Leveraging the Cloud for Intelligent Clinical Data Registries

Marty Humphrey
Department of Computer Science
University of Virginia
Charlottesville, Virginia, USA

Vincent Lin
Department of Computer Science
University of Virginia
Charlottesville, Virginia, USA

Shweta Notani
Department of Computer Science
University of Virginia
Charlottesville, Virginia, USA

Jose Mattos
Department of Otolaryngology - Head and Neck Surgery
Division of Rhinology and Endoscopic Sinus Surgery
University of Virginia
Charlottesville, Virginia, USA

ABSTRACT

Public cloud platforms provide an amazing set of capabilities, but it can be an overwhelming challenge to create a design, implementation, and deployment that properly leverages today's existing public cloud capabilities while not precluding the use of near-future new services and infrastructure. We tackle this challenge in the context of clinical data registries, and create *Cloud-based Patient Outcomes Platform (CPOP)*, our scalable public cloud application for clinical patient data. Doctors are able to visualize collected medical data in different chart formats and patients are able to check their data and submit medical survey forms. The specific domain of interest in this paper is Chronic Rhinosinusitis (CRS), a largely under-recognized chronic disease in our society. The primary barrier to quality improvement in CRS is the difficulty in collecting data from patients, tracking appropriate follow-up time intervals, and analyzing outcomes results in a prospective and ongoing fashion. We describe key aspects and design experiences of CPOP-CRS in Amazon Web Services. We also provide quantitative evaluation of a key feature of CPOP-CRS, which is the ability of CRS doctors to upload an audio clip of a doctor-patient interaction, and have the cloud render a text-based representation, and show a word error rate of 15.6%. We outline next steps in the development of the CPOP/CPOP-CRS, and provide guidance for other users considering the public cloud for their next parallel and cloud-based Bioinformatics and Biomedicine project.

CCS CONCEPTS

• **Computer systems organization** → **Cloud computing**; • **Applied computing** → *Health care information systems.*

KEYWORDS

cloud computing, Amazon Web Services, clinical data registries

ACM-BCB '19, September 7–10, 2019, Niagara Falls, NY, USA

ACM ISBN 978-1-4503-6666-3/19/09…$15.00
https://doi.org/10.1145/3307339.3343464

ACM Reference Format:
Marty Humphrey, Vincent Lin, Shweta Notani, and Jose Mattos. 2019. Leveraging the Cloud for Intelligent Clinical Data Registries. In *10th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics (ACM-BCB '19), September 7–10, 2019, Niagara Falls, NY, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3307339.3343464

1 INTRODUCTION

Intuitively, it may appear to be relatively easy to build applications for "the cloud". Amazon, Microsoft, and Google, among others, have created *public* clouds, (Amazon Web Services (AWS) [4], Microsoft Azure [15], and Google Cloud Platform [6], respectively), in which essentially anyone with the equivalent of a credit card can rent computing infrastructure on demand. These public clouds are in constant competition with each other to develop new services and to improve end-user experiences. As a result, public clouds today provide a rich set of capabilities upon which to create a wide set of complex applications.

In reality, cloud applications are often much more difficult to design, implement and test than their non-cloud counterparts. For example, a rapidly-evolving infrastructure facilitates, for better or for worse, a software design challenge of "because they can, they should" (e.g., *should* scale, *should* support cross-domain identity and authorization, *should* integrate intelligent machine learning algorithms and voice control). This expansive nature of many potential requirements of cloud applications is too often overwhelming and can lead to suboptimal application design and implementation.

In this project, we tackle this challenge in the context of clinical data registries, in which multiple clinics and physicians store and analyze their patient data in a virtual centralized manner. The collective data can be studied to determine emerging patterns and/or establish best practices for improving patient care and outcomes. A critical requirement is to securely store data in accordance with environmental standards (e.g., HIPAA [21]), whereby authentication and authorization requirements are satisfied without sacrificing ease-of-use requirements of multiple stakeholders.

In this paper, we describe our scalable web application for doctors and patients that leverages cloud computing services to collect and visualize patient clinical data. Our general cloud-based architecture is entitled *Cloud-based Patient Outcomes Platform (CPOP).* CPOP supports different features for doctor and patient accounts. Doctors are able to visualize collected medical data in different chart formats and submit audio files for transcription. Patients are able to check

their data and submit medical survey forms. We discuss the overall architecture of our system as well as individual components and the role of cloud computing services within them.

The specific domain of interest in this paper is Chronic Rhinosinusitis (CRS) [13], a largely under-recognized chronic disease in our society. In North America, the estimated prevalence of CRS is 10% to 15%, similar to chronic low back pain and adult asthma. CRS patients suffer from daily facial pain and pressure, inability to breathe through their nose, thick and discolored nasal drainage, and decrease in their sense of smell. CRS also has a significant economic impact, with annual direct and indirect costs estimated between $1 billion and $2.3 billion respectively. The primary barrier to quality improvement in CRS is the difficulty in collecting data from patients, tracking appropriate follow-up time intervals, and analyzing outcomes results in a prospective and ongoing fashion. In current practice this requires a significant amount of time and research resources, which the vast majority of providers do not possess. In addition, it can be extremely difficult to share data and/or insights amongst CRS physicians because of regulatory and other constraints.

