Scalable Quality Assurance for Neuroimaging (SQAN): Automated quality control for medical imaging

Arvind Gopu[a], Michael D. Young[a], Andrea Avena Koenigsberger[a], Raymond W. Perigo[a], John D. West[b], Meenakshisundaram Paramasivam[c], Soichi Hayashi[a], and Robert Henschel[a]

[a]Indiana University, 2709 E 10th St, Bloomington, IN, USA
[b]Center for Neuroimaging, IU School of Medicine, 355 W 16th St, Indianapolis, IN, USA
[c]Dept. of Radiology & Imaging Sciences (RADY), IU School of Medicine, 950 W Walnut St, Indianapolis, IN, USA

ABSTRACT

Medical imaging, a key component in clinical diagnosis of and research on numerous medical conditions, is very costly and can generate massive datasets. For instance, a single scanned subject produces hundreds of thousands of images and millions of key-value metadata pairs that must be verified to ensure instrument and research protocol compliance. Many projects lack funds to reacquire images if data quality issues are detected later. Data quality assurance (QA) requires continuous involvement by all stakeholders and use of specific quality control (QC) methods to identify data issues likely to require post-processing correction or real-time re-acquisition. While many useful QC methods exist, they are often designed for specific use-cases with limited scope and documentation, making integration with other setups difficult. We present the Scalable Quality Assurance for Neuroimaging (SQAN), an open-source software suite developed by Indiana University for protocol quality control and instrumental validation on medical imaging data. SQAN includes a comprehensive QC Engine that ensures adherence to a research study's protocol. A modern, intuitive web portal serves a wide range of users including researchers, scanner technologists and data scientists, each of whom approach QC with unique priorities, expertise, insights and expectations. Since Fall 2017, a fully operational SQAN instance has supported 50+ research projects, and has QC'd ∼3.5 million images and over 700 million metadata tags. SQAN is designed to scale to any imaging center's QC needs, and to extend beyond protocol QC toward image-level QC and integration with pipeline and non-imaging database systems.

Keywords: Quality Assurance, Automated Quality Control, Web Portal, node.js, AngularJS, Vue.js, MongoDB

1. INTRODUCTION

Medical imaging is integral to the clinical diagnosis and research of neurodegenerative diseases like Alzheimer's and Parkinson's, as well as brain cancers [1,2,3]. Medical imaging techniques – including Magnetic Resonance Imaging (MRI), Computed Tomography (CT), and Positron Emission Tomography (PET) – generate massive datasets in the order of tens to even hundreds of thousands of images for a single subject, and studies include hundreds or even thousands of subjects [4,5]. Further, imaging datasets from radiological scanners involve a complex hierarchical structure. Each subject in a research study or clinical trial undergoes one or more scan-sessions, each of which comprise several series of scans. Each of those series can contain thousands of images with dozens of metadata tags in each image.

Medical imaging is expensive, and many projects lack funds to reacquire images if quality issues are detected after the acquisition period. Quality assurance (QA) is a fundamental first step in guaranteeing reliable and reproducible medical imaging research [6,7,8]. Importantly, QA requires continuous involvement by all stakeholders, and the use of appropriate quality control (QC) methods that rapidly identify data in need of post-processing correction or re-acquisition. Statistical inaccuracies from incorrect imaging parameters, low image quality, scanner software updates [9], and motion artifacts [10] can all contribute to noise in data, leading to unreliable and

Further author information and inquiries, contact agopu@iu.edu.

non-reproducible results [6,7]. Further, because imaging data are often made available to the general scientific community, results may also impact downstream research [5].

While many useful quality control (QC) methods exist, they are often limited in scope or require time-consuming manual techniques [8,11,12]. Existing automated or semi-automated QC procedures tend to be complex and lacking in documentation; they are generally designed for specific use-cases (i.e. scanner-specific, image-specific), making integration with other setups difficult. Simple QC processes performed at some institutions include the following: 1) verifying that a small number of metadata tags have a certain value or range of values, via administrative configuration on imaging archival systems like XNAT or PACS [13, 14]; 2) verifying the criteria in item 1 via custom shell scripts or the like; 3) relying on spot checks of random metadata tags for a small sample of images belonging to a sample of subjects who were imaged for a particular research study. Automated and flexible QC on imaging protocols that supplement existing imaging QA/QC methods represent a big step forward in ensuring improved image quality; this is particularly critical for multi-center projects with heterogeneous data. [11,12,15] Here we present the open-source Scalable Quality Assurance for Neuroimaging (SQAN - pronounced "scan") software suite for automated, highly customizable, web-based protocol quality control and instrumental validation on medical imaging data.

1.1 Origins of SQAN: Project team background

In late 2015, the Indiana University Scalable Compute Archive (IU SCA) team and the Indianapolis-based Dept. of Radiology and Imaging Sciences (RADY) imaging center began discussing the latter's quality control needs for research and clinical trial datasets collected over time. Most of the data were archived on a PACS based setup while a subset of data were also archived on an XNAT instance. That coupled with the sheer number of metadata tags that must be evaluated to determine protocol compliance led us to conclude that manual QC checks of RADY datasets was implausible at scale. Existing QC methods we explored also required significant overhead or unsustainable manual steps. We identified the need for a system that ingests metadata automatically and applies quality control algorithms. We also determined the importance of a modern web portal user interface in making such a system viable, irrespective of the computer skills (or lack thereof) of the user base. This was important in the context of our RADY colleagues' research clients and staff technologists, as well as in the case of other medical centers who may adopt SQAN for their QC needs in the future. We recognized the need for a wide array of skills and experience in designing and developing an automated-yet-flexible, managed QC system for imaging protocols, including imaging operational expertise, computer science (software development & deployment skills), and neuroscience domain expertise.

The RADY imaging center is one of the leading medical imaging facilities in the Midwestern US; it has supported a large number of research studies and several clinical trials over the past 15 years. The center is equipped with multiple Siemens scanners for MRI, PET, and CT imaging modalities, and employs 6 professional technologists and dedicated information technology (IT) staff. The Scalable Compute Archive (SCA) - both the name of a software suite and the team that manages it - builds, delivers, and operates customized web user interfaces, secure data management systems, and integrated scientific software application pipelines. SCA systems - supporting astronomy* [16, 17, 18, 19, 20], IU's High Performance Computing (HPC) community at large†, neuroscience‡, dynamic image visualization [21], electron microscopy (IU EMCenter portal operational 2013-18, retired in 2019 [22]) - are securely accessible from any web-enabled device at any time. Several SCA projects have openly-available published papers§, and some of these projects are already open source. SCA team members have backgrounds spanning a broad spectrum (including computer science, neuroscience, astronomy, informatics, and Linux system administration), with 90+ person years of experience enabling scientific research at IU. Together, we deduced that the two teams possessed the skills necessary to collaborate successfully on designing and developing an automated, flexible, managed QC system for imaging protocols. SQAN is the end

*(a) ODI-PPA: portal.odi.iu.edu (b) GCS-SCA: gcs.sca.iu.edu (c) SpArc: sparc.sca.iu.edu (d) BDBS-SCA: bdbs.sca.iu.edu/

†(a) HPC everywhere: hpceverywhere.iu.edu (b) ImageX dynamic image visualization: imagex.sca.iu.edu

‡(a) SQAN demo: sqan.sca.iu.edu (b) Brainlife.io: brainlife.io (c) Connectivity pipeline: github.com/IUSCA/IUSM-connectivity-pipeline

§List of SCA publications: sca.iu.edu/publications

product of these two teams investing ∼3500 person-hours on software development and ∼1500 person-hours on collaborative design and testing.

Since Fall of 2017, a fully operational production SQAN service instance has supported the RADY imaging center, providing QC for their research projects; it has QC'ed 3.5 million images and validated 700 million metadata tags spanning MRI, CT, and PET modalities. In fall 2019, we transitioned the project from an IU-internal project to an open-source project hosted on GitHub[¶] with the longer-term goal of making it a self-sustaining project supported by a vibrant community of developers and maintainers. SQAN is built on the foundation of open-source software and platforms including GitHub [23], node.js [24], AngularJS [25], MongoDB [26], Vue.js [27 and docker [28]), and can benefit any imaging center. Currently, Harvard Medical School/Brigham and Women's Hospital (BWH) and Dartmouth-Hitchcock Medical Center are evaluating SQAN for their protocol compliance needs. A demo SQAN portal is available at `https://sqan.sca.iu.edu`.

We describe the technical design and architecture of SQAN in Section 2, including detailed design considerations associated with various facets of the software suite. In Section 3, we describe the features offered by the web portal, including usability considerations. We conclude this paper with a short summary of future goals we envision pursuing.

2. SQAN DESIGN & ARCHITECTURE

In this section, we describe the design and architecture of the SQAN software suite. Before we begin, it is worth expanding on the precise need for automated protocol QC, especially due to the the complex hierarchical structure of the imaging data acquired by radiological scanners. Research projects (and clinical trials) are usually assigned a unique identifier, referred to herein simply as *research*. Each subject in the research study undergoes one or multiple scan-sessions, here referred to as *exams*. An *exam* comprises several *series* of scans (with changes to scanner settings per series). Each of those *series* contains hundreds or thousands of images, and each of those images, in turn, contains dozens of points of metadata. Figure 1 offers a graphical representation of the complex structure of data produced by a single exam; a typical imaging center can run several thousand of such exams each year, illustrating the scope of QC for protocol compliance.

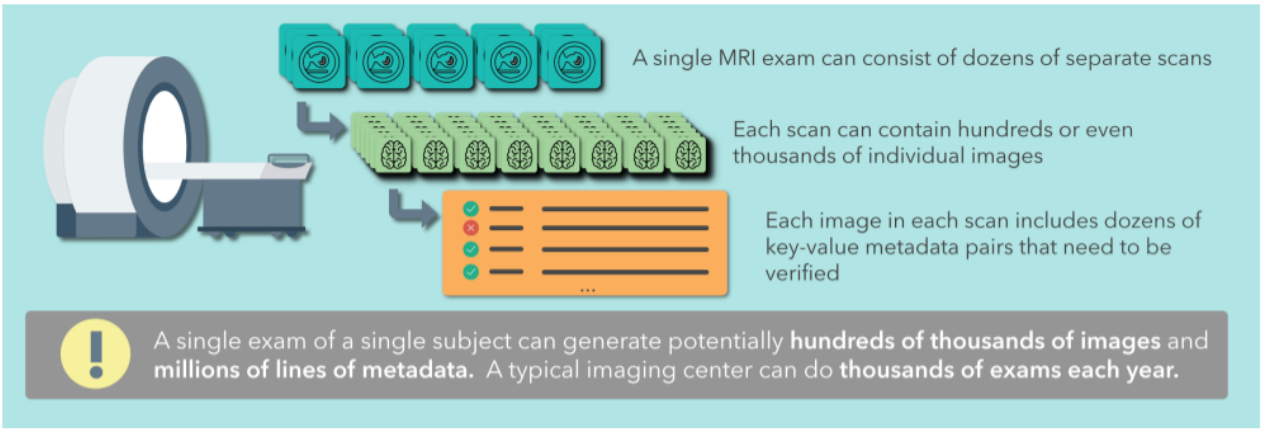

Figure 1. An illustration of the complexity & volumes of imaging data produced from a single exam.

