient records per hour, where each record represents thousands of multimodal measurements, text notes, and high-resolution data. Collectively, these records can approach a terabyte of data. The platform can further perform cohort generation and preliminary dataset analysis in 2-5 minutes. As a result, multiple users can collaborate simultaneously to iterate on datasets and models in real time. We anticipate that this approach will accelerate clinical AI model development, and, in the long run, meaningfully improve healthcare delivery.

Index Terms—Artificial intelligence, distributed computing, electronic healthcare records, medical information systems.

I. INTRODUCTION

Recent advances in artificial intelligence (AI) in healthcare hold the potential to increase patient safety, augment efficiency, and improve patient outcomes. In clinical care, AI technologies can aid physicians and other healthcare practitioners in diagnosis and treatment selection, risk prediction and stratification, and improving care delivery [1]. In addition, AI technologies can provide faster and more efficient access to data and tools for medical researchers, improved communication and engagement of patients in their care, more efficient hospital operations for healthcare administrators, cost savings for insurance companies, and opportunities for entrepreneurs [2], [3]. The power of AI technologies in healthcare was demonstrated in the COVID-19 pandemic with the adoption of AI tools for contact tracing, drug discovery, clinical decision-making and prediction, and knowledge dissemination [2]. Yet, most sophisticated AI models exist only in high-profile publications, and only a few are implemented in clinical practice [4], [5].

The barriers to translating data science research into patient care are inadequate data quality, scarce resources, and high patient confidentiality needs. With the Health Information Technology for Economic and Clinical Health Act of 2009, many institutions have transitioned to electronic health records (EHR) that provide a rich medical data source. Most EHR systems store patient data in heterogeneous formats, sometimes combined with legacy systems. The EHR data have a significant degree of missingness, misclassification, and errors [6]. Furthermore, high-dimensional data like continuous physiological data, EEG, genetic, and diagnostic tests provide valuable clinical insights but require large storage and processing capacity. Harmonizing these diverse data sources can be a very time-consuming and resource-heavy process. Moreover, healthcare is a highly regulated industry, and there are complex requirements for patient safety, confidentiality, and ethics [7]. These characteristics of the healthcare landscape present significant challenges for the rapid and efficient development and deployment of AI technologies.

We aimed to create a system that can automate clinical data acquisition and processing efficiently in machine-learning-ready formats, enabling near-real-time generation of datasets for subsequent analysis and development of AI models. At the same time, scalability to overcome significant data needs and generalizability to handle heterogeneous projects are requirements for this system. In addition, we followed the strict data governance and patient privacy requirements in our institution. Achieving these objectives yields a low-friction platform that focuses on iterative collaborative processes, making the AI workflow more productive and efficient. Shown in Fig. 1 is the overview
of the system architecture of the platform, Medical Record Longitudinal Information AI System (MERLIN). The system follows the traditional design of a web application, with a user-facing application layer, a security layer, data controllers and data processors serving as the logic layer, and the data storage layer. Key to the platform's design is the creation of many stateless, independent Python microservices in Docker containers that enable data processing, dataset generation, model development, and interaction with applications. These stateless modules interact with multiple stateful storage services, such as a PostgreSQL database, Simple Storage Service (S3) compatible storage, a RabbitMQ message queue, and networking services, such as NGINX. The core advantage in the platform's development was the early translation of data to atomic representations and data processes with parallelized microservices which enabled substantive benefits at later stages of dataset and model creation.

The rest of this paper is organized as follows: Section II presents an overview of some of the related research; Section III describes the major challenges of the biomedical data analysis workflows; Section IV provides the platform architecture overview; Section V provides a detailed description of each of the platform components, and Section VI presents the platform workflow. The benchmarking tests and results are presented in Section VII; the discussion is in Section VIII, and the conclusions are presented in Section IX.

II. RELATED WORK

Multiple systems exist that can automate data collection and warehousing; recent works demonstrate progress in data translation, and the AI model lifecycle and the most relevant systems are discussed below (Table 1).

Large research patient data registries can store and provide research data access efficiently. One of the most widely used general medical data platforms implemented in many institutions is i2b2 [8]. It supports multiple uses by researchers, clinicians, data scientists, and organizations. The platform has flexible ontology-driven data access patterns and community-developed plugins and interfaces, which enable a variety of configurations. Data from multiple sources with varied terminologies and formats undergo a series of translations that map the original source data into common representations to better facilitate subsequent data analysis [9]. Depending on the specific implementation, the platform’s functionality can be extended using multiple plug-ins and third-party software to enable complex data operations, including statistical analysis, AI, natural language processing, patient recruitment, and clinical trial management [10]. Currently, i2b2 is used in over 250 locations worldwide [11].

The UKBiobank [12], is one of the largest biomedical databases and contains the clinical, imaging, and genomic data of 500,000 participants from the United Kingdom. Recently, a cloud-based research platform was added that provides access to data and diverse analysis tools, such as genomic software (PLINK, SAIGE), Python analytical libraries (NumPy, Pandas), machine learning (ML) frameworks (TensorFlow, Scikit-Learn) and image processing (nipype). To date, more than 3,000 studies have been published on a wide range of topics.

The program AllofUS [13] was launched in 2017 with the goal of developing a research cohort of 1 million people from diverse backgrounds living in the United States. The main goal of the platform is to provide access to registered researchers and make available the research results to the participants. The research workbench provides integrated tools for data processing and analysis, including R, Python, Jupyter notebooks, and compute resources in an online cloud environment. This platform has enabled multiple projects on broad topics such as polygenic risk scores, epidemiology, women’s health studies, and sleep disorders.