In this paper, we utilize our general CPOP architecture to create an application focusing on CRS – the application is called CPOP-CRS. We present the specifics of prototyping CPOP-CRS in Amazon Web Services (AWS), including an early evaluation of the speech-to-text capability in CPOP-CRS that leverages the recently-launched Amazon Transcribe service [3], observing a word error rate of 15.6%. We outline next steps in the development of the CPOP/CPOP-CRS, and provide guidance for other users considering the public cloud for their next parallel and cloud-based Bioinformatics and Biomedicine project.

The paper is organized as follows. Section 2 describes the benefits of a cloud-based clinical data registry. Section 3 provides more details on Chronic Rhinosinusitis (CRS), and specifically how CRS treatment could benefit from a cloud-based clinical data registry. Section 4 describes the CPOP system design, including the assumptions and requirements. Section 5 details a number of important CPOP use-cases. Section 6 contains the details of the prototype implementation of CPOP-CRS. Section 7 concludes.

2 BENEFITS OF A CLOUD-BASED CLINICAL DATA REGISTRY

In general, the practice of robust, prospective quality improvement for individual providers caring for patients can be challenging. Currently, typically, if a doctor wishes to track patient outcomes she must ask patients to complete patient reported outcome metrics (PROMs) while waiting in the doctor's office. The doctor must then manually input the PROM scores into some type of data base software for storage. When a doctor wants to know the outcomes of his patients, the data must then be imported into a statistical software package for analysis. Considering that a busy clinician can see hundreds of patients every month, this task is overly burdensome and rarely attempted. If one then wishes to track patients over time, the doctor must keep track of the date the patient had treatment and then obtain follow up questionnaires at the appropriate time interval which further complicates the matter.

A cloud-based system simplifies and streamlines the process of patient data collection to the point where the work and time burden would be minimal for both the physician and patient. (See [11] for an overview of the potential and open issues for using cloud computing for health informatics). In this scenario, when a doctor sees a patient in their office, and the patient is a candidate for treatment, the doctor enters all of the necessary basic information into a system. The goal at this point is to pare down the information required so as to minimize physician burden. Furthermore, the interface would be customizable for the individual doctor (e.g., interactive forms, spreadsheets, etc.) in order to facilitate data entry. Once the basic information is entered by the physician, then the patient is able to input data into the cloud. Similarly, with a cloud-based application, the patient is able to enter information outside the confines of the doctor office. The interface can be customized for the patient (e.g., mobile app or website, automatic via wearable device, browser-based forms, etc.). The application could track when a patient had treatment and trigger a follow-up evaluation at the appropriate time interval.

A cloud-based application could immediately perform doctor-specific and patient-specific analysis and present the information back to the doctor and patient securely. This eliminates the need for individual providers to analyze their own data, as this service is provided automatically by the cloud. An individual provider can immediately know how the outcomes of his patients compare to the aggregate outcomes of all users. Similarly, a patient can understand how her outcomes compare to all of the other patients whose data is captured by the application. Data privacy and protection is of utmost importance. Even in the cloud, with proper security procedures, only the individual who is entering information into the cloud can know the results of his own data analysis. That is to say that an individual, whether a doctor or patient, can know how his outcomes compare to the aggregate cloud data, but other cloud users cannot know any other individual's data. In this way individual data and privacy is protected in a manner that still allows for meaningful comparisons and quality improvement.

Perhaps the most significant outcome via a cloud-based approach involves analysis based on multiple patients and/or multiple doctors. Patient privacy and confidentiality will be ensured through a combination of state-of-the-art security design principles, analysis and mechanisms (e.g., strong cryptography, best practices for de-identifying patient data, etc.) With such security in place, a doctor is able to see aggregate analysis on their patients. A cloud-based system has the potential to discover and learn of similar data/patients being monitored by other doctors – in this situation, the doctor can be alerted to the possibility of combining analysis over a larger population (securely and subject to patient approvals).

Our project shares many goals and design principles with other projects, but with key differences. For example, a proof-of-concept design is described in [17] that focuses on attaching unobtrusive sensors to existing medical devices, whereby these sensors relay their data to a cloud. This project originated before the recent advances in public cloud computing, so it is not clear how it would map to individual public cloud services and/or meet application requirements in general (the same could be said about our own earlier work [8]). An Internet-of-Things (IoT) approach to health informatics has significant transformative potential and is the focus of many

projects (e.g., [7, 23]); our project is more focused on human-in-the-loop data collection and analytics. SAFTINET [19] is a federated architecture for distributed query processing; however it is based on a Grid style of computing rather than a virtually-centralized cloud architecture. Similar projects exist [9, 22]. OpenSpecimen [14] is representative of open source projects aimed at similar goals but not necessarily with the intent of generalizability across multiple topics/scopes.

3 WHY CHRONIC RHINOSINUSITIS (CRS)?

CRS is an important and largely underreported issue for quality of life worldwide, and concerns exist regarding quality of care and treatment outcomes in CRS. The treatment of CRS consists of initial medical therapy, usually in the form of topical and systemic medicines like antibiotics and corticosteroids. Rhinosinusitis is the sixth leading cause of outpatient physician visits and is the diagnosis responsible for the most antibiotic prescriptions in the United States [18]. Patients who fail to sufficiently improve on "appropriate medical therapy" are candidates for endoscopic sinus surgery (ESS). ESS has been shown to improve symptomatology, quality of life (QOL), and work productivity.