SQAN includes a versatile QC engine that ensures adherence to a research study's protocol by comparing pre-configured facets (metadata tag values) of each scan for individual subjects to a *template* (or expected values). It ensures that all series of scans required across each modality (e.g., MRI, PET, CT) by a study's pre-defined protocol are present with the expected image counts, and that values match exactly, or within a percentage threshold of the *template* or expected value. SQAN safely ignores keywords expected to differ

[¶]SQAN source code available at: `https://github.com/IUSCA/SQAN`

between the *template* and subject *exam* (e.g., timestamps, subject demographics). Determining which keywords should be checked, and the type and thresholds of the checks, requires significant engagement and feedback from researchers, scanner technologists, and data scientists. We developed extensive measures to address the edge-cases and quirks of the DICOM®[‖] imaging standard as implemented by multiple manufacturers and utilized by researchers with a variety of goals. SQAN also features a modern web portal that is intuitive to a wide range of users including researchers, scanner technologists, and data scientists, each of whom approach QC with unique priorities, expertise, insights, and web portal expectations. It is worth noting that at this stage, the QC performed by SQAN is evaluated over each image's metadata, comprised by a key-value pair. The actual image files are not involved in any process described herein. Henceforth, we will refer to an image's metadata as an *image*.

The SQAN software suite includes the following components:

1. A MongoDB database that reflects the hierarchical nature of the *images* being stored for QC.
2. An incoming application that pulls metadata tags (key-value pairs) off files on disk on demand, in batch, or by querying an Orthanc receiver, and stores that metadata in a database.
3. A QC Engine that compares pre-specified metadata tags in every *image* in a subject's scan to a specified *template* (or expected values) based on predetermined QC criteria, and stores the results in a database.
4. A modern web portal that serves as a single point of access to data, *templates*, and QC results for a wide user base spanning faculty members who lead research studies, imaging center staff, and research associates.
5. An Application Programming Interface (API) that liaises between the web portal and the other back-end components including the database.

We describe these components in further detail below.

2.1 Data/metadata management

Research groups and medical imaging centers vary widely in the tools and techniques they use to acquire data. There are several manufacturers of imaging hardware (e.g. Siemens, Phillips, GE), and the data archival systems they feed may be a known entity such as XNAT or PACS, or perhaps custom solutions that simply create a hierarchy of physical directory trees on a filesystem. SQAN is designed to plug into any of these setups with no effort, or with, at worst, minimal effort. We describe the data access/transfer methods for ingestion that we currently support in SQAN below.

2.1.1 Ingestion via automated filesystem scans

With some setups, the datasets coming off the scanner may simply be stored onto a server's hard disk or a network mounted filesystem, with a directory tree organizing files in a defined hierarchy. A simplified example: a top level research directory which contains sub-directories for each subject ID, each subject directory contains sub-directories for each exam, each exam directory contains sub-directories for individual series of scans, and each of these series directories contain DICOM® files. This filesystem must be made available to the SQAN instance, either by copying data from the scanner server to the server running SQAN, or by exporting this filesystem via a network share and mounting it on the server hosting SQAN.

2.1.2 On-demand, manual ingestion

In some cases, a user may need to manually ingest data into the system (e.g. previously-archived data or datasets that don't follow a strict hierarchy on disk). Assuming the filesystem is accessible, this can be accomplished via the Linux command line on the host running the SQAN instance (or via a form within the portal - see Section 3.7.2). If data are ingested manually, the user or administrator performing the ingestion must also provide contextual identifiers such as the research study, subject ID, etc. The data can be provided as a directory of DICOM® files or as a `tarball` (.tar) archive.

[‖]DICOM® is the registered trademark of the National Electrical Manufacturers Association for its standards publications relating to digital communications of medical information.

2.1.3 Orthanc data receiver and metadata extraction used on the RADY SQAN instance

On the production RADY SQAN instance, the transfer of data from the imaging center is accomplished via a local instance of the open-source Orthanc[29] DICOM® image server (with an anonymizer service in between - see Section 2.5.1 for details). This instance is run within a Docker container 28, as the SCA infrastructure runs on CentOS Linux and not Orthanc's native Debian Linux environment. This approach allows more granular control over access to the Orthanc server's network access and therefore greater security, while also enabling more flexibility in deployment. In this Orthanc based setup, there is added bonus of the scanner technologist (or whoever sends data) being able to mark a dataset as a gold-standard *template* for a specific *research* study or an entire modality; the metadata tag values extracted from these *template* datasets can be used to perform QC checks in subsequent stages.

We initially utilized an Orthanc Application Programming Interface (API) call to translate the DICOM® image headers into a Javascript Object Notation (JSON)-encoded set of key-value pairs; however, we found that this method had several shortcomings, particularly when it came to private tags. Private tags are metadata keys defined for a particular instrument and not present in the standard DICOM® dictionary. To ensure that no private tag information is lost, we extract and store a full dump of the image headers utilizing our own routines.

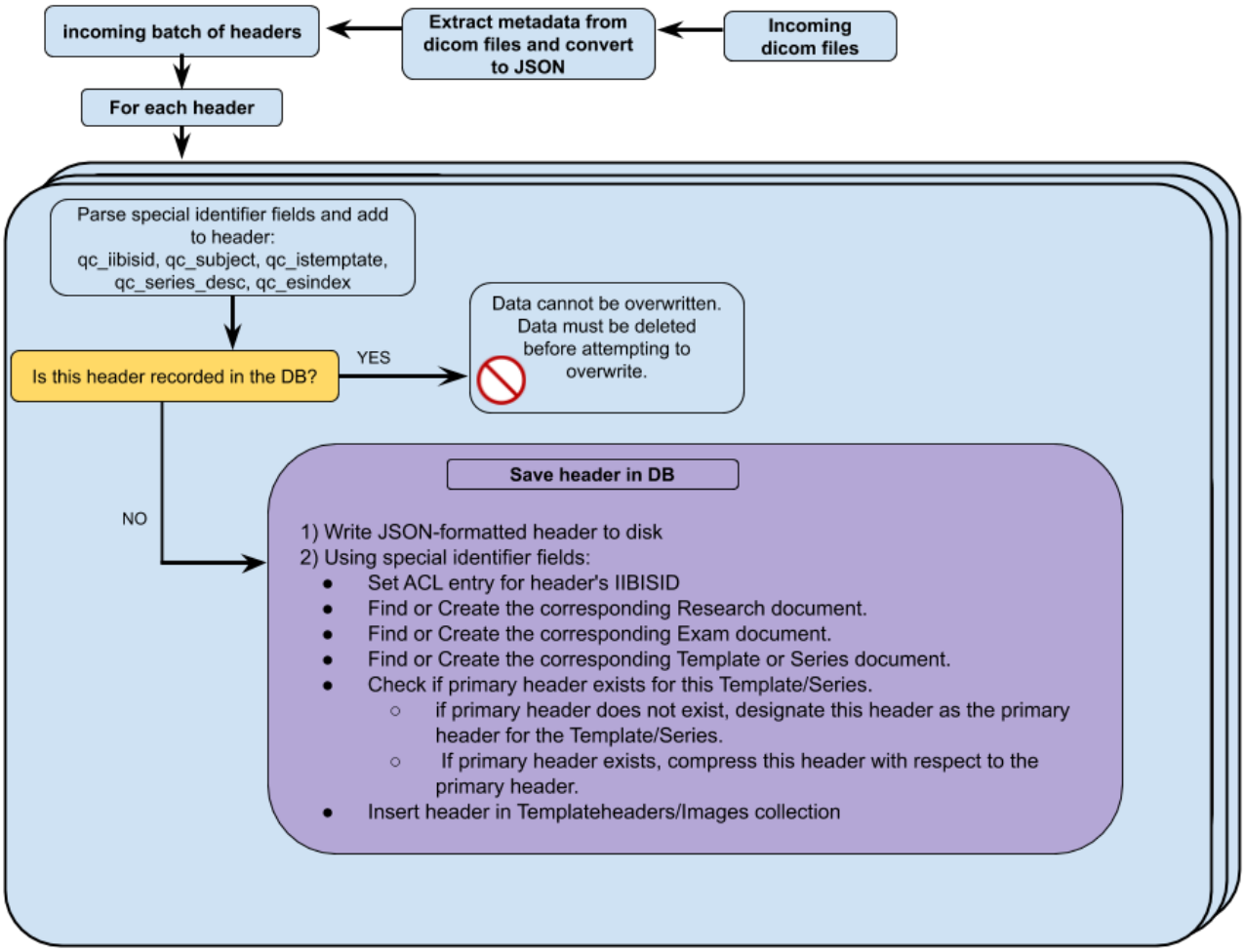

Figure 2. Process flow for extracting metadata from images and ingestion into SQAN.

