Web Services for Image Analytics to Enable Feedback Control of Remote Instrumentation in CNT Growth.

A Thesis presented to

the Faculty of the Graduate School

at the University of Missouri

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

---

by

Divya Mishra

Dr. Prasad Calyam, Thesis Supervisor

DECEMBER 2023

The undersigned, appointed by the Dean of the Graduate School, have examined the thesis entitled:

WEB SERVICES FOR IMAGE ANALYTICS TO ENABLE FEEDBACK CONTROL OF REMOTE INSTRUMENTATION IN CNT GROWTH.

presented by Divya Mishra,
A candidate for the degree of Master Science and hereby certify that, in their opinion, it is worthy of acceptance.

Dr. Prasad Calyam

Dr. Matthew R. Maschmann

Dr. Filiz Bunyak

Dr. Kannappan Palaniappan

ACKNOWLEDGMENTS

Above all, I want to extend my heartfelt appreciation to my mentor, Dr. Prasad Calyam. His consistent encouragement, valuable advice, and role as a guiding light have significantly influenced my thesis journey. His passion for academic excellence and the cultivation of a curious mindset have profoundly impacted my academic path. This thesis stands as a testament to the collaborative spirit of our research work, showcasing the victories and obstacles that characterize our lab's endeavors.

I am deeply thankful for the privilege of being part of the VIMAN and CERI labs. The dynamic and cooperative environment within these labs has played a vital role in my personal and career development. The excellence of this thesis reflects the commitment and enthusiasm of the remarkable individuals comprising these lively research communities.

I want to extend my heartfelt gratitude to my fellow lab members. Your companionship, constructive input, and priceless perspectives have been truly cherished. Your support and motivation have been a consistent wellspring of strength. The accomplishments and considerations of this thesis mirror our collective journey of learning and our unwavering quest for knowledge.

Furthermore, I want to acknowledge the faculty and staff of the department for their indispensable resources and guidance that paved the way for the accomplishment of this thesis. Your consistent dedication to cultivating an environment of scholarly distinction has deeply influenced my endeavors.

Last but not least, I am immensely thankful for the unswerving love and encouragement bestowed upon me by my beloved parents, family members, and friends. Their unwavering support enabled me to tirelessly pursue the successful completion of this thesis.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

ABSTRACT

Remote instrumentation of scientific instruments e.g., scanning electron microscopes today involves inefficient, manual methods for remote instrument control and handling of large image data sets for analysis at high performance computing locations. There is a need to develop web services that enable fast and secure remote instrument control and automated image analytics to foster rapid innovations in scientific discoveries in e.g., manufacturing, and biomedical applications. In this thesis, we present a set of novel web services that are part of a Remote Instrumentation Science Environment (RISE) that features chatbot guided capabilities to navigate a number of functions related to remote instrument control and image analytics. Specifically, we detail web services for secure automation of image acquisition tasks, and enhanced accessibility to improve the iterative user actions involved in achieving an image analytics objective. We implement our RISE web services in a real-world testbed featuring remote instrumentation programmable interfaces, and cloud computing resources for a Carbon Nano Tube manufacturing application. Usability study of our implementation demonstrates benefits of our solution for users with varying skill levels, and shows how our approach expedites gathering experiment results, improves networked resource usability, streamlines processes and increases accuracy of image analytics.

Chapter 1

Introduction

Scientific experimentation in e.g., manufacturing and biomedical applications often relies on the utilization of specialized instruments such as scanning electron microscopes (SEM) and transmission electron microscopes (TEM) [1, 2]. These instruments demand expertise and effort to manually operate the instruments and transfer data collections to high-performance computing (HPC) locations for image analytics [3, 4] as shown in Figure 1.2. Such a process for image acquisition and analytics is susceptible to errors and is highly time consuming due to the slow and inefficient mechanisms to perform iterative instrument control by being present at the instrument site, and also to manually transfer the large amount of collected image data to HPC Locations for analysis.