While the above tenets in the treatment of CRS are generally agreed upon, wide practice pattern variations exist for both the medical and surgical management of CRS. Currently, no standardized medical regimen for CRS exists so it is difficult to guage the efficacy or appropriateness of varying combinations, dosages, and durations of medications. Additionally, the level of available evidence for the different medical treatments for CRS varies widely and is not of the highest quality [12, 18]. There is also significant variation in the utilization of surgery for CRS, with greater geographic variation than gallbladder surgery, back surgery, and coronary artery bypass grafting [12, 18]. The presence of these large practice pattern variations raises concerns regarding the overall quality of care provided for this common inflammatory disease. Consequently, given the high prevalence of CRS and its large economic burden on society, the treatment of CRS should represent is a high priority target for quality improvement (QI) initiatives.

Significant barriers impede large scale, real world assessment of patient outcomes in CRS. The main outcomes of interest in CRS are patient reported outcome metrics (PROMs), which rely on patients completing questionnaires exploring their symptoms and QOL. Furthermore, obtaining serial measures from patients is of utmost importance in order to judge the impact of treatment over time. As such, recruiting patients into prospective studies, tracking patients longitudinally, and obtaining PROMs in a robust, timely fashion from multiple institutions can amount to an enormous and expensive task. One of the primary barriers to this type of data collection is that it requires significant research resources, in the form of clinical research coordinators/assistants and an integrated electronic data collection platform to make data collection feasible. While isolated institutions or departments might have such resources, most providers do not and hence are unable to contribute data or design QI initiatives. The difficulty of this endeavor is evidenced by the fact that, to date, no concerted QI initiative or centralized QI registry in CRS exists.

4 CPOP SYSTEM DESIGN

The first system design requirement for a cloud-based application is usually the ability to scale – with the number of users, size of data, frequency of patient interaction with the system, etc. CPOP is no different in this regard. As such, the foundation of CPOP's system design is the cloud design pattern for scalability shown in Figure 1. Interaction with the backend information system is regulated with a frontend load balancer, which directs client requests to one of the Web apps (running within a general Web server framework) to provide the fastest response. The diagram depicts the general situation whereby a user (doctor or patient) makes a request to CPOP that requires a non-trivial response; in this case the work request is enqueued for subsequent servicing by a separate collection of "Worker App" servers. While not shown in Figure 1, the "Web App" can interact directly with the backend storage system for less compute-intensive operations. Further performance enhancements could be explored as necessary (e.g., Redis [16]). **To simplify the discussion, in this section, we will assume the use of Amazon Web Services as the underlying public cloud, although certainly other public clouds would most likely provide a reasonable substrate. For example, the Work queue pattern of Figure 1 could be implemented via the AWS Simple Queue Service (SQS), but the Google Cloud Platform and Microsoft Azure have similar services**.

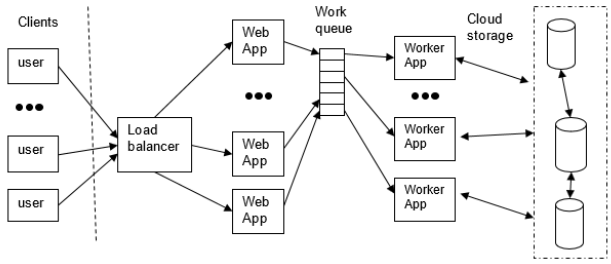

Figure 1: System Architecture for Scalable Cloud Application (used by CPOP)

This cloud application design facilitates independent scaling (and de-scaling) of Web-facing servers and back-end computational services. The number and size of the Web-facing services is primarily a function of the current (or anticipated) number/activity of the doctors and patients (and system administrators). The number and size of the "Worker App" servers are an indirect function of the activity of the Web-facing services and any periodic data analysis jobs that are needed by the CPOP application. Examples of periodic jobs include both application-centric jobs such as a holistic analysis of all patients on a particular drug/treatment as well as non-application-centric jobs such as general data integrity/completeness jobs ("is the data consistent across all dimensions?") and security-analytic jobs ("is there any evidence of intrusion and/or unauthorized data access?"). Note that ensuring system integrity and patient privacy is further discussed in Section 4.2, below.

A public cloud platform is chosen in CPOP in part so that the computation and data resources to fulfill the application goals and requirements can be selected and engaged dynamically and best-suited for the given computational need. Although not shown in

Figure 1, the architecture leverages underlying public cloud capabilities designed for high-availability and high-durability. This includes multi-region automatic replication of stored data. This architecture supports the widest range of computational abstractions upon which to fulfill the application goals and requirements. For example, the initial prototypes should utilize Infrastructure-as-a-Service (IaaS) mechanisms and abstractions (i.e., virtual machines as the Web Apps and the Worker Apps). However, Platform-as-a-Service (PaaS) approaches, container-based approaches (e.g., Docker containers), and "cloud function"/"serverless'' approaches (e.g., AWS Lambda) could be utilized in subsequent design phases to both speed development time and reduce deployment cost as new requirements emerge.

Software components that could be utilized for the Web-facing functionality include Node.js and Angular. Hadoop and Spark are among the technologies used for back-end data analytics.

4.1 Information Model

An important design principle for many successful software applications is simplicity and/or complexity management. The key data objects of CPOP design and development are:

- Patient: name, age, gender, DOB, etc., Doctor overseeing participation in study, and PROMs
- Doctor: name, patients in study
- PROM: patient, PROM date, PROM type, PROM data

Additional objects/fields can be introduced as necessary beyond this core functionality. The information model and its implementation should be architected for configurability and extensibility to facilitate adoption in other computing applications/scenarios.