2.1.4 Incoming metadata ingestion service

Irrespective of the data access/transfer method utilized by an imaging center, we have an *Incoming* service written in Javascript (specifically, in node.js [24]) that helps with the *Extraction*, *Transformation* and *Loading* (ETL) process 30. The *Incoming* service ingests metadata tags (key-value pairs) from DICOM® files into a MongoDB database [26]. As the *images* are ingested, a number of decisions are made and processes run (see Figure 2). Additional keywords are added to the header to aid with QC processes, and some header keywords are transformed or split (e.g., timestamps, subject IDs). Checks for duplicate *images* are made, and upstream records (*series*/*exam*/*research*) are created, as needed, for new datasets. If the *image* has been designated as a *template* by the sender, it is marked and separated to be used for later processing. Access controls are created and applied based on the *research* identifier. When the insertion of the *image* record is complete, the incoming application proceeds to the next one;

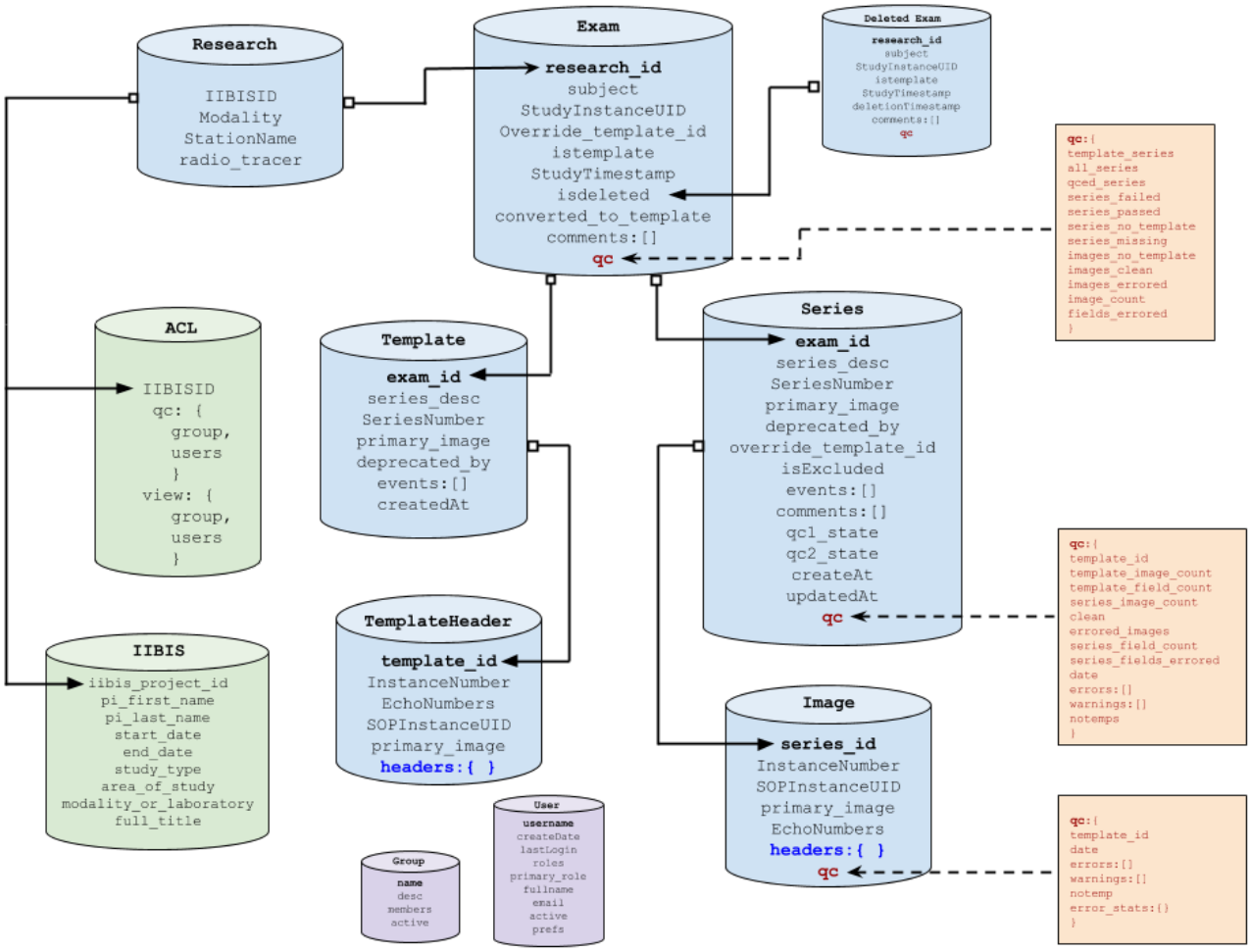

Figure 3. Current SQAN MongoDB schema

In setups involving an Orthanc receiver, this service has the additional capability to poll periodically for newly-arrived data. When Orthanc returns a non-null response, the incoming service saves the returned metadata in batches (in the case of our production RADY SQAN instance, of 1000) *images* to a location determined by the *research* ID, subject ID, and the imaging modality. When a *image* record is successfully inserted, the Incoming service instructs Orthanc to delete the local copy of the *image* from its records. When the current batch operation is completed, the incoming service immediately requests another batch of images from Orthanc and continues

in that manner until an empty set is returned, at which point it resumes polling at a defined interval, typically every few seconds.

2.2 MongoDB database structure

As noted above, the SQAN metadata tag/value pairs are stored in a MongoDB database. Unlike the strict schemas of traditional database systems, MongoDB allows for much greater freedom when dealing with heterogeneous datasets and enables SQAN's design goal of compatibility with a wide variety of instrumental sources. The data structure itself reflects the hierarchical nature of the *images* being stored for QC. The *research* collection has an entry for each unique modality–the group conducting the research, the instrument being used, the location of the instrument, and any radiotracers used (typically for PET instruments). Each entry in the exam collection corresponds to a single instance of a subject being scanned. Some *exams* are designated as *templates* by the imaging center and used in later QC steps. Each *exam* consists of multiple *series* which use different modes of scanner operation and data collection, and are stored in their own collection in the database. Each *series* is made up of multiple *images* and the *image* collection contains details about each *image*, its QC status and a complete audit trail (i.e., SQAN processing timestamps, user comments, QC audit information – see Section 3.3 and Figure 6 for more detail). The metadata headers extracted from each *image* are stored in a sub-document in each *image* entry. Figure 3 illustrates the database schema in full.

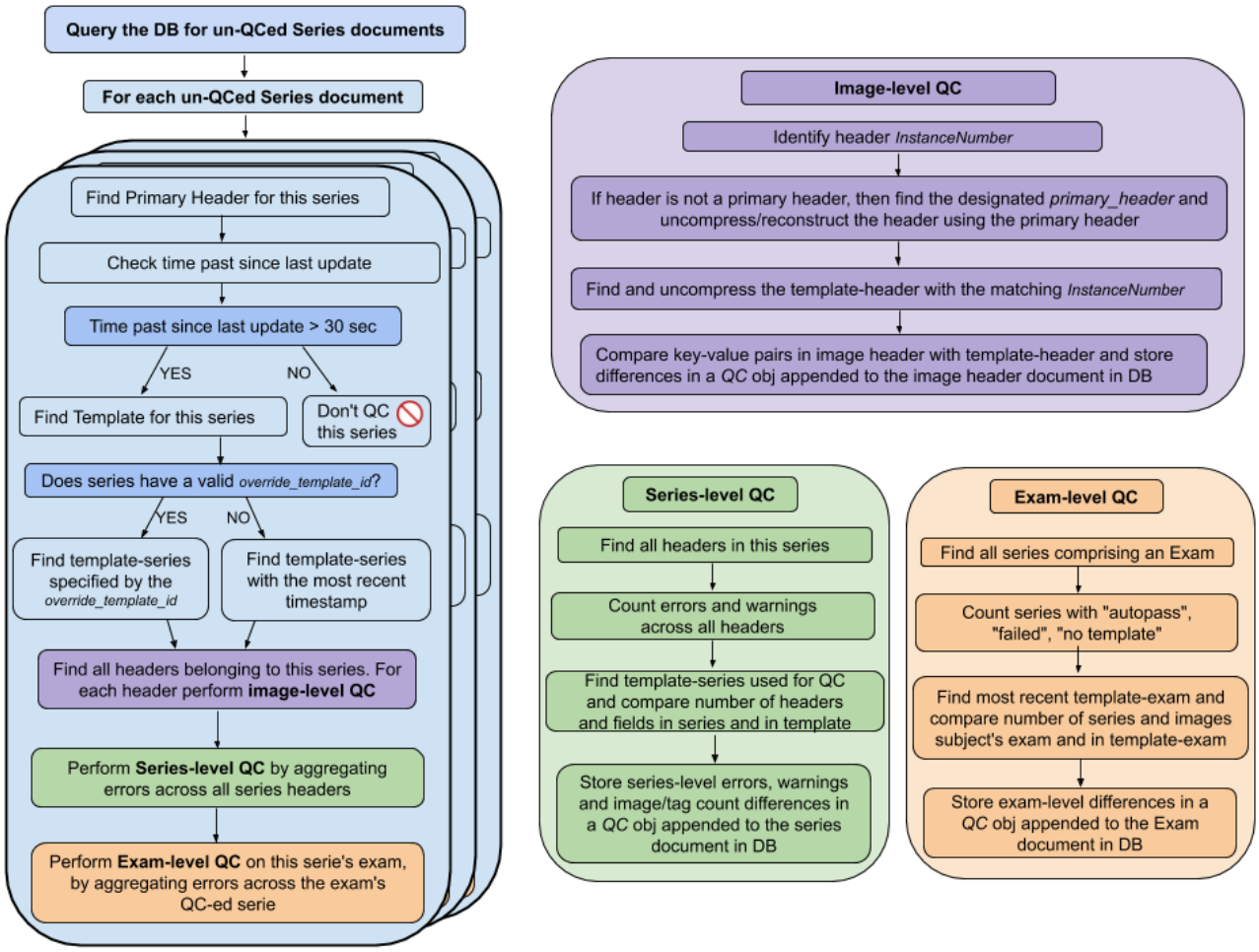

Figure 4. QC Engine process flow based on *template* datasets, used for RADY SQAN instance.

2.2.1 Metadata compression

During initial development and early production operations, we stored the entire metadata set for every *image* in the database. It quickly became apparent that this approach was unsustainable as the database grew unwieldy and unresponsive. In response, we went through a comprehensive code and data structure refactoring process. We developed updated routines that allowed us to compress the stored metadata by an order of magnitude. We accomplished this by recognizing that while each *image* contains dozens or even hundreds of keywords, only a few of these differ from *image* to *image* in a single scan (*series*). Before ingesting an *image*, our incoming application queries the database to check if there is a matching "primary" *image* (from the same *series*) for the one being processed. If there is no existing primary (i.e., this is the first *image* ingested from this *series*), then the *image* being processed is designated as the primary and stored in full. If a primary is found, then a differential analysis is performed on every metadata keyword and only those that differ from the primary are retained for insertion into the database.