A number of disease- or subject-specific platforms exist. For example, UNRAVEL [14] is a platform developed using data from EHRs, genetics, and tissue samples from patients with cardiomyopathy. The system contains data from 828 patients with cardiomyopathy and their relatives; all data is processed and available for use. In addition, the platform has an integrated tool for text mining and natural language processing (NLP).

Some of the platforms are designed to serve institution-specific projects. HealtheDataLab [15] is a platform designed to develop clinical ML models using EHR data. It uses cloud infrastructure with the integration of multiple software tools in R and Python languages which can be used by data scientists. Similarly, Isthmus [16] is a platform centered on the development and deployment of AI models into clinical practice. Neptune [17] is a single-institution research data warehouse developed to interact efficiently with EHR data. It uses an atomic data design, which allows for attribute representations of data to be stored in their most granular form. This process aids in the harmonization of data from multiple sources. This approach has allowed for efficient data provision for multiple purposes for research projects and participation in national data marts.

A recent platform for medical imaging AI, Tomosuite [18], (created by R.Y. Cohen), utilizes a microservice architecture with an emphasis on the early and accurate translation of data to machine-amenable formats to lower the barriers to entry for AI development. This platform reduces data acquisition, cohort
creation, annotation, and model training time by enabling the continuous development of both data cohorts and models. The platform was designed to serve the needs of radiology research groups, and it integrates ML tools for image analysis which can be used by both data scientists and radiologists.

We sought to create a framework agnostic to projects and data sources, which can take advantage of atomic data representations and a microservice architecture to serve the needs of a wide variety of projects. This platform will significantly reduce the time to model completion; in addition, the integrated Python ML tools allow both data scientists and physicians to work on multiple projects. Thus, MERLIN creates opportunities for faster AI integration in clinical care to meaningfully impact patient outcomes.

III. BOTTLENECKS IN THE BIOMEDICAL DATA ANALYSIS WORKFLOW

Solving challenges that occur late in the workflow by addressing them in earlier stages enables the platform’s scale and efficiency. This allows for simple algorithms to accomplish formerly complex needs within the workflow. The stated goals for the platform will be distilled into three measurable objectives pertaining to biomedical data analysis; then, the principles for handling both data and algorithms that govern the platform design will be presented; and finally, we will show how we applied those principles to implement the platform and answer our established objectives.

The challenges associated with healthcare data have been extensively discussed [6] and include missing data, redundant data, lack of structure, and standardization. For example, “preterm”, “Pre-Term”, and “pregnate” all refer to the same concept, preterm delivery, despite appearing differently within the EHR. Likewise, the process of ingesting data can be time-consuming and resource-intensive when new, often unstructured, data becomes available for use on new projects. Poor standardization within source data can propagate into the data warehouse and beyond into subsequent datasets. As a result, such patient cohorts would need to be verified due to the risk of variable representations within derived values. Furthermore, downstream modeling thus becomes more laborious due to those data irregularities.

To reduce many of these bottlenecks, our platform focuses on addressing later challenges by minimizing their impact during earlier stages of the workflow.

Development efforts were guided by three primary objectives to achieve these aims, which will be used as touchstones in evaluating the platform:

- **Objective 1:** Acquire, ingest, and standardize data automatically into a machine-analysis-ready form.
- **Objective 2:** Generate data cohorts efficiently and comprehensively. This can enable researchers to work dynamically.
- **Objective 3:** Generate standardized datasets in a format for common analysis packages, such as NumPy, CSVs, and Excel. This dataset creation process should occur in near-real-time.

Additional considerations include the amount of data, which may be in the hundreds of thousands of patients, the security and regulatory needs for the platform’s execution, and the management of people and projects within the system so that
the created tools can serve as an interactive workbench for researchers.

IV. PRINCIPLES FOR DATA HANDLING AND ALGORITHM DESIGN

The principles that govern the development of the platform are divided into principles for the handling and storage of data and principles for the development and organization of algorithms.

A. Principles for Data Storage

- **Atomicity/Indivisibility of Data Attributes:** Once data are translated, they must be stored as data attributes that cannot be broken down into meaningful sub-components. For example, “Test Result”: “positive covid 19 antibody test” can be broken down into sub-components, whereas each item in the collection of “Test Type”: “covid 19”, “Test Subtype”: “antibody”, “Test Result”: “positive” cannot be further broken down into meaningful sub-components.
- **Consistency:** The same value should have the same meaning across the entire dataset, and multiple values cannot be used to represent the same concept. For example, “Positive”, “Pos”, “True”, and “Yes” are all results for a Covid 19 test that indicate a positive result. Instead of this variety in presentation, the system instead stores the boolean value True to indicate the test was positive, False if it was negative, and Null if it was inconclusive.
- **Translatability:** Data that are not stored as numeric values must have a complete mapping or methodology provided to convert these data to a numeric format. Data that are stored numerically may need a mapping to a human-readable value. For example, “low”, “medium”, “high” might map to 0, 1, 2.
- **Traceability:** Data must be tracked through its entire lifecycle within the system. Data that has been translated must maintain a reference to its source data. Data that exists in a version-controlled dataset must maintain a reference to the translated data from which it was drawn. All models must maintain a reference to the dataset it was developed with.