To ensure patient privacy, the Patient record should be split, with cloud-resident patient-identifying data being replaced with a token generated by CPOP for use only within CPOP. The mapping from specific user to this token should exist only with the doctor of record for each patient. This precise mechanism should be designed during the very beginning of any CPOP instantiation to meet the security and privacy requirements of the participating enterprises. Conceptually, this mechanism is the equivalent of secure enterprise-resident storage system behind enterprise security mechanisms.

4.2 Security Architecture Design

A comprehensive end-to-end security solution is necessary for CPOP:

- Authentication: for humans (doctors, patients, system administrators), default multi-factor authentication (MFA) should be implemented (typically username/password and cellphone token and/or Google authenticator). All machines/software components will utilize SSL/TLS. In addition, the re-use of existing IDs should be explored – e.g., via AWS Identity and Access Management (IAM) and/or AWS Cognito.
- Authorization: arobust role-based, standards-based authorization infrastructure such be used (e.g., AWS Identity Management (IAM)) along with datastore protections (e.g., SQL protection mechanisms should be used in DB-based designs; similar security mechanisms should be used if/when a noSQL such as MongoDB is used)
- Logging and Auditing: In addition to traditional local/custom logging on a per-component basis, public cloud logging mechanisms (e.g., AWS CloudWatch) should be used for a systematic and integrated view of the end-to-end system.
- Encryption: SSL/TLS should be used end-to-end for confidential communications between components of the system. Data should be encrypted at rest within the cloud. Public cloud support for encryption keys (e.g., AWS Key Management System (KMS)) should be used.
- Network ports usage: Public clouds provide ample support to control network traffic. For example, AWS provides Security Groups that are used to control data into/out of a service based on port and type of traffic (e.g., HTTPS/443). These security groups should be used extensively, configured by default to only allow traffic that is necessary to enter/exit the software component in question.
- Intrusion Detection and Prevention: Application-specific measures should be designed and implemented at critical entry points into the system (e.g., member login page, access to back-end data, etc.) The system architecture should be monitored in part via specific public cloud services such as AWS CloudTrail, which provides secure logging and auditing for invocations of the AWS APIs. This should be used to monitor the running system to ensure that the current system status is as designed. If using AWS, AWS Config should be used to observe security groups and network configurations and to automate the mitigation of non-compliant settings and alert project staff.
- Virtual Private Network (VPN): A mechanism such as AWS Virtual Private Cloud (VPC) mechanisms should be used to isolate and regulate network traffic in addition to the port blocking mechanisms discussed above.

A requirement of the CPOP user interface design is to comprehensively validate data fields during the entering of new PROMs. This includes mechanisms to verify/validate entered text that, while syntactically valid, appear to be outside the norms of previous data input by the particular user. In this case, unexpected values are be confirmed via pop-up windows.

4.3 Performance Design

CPOP must provide quick responses to the end-users (doctors and patients). In a typical CPOP system design and implementation, it should be the goal that doctors can login and complete data entry within 2 minutes. That is, given the anticipated user community for CPOP, it is important the that *first* interactions provide a reasonable, but not perfect, user interface, else risk rejection from the potential user community. In subsequent versions that introduce patient-facing functionality, the goal should be to support patient login and patient data entry within 5 minutes. User studies and feedback should confirm/refine these goals throughout development.

The performance-monitoring capabilities of public clouds (e.g., AWS Cloudwatch) should be leveraged on a per-component basis to identify actual and potential bottlenecks and to ensure end-to-end performance goals are met. A CPOP system will automatically scale (see Figure 1) to minimize cost while still achieving performance goals. Auto-scaling an end-to-end cloud application is challenging

given that the ultimate performance is a user-perceived subjective metric (combining response time and user interface issues) that cannot be easily directly measured (but is rather often inferred from other observations, such as patient drop-out rate during the life of the application). Furthermore, the best auto-scaling system anticipates near-future resource capacity necessary to meet behavioral goals; such future requirements can be periodic and observed but are frequently incorrect. Therefore, whenever in doubt, the CPOP system should be conservative and overprovisioned to ensure the highest probability of providing a high-quality end-user experience.

The performance of the background (e.g., nightly) CPOP analytic jobs is unlikely to be a significant challenge, given the nature of the computation (i.e., they are not user-facing and/or interactive). Nightly and/or weekly reviews of the duration of these jobs will drive capacity changes for the next set of jobs.

The goal of CPOP in most situations is 99.9% availability and 99.999% durability. This goal is readily derived from the corresponding properties in the Service Level Agreements (SLAs) of the constituent cloud components being utilized in combination with the application-specific functionalities and algorithms of CPOP.

Redundancy and replication are built into many of the underlying cloud compute/data services and should be heavily utilized. The load balancer of Figure 1, as the main entry point, is (logically) a potential single point of failure. Public cloud history has shown that this service has not failed in the recent past. Additionally, by relying on such a heavily-utilized component of the public cloud, other concurrent users of this service are apt to make the public cloud provider aware of such failure (in the unlikely event that the public cloud has not discovered this already). This is another example of how CPOP has been designed to leverage the experience and general use of a shared computing infrastructure.

Secure backups of the CPOP data should occur on a nightly basis. These backups should be encrypted with a separate and unique encryption key (distinct from the keys used within the running CPOP). Backups should be replicated into multiple regions of the public cloud. If deemed necessary, these backup should be replicated to other clouds and/or enterprise servers.