2.3 SQAN QC Engine

After a dataset has been fully ingested, the SQAN QC Engine applies QC processes identified as being appropriate for that dataset based on a matching *template* or criteria specific to the *research* study, modality and *exam* timeframe, depending on the deployed instance. In setups involving an Orthanc receiver, we had to handle an additional limitation of Orthanc's data transfer routines: there is no simple method to determine when a dataset has fully arrived. We do not know, a priori, how many *images* comprise a given *series*. Moreover, images from a radiological scan arrive in no particular order: while *images* pertaining to a given *series* are consecutively numbered, they are ingested in random order; the "primary" defined above could be any *image* from the *series*. We addressed this by introducing a delay so QC is only initiated on *images* if (a) they have no QC status, and (b) no *images* belonging to that *series* have been ingested within the last N seconds (we set N=30 on our production RADY SQAN instance). Networking issues and/or excessive queuing at the Orthanc sending server or the anonymizer service can still result in dispersed datasets, so the arrival of additional *images* after QC has been performed results in the QC status being reset.

Table 1. QC Operations by level.

Level	**QC Checks**	**~# of times performed (per *exam*)**
Metadata tag	Consistent data type Within thresholds (number) Exact match (string)	>1,000,000
Image	Missing keywords Extra keywords Aggregates metadata QC	>10,000
Series	Missing *Images* Extra *images* Aggregates *image* QC	>10-20
Exam	Missing *series* Extra *series* Aggregates *series* QC	1

The QC Engine polls the database for any datasets (*series*) with no QC status set. Upon the return of data, it performs QC operations hierarchically. The starting point of the QC is performed at the level of *image* metadata. The compression process described above is reversed and the full header is decompressed and recreated for every *image*; the decrease in database size and performance gains far out-shadow the minimal overhead in

this step. A matching template-series is identified for comparison, and the corresponding *image* in the template-series matching the *image* to be QC'ed is retrieved. If the SQAN instance is template-less, then a *template* is constructed based on QC criteria defined by the user. A check of each keyword is done according to the data type of the keyword value. String values (e.g., software versions, scanner flags) typically require exact matches, whereas numerical values (e.g., slice thickness, echo times) have a threshold value that can result in either a warning or an error depending upon the severity of the deviation.

The QC Engine also knows which keywords are expected to vary between the *template* and a subject's exam, and can be safely ignored (e.g., patient demographics, timestamps, etc.) Additionally, different modalities (i.e., MRI, PT, or CT) have different sets of valid/invalid keywords, specialized handlers, and thresholds. Determining which keywords should be QC'ed or ignored, which threshold values to use, and which data types were valid for different keywords required extensive feedback from researchers, technologists, and data scientists, each of whom had differing priorities and expectations. This feedback process is vital to the development of a robust and reliable QC system, as false positives/negatives can poison results, making the system either overly sensitive or blind to systemic issues.

After QC has been performed at the imaging-level, the QC Engine then runs a series-level QC validation. This checks for missing *images* when compared to the template-series, and collates the QC state of each *image* in the *series*. When each *series* in a single subject *exam* has been QCed, the *exam* undergoes a final QC check, looking for missing (or extra) *series* when compared to the template-exam, and the QC states of each *series* in the *exam* are aggregated. Statistics are calculated and a final QC state is assigned to the *exam*. At this point every keyword in every *image* in every *series* of the *exam* has been validated and the QC process is complete. This is repeated for every *exam* requiring QC. See Table 1 for a breakdown of the quantity of QC procedures at each level and Figure 4 for an illustration of the QC process flow.

2.4 Web portal design including user experience/usability (UX) considerations

The SQAN web portal is written using the AngularJS client-side user interface framework [25], and is coupled with an Application Programming Interface (API) also written in Javascript as a node.js [24] Express app. We use the bootstrap and font-awesome javascript libraries [31, 32]to maintain look and feel across the portal. We are currently transitioning the user interface to Vue.js [27]; see Section 4.1.1 for more details.

Many existing QC solutions (e.g., XNAT's protocol check pipeline [33]) include useful QC algorithms but pose several challenges. Typically, such solutions require a time-consuming setup process, and may not be fully documented. Additionally, they are generally designed with a specific use-case in mind (i.e., specific scanner or image protocols), which can make integration with a different environment difficult. Even if these obstacles are overcome, common maintenance tasks (like altering QC check fields, adding *templates*, or updating parameter thresholds) may require a system administrator to step in.

SQAN portal's intuitive web interface offers significant usability improvements over many existing solutions, and empowers users to customize QC criteria themselves. The portal design and development also benefited from the significant engagement with researchers, scanner technologists, and data scientists that we had described earlier. One of the guiding principles behind the portal design is that it must be flexible enough to enable both researchers and technologists to perform adequate QA assessments.

As general protocol adherence is essential to both groups, it is thus the foremost goal of the system. SQAN allows technologists to track down issues with the scanner that might otherwise go unnoticed. This empowers technologists to be proactive in rescheduling scans for upcoming studies and avoid having to repeat data acquisitions; this can save the imaging center thousands of dollars. SQAN can also save researchers and staff hundreds of hours of data processing by identifying data that needs to be corrected or reacquired before it is included in any analysis; importantly, it ensures that the data being analyzed (and possibly published) is accurate and reliable. Researchers can also potentially use the data captured by the system to correct data during post-processing. The system's *research* summary feature, important to both researchers and technologists, allows users to identify problematic *series* or subjects that are not conducive to imaging research. Identifying these situations can allow researchers to allocate resources more effectively in the future. In Section 3, we describe the web portal's features, functionality and specific usability considerations.

2.5 Additional design considerations

In this section, we describe several other design considerations that we identified and addressed in order to improve SQAN's robustness and viability beyond our initial RADY production use-case. Some of these are operational considerations aligned with modern service deployment best practices while others are more specific to medical imaging.

2.5.1 Handling ePHI

Medical imaging data typically includes electronically protected health information (ePHI - e.g., patient name or date of birth). In our early requirements-gathering phase, we determined that QC at the instrument and protocol compliance level does not benefit from these elements of the data. Therefore, we introduced an anonymizer software layer in between the RADY scanners producing/sending the data and our SQAN service stack, thus ensuring that patient privacy is not compromised. After newly acquired scans are archived locally, they are transmitted to a ClinicalTrialProcessor (CTP) [34] anonymization service. CTP strips identifiable ePHI out of DICOM® metadata. A sister technical support group at Indiana University operates a CTP service instance for our RADY collaborators, thus allowing anonymized DICOM® data to be sent to SQAN via Orthanc for QC purposes. When SQAN retrieves an image from its local Orthanc server, it accesses only the image's metadata. While we have future plans to integrate QC procedures on the imaging data itself, at this time that data is discarded and only the metadata is retained for QC processing.

2.5.2 Backup and Disaster Recovery

After the metadata are written into the database, a JSON-formatted dump of each image's extracted headers is stored to local disk in a directory structure that describes the research, patient, *exam*, and *series* that the *image* came from. At regular intervals, these header dumps are collated, compressed, and sent to Indiana University's Scholarly Data Archive (SDA), a tape-based, distributed archival storage system. In the event of database corruption or loss, the entire metadata archive can be restored by retrieval and reprocessing without requiring mass data resends from the instrumental archives. The SQAN incoming application can be run in a recovery mode where it extracts a tarball or scans a directory tree on disk and processes every JSON file found instead of pulling them from the Orthanc instance.

In addition, the MongoDB content is backed up and archived on a rotating schedule. This includes not only the *image* headers and the QC results, but also the portal's user and access control tables, and information on each research group.

2.5.3 Performance considerations

From the beginning, SQAN was developed with QC performance as a significant design factor. Here, we discuss the reasoning behind and advantages of our choices for database (MongoDB) and runtime environment (node.js), and demonstrate the speed, efficacy, and scalability of our QC procedures. As an example, within our currently deployed production instance of SQAN there are >3.5 million images, and >700 million metadata key-value pairs. Testing has shown that if the QC status is completely reset for the entire database, our QC procedures can automatically process all existing records within just a few minutes. One of our design goals is for SQAN to be capable of scaling to real-time multi-center operations and be able to keep pace with the QC needs of any existing medical imaging operation.

Non-relational database: Our decision to utilize MongoDB as the repository of the imaging metadata was driven by the flexibility offered by its design and structure. MongoDB is an open-source, high-performance, industry-standard, document-driven database. In a traditional relational database (e.g., MySQL), a single record is represented by a row in a table, and the columns of the table define what data can be stored in each row. Relational databases are well-suited for homogeneous datasets with a small number of potential values. However, our dataset is heterogeneous, as different manufacturers and research centers can customize the DICOM® files to create private "tags" (key-value pairs) within the metadata. In addition, there are hundreds of potential tags in each header, not all of which are utilized on each *image*. MongoDB's design lets each document in a collection include only those keys that are defined. Likewise, some QC records themselves vary greatly depending on cascading QC results and statistics appropriate to a certain *exam*. QC records may also contain a technologist

or researcher comment, typically about the acceptability of a failed QC state (or, on rare occasions, the unacceptability of a successful QC state). Finally, our decision was driven in part by our existing familiarity with MongoDB from previous projects, and our ability to leverage existing codebases and frameworks.

node.js API: The decision to utilize MongoDB also helped to drive our choice of node.js as the back-end runtime environment, commonly referred to as Application Programming Interface (API). MongoDB stores its data in modified JSON structures; using a Javascript framework to access the database makes it straightforward to store and retrieve database records without format changes. Utilizing a runtime-interpreted language sped up the development process by allowing us to iterate quickly through our coding/deployment/testing cycles. As with MongoDB, node.js is well-supported, open-source, freely available, and used across multiple industries. To access the MongoDB instance from within our node.js applications, we made use of the mongoose.js [35] framework, which provides schema support and helper functions for processing data and retrieving related records.

Batch processing all *images* in a *series*: Those unfamiliar with Javascript development should be aware that it is, by nature, non-blocking and asynchronous. That is, unless steps are taken to prevent it, the interpreter will call a function and proceed to the next line without waiting for the called function to return. There are many advantages to this approach, including responsiveness and a reduced risk of computational bottlenecks, but there are drawbacks. In particular, we noted that our QC procedures were running inefficiently due to this asynchronous behavior. A set of *images* would be retrieved for QC, and the series-level QC would be performed as each *image* completed. This ensured that the series-level QC was always accurate, but it essentially doubled the time required to perform QC procedures on that set of *images* To avoid this we made use of the node.js *async* [36] library, which allowed us to batch process all the *images* in a *series* and run the series-level QC a single time. A similar approach was used in other parts of the incoming and QC systems that performed batch processing, and the result was that the performance of the QC process was effectively tripled.