B. Principles for Modular Algorithm Development

Every component of the platform architecture is a containerized module that performs simple and small-scale individual tasks without cross-module dependencies. This reduces bottlenecks and enables parallelization that can take advantage of cloud-computing resources and distributed workloads. At the core of the system are common shared resources:

- **PostgreSQL Database:** Storage of most data in a relational database, including raw EHR, translated data, manual data collection, projects, and management records.
- **Simple Storage Service (S3):** High-resolution data, large text files, version-controlled datasets, and models are stored in an S3 resource. This resource provides file-like storage and is common in services like Amazon Web Service (AWS).
- **Messaging Queue Service (MQ):** A service for worker modules to post and read jobs to and from queues like a messaging board. When modules complete their tasks, they inform the message queue resource the job is complete and receive the next job to work on. Many modules, especially copies of the same module, may all operate on the same queue at once to provide large scalability of processing capacity.
- **Networking Services:** Some modules serve user-facing applications and need to communicate with other networked devices, so they interact with the stack of network services, including NGINX, to connect to their endpoints.

On top of this layer are hundreds of worker modules, which are stateless services that can only communicate with the MQ resource and the shared storage resources, but not with each other. This is shown in Fig. 2. These modules run in virtualized Docker containers and can run on distributed compute systems so long as they can communicate with the core resources. Functions that a user of the system may view as a single task, such as pulling data for specific patients from the EHR, creating datasets, or running basic statistics, are not handled by a single module but rather by a collection of many modules, often in parallel, before the results are returned to the user. Each module performs small, simple parts of a workflow that in aggregate, performs larger tasks. For example, one module’s function may be to analyze a request to pull data from the EHR and split it into appropriately sized requests for another module that actually pulls the data from the EHR. Another module might upload the results from the EHR into a staging database before additional specialized modules process that data. This design enables an ensemble behavior that allows an automated system controller such as Kubernetes to dynamically scale copies of modules with the immediate computational needs within the platform, swarming large jobs with temporary resources at appropriate scales.

The centralization of the network stack allows the system to maintain one path in and one path out, thereby simplifying...
Multimodal data sources feeding into a centralized data collection process in the platform.

V. APPLYING PRINCIPLES TO CREATE AN INTERACTIVE WORKBENCH FRAMEWORK

The implementation of these principles is reflected throughout the development of this platform and in the various workflow components. For each of the following workflow sections, modules are developed that allow developers to “plug-and-play” with specific needs for new sources; module templates serve as a framework that allows this platform to be readily adapted to new data, sources, and needs.

For example, in the data ETL pipeline, each data source (such as an EHR, legacy database, or live data collection) has a unique processing component which is wrapped by a common framework. Including a new data source is, therefore, as simple as using this framework to duplicate a template and add the custom code needed to interpret the new source.

The core layer of the platform is the data collection, translation, and analysis pipeline. On top of this core layer are layers for management and organization, security, and applications to interact with this environment. Because this platform focuses on addressing early needs in the pipeline, the methods in the core layer will first be discussed in the order that data flows through it, followed by the methods included in the layers on top of it.

A. Data Collection Processes

Data can be collected from multimodal sources. The platform includes pathways for real-time and retrospective data collection. Tools were created to enable bedside structured data entry, which along with a timestamp, are continually streamed back to the server application. Shown in Fig. 3 are also software wrappers that encapsulate medical device recording software and stream continuous waveform data, such as 300Hz ECG data, back to the server.

The platform also supports querying and pulling data from the hospital’s EHR infrastructure. Given the depth of the EHR data, these sources also represent a large trove of longitudinal data on vast retrospective patient cohorts. There are sometimes errors with data stored in the EHR, and the platform includes a QA/QC pipeline to detect and mitigate many of these errors. Any manual corrections obtained during the QA/QC process are recorded and treated as an additional data source.

Additionally, for collaborative efforts, external data may be uploaded and stored. These data will be processed and transformed into the same standardized formats as internal data.

Every data resource has its data source controller, which can initiate a query (for resources such as the EHR) and receive data, including streaming capability in the case of medical device recording.

These data are recorded and stored in the S3 resource. To avoid bottlenecks in the case of large tasks, the data are first stored before being processed to separate the operation to receive data from their interpretation. When the data is ready to be processed, these controller modules publish a message to a messaging queue to inform the collection of data upload controllers to ingest the files into the proper locations for the subsequent data ETL modules. This upload controller receives the instructions from the message queue informing it what type of data was received and where its location for processing is, such as in the source database for most data or reformatted data in S3 storage for waveform analysis. Once the data is present, it creates a series of messages in the appropriate message queues pointing to small blocks of unprocessed data.

Once data has passed through this stage, it is ready to be translated and interpreted.

B. Data ETL Processes

To facilitate the data translation process, a python library was developed. This library standardizes the creation of stateless Docker modules, which handle all communication with the message queue, acquisition of source data, and the insertion and conflict resolution for ingested data. For each new data item, a developer needs only to define the internal logic in a single function which translates a single row of source data into a list of output data elements.

The ETL modules adhere to the data principles established above. Output data are consistent in representation (numeric or decimal is preferred, short text if required), in unit (standardized to the metric system), and if text data, in choices. Concepts typically stored as long strings in the source data, such as the formulation of drugs in the EHR, are broken down into their components. For example, the value “BUTALBITAL-ASPIRIN-CAFFEINE 50 MG-325 MG-40 MG CAPSULE” is broken down into three linked entries (medication class, dose, units): (“aspirin”, 325, ‘mg’), (“butalbital”, 50, ‘mg’), and (“caffeine”, 40, ‘mg’). These three entries are trivially reassembled to indicate that the patient received these three drugs simultaneously.

All modules are designed with medically relevant knowledge stored in JSON configuration files. This decreases required maintenance efforts for developers and allows for the creation
of interfaces that let clinicians update modules without the need for programming expertise. If data is found that cannot be translated with the framework in the configuration file, it is flagged for QA/QC review. Furthermore, if data is translated but produces unexpected results, it is automatically sent to the QA/QC pipeline and may be withheld based on the severity of the deviation.