5 CPOP USE CASES

5.1 New Patient Registration

The process to register a new patient in CPOP begins when the authorized member of the doctor's office logs into the platform to perform a tentative registration. This records basic information into the backend data store and generates a unique https URL that is emailed via a public cloud email server to the potential new patient. The patient then clicks on the URL to continue the registration. The security and best practices of this pattern have been well-established through its use in a large number of Web systems. During new patient registration, a unique identifier is generated that is forever used as the basis to tag patient-specific information in the cloud. As described in Sections 4.1 and 4.2, a split design will ensure the confidentiality and privacy of certain patient data. The browser-based user interface should be designed/implemented in a responsive web framework (e.g., Angular), which will run via the javascript engine in the client browser, communicating via the load balancer to a Web app (which simply may bypass the second layer of servers and interface directly to the back-end storage). Once all patient data has been stored, the public cloud email server should be utilized to send appropriate email to the doctor, who then follows a similar flow (and similar software components in the cloud) to complete the registration. Note that this protocol is designed to ensure that the physician has not put ANY patient protected information into a 3rd party system until consented to the patient.

5.2 New Patient Reported Outcome Metrics (PROMs)

The timely and correct collection of patient reported outcome metrics (PROMs) occurs when the patient logs into the CPOP portal and is taken to the data entry Web pages. A patient can initiate this themselves, but it is anticipated that most patient PROMs should be entered by the patient via CPOP sending email/SMS to the patient as a reminder (this would be generated directly/indirectly by one of the nightly analysis jobs in CPOP in the public cloud). Ease of use and data correctness/integrity are significant design considerations, especially considered an anticipated wide variety in "comfort level'' of a given patient with regard to information technology. The CPOP user interface for a particular person should learn and adapt to patient-specific preferences and behaviors. For example, as data is entered, CPOP will compare the new data with previous data entered by the same patient – e.g., a value that is outside the expected value for a generic patient might indeed be more typical of this particular patient (and thus not subject to an "are you sure?'' confirmation window). This functionality should be instrumented to monitor engagement properties over time, providing critical mechanisms by which to identify and refine problematic CPOP interfaces.

5.3 Doctor Reviews Patient Status

The CPOP Web functionality for a Doctor to review patient status (either a single patient or collectively) should be tailored to support its three main use-cases: [a] as requested via an nightly analytical job in the CPOP cloud backend; [b] as requested as a result of new PROMs, either via direct patient request or via real-time CPOP analysis at the time of patient input; [c] not as a direct result of new data, but rather as part of normal Doctor periodic review. After authenticating, the Doctor should be presented with information based on the particular use-case – the design will ensure efficient, correct, and secure analysis and input from the Doctor (e.g., the particular data of concern, the reason for concern, possible actions to take). Information/requests from the Doctor to a particular Patient should be recorded and facilitated in CPOP – such info becomes part of the overall data/information associated with the patient and usable for longitudinal studies. Doctors/patients will not be artificially forced to perform all interactions within the scope/mechanisms of CPOP; rather, such behavior should be expected and reflected in the data/information stored within CPOP. Any such synchronous/asynchronous interactions should be recorded and facilitated as appropriate by CPOP.

5.4 Patient Reviews Patient Status

A Patient must be given an enjoyable mechanism and user interface, broadly, in CPOP else a patient may discontinue entering new PROMs. A patient reviewing their status – either individually or in comparison to others – should be key to continued user engagement. This capability can be directly engaged at any time (i.e., not solely when a patient enters new PROMs). A significant challenge should be to determine and present only those other patients (anonymously) that are relevant to the particular patient. For example, there is significant risk of patient withdrawal in the use of CPOP if the patient is falsely compared – positively or negatively – to other patients who are misidentified as being in the same equivalence class as themselves. CPOP should provide context for comparisons and will provide mechanisms for patient-to-doctor communication tailored on the comparisons presented in the user interface. Furthermore, CPOP should learn and modify its behavior in this functionality to attempt to avoid such false positives/negatives, contextualized to both patient and doctor. The correct and appropriate functionality of this capability is arguably one of the most important features of CPOP.

5.5 Doctor Seeks Permission from Another Doctor

A key to the success of CPOP will be the volume of data held within CPOP. A patient's data will (by default) be securely and anonymously available to other patients and doctors (acknowledging the constraints and issues of CPOP and its operating environments). However, there will be situations whereby it might be important to de-anonymize certain patient data – and given that this mapping only exists with the patient's doctor, there will be situations whereby one doctor might need to engage another doctor regarding a particular patient. CPOP should facilitate at least the initial contact (e.g., by reference to the unique CPOP ID of the patient in question). Due to the complexity issues involved, the degree to which subsequent discussion/negotiation occurs in this dialogue is an open issue for CPOP.

6 IMPLEMENTATION OF CPOP-CRS

In this section, we describe our prototype implementation of CPOP as applied to CRS. We were given prototype, anonymized sample medical data. The data consisted of 100 patients with patient characteristics including age, race (White, African-American, Asian, Multiracial), ethnicity (Hispanic or not), gender, nasal polyp (yes/no), allergies (yes/no), asthma (yes/no), and aspirin sensitivity (yes/no). The dataset also included 22 factors including running nose, cough, sneezing, fatigue, and sadness that patients rated from 1-5, with 1 being no problem and 5 being very severe. This is the SNOT-22 score [10], and patients filled out this survey before and after sinus surgery. While the data set included other information including CT scans of sinal opacification, endoscopic scores, and a satisfaction survey of the surgery, we only visualize SNOT-22 factors based on patient characteristics for our proof of concept system because not all patients opted to undergo surgery.