3. WEB PORTAL VIEWS & FEATURES

In this section, we describe SQAN's intuitive web portal interface in detail, including how it is designed to empower a wide spectrum of users and usability considerations. Our goal from the outset was to enable as many functions from within the portal as possible so users never have to contact an administrator for basic operations. Accurately and intuitively conveying the condition and status of large datasets is an ongoing challenge in the field of user interface (UI) and user experience (UX) design and development. In designing our web portal, we considered the hierarchical data structure across all medical imaging modalities. Typically, a *research* (study) is defined by a group of subjects and an imaging protocol. The imaging protocol mandates what acquisition series (i.e., a specific set of scanner parameter settings) are performed during an exam, in what order, and how many *images* are acquired in each of those series. A subject can undergo one or more exams within a given *research* study; therefore, a subject's exam is expected to contain the acquisition series that are determined in the research imaging protocol. Below, we describe each major web portal view, including specific usability (UX) considerations that were factored into our design.

3.1 Exams view

Given the hierarchical nature of our datasets, we took a similar approach in developing the web portal through which users (i.e., researchers, technologists, and administrators) would interact with the metadata, view the QC statuses, and view any warnings or errors. Determining at a fundamental level how our system would be used helped drive the design process. Researchers informed us that they were most concerned with the overall QC status of each subject's scan, so we designed the landing UI element to display the subject-level QC statuses, where subjects are grouped by *research* and the subject QC statuses are summarized by color-bars indicating the proportion of *images* identified with the different types of image-level QC states, such as pass, fail, no template, etc. (see Figure 5). The user can request running the QC engine on a particular *exam*, for example if an updated set of *templates* is available (or if an older *template* is more appropriate), or if a QC engine bug affecting a specific dataset is fixed. These procedures are all available to authorized users via the web interface. Additionally, users can request the creation of a *template* based on a subject's *exam* if it is identified as an exceptional quality dataset worthy of serving as a *template*. Both of these actions are stored and displayed, allowing us to keep track of the history of changes for a particular dataset or its *template*.

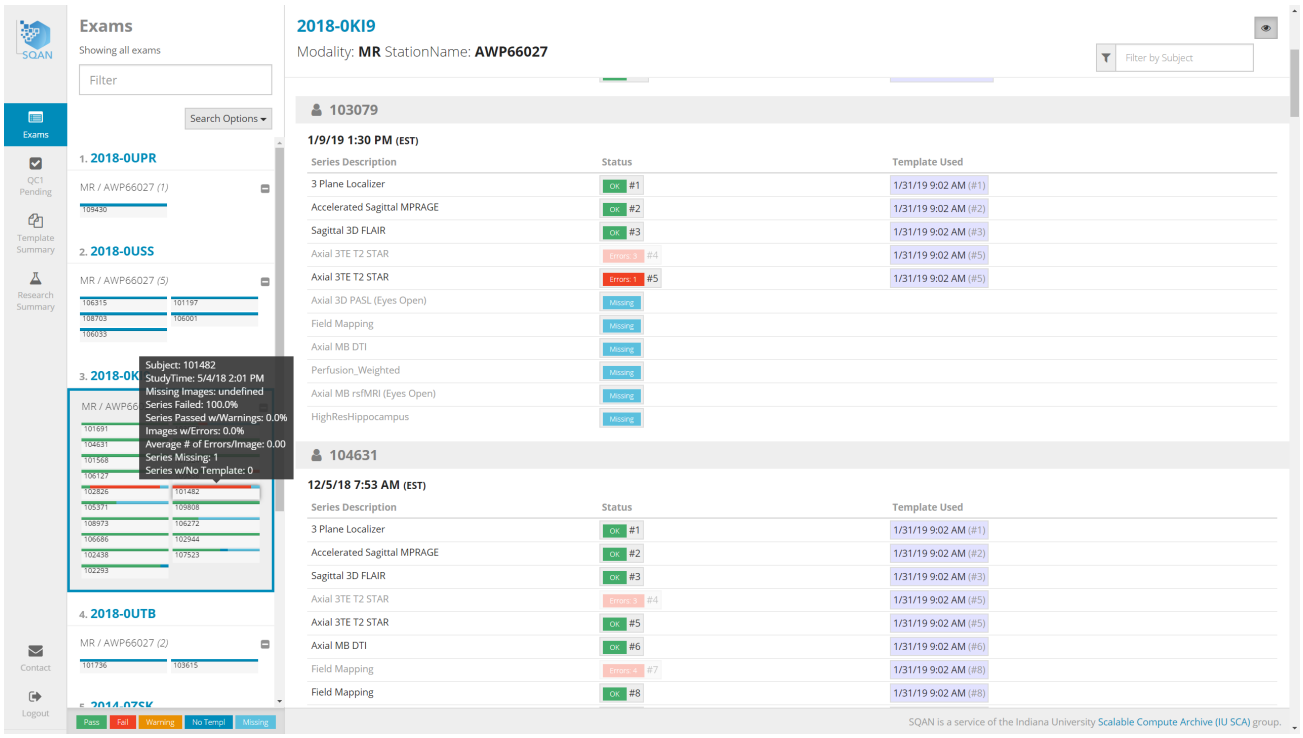

Figure 5. SQAN screenshot: Exams view with additional pop-up information about error conditions.

We addressed the following additional UX considerations in the Exams view. A researcher can mouse-over an element to receive a more detailed statistical QC summary for that *exam*. Note here that, as with all statistics, there are many different ways of presenting the same data. Consider a scan where each *image* contains 100 key-value pairs and a single value is found to differ from the corresponding *template* in every *image*. One can interpret this as a 100% error rate as all *images* contain an error; an alternative interpretation is that the overall error rate is 1%, as the error appears in one out of 100 key-value pairs. This simple example illustrates non-trivial decisions that must be made to display and summarize the dataset's QC status; importantly, these decisions have real consequences for user-response to QC issues. We took a measured approach to displaying QC statistics with particularly careful consideration of word choice, coloration, and warning size and location. The exam-level QC listing also includes a responsive search bar and filtering options (date range, modalities to display, and sorting order for the *research* studies and the subject listing within each *research* study), which allows rapid drill-down to the desired dataset.

3.2 QC pending view

The QC Pending view is a variant of the Exams view described above but only lists the *research* studies, and subjects/*exams* within them, that are believed to have QC issues and therefore require attention. While the Exams view is useful to review the QC status of an entire study or a *series* of subjects within one modality (e.g., MRI) within a study, the QC Pending view allows a technologist to review problematic datasets quickly and to contact responsible parties (e.g., the *research* study's PI) to solicit advice on how to proceed.

3.3 Series detail view

In both the Exams and QC Pending views, users can click on an *exam* or *research* to see the list of *series* acquired in that *exam*, the series-level QC status, and the *template* that was used to perform QC. The details of each *series* can then be opened to see the image-level QC status for every *image* in that *series* (Figure 6); finally, for each *image* the user can click to view the entire set of metadata key-value pairs and their QC states. At each QC-level, a summary of the QC status of the immediate lower level is clearly indicated.

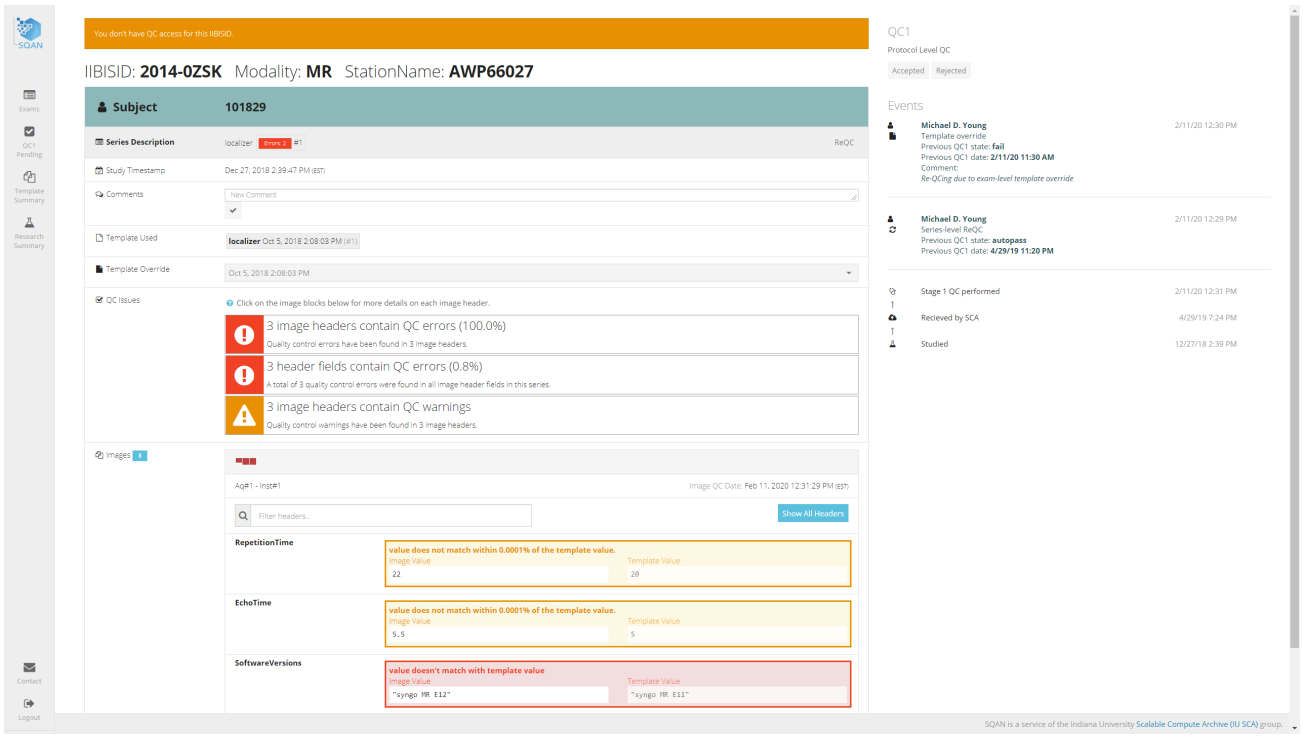

Figure 6. SQAN screenshot: Series detail view with clear warning/error highlights & audit trail.