Modules are specialized to perform small and specific ETL tasks. For example, one module handles ICD code extraction from the EHR data; another handles vital signs from the EHR flowsheet data, and another processes waveform data from live data collection. This is done so that Kubernetes can dynamically scale the number of each of these modules with the amount and kind of data waiting to be translated. This is also shown in Fig. 2. The entire system contains many dozens of ETL modules based on the data form and type.

Except for data passed to the QA/QC pipeline, once data have moved through this phase, they are stored in the data warehouse in adherence with the outlined data principles to address Objective 1.

C. Quality Assurance and Quality Control Processes

The QA/QC pipeline begins with one of the following pathways:

- **Errors in the ETL Pipeline**: Triggered when data cannot be parsed in the ETL processes. This includes cases when a processed value lies outside of an expected range.
- **Data Warehouse Validation**: Data is selected at random to ensure the ongoing quality of the data warehouse.
- **Dataset Validation**: A user specifically requests validation for a generated dataset.

The majority of the QA/QC processes pertain to the first pathway. If the integrity and accuracy of a data point are in question, the ETL modules carry out a diagnostic process on the data and stage it for manual mitigation, as shown in Fig. 4.

Once data are moved to the QA/QC pipeline, the mitigation phase begins. Data are inserted into a special QA/QC database, in different tables based on their source. Data coming from the ETL process maintain links to the source data in question along with a status message from the ETL module describing the source of the problem.

The clinician uses available tools, such as their access to the EHR and their medical background, to decide what the correct input or parsed value should have been. They can submit a correction to the source data, which is used as a secondary source of input that overrides the source data. In addition to the correction, the physician’s approval and timestamp are recorded by the system.

If a physician believes that an ETL module produces a systematic error, they can issue a HALT command to terminate those modules. Once developers have upgraded the module, all data processed by that module will be re-processed.

In the case of a QA/QC process on a dataset, the results of the operation are also stored with the dataset.

D. Dataset Creation Processes

The workflow components above address Objective 1. The benefits of early attention to data quality and translation can now be realized as Objective 2 is addressed.

Dataset creation is designed to be an interactive process for researchers. Researchers can define complex variables and custom phenotypes to create datasets from the atomically stored data elements in the data warehouse. Variables are defined as a

- **Data Source**: Either a table and field or a set of custom variables.
- **Operation**: A function to apply to the data. It may be as simple as “Identity” to return the existing value, more complex like “Like”, which is similar to the SQL Like operator, or even “Time Series” to aggregate entire series with timestamps, values, and dates. Numerous operations are defined within the system, and developers can contribute to a library available for all users.
- **Value**: An externally provided value or list of values to be used with the operation. For example, ‘O10.’ might be provided to the “Right Like” operator to yield all ICD codes that begin with ‘O10.’.
- **Timeframe**: A limit on when the operation can be applied. These can be patient-dependent, such as “During Pregnancy” or “Before Visit”, or during specified time frames, such as after January 1, 2015.
- **Constraints**: A list of other variables to use as constraints on the current operation.
- **Mapping Scheme**: An optional choice that lets users define how data can map to numeric values. The default is stored mappings that are created at the same time as translation modules are developed, but users may choose from common orderings, alphabetical order, or create their own.

When the user launches the dataset generation process, a series of modules extract the relevant information from the data warehouse with a series of parallel SQL commands and data stores in S3. The commands are optimized to limit both the complexity of the queries on the database and the size of the data being extracted from it. Subsequently, the dataset is compiled into a single CSV or NPY file available for further use in the application or user download. Two copies of the dataset are...
stored: one is a human-readable document with standardized strings and text, and the other is machine-analysis-ready, which uses the mapping scheme for each variable to create a numeric dataset. Both copies of the dataset are version-controlled with additional references and logging stored in a database table.

Following the dataset’s creation, the next phase, dataset analysis, is automatically triggered.

E. Dataset Analysis Processes

The presence of machine-readable datasets simplifies the automation of basic statistics, with the target of satisfying Objective 3. A series of modules work in concert to create a dataset summary and basic statistical analysis upon dataset creation. Users of the system can define additional statistics, or discrete or continuous groups within the data for more advanced summary and statistics usage.

The dataset summary includes mean, median, standard deviation, extreme values, and percentile distribution for each variable and group. These data can be used to create graphs within software applications. The statistics module creates correlation maps in addition to user-specified statistical tests. These include t-tests, Pearson and Spearman correlation, and ANOVA tests. All results are exportable in CSV files.

When researchers are satisfied with a dataset, they can use modules within the platform to train many deep learning models in parallel. The built-in model zoo includes FCDNNs, CNNs, LSTMs with Attention Mechanism, Transformers, and other common models. Model checkpoints are saved, and researchers can track training progress. Users may also download generated Python code and model weights to further adapt the trained models. Models may also be exported as Docker images for use in applications.

This completes the data collection, translation, and analysis core layer. In order to operate within a HIPAA-compliant environment and facilitate interaction with data and the workbench environment, additional layers are needed.

F. Security and Integrity Layer

All actions within the platform are time-stamped and logged. All data is stored with a linking identifier back to its source as well as the version of the module used to translate it. All datasets are version controlled and maintain links to the versions of the modules used to translate the data from which it was derived. This allows data at any stage in the pipeline to be traced to the source data it originated from and what code processed it along the way. Combined with the logs of user activity and limits and permissions for users to perform necessary functions, this contributes to the system’s design for accountability and integrity.

Furthermore, the system logs all activity in and out of the platform. It also enforces that all data is associated with IRB protocols, and that projects, researchers, and data (or datasets) all must belong to the same IRB protocol to be accessed. Only individuals with administrative privileges can control user access. This is done to provide IRB and HIPAA compliant operation.