However, remote instrumentation can provide benefits to researchers to: (a) gain access to networked instruments for preparing samples, configuring image analysis parameters (e.g., zoom, focus, contrast) to steer the experiment to meet a specific scientific objective, and (b) help with automation of image data collection and transfer to HPC centers for storage and AI/ML models training for image analysis. Such a remote instrumentation poses many challenges, especially to orchestrate iterative workflows involved with scientific experiments, where intermediate images need to be analyzed to determine the instrument

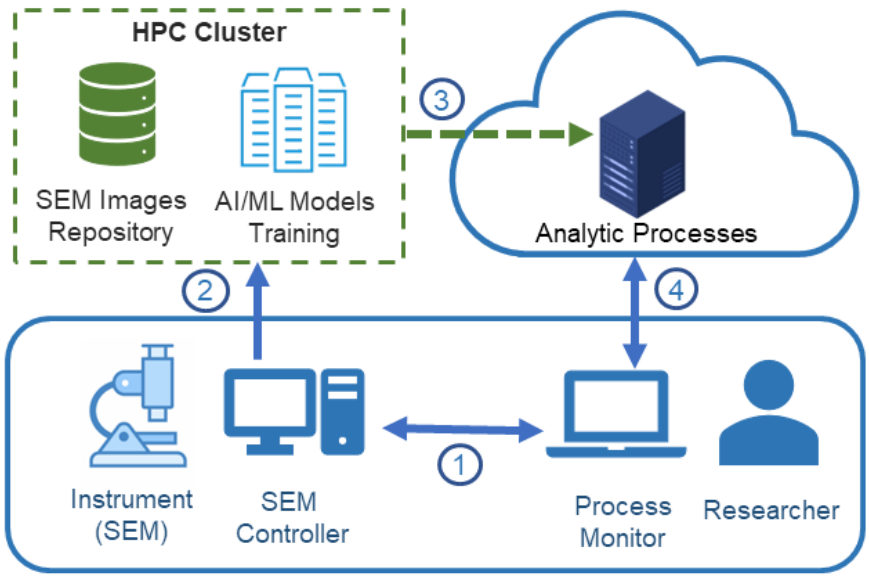

Figure 1.1: Overview of remote instrumentation process for image analytics.

settings for the next iteration and so on [5]. In addition, it is important to identify automation steps and enable convenient, yet secure remote instrument control so that user collaboration is possible when dealing with real-time experiments. Similarly, automation for centralized images storage, and image analytics through the development of relevant web services and guided user interfaces for the users can help facilitate faster innovations in scientific discoveries. In the automation for image analytics, AI/ML-based processes can be set in place to analyze incoming data and provide researchers the ability to validate insights in real-time and make quick decisions on the adjustment of instrument settings to steer the experiments in the right direction.

In this thesis, we present a set of novel web services that are part of a Remote Instrumentation Science Environment (RISE) that features chatbot-guided capabilities to navigate a number of functions related to remote instrument control and image analytics. Specifically, we detail RESTful web services for secure automation of image acquisition tasks, where remote access and control of scientific instruments is performed using Python programmable interfaces of specialized SEM instruments. In addition, we show how our proposed web services can provide enhanced accessibility to improve the iterative user actions involved in achieving an image analytics objective in e.g., a Carbon Nano Tube (CNT) manufac-

turing application. In this case, we detail how AI/ML-based methods [6] can allow RISE users to perform real-time image analytics, and discern critical process parameters to steer experimental control measures through an interactive chatbot-guided interface. Further, we describe the various security considerations for secure remote instrumentation session connections, data encryption, secure authentication, and stringent access control in the image data acquisition, storage and analysis functions.

We implement our RISE web services in a real-world testbed featuring remote instrumentation programmable interfaces, and cloud computing resources for a CNT manufacturing application. Specifically, we show the testbed offers a number of services such as the data storage service, an AI/ML data analysis service, a web portal featuring chatbot-guided interfaces for instrumentation setup and operation, and an integrated SEM via a cloudlet component. To validate our RISE web services benefits, we performed a usability study of our implementation for users with varying skill levels. The experiments involved users accessing the RISE web services via a web portal, allowing them to interact with the system for CNT image analysis, review chatbot-provided analysis to fine-tune zoom, focus, and contrast settings on the remote instrument. The results of our usability study show how our approach expedites gathering experiment results, improves networked resource usability, streamlines processes and increases accuracy of image analytics.

The remainder of this thesis is organized as follows: Chapter 2 presents related work. In Chapter 3, we detail the RISE web service architecture. In Chapter 4, we describe our evaluation of the RISE web services by describing the tested and the corresponding usability study. Chapter 5 concludes the thesis.

1.1 Problem Motivation

Imagine a world where researchers could break free from the confines of laboratories and access cutting-edge scientific instruments from anywhere in the world. This vision forms

the heart of the RISE (Remote Instrumentation and Sensing Environment) thesis, driven by a compelling problem motivation.