The automated QC state of an *exam* can be updated manually: a user may determine that a particular QC condition will not affect their *research* goals, and so a *series* may be marked as "passing" QC. Alternatively, if an updated set of *templates* has been received by SQAN, or if an older *template* is more appropriate, an *exam* may have its QC status recorded and then reset to undergo QC processing with another *template*. These procedures are all available to authorized users via the web interface. In addition, users are required to enter comments when manually updating a QC status; users may also alert/ask questions of responsible parties. All events and changes performed on the *series* detail page are stored and displayed, allowing us to keep track of the history of the QC statuses for any given *series*.

We addressed the following additional UX considerations. Based on feedback from our users, we updated how the audit trail of events is displayed, especially if manual QC overrides are applied. We also updated the portal so QC issues are listed with color-coding and added QC-status-appropriate icons along with human-readable error messages, as shown in Figure 6. We use color-coding on the *image* listing and include a grey-colored box for missing *images*. The header listing (key-value pairs) only lists header tags with error or warning conditions by default, but can be expanded to show the entire header; a filter text box allows the user to navigate quickly to a particular header.

3.4 Research Summary view

The Exams and QC Pending views described above enable users to review the detailed QC status for their datasets. We determined that researchers, in particular Principle Investigators (PIs) and their associates, also often want a high-level summary of an entire *research* study in a small amount of user interface real estate. They want to be aware of one or more problematic subjects whose data they should exclude from their eventual analyses, or perhaps even of a particular acquisition series in which several subjects had trouble. In the Research Summary view, such users can select a specific *research* study and view a tabular listing of all subjects and acquisition *series* associated with that study, including a color coded status indicator for each subject and *series*. Clicking on a status indicator takes the user to the *series* detail view for that subject within that *research* study.

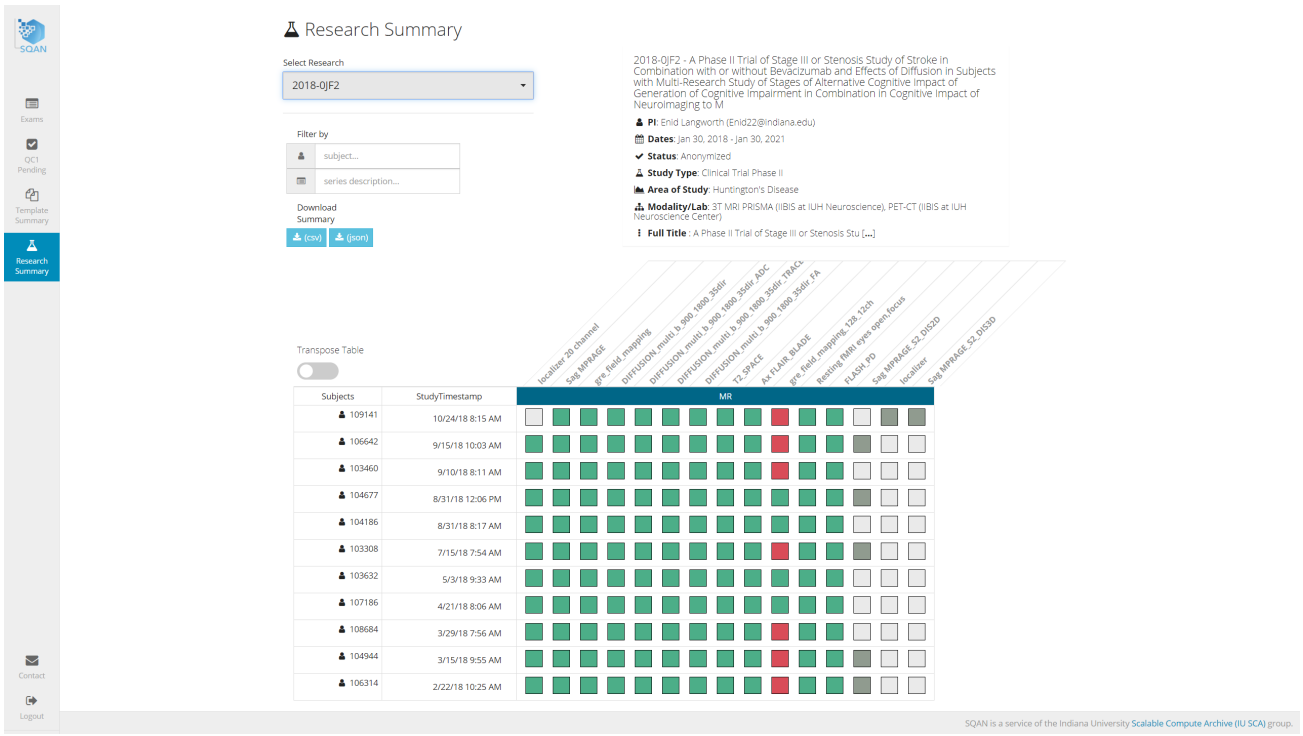

Figure 7. SQAN screenshot: Research summary view with a high level overview of a *research* study.

We addressed the following additional UX considerations for the Research Summary view. Users are able to filter by subject or the name of the acquisition *series*. We also provide the ability to download a CSV or JSON file of the data presented in the view, for e.g. to enable automated exclusion of a dataset from subsequent pipeline processing. We determined that some researchers want the subject listing to flow from top to bottom, while others want it to flow left to right, so we included a *transpose* button to change the row/column ordering of the view. This flexibility is useful in situations where a large number of subjects or *series* descriptions extends the table beyond the browser display area (horizontally on a laptop or large screen monitor and vertically on a mobile device).

Our close operational relationship with the RADY imaging center allowed us to include an optional feature, where we collate QC information displayed on the Research Summary view with information from a RADY-hosted *research* study database that allows contextualization of a particular study. We pick up this additional information by making a query out to a REST API on their end. Then we list the title of the research study, the PI's name, start and end dates for data acquisition for that study, study type (e.g., basic science human research vs. animal cognition), and modalities.

3.5 Template Summary view

SQAN currently allows QC criteria to be set entirely via *template* datasets sent specifically by the imaging center. There are occasions when a technician has sent a *template* over to SQAN and wants to verify it was received properly; there are other cases when an error condition is identified in a previously registered *template* within SQAN. Our portal includes a Template Summary view for this reason (Figure 8). Every *template* ever received is displayed with the corresponding *research* study name, modality, station (scanner) name, radio tracer, and the number of *templates* (referred to as "# Study Instances") available in the system for that *research* study. *Templates* are designated either from a subject's anonymized data or a phantom scan. A *research* can have multiple template-exams, identified by the imaging acquisition time-stamp (as *templates* have no subject ID). As shown in Figure 8, multiple *templates* for a given *research* study are displayed under different tabs. The user can view the details of a particular *template*, including how many times that *template* has been used to QC imaging

datasets, and the image count within that *template*. A technologist with sufficient authorization can delete a *template* or a specific acquisition *series* within a *template*, e.g., if they intend to resend that *template* (to correct errors) or provide an alternative *template* for that *research* study. We also list *research* studies for which we have no *templates* in our database in order to assist a technologist in sending *templates* for those studies.

We addressed the following additional UX considerations. Users can filter by *research* study ID (also referred to as IIBIS ID by our RADY colleagues). They can also sort by any of the fields on the *template* summary table header (e.g., modality, radio tracer). Entries are demarcated by background color differences to make them easy to parse for the user.

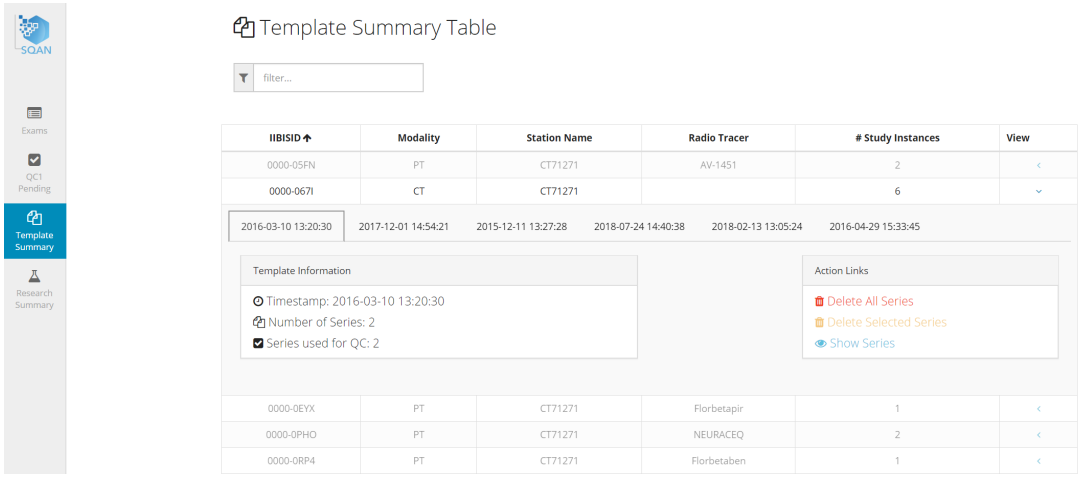

Figure 8. SQAN screenshot: Template summary view with multiple *templates* in one study.

3.6 Administrative views

3.6.1 User and access control management

SQAN allows authorized administrative users to control other user accounts, including the data access groups each user is part of and the Access Control List (ACL) that maps *research* studies to one or more user groups, authorizing them to view/comment and/or perform QC operations on that study. A screenshot of this view is shown on Figure 9.

3.6.2 Data transfer monitoring (in automated setups)

As described previously, on our production RADY SQAN instance, image data are shipped directly from the scanner server (by an Orthanc sender via an anonymizer service) to an Orthanc receiver on our end. We identified the need for a monitoring the data transfer & ingestion for both that setup and any other instance where image data are ingested automatically and/or periodically. To address this, we have a data transfer monitoring administrative view within the portal. A screenshot of this view is shown on Figure 10.

3.7 Early adopter features

We briefly describe recent feature additions and updates in the SQAN codebase. In particular, we invited potential collaborators from Harvard Medical School/Brigham and Women's Hospital (BWH) and Dartmouth-Hitchcock Medical Center for separate 2-day hackathons. In these sessions, we deployed prototype instances of SQAN and ingested sample data on local hardware at the respective institutions. This enabled us to have productive discussions and collate specific requirements. Two such requirements were aligned with features we had intended to add to SQAN even prior to the hackathons, and were implemented very recently. We expect to further refine these features and to make them more robust before we consider them ready for production deployment; for now, we deem them early adopter features.