Combined with logs, this can facilitate auditing and regulatory needs for the platform.

G. Application Layer

Numerous applications make up the platforms’ ecosystem. These range from data-collection-oriented tools, research tools, and even management applications.

Data collection applications operate locally on computers for bedside use. Following authentication, users select an appropriate project and create and update entries for patients. The data generated by the tool are structured and standardized. Structured data entry types can be created by authorized users on a centralized web client application.

Traditionally, in live recording sessions, users may not be able to record all aspects of an event at the exact moment it occurs. Errors in event timestamps may arise from recording delays. To minimize this error, in the platform’s data collection applications, users press a button that immediately records the timestamp of the event. They then are able to fill in any supplemental numerical values at the first available moment that they can safely do so.

These applications store temporary anonymized backups of data collection locally in the event of a network outage and opportunistically stream data to the server when network connectivity returns. The interface of this application was designed to maximize usability and efficiency for bedside use.

The data collection applications also integrate with other solutions that record live data from medical devices and are able to initiate and control streams back to the platform. Other desktop solutions designed for data collection can also be integrated, and some may get software wrappers or daemon applications to manage the streams to the platform.

A web-based management application allows authorized users to create, update, remove, and manage users, projects, and data. This application provides access to logging data and source data so that administrative functions are available to principal and system administrators.

A REST API was created that allows for new tools to be created and integrated into the system. Provided proper authentication and privileges, it can interact with all elements of the system functionality, enabling developers to further expand the capabilities of the platform environment. This API even can enable federated learning efforts between two deployments of the platform.

VI. WORKFLOW

The implemented platform, MERLIN, has the following benefits:

- **Project Agnostic Design:** Data are stored and can be reused in multiple projects. The system is adaptable to various needs of a diverse research environment and need not be rebuilt for each new project.
- **Quality Assurance and Quality Control:** This platform is transparent to investigation into data quality and error. It contains methods and tools to evaluate and mitigate data translation and curation issues.
Fig. 5. The processes involved with the data lifecycle within MERLIN. This includes data collection, ETL, QA/QC, dataset creation and analysis.

- **Minimization of Effort for Both Implementation and Usage:** The system is efficient to build, maintain and update and has intuitive interfaces. This reduces the effort needed to develop and interact with the system.

- **Modular Construction:** The modular nature of the platform’s design enables its functionality to be easily expanded with the addition of further modules. This design allows research efforts to start quickly. When new ETL modules are created, they can be added to the platform without additional modification or re-processing of data, allowing the cohort of available data sources to be expanded dynamically. Furthermore, new analysis and modeling approaches can be introduced with new modules. This allows the platform to grow with the needs of the research group.

- **Minimization of Required Human Intervention:** Researchers spend minimal time performing tasks such as data translation and tracking and instead focus efforts on data analysis and problem-solving.

Due to the modular nature of the system construction, its design is amenable for use with cloud technologies such as Amazon Web Services (AWS), Google Cloud Platform (GCP), and Microsoft Azure to scale with operational needs.

This system follows the traditional design of a web application, with a user-facing application layer, a security layer, data controllers and data processors serving as the logic layer, and the data storage layer.

An expanded view of the Data Processing core layer is shown in Fig. 5, separated into the five stages addressed in Section V: data collection, ETL, quality assurance and quality control, dataset creation, and dataset analysis. Modules for these stages are highly independent of the user experience in attached applications; instead that is left to the interfaces created to interact with and control the system.

Of those interfaces, the web client portal serves as the main application to access and interact with projects and data. Using it, researchers can pull data from the EHR; interact with the QA/QC pipeline; create and manage projects; design, pull, and analyze datasets, track model progress and training, manage people, data, and projects under IRB protocols; and conduct system auditing all in one place. This portal is designed to simplify the user experience, allowing them to spend their time interacting with data instead of managing computers and data cleaning.

The platform’s performance is not specialized in one area. Instead, it serves as a workbench that enables clinicians and data scientists to create cohorts, define phenotypes, and analyze data in methods amenable to the needs of specific projects. This is because the platform stores data in atomic forms so that they can be combined on-the-fly to suit the needs of research efforts. This allows the inclusion of new variables into models in mere minutes. This even has the benefit that suboptimal results can be detected early, and mitigations, such as thoughtful changes to cohorts’ specifications or variables, can be implemented.

MERLIN has been used to date on multiple projects [19], which took advantage of components as they became available.
throughout its development. The system has parsed the medical records of over 200,000 patients from 1989 to the present day.

VII. EVALUATION

A. Method

The platform was benchmarked using a simulated patient data resource. For each data type the platform ingests, a generator function was written to create random strings that looked like real data with the statistical properties of real patient cohorts. This synthetic resource could be queried like a real EHR resource. The records of 79,143 patients (obtained using IRB #2020P002859) were used to calculate the mean and standard deviation of the size of each string so that the generator would produce data with this mean and standard deviation. For each category, entries were sorted from the most to the least common, and the generator drew from these lists of words to create new random entries. This ensured that the ETL modules were benchmarked at their data processing rate instead of directing malformed data entries to the QA/QC process, which would have given a false sense of platform efficiency.