Figure 2 shows the overall architecture for CPOP-CRS on top of Amazon Web Services (AWS). The typical workflow for CPOP-CRS begins with the user going to a login/sign up site provided

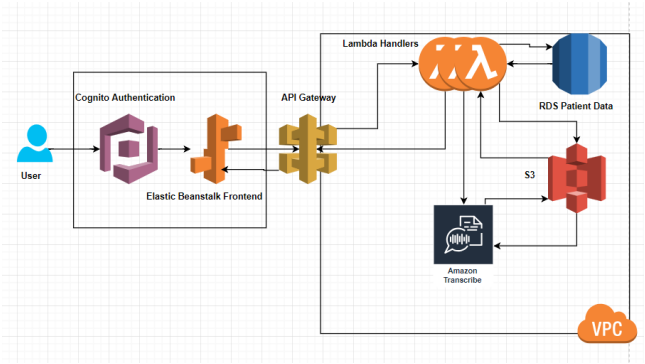

Figure 2: AWS-based System Architecture for CPOP-CRS

by Amazon Cognito. Upon successful login, the user will receive a JSON Web Token (JWT) from Cognito, which is used to determine which pages the user can access and what API endpoints can be called. Many actions including submitting survey data, visualizing existing patient data, submitting an audio file for transcription, and retrieving a text transcription are performed by calls to a RESTful API on API Gateway. The following sections discuss components and performable actions in more detail.

6.1 Database

In order to better integrate with the AWS cloud services we are using in our system and to take advantage of managed scaling, we decided to put our data into a cloud database. Using the database product guide provided by Amazon [5], we decided to use Amazon Relational Database with a MySQL engine over noSQL alternatives such as DynamoDB, Elasticache, and Neptune because our original data has a strict schema and is best described relationally. We did not choose Amazon Aurora because it seemed too experimental. Our original data fit nicely into our database, but we did rename SNOT-22 columns from undescriptive headers such as SNOT1 and SNOT2 to more meaningful column names such as sneezing and coughing. For security purposes, we made our database accessible to only resources within the Virtual Private Cloud (VPC) it is contained in.

6.2 Cognito

We use Amazon Cognito for simple and secure user sign-up, sign-in, and access control. We first created a user pool for storing valid users of CPOP-CRS. We then integrated sign-in and sign-up pages that Cognito provides into our application. Users are able to sign up with an email, password, name, and optional phone number. Users who sign up using the sign-up page are automatically assigned a patient role. We use a manual method to create doctor accounts since we do not want an unknown user to register as a doctor. Once a user has successfully signed up, they can log into the app. Cognito will provide the logged in user with a JWT, which is checked by the app to display different pages to different roles. Furthermore, before data-sensitive API calls, the app will check the token for the correct role before invocation. We hash the username associated with the JWT and provide this value to the get patient data API method to allow patients to check their current medical data.

6.3 API

Since our RDS database is not accessible publically, we need to create an API within the VPC that is exposed to the public. We decided to create a RESTful API using AWS Serverless Application Model (SAM) [2], a framework that simplifies building serverless applications on AWS. SAM packages up our function handlers and dependencies, and it converts our configuration YAML file into a CloudFormation template. The CloudFormation template in turn creates a stack by creating Lambda functions for all our packaged handlers and setting up a corresponding API in API Gateway with the routes we defined.

By using SAM, we were able to save time by avoiding manually creating Lambda functions and setting up API Gateway. Our design of creating such an API makes the project more robust as we can use the same data interaction model even if we change the user interface. Figure 3 describes our API. We note that we needed to configure Cross Origin Resource Sharing and have the back end return the necessary headers in order to get POST requests to work.

API Method	HTTP Method	Route	Parameters	Returns
Get patient data	GET	/patient/:userID	:userID – hash of Cognito username	JSON of the specified patient's data
Get chart data	GET	/chart/:type/:symptom/:characteristic	:type-baseline or follow-up :symptom-SNOT symptom :characteristic-patient characteristic	Google Chart friendly JSON of requested data
Submit data-Insert new patient data to RDS or update if data already exists	POST	/patient/:userID	:userID – hash of Cognito username JSON with new patient data	Success/error
Audio upload-upload audio and start Transcribe	POST	/upload	Audio file binary data, number of speakers, and username hash	Success/error
Get transcriptions	GET	/transcription/:userID	:userID – hash of Cognito username	JSON containing transcription file names for user
Get transcription	GET	/transcription/:userID/:file	:userID :file – file name	JSON of transcription

Figure 3: CPOP-CRS API

6.4 Data Visualization

In order to allow doctors to look at trends and gain insights, we support visualization of data shown in Figures 4 and 5.

We have drop down options for the data type (before or after surgery), symptom (the SNOT-22 factors), and the seven patient characteristics described in the introduction. These drop downs use AJAX calls to our API to retrieve the relevant data from our RDS database and dynamically render charts. Since we use Google Charts, the Lambda handlers of the API process and format data from SQL queries to match the format Google Charts expect. We support column, bar, line, and pie charts (note that pie charts do not split the data by patient characteristics).