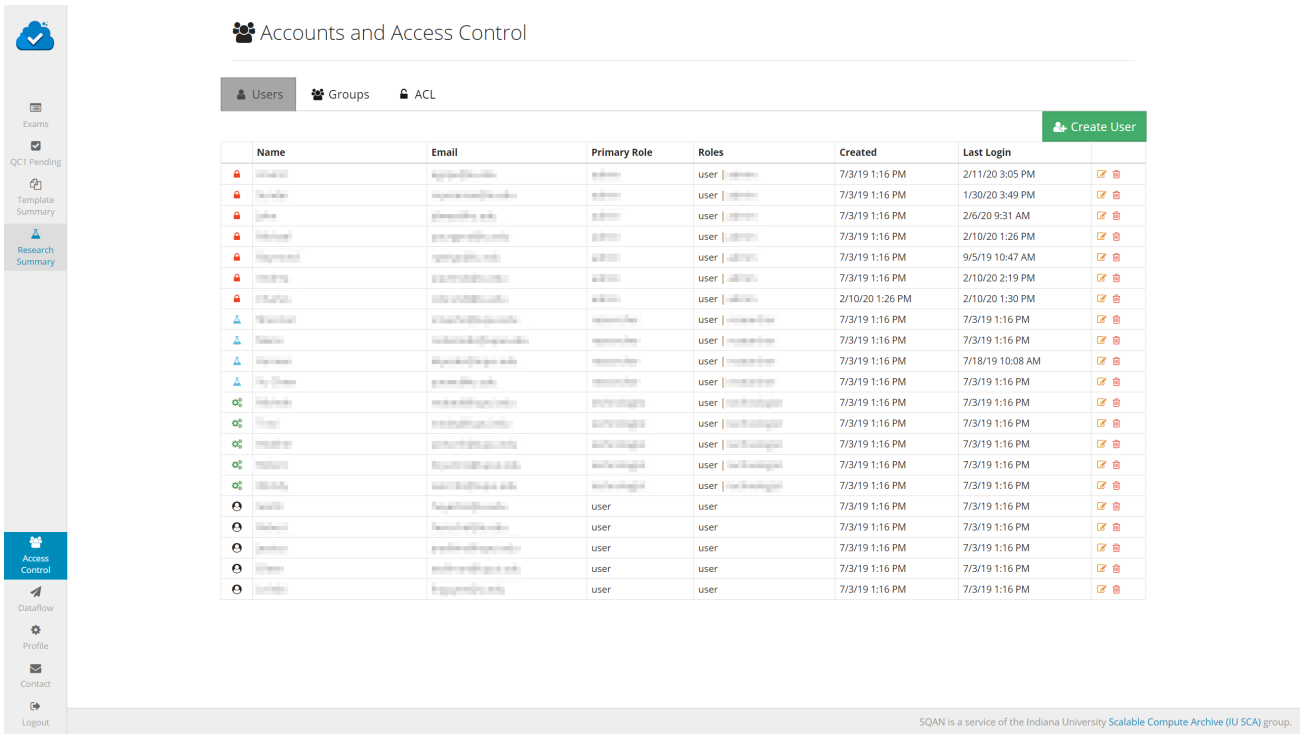

Figure 9. SQAN screenshot: Administrative view for access control.

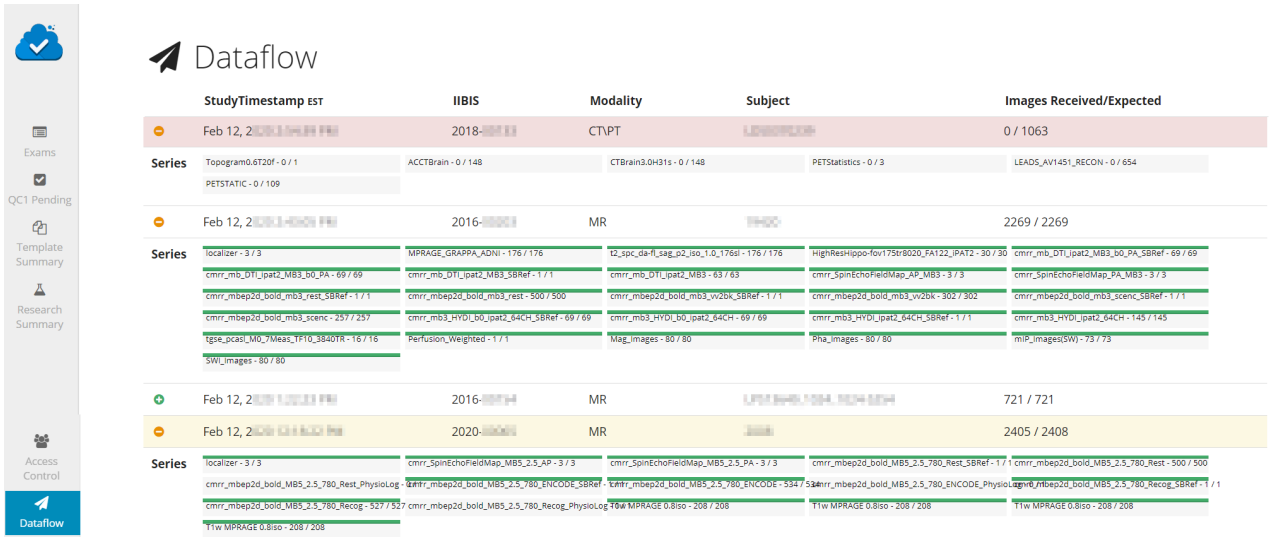

Figure 10. SQAN screenshot: Administrative view to monitor data transfer & ingestion into SQAN.

3.7.1 Custom template-less QC criteria

SQAN's QC engine used on our production RADY SQAN instance relies on QC criteria derived from a combination of metadata-tag value expectations per modality and a *template* dataset uniquely matched to an *image* dataset being QCed. However, under the hood the QC engine was already agnostic to the where the list of precise tags to be QCed (or not) came from. The tag value expectations were hard-coded within the QC engine code. In winter 2019, we modified the QC engine so that both the tag value expectations as well as which tags should be QCed can be set within the portal. The portal user interface allow a user to setup two complementary

custom QC configurations:

- *Blacklist* model in which the initial assumption is that all keywords in a given modality should be QC unless the user specifically asks to exclude certain metadata tags from being QCed. This is similar to the *template* based QC executed on our production RADY SQAN instance. There is a heavier burden on the user to exclude tags and precisely define acceptable QC criteria for all the other keywords. Anyone looking for comprehensive protocol QC would find this model valuable despite the need for an initial investment of time to set the criteria.
- *Whitelist* model in which the user starts from scratch, picks tags to be QCed, and then assigns acceptable QC criteria to those keywords. This model is similar to some of the existing automated QC options we discovered in our research prior to our work on SQAN. Anyone looking to keep the initial investment of time minimal while still reaping the benefits of protocol compliance would find this model useful.

Our overall goal is to enable as broad a spectrum of QC options as possible so any imaging center that adopts SQAN can choose the method that fits their usecase best.

3.7.2 Manual image upload

We also developed a Data Upload view within the portal that allows authorized users to point the portal at a specific disk location where a dataset can be accessed for ingestion. Upon selecting or entering a *research* study and a subject name, SQAN will ingest the dataset and perform automated QC. This feature is useful for imaging centers that do not setup an automated ingestion system from the scanner (for e.g., via Orthanc) or via periodic ingestion workflow (e.g., scheduled `cron` jobs); it is also useful for a technologist who is trying to debug ingestion of a particular dataset repeatedly.

4. CONCLUSION & FUTURE PLANS

Quality assurance (QA) is a fundamental first step in guaranteeing reliable and reproducible scientific research, and requires continuous involvement by all stakeholders and the use of appropriate quality control (QC) methods to identify data quality issues. Medical imaging, fundamental to clinical research and diagnosis, is costly and generates large datasets, thereby making the need for quality assurance (QA) through adequate quality control (QC) methods even more critical. Existing medical imaging QC methods are designed for specific use-cases with limited scope. In this paper, we have presented Scalable Quality Assurance for Neuroimaging, an open source, automated, flexible QC software suite that has supported 50+ research projects QCing >3.5 million images and >700 million metadata tags.

4.1 Future plans

Our immediate plans for SQAN include continuing our collaboration with Harvard Medical School/Brigham and Women's Hospital (BWH) and Dartmouth-Hitchcock Medical Center while continuing to pursue the larger goal of building a vibrant community of developers/maintainers and setting up an independent governing body. We expect to polish the early adopter features described in the previous section and to also finish the user interface refresh (described further below) in the near future. In the longer-term future, we intend to expand the capabilities of the SQAN project to extend beyond metadata/protocol QC including but not limited to the areas described below.

4.1.1 User interface refresh (Vue.js)

The current SQAN UI is built atop the once-actively-maintained AngularJS framework [25], which will not be maintained post-mid-2021. The SCA team has switched to the better-supported Vue.js framework [27] for new projects, and has transitioned other existing projects to Vue.js. The SQAN user interface is currently being rewritten in Vue.js, providing a more stable, secure, and performant experience; this transition is expected to be completed by April 2020, and is expected to better accommodate third-party community contributions adapting to diverse end-user requirements.

4.1.2 Imaging level QC

While the protocol compliance QC that SQAN performs is an essential first step in data quality assurance, noise and image artifacts often prevalent in medical imaging are not necessarily flagged by this stage of QC. The noise and artifacts may originate from a subject's motion or from the instruments themselves. Many techniques have been developed over the years to detect and attempt to correct such artifacts [37, 38]. It is our goal to expand SQAN to be able to detect and catalogue these QC issues and to incorporate existing QC tools and methods developed by the research and clinical imaging community. The first step is to connect SQAN to an imaging database to gain access to imaging. Rather than create our own system, we will draw on the very mature and robust XNAT [13] system—already in production at IU—to accomplish this task. Once that is accomplished, we will draw on the already-in-use ImageX system [21] or the XNAT viewer to allow users to look for artifacts within the image data. We will also draw on existing QC tools for imaging types (for e.g. fMRI, DWI, etc.) to incorporate those into our system. Our long term goal is to make SQAN a *one-stop shop* for all imaging QA needs.