To evaluate Objective 1 (Data Acquisition and Ingestion), the platform was benchmarked by running synthetically generated data from a test-EHR system through the intake pipeline. This was done for varying numbers of patients: 1, 5, 10, 50, 100, 500, and 1000 to evaluate how the system can scale with a simultaneous load of patients to be ingested. Each patient was simulated with between 2000 and 5000 entries representing approximately 8.73GB of data for the 1000 patient test. For this test, the number of working modules was held constant, and the time until completion, CPU and memory usage was reported. The objective is that the system can perform in close to linear computational time. A second test was performed in which the number of patients was held constant at 1000, and instead, the number of nodes was varied between 1 and 3 to observe how the platform’s performance changes with horizontal scaling. Each node contains 100 worker modules. The time until completion, CPU usage per node, and memory usage per node were reported. The objective is that the system will decrease each of these metrics with additional nodes. The measured operation is a one-time ingestion cost per patient.

To evaluate Objective 2, the platform was benchmarked for the creation of datasets. Although benchmarks attempt to capture the system performance, this step is highly use-case dependent and is influenced by the complexity of the query. The test scaled the number of variables and measured the response time. The variables added were drawn from a set of functions that required an expected average amount of computation, neither overly simple like the identity function, nor overly complex. This test was performed for 1, 5, 10, 50, 100, 500, and 1000 unique variables. Response time, CPU, and memory usage were measured. This is not a one-time cost and occurs each time a user requests a new dataset. Therefore, it is far more important that this metric is lower than the tests evaluating Objective 1.

To evaluate Objective 3, the dataset summary and statistics packages within the platform were benchmarked. While this result is highly variable between use-cases, success in this test demonstrates that the data is readily available for use in analysis packages. This test was performed for 1, 5, 10, 50, 100, 500, and 1000 unique variables, using the datasets generated in the previous test. Response time, CPU, and memory usage were measured.

B. Results

Testing used up to three machines (nodes) with 8-Core 2.3GHz CPUs, and 6GB of allocated RAM, all throttled with a network speed of 100Mbps both for upload and download. Each node had 100 worker modules running. Fig. 6 shows the dataset ingestion test scaling up the number of patients to ingest for a single node. Fig. 6(a) shows that the process finishes in linear time, with 1000 patients processed in just over 4 minutes. This ingestion and translation process is a one-time cost in the system, and after, those patient records can be used repeatedly on multiple projects. Fig. 6(b) and (c) show that the system is linear in CPU and constant in memory, respectively.

Fig. 7 illustrates the horizontal scalability of the system, with the number of ingested patients held constant at 1000. Fig. 7(a) shows a decrease in computation time with increased nodes. However, the long tails in Fig. 7(b) indicate that the process of ingesting and translating data took longer than acquisition. This can be explained by a rate limit on write operations for the database resource that the platform was provisioned. Nevertheless, the expectation is that when unconstrained, these processes will benefit from a decreased run-time. Like the previous test, Fig. 7(c) shows that the system is constant for memory, consistent with the design as stateless services.

In its benchmarked deployment, MERLIN processed approximately 15,000 patients per hour. By comparison, typical processes can take weeks to months. MERLIN’s processing capacity can be expanded with increased resources.

These tests suggest that previously stated Objective 1 has been accomplished for the needs of a general use case. In these tests, a longer ingestion time for a large number of patients is acceptable because it is a one-time cost, whereas the subsequent steps should happen faster.

This is demonstrated in Figs. 8 and 9. Fig. 8 benchmarks the dataset generation process. In this test, the number of computed variables is scaled up. The test was conducted holding the number of output rows constant, 100,000 synthetic entries across 80,000 synthetic patients in each test (simulating 1.25 events of interest per patient), despite an increasing number of variables. The runtime, CPU, and memory usage all scale linearly. With 100 computed variables, the generation time is only slightly over 30 seconds. Indeed, even with 1000 computed variables, this process takes 2.5 minutes. By comparison, prior manual methods could take days to weeks to properly regularize data and build custom pipelines for each project, depending on data formats and quality. This suggests that Objective 2 has been accomplished.

Fig. 9 shows the benchmark test for the preliminary summary and statistics packages to run in the analysis pipeline. This confirms that the datasets generated are readily available for analysis, given that they are stored as CSVs with numerical
Fig. 6. Benchmark of the ingestion process for across a single node. In this test, the number of synthetic patients is varied while the number of compute nodes is held constant.

Fig. 7. Benchmark of the ingestion process across multiple nodes. In this test, the number of synthetic patients is held constant at 1000 patients.

Fig. 8. Benchmark of the dataset generation process.

Fig. 9. Benchmark of the dataset analysis process.
representation or NPY files. This test shows that runtime, CPU, and memory all scale linearly, although the CPU usage is subject to fluctuation based on the dataset’s content. Even for 1000 computed variables, this process took only 45 seconds, and for 100 computed variables, a mere 7 seconds. This suggests that Objective 3 has been accomplished.

VIII. DISCUSSION

We present an interactive platform, MERLIN, that relies on atomic data representation, a scalable microservice architecture, and methodology to accelerate data acquisition, dataset development and analysis, and AI model development. Every component of the platform architecture is a containerized module that performs small-scale individual tasks without cross-module dependencies. This reduces bottlenecks and enables parallelization to take advantage of cloud-computing resources and distributed workloads.

One of the main advantages of the MERLIN platform is that it can perform data ingestion, processing, and analysis in near real-time, which is achieved by parallelization both within and between worker module types. This includes multiple copies of the same module performing the same tasks in parallel, such as interacting with the EHR database, as well as multiple different types of modules performing different steps at the same time, such as ingestion and analysis. The short turnaround times for dataset generation and analysis mean that cohorts can be developed and analyzed rapidly.