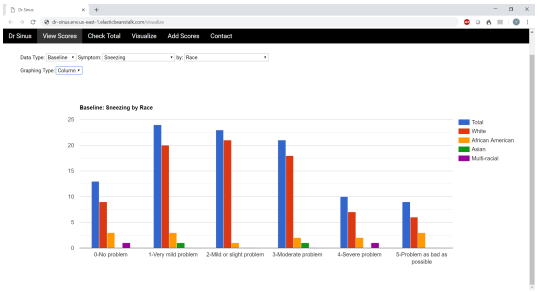

Figure 4: CPOP-CRS Column Graph Visualization

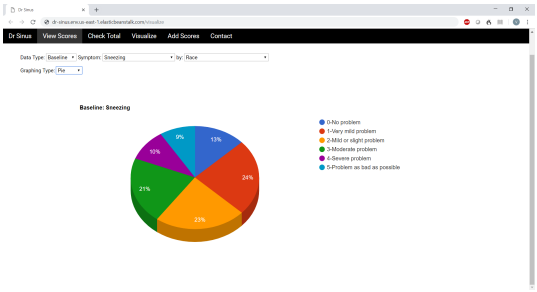

Figure 5: CPOP-CRS Bar Graph Visualization

6.5 Audio Transcription

With the popularity of electronic health records, it is likely that a doctor may want a transcript of conversations with patients. We assume a doctor already has an audio file to be transcribed. The doctor can use the form provided on CPOP-CRS to submit an audio file to be transcribed and specify the number of speakers (up to 10). This action calls our API which uploads the file to S3 and then runs a script to start an Amazon Transcribe job on the file. Transcribe then writes a transcript JSON file back to S3. Going to the Transcripts page provides a drop down menu with the names of transcription files for the current user. Selecting a transcript retrieves the transcript from S3, runs it through a Javascript script to parse into human readable format, and displays the transcript with speaker identification as applicable.

As a new cloud service released in April 2018 [3], Transcribe is difficult to use programmatically because of sparse documentation. Furthermore, we were unable to find any evaluation of the accuracy of transcription. Qualitatively, we noticed from our usage that Transcribe had difficulties in transcribing punctuation including commas and periods. We more empirically evaluate Transcribe by using the popular metric word error rate (WER), defined in Equation 1 where S is the number of substituted words, D is the number of deleted words, I is the number of inserted words, and N is the total number of words.

$$WER = \frac{S + D + I}{N} \tag{1}$$

We recorded short (less than 5 minutes) single speaker speeches in an environment without background noise and ran Transcribe on the samples. We then removed punctuation, converted both the ground truth and transcription to lower case, and used Martin

Thoma's WER calculation Python script on the data [20]. The results of five representative interactions are shown in Figure 6 with an average WER of 15.6%.

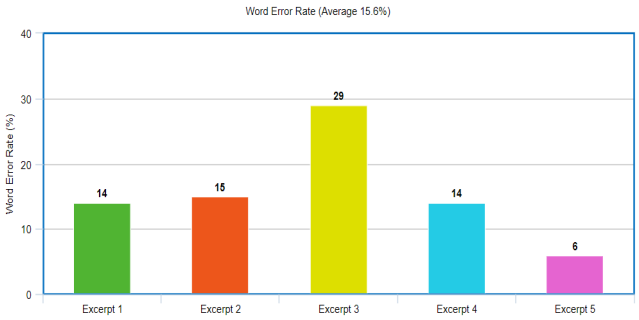

Figure 6: Word Error Rate for 5 Transcripts

We also noticed that Transcribe is slow as it took 1 minute to transcribe a 30 second clip, 2 minutes for a 45 second clip, 5 minutes to transcribe a 3 minute clip, and 10 minutes to transcribe a 30 minute audio clip. It was confirmed by an Amazon developer that Transcribe is optimized for longer audio segments [1].

We also investigated streaming transcription and only found minimal Java documentation. While we were able to get a standalone Java streaming transcription application to work, we do not integrate it into CPOP-CRS currently.

7 CONCLUSION

The cloud facilitates a diverse set of applications, but designing and implementing a software cloud project will remain challenging for the near future. In this paper, we argued that the public cloud is an important substrate for health informatics applications. We presented CPOP, a Cloud-based Patient Outcomes Platform. In addition, we described how we prototyped the CPOP architecture for CRS (CPOP-CRS). CPOP-CRS supports user authentication and authorization, allows retrieving and updating medical information from a cloud database, and has managed load balancing and auto recovery. Patients are able to check and update their medical data while doctors can visualize data in a variety of charts. Doctors can also upload audio files for transcription with identification of speakers. We also evaluated Amazon Transcribe and found it to have a word error rate of 15.6%. In the future we will expand the capabilities of CPOP (and CPOP-CRS) to further leverage emerging capabilities in public clouds.

Our next steps are to utilize CPOP-CRS in a real clinical setting. Inevitably, new feedback and requirements will arise that will further the project goals and ambitions.

ACKNOWLEDGMENTS

The authors thank Akanksha Nichrelay and Arjun Malhotra for contributions to earlier versions of this work.