4.1.3 Integration with other non-imaging databases

A critical aspect of imaging research is to be able to relate imaging metrics to other biomarkers. These can include demographics, genetic information, neuropsychiatric score, and medical records, just to name a few. As such, it is important that all of this data be available to researchers. To that end, we plan to connect SQAN with these various databases, for e.g. REDCap [39], so that once imaging QC has been completed the user can easily connect the vetted imaging data with important metrics from other sources and more rapidly engage in the desired analyses.

4.1.4 Software algorithms/pipelining

The Quality Assurance capabilities provided by SQAN needs to lead to better data outcomes and better processing. To that end, we plan to link SQAN with pipelining processes in order to automate data processing as it is obtained, and to identify the correct parameters to use in those processes when possible. Pipeline systems exist currently, for e.g. XNAT [40] or brainife.io, an IU project mentioned earlier. Our goal is to interface with those rather than build pipelines into SQAN itself. The ideal flow would be as follows:

1. Raw data are ingested into SQAN and metadata is checked to make sure correct protocols have been followed.
2. Once protocol and metadata have been verified, the raw imaging data will be made available for manual viewing or automated QC checks, if available.
3. With both QC steps passed, a call will be made to the appropriate processing pipeline to begin working on the data.
4. Processed data will be subject to further QC checks.
5. Final vetted data will be made available to researchers.

To accomplish this ideal workflow, SQAN will need to be interconnected with imaging databases, pipelining systems, and computational infrastructure. Our plan is to develop these interconnections at IU (and at our current collaborator institutions) where each of these pieces is available in some capacity. Once complete, we will have a fully automated imaging data system that will ingest raw data and disseminate final processed data that has been fully vetted by a robust QC system. Such a system would be able to make imaging centers—especially ones responsible for serving data in multi-center imaging projects—more efficient, and the data more robust for critical analyses.

4.1.5 Expanding SQAN to other imaging disciplines

Medical imaging goes beyond the scope of neuroimaging QC that SQAN currently supports. To that end, we expect work with imaging centers within IU and beyond to identify instruments and imaging types (for e.g. electron or light microscopy, both of which are heavily used at IU) that could benefit from QC functionality provided by SQAN.

ACKNOWLEDGMENTS

Our deepest gratitude to researchers at the IU Center for Neuroimaging; Technologists and staff at the RADY imaging center; Collaborators Dr. Sylvain Bouix and Dr. James Ford who are currently evaluating SQAN; and Dr. Harmony Jankowski for taking the time to review the manuscript of this paper.

REFERENCES

[1] Barthel, H., Schroeter, M. L., Hoffmann, K.-T., and Sabri, O., "Pet/mr in dementia and other neurodegenerative diseases," in [*Seminars in nuclear medicine*], **45**(3), 224–233, Elsevier (2015).

[2] Assaf, Y., Johansen-Berg, H., and Thiebaut de Schotten, M., "The role of diffusion mri in neuroscience," *NMR in Biomedicine* **32**(4), e3762 (2019).

[3] Dumoulin, S. O., Fracasso, A., van der Zwaag, W., Siero, J. C., and Petridou, N., "Ultra-high field mri: Advancing systems neuroscience towards mesoscopic human brain function," *NeuroImage* **168**, 345–357 (2018).

[4] Smith, S. M. and Nichols, T. E., "Statistical challenges in "big data" human neuroimaging," *Neuron* **97**(2), 263–268 (2018).

[5] Borghi, J. A. and Van Gulick, A. E., "Data management and sharing in neuroimaging: Practices and perceptions of mri researchers," *PloS one* **13**(7) (2018).

[6] Lu, W., Dong, K., Cui, D., Jiao, Q., and Qiu, J., "Quality assurance of human functional magnetic resonance imaging: a literature review," *Quantitative imaging in medicine and surgery* **9**(6), 1147 (2019).

[7] Schwartz, D. L., Tagge, I., Powers, K., Ahn, S., Bakshi, R., Calabresi, P. A., Todd Constable, R., Grinstead, J., Henry, R. G., Nair, G., et al., "Multisite reliability and repeatability of an advanced brain mri protocol," *Journal of Magnetic Resonance Imaging* **50**(3), 878–888 (2019).

[8] Marcus, D. S., Harms, M. P., Snyder, A. Z., Jenkinson, M., Wilson, J. A., Glasser, M. F., Barch, D. M., Archie, K. A., Burgess, G. C., Ramaratnam, M., et al., "Human connectome project informatics: quality control, database services, and data visualization," *Neuroimage* **80**, 202–219 (2013).

[9] Greve, D. N., Mueller, B. A., Liu, T., Turner, J. A., Voyvodic, J., Yetter, E., Diaz, M., McCarthy, G., Wallace, S., Roach, B. J., et al., "A novel method for quantifying scanner instability in fmri," *Magnetic resonance in medicine* **65**(4), 1053–1061 (2011).

[10] Power, J. D., Barnes, K. A., Snyder, A. Z., Schlaggar, B. L., and Petersen, S. E., "Spurious but systematic correlations in functional connectivity mri networks arise from subject motion," *Neuroimage* **59**(3), 2142–2154 (2012).

[11] Wyman, B. T., Harvey, D. J., Crawford, K., Bernstein, M. A., Carmichael, O., Cole, P. E., Crane, P. K., DeCarli, C., Fox, N. C., Gunter, J. L., et al., "Standardization of analysis sets for reporting results from adni mri data," *Alzheimer's & Dementia* **9**(3), 332–337 (2013).

[12] Alfaro-Almagro, F., Jenkinson, M., Bangerter, N. K., Andersson, J. L., Griffanti, L., Douaud, G., Sotiropoulos, S. N., Jbabdi, S., Hernandez-Fernandez, M., Vallee, E., et al., "Image processing and quality control for the first 10,000 brain imaging datasets from uk biobank," *Neuroimage* **166**, 400–424 (2018).

[13] "XNAT: An open source imaging informatics platform developed by the Neuroinformatics Research Group at Washington University.." https://www.xnat.org/about.

[14] "Picture Archiving and Communication System (PACS).." https://en.wikipedia.org/wiki/Picture_archiving_and_communication_system.

[15] Friedman, L., Stern, H., Brown, G. G., Mathalon, D. H., Turner, J., Glover, G. H., Gollub, R. L., Lauriello, J., Lim, K. O., Cannon, T., et al., "Test–retest and between-site reliability in a multicenter fmri study," *Human brain mapping* **29**(8), 958–972 (2008).

[16] Perigo, R. L., Gopu, A., Young, M. D., and Bao, Y., "Toward sustainable deployment of distributed services on the cloud: dockerized odi-ppa on jetstream," (2018).

[17] Gopu, A., Hayashi, S., Young, M. D., Harbeck, D. R., Boroson, T., Liu, W., Kotulla, R., Shaw, R., Henschel, R., Rajagopal, J., Stobie, E., Knezek, P., Martin, R. P., and Archbold, K., "ODI - Portal, Pipeline, and Archive (ODI-PPA): a web-based astronomical compute archive, visualization, and analysis service," (2014).

[18] Young, M. D., Rhode, K., and Gopu, A., "A science portal and archive for extragalactic globular cluster systems data," (2015).
[19] Pilachowski, C., Hinkle, K., Brokaw, H., Young, M. D., and Gopu, A., "Preservation of 20 years of spectrographic data," (2016).
[20] Young, M. and Michael, S., [*Big Data Challenges in the Blanco DECam Bulge Survey*], vol. 512 of *Astronomical Society of the Pacific Conference Series*, 189 (2017).
[21] Young, M. D., Perigo, R. L., and Gopu, A., "Imagex: a full stack imaging archive solution," (2018).
[22] Morgan, D., Gopu, A., Young, M. D., and Hayashi, S., "Emc-sca: Organizing, managing and searching large collections of images: A new resource to handle high-throughput imaging," (2014).
[23] "GitHub: A secure, scalable developer platform.." https://https://github.com.
[24] "NodeJS JavaScript Runtime." https://nodejs.org.
[25] "AngularJS client-side user interface framework." https://angularjs.org.
[26] "MongoDB: a general purpose, document-based, distributed database [...]." https://www.mongodb.com.
[27] "Vue.js: The progressive JavaScript client-side user interface framework." https://vuejs.org.
[28] Merkel, D., "Docker: Lightweight linux containers for consistent development and deployment," *Linux J.* **2014** (Mar. 2014).
[29] "Orthanc: open-source, lightweight DICOM server for healthcare and medical research.." https://www.orthanc-server.com.
[30] "Extract, Transform, Load (ETL)." https://en.wikipedia.org/wiki/Extract,_transform,_load.
[31] "Bootstrap: An open source toolkit for developing with HTML, CSS, and JS.." https://getbootstrap.com.
[32] "Get vector icons and social logos on your website with font awesome, the web's most popular icon set and toolkit.."
[33] "XNAT Protocol Check.." https://wiki.xnat.org/docs16/3-administrator-documentation/pipeline-engine/configuring-the-protocol-check-pipeline.
[34] "MIRC Clinical Trials Processor (CTP): a stand-alone image processing application for imaging clinical trials data.." https://mircwiki.rsna.org/index.php?title=CTP-The_RSNA_Clinical_Trial_Processor.
[35] "Mongoose: Elegant mongodb object modeling for node.js." https://mongoosejs.com.
[36] "Async: A utility module which provides straight-forward, powerful functions for working with asynchronous JavaScript.." https://caolan.github.io/async/v3.
[37] Power, J. D., Mitra, A., Laumann, T. O., Snyder, A. Z., Schlaggar, B. L., and Petersen, S. E., "Methods to detect, characterize, and remove motion artifact in resting state fmri," *Neuroimage* **84**, 320–341 (2014).
[38] Power, J. D., Plitt, M., Gotts, S. J., Kundu, P., Voon, V., Bandettini, P. A., and Martin, A., "Ridding fmri data of motion-related influences: Removal of signals with distinct spatial and physical bases in multiecho data," *Proceedings of the National Academy of Sciences* **115**(9), E2105–E2114 (2018).
[39] "REDCap: A secure web application for building and managing online surveys and databases. ." https://www.project-redcap.org.
[40] "XNAT Pipeline Processing.." https://wiki.xnat.org/documentation/xnat-administration/configuring-the-pipeline-engine/installing-pipelines-in-xnat.