The more traditional approach for cohort generation available in large academic centers is utilizing institutional pipelines, such as i2b2 [8], which can provide project-specific data marts, and the processing times may take up to several weeks. In addition, if the parameters of the dataset need to be changed subsequently, the full process has to start from the beginning. If investigators do not wish to use these institutional pipelines, they are often only left with the option to query a data warehouse directly [12], [13], [17]: however, this process requires highly specialized skills such as expertise in SQL, and the permission to perform these tasks may be limited in time and data size. Therefore, by using a parallelization approach, the MERLIN platform allows researchers to apply their domain-specific knowledge to problems in near real-time and complete tasks in hours and days instead of months or even years.

The standardized nature of data storage in MERLIN means that even with the inclusion of new data sources, the dataset generation and analysis workflows do not require any change to accommodate the increased data available. Likewise, if the dataset generation functions or analysis modalities are improved, data need not be re-imported or processed to take advantage of new capabilities. Currently, most warehouses provide partially processed data [9], and individual users are required to perform complex ETL tasks for every project. For example, using ALLOfUs, the data for the height and weight of 182,740 participants required significant processing and harmonization [20]. As these variables are used for multiple projects, the same work will be repeated by multiple investigators.

To improve efficiency, we process the data to a high level of granularity before storing them in our database, thus making them readily available in a machine-learning-ready format. This allows us to perform the task of data processing only upon initial ingestion. As a result, dataset generation is also expedited. This approach has demonstrated resource- and time-efficient data management for multiple projects in other institutions and enabled a small informatics team to process a very large number of requests [17]. Additional data can easily be added without the need to re-design the whole data storage process [18]. Implementing an atomic data approach in large public data repositories would result in significant cost and time savings for multiple users and lower the barrier to entry in biomedical research.

MERLIN enables interactive and iterative research pipelines utilizing data from different sources. This platform has the further benefit that it is not specialized to a single task or workflow, which would necessitate rebuilding it for each new project. Instead, new interfaces or APIs can be developed to add new data. This modular design allows new functionalities, applications, and even third-party software to be added seamlessly, and the value of this approach has been demonstrated by others [10]. Currently, data analysis and AI model development are reserved for teams that include highly skilled statisticians, data scientists, and engineers. Moreover, while data analysis and ML tools are integrated into many platforms [11], [12], [13], [17], their users need significant technical expertise. Our approach lowers the barriers to entry and enables clinicians without an advanced computer science degree to operate model development pipelines. We incorporated common statistical tests and ML models via a clinician-oriented interface. Subsequent analysis is expedited by the atomic, ML-ready data design, which allows the data to be used in the models directly. This approach is resource- and time-efficient [18] and will enhance the democratization of data science.

One limitation of our platform is the omission of NLP-based models, which can serve to generate complex phenotypes [21]. The MERLIN framework allows for NLP model inclusion which could serve as the inner function of ETL modules referenced in Section V-B, so long as the output predictions from these models follow the outlined principles for data representation. Furthermore, the included logging and data linking would allow quick references between NLP models and data predicted by those models. In this fashion, NLP models could be incorporated into our framework to leverage their capabilities. NLP-based models were not included in the current implementation due to their fragility with small perturbations, new concepts, or concept drift [6], [22], which creates a more significant burden on the QA process in part due to the institution-specific clinical notes structure. For older long-form and unstructured text data, including these models may be necessary. Still, the number of usable patients gained through this approach may be a minimal addition to the study cohort and may not justify the development effort.

In addition to MERLIN’s generalizability over its input, it is both able to scale in large environments and can be performant in
resource-limited settings such as academia. This gives it the ability to work within the constraints of small research labs and grow with their needs. Our system is designed to utilize the available resources, handle more concurrent data streams, and run faster if migrated to larger compute centers. The platform can grow without many bottlenecks encountered in other solutions due to algorithmic modularity and separation of each task. Instead of one long, expensive job as found in monolithic architectures, tasks are divided into hundreds of smaller jobs, which may take advantage of hardware parallelization in a more resource-savvy way. Ultimately, this system may benefit from the infrastructure available in AWS, GCP, and Microsoft Azure.

MERLIN’s usability is not only due to its efficiency and scalability but also its attention to interfaces and integration. Key to the rapid and efficient use of the platform is the minimization of barriers in interfaces to collect data, generate datasets, and visualize data during the analysis process.

IX. Conclusion

We created a versatile platform to address diverse needs throughout the healthcare AI research pipeline. It is designed to facilitate data collection from multimodal sources, such as EHRs and live data from devices, into a high-fidelity centralized repository with rigorous quality control in ML-ready formats. This methodology enables the rapid creation, evaluation, and analysis of datasets drawn from this repository with version control and logging, which can be directly used to train models, discover relationships in the data and produce results efficiently. We anticipate that this approach will spearhead real-world AI model development and meaningfully improve healthcare delivery in the long run.

Acknowledgment

The work presented in this paper has been aided by numerous interns and students over the past two years. Most notably, Braden Eberhard, Jonathan Dong, and Sheridan Rea helped write the code. In addition, Dr. Kathryn Gray, Dr. Heidi Lightfoot, Logan Scoon, Alexi Gutierrez, Hassan Hassan, and Dr. Christopher Mukasa have helped in the multiple stages of data collection, interpretation, and translation.

References

[1] J. He, S. L. Baxter, J. Xu, J. Xu, X. Zhou, and K. Zhang, “The practical implementation of artificial intelligence technologies in medicine,” Nature Med., vol. 25, no. 1, pp. 30–36, Jan. 2019, doi: 10.1038/s41591-018-0307-0.

[2] D. Calandra and M. Favaretto, “Artificial intelligence to fight COVID-19 outbreak impact: An overview,” Eur. J. Social Impact Circular Economy, vol. 3, no. 1, pp. 84–104, 2020.