REFERENCES

[1] Amazon, Inc. 2018. Transcribe is very slow. Retrieved June 20,2019 from https://forums.aws.amazon.com/thread.jspa?messageID=854301

[2] Amazon, Inc. 2018. What Is the AWS Serverless Application Model (AWS SAM)? Retrieved June 20,2019 from https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/what-is-sam.html

[3] Amazon, Inc. 2019. Amazon Transcribe Now Generally Available. Retrieved June 20,2019 from https://aws.amazon.com/blogs/aws/amazon-transcribe-now-generally-available/

[4] Amazon, Inc. 2019. Amazon Web Services (AWS). Retrieved June 20,2019 from https://aws.amazon.com/

[5] Amazon, Inc. 2019. Databases on AWS. Retrieved June 20,2019 from https://aws.amazon.com/products/databases

[6] Google, Inc. 2019. Google Cloud Platform. Retrieved June 20,2019 from https://cloud.google.com/

[7] Moeen Hassanalieragh, Alex Page, Tolga Soyata, Gaurav Sharma, Mehmet Aktas, Gonzalo Mateos, Burak Kantarci, and Silvana Andreescu. 2015. Health monitoring and management using Internet-of-Things (IoT) sensing with cloud-based processing: Opportunities and challenges. In *2015 IEEE International Conference on Services Computing*. IEEE, 285–292.

[8] Marty Humphrey, Jacob Steele, In Kee Kim, Michael G Kahn, Jessica Bondy, and Michael Ames. 2013. CloudDRN: A Lightweight, End-to-End System for Sharing Distributed Research Data in the Cloud. In *2013 IEEE 9th International Conference on e-Science*. IEEE, 254–261.

[9] Rainu Kaushal, George Hripcsak, Deborah D Ascheim, Toby Bloom, Thomas R Campion Jr, Arthur L Caplan, Brian P Currie, Thomas Check, Emme Levin Deland, Marc N Gourevitch, et al. 2014. Changing the research landscape: the new York City clinical data research network. *Journal of the American Medical Informatics Association* 21, 4 (2014), 587–590.

[10] Joshua L Kennedy, Matthew A Hubbard, Phillip Huyett, James T Patrie, Larry Borish, and Spencer C Payne. 2013. Sino-nasal outcome test (SNOT-22): a predictor of postsurgical improvement in patients with chronic sinusitis. *Annals of Allergy, Asthma & Immunology* 111, 4 (2013), 246–251.

[11] Mu-Hsing Kuo. 2011. Opportunities and challenges of cloud computing to improve health care services. *Journal of medical Internet research* 13, 3 (2011), e67.

[12] Jose L Mattos, Charles R Woodard, and Spencer C Payne. 2011. Trends in common rhinologic illnesses: analysis of US healthcare surveys 1995–2007. In *International forum of allergy & rhinology*, Vol. 1. Wiley Online Library, 3–12.

[13] Mayo Clinic 2019. Chronic sinusitis. Retrieved June 20,2019 from https://www.mayoclinic.org/diseases-conditions/chronic-sinusitis/symptoms-causes/syc-20351661

[14] Leslie D McIntosh, Mukesh K Sharma, David Mulvihill, Snehil Gupta, Anthony Juehne, Bijoy George, Suhas B Khot, Atul Kaushal, Mark A Watson, and Rakesh Nagarajan. 2015. caTissue suite to OpenSpecimen: Developing an extensible, open source, web-based biobanking management system. *Journal of biomedical informatics* 57 (2015), 456–464.

[15] Microsoft, Inc. 2019. Microsoft Azure. Retrieved June 20,2019 from https://azure.microsoft.com/

[16] Redislabs 2019. Redis is an open source (BSD licensed), in-memory data structure store. Retrieved June 20,2019 from https://redis.io/

[17] Carlos Oberdan Rolim, Fernando Luiz Koch, Carlos Becker Westphall, Jorge Werner, Armando Fracalossi, and Giovanni Schmitt Salvador. 2010. A cloud computing solution for patient's data collection in health care institutions. In *2010 Second International Conference on eHealth, Telemedicine, and Social Medicine*. IEEE, 95–99.

[18] Luke Rudmik, Jose L Mattos, Janalee K Stokken, Zachary M Soler, R Peter Manes, Thomas S Higgins, Michael Setzen, Jivianne Lee, and John Schneider. 2017. Rhinology-specific priority setting for quality improvement: a modified Delphi study from the Quality Improvement Committee of the American Rhinologic Society. In *International forum of allergy & rhinology*, Vol. 7. Wiley Online Library, 937–944.

[19] Lisa M Schilling, Bethany M Kwan, Charles T Drolshagen, Patrick W Hosokawa, Elias Brandt, Wilson D Pace, Christopher Uhrich, Michael Kamerick, Aidan Bunting, Philip RO Payne, et al. 2013. Scalable Architecture for Federated Translational Inquiries Network (SAFTINet) technology infrastructure for a distributed data network. *EGEMS* 1, 1 (2013).

[20] Martin Thoma. 2018. Word error rate calculation. Retrieved June 20,2019 from https://martin-thoma.com/word-error-rate-calculation

[21] U.S. Department of Health & Human Services 2019. Health Information Prvacy. Retrieved June 20,2019 from https://www.hhs.gov/hipaa/index.html

[22] Jiawei Yuan, Bradley Malin, François Modave, Yi Guo, William R Hogan, Elizabeth Shenkman, and Jiang Bian. 2017. Towards a privacy preserving cohort discovery framework for clinical research networks. *Journal of biomedical informatics* 66 (2017), 42–51.

[23] Ya-Li Zheng, Xiao-Rong Ding, Carmen Chung Yan Poon, Benny Ping Lai Lo, Heye Zhang, Xiao-Lin Zhou, Guang-Zhong Yang, Ni Zhao, and Yuan-Ting Zhang. 2014. Unobtrusive sensing and wearable devices for health informatics. *IEEE Transactions on Biomedical Engineering* 61, 5 (2014), 1538–1554.