[3] S. Secinaro, D. Calandra, A. Secinaro, V. Muthurangu, and P. Biancone, “The role of artificial intelligence in healthcare: A structured literature review,” BMC Med. Inform. Decis. Mak., vol. 21, no. 1, Apr. 2021, Art. no. 125, doi: 10.1186/s12911-021-01488-9.

[4] P.-H. C. Chen, Y. Liu, and L. Peng, “How to develop machine learning models for healthcare,” Nature Mater., vol. 18, no. 5, pp. 410–414, May 2019, doi: 10.1038/s41563-019-0345-0.

[5] T. Panch, H. Mattie, and L. A. Celi, “The ‘inconvenient truth’ about AI in healthcare,” NPJ Digit. Med., vol. 2, 2019, Art. no. 77, doi: 10.1038/s41746-019-0155-4.

[6] M. Ghassemi, T. Naumann, P. Schulam, A. L. Beam, I. Y. Chen, and R. Ranganath, “A review of challenges and opportunities in machine learning for health,” AMIA J. Summits Transl. Sci. Proc., vol. 2020, pp. 191–200, 2020.

[7] D. S. Char, M. D. Abràmoff, and C. Feudtner, “Identifying ethical considerations for machine learning healthcare applications,” Amer. J. Bioeth., vol. 20, no. 11, pp. 7–17, Nov. 2020, doi: 10.1080/15265161.2020.1819469.

[8] S. N. Murphy et al., “Serving the enterprise and beyond with informatics for integrating biology and the bedside (2B2),” J. Amer. Med. Inform. Assoc., vol. 17, no. 2, pp. 124–130, Apr. 2010, doi: 10.1136/jamia.2009.00893.

[9] J. M. Overhage, P. B. Ryan, C. G. Reich, A. G. Hartzema, and P. E. Stang, “Validation of a common data model for active safety surveillance research,” J. Amer. Med. Inform. Assoc., vol. 19, no. 1, pp. 54–60, Feb. 2012, doi: 10.1136/amiajnl-2011-000376.

[10] V. M. Castro et al., “The mass general Brigham biobank portal: An i2b2-based data repository linking disparate and high-dimensional patient data to support multimodal analytics,” J. Amer. Med. Inform. Assoc., vol. 29, no. 4, pp. 643–651, Mar. 2022, doi: 10.1093/jamia/ocab264.

[11] The i2b2 transSMART foundation, 2020, Accessed: Nov. 03, 2022. [Online]. Available: [i2b2transmart.org](http://i2b2transmart.org).

[12] C. Sudlow et al., “U.K. biobank: An open access resource for identifying the causes of a wide range of complex diseases of middle and old age,” PLoS Med., vol. 12, no. 3, Mar. 2015, Art. no. e1001779, doi: 10.1371/journal.pmed.1001779.

[13] All of Us Research Program Investigators et al., “The ‘All of US’ research program,” New England J. Med., vol. 381, no. 7, pp. 668–676, Aug. 2019, doi: 10.1056/NEJMsr1809937.

[14] A. Sammani et al., “UNRAVEL: Big data analytics research data platform to improve care of patients with cardiomyopathies using routine electronic health records and standardised biobanking,” Netherlands Heart J., vol. 27, no. 9, pp. 426–434, Sep. 2019, doi: 10.1016/j.OTHJ.2019.09.004.

[15] L. Ehwerhemuepha, G. Gasperino, N. Bischoff, S. Taraman, A. Chang, and W. Feaster, “HealthDataLab - a cloud computing solution for data science and advanced analytics in healthcare with application to predicting multi-center pediatric readmissions,” BMC Med. Inform. Decis. Mak., vol. 20, no. 1, Jun. 2020, Art. no. 115, doi: 10.1186/s12951-020-01153-7.

[16] A. Arora et al., “ISTHMUS: Secure, scalable, real-time and robust machine learning platform for healthcare,” Oct. 2019. Accessed: Jul. 29, 2022. [Online]. Available: http://arxiv.org/abs/1909.13343.

[17] S. Visweswaran et al., “An atomic approach to the design and implementation of a research data warehouse,” J. Amer. Med. Inform. Assoc., vol. 29, no. 4, pp. 601–608, Mar. 2022, doi: 10.1093/jamia/ocab204.

[18] V. Ranganath et al., “A methodological approach for integrating a research data warehouse linking disparate and high-dimensional patient data to support multimodal analytics,” J. Digit. Imag., vol. 36, pp. 700–714, 2023, doi: 10.1007/s12069-022-01649-0.

[19] V. Kovacheva, B. Eberhard, R. Y. Cohen, M. Maher, R. Saxena, and K. J. Gray, “Prediction of preeclampsia from clinical and genetic risk factors in early and late pregnancy using machine learning and polygenic risk scores,” medRxiv, doi: 10.1101/2023.02.03.23285585.

[20] M. S. Khan and R. J. Carroll, “Inference-based correction of multi-site height and weight measurement data in the all of US research program,” J. Amer. Med. Inform. Assoc., vol. 29, no. 4, pp. 626–630, Mar. 2022, doi: 10.1093/jamia/ocab251.

[21] J. Wong, M. M. Horwitz, L. Zhou, and S. Toh, “Using machine learning to identify health outcomes from electronic health record data,” Cureus Epidemiol. Rep., vol. 5, no. 4, pp. 331–342, Dec. 2018, doi: 10.7205/cureus.0000165-9.

[22] C. Aspillaga, A. Carvallo, and V. Araujo, “Stress test evaluation of transformer-based models in natural language understanding tasks,” Mar. 2020. Accessed: Jul. 29, 2022. [Online]. Available: http://arxiv.org/abs/2002.06261.