The Geographical Challenge: Traditionally, scientific experiments, especially those reliant on scanning electron microscopes (SEMs), were bound by geographical limitations. Researchers needed to be physically present in proximity to these specialized instruments. This constraint not only limited access but also hindered global collaboration in the pursuit of scientific breakthroughs. Recognizing this challenge, the RISE thesis sets out to redefine scientific experimentation by enabling remote access.

Unlocking Efficiency: Another pivotal issue lies in the inefficient utilization of SEM resources. The manual operation of SEMs led to substantial idle time and scheduling conflicts, resulting in increased costs and delays. Within this challenge, RISE sees an opportunity for optimization. By automating SEM control and data collection, RISE aims to maximize resource utilization, making scientific exploration not only more accessible but also more cost-effective.

Streamlining Workflows and Data Management: The intricate dance of SEM operation and data collection presented its own set of hurdles. Human errors were not uncommon, and the process was labor-intensive and time-consuming. RISE recognizes the need for streamlined workflows. Through intelligent agents and automation, it seeks to transform the experiment setup, monitoring, and data collection processes. This transformation promises increased efficiency and accuracy.

Fostering Collaboration Across Borders: In a world connected by the digital thread, collaboration knows no boundaries. However, the physical constraints of SEM usage hindered this collaborative spirit. RISE's vision is to create a secure and collaborative workspace where researchers from different corners of the globe can seamlessly share data, insights, and experiment setups. The goal is to foster a new era of teamwork, transcending geographical limitations.

Elevating Data Management and Quality: In the world of SEMs, data reigns supreme. The immense volume of data generated posed challenges in management and quality assurance. The RISE thesis recognizes this, introducing intelligent feedback mechanisms and deep learning-based image analytics to enhance data quality. The objective is clear: to transform raw data into valuable insights.

Data Security: Yet, amidst all this transformation, one principle remains steadfast data security. RISE places data integrity and confidentiality at the forefront. With robust security measures, including encryption, the architecture ensures that sensitive research data remains impervious to prying eyes.

In essence, the RISE thesis is driven by a grand vision—to democratize scientific experimentation, to optimize resource utilization, to streamline workflows, to foster global collaboration, to elevate data quality, and to safeguard data security. It is the embodiment of a world where science knows no boundaries, where researchers are empowered, and where breakthroughs are boundless.

1.2 Research Aim

The ultimate goal of this research project is to reshape the landscape of scientific experimentation through the development and deployment of the RISE (Remote Instrumentation and Sensing Environment) architecture. At its core, this goal strives to overcome traditional limits, allowing scientists to conduct experiments with unparalleled efficiency, accessibility, and security. Importantly, web services play a critical role in achieving this goal. They serve as the RISE ecosystem's entry point, allowing global access to cloud-based resources and powerful scientific instruments. Web services simplify experiment configuration, monitoring, and real-time feedback by providing a user-friendly web portal, thereby optimizing the experimentation process. Furthermore, by automating instrument control and data collecting, these services increase resource efficiency. They also improve data management

and quality assurance by allowing for safe data storage, retrieval, and analysis. Web services create a culture of global collaboration among researchers by providing secure web-based collaboration tools. Finally, they serve as the foundation for strong data security measures, ensuring the integrity and confidentiality of essential research data. In essence, the RISE thesis seeks to empower scientists all over the world using web services, thereby reinventing the way scientific experiments are carried out.

1.3 Tools and Technologies Involved

This section talks about tools and technologies used in the RISE project.

1.3.1 Node.js

Node.js is a runtime environment that allows developers to run JavaScript code outside of web browsers. In this project's dependencies, Node.js serves as the foundation for the back-end server. The ‘"aws-sdk"‘ and ‘"@aws-sdk/client-s3"‘ dependencies enable interactions with Amazon Web Services, particularly for managing S3 storage. ‘"bcrypt"‘ is utilized for secure data hashing and encryption, while ‘"body-parser"‘ parses HTTP request bodies. ‘"cors"‘ facilitates cross-origin requests for frontend-backend communication. ‘"dotenv"‘ securely manages environment variables. Express.js, driven by Node.js, serves as the web application framework, and ‘"express-session"‘ handles user sessions. ‘"jsonwebtoken"‘ manages JSON Web Tokens for authentication. ‘"mongoose"‘ provides an interface for MongoDB databases, and ‘"sharp"‘ aids in image processing. Node.js, along with these dependencies, empowers the backend server to handle various tasks, from data management to security and image manipulation.

1.3.2 React.js

React is a JavaScript library used for building user interfaces. It simplifies the process of creating interactive and dynamic web applications. In this project's dependencies, React (‘”react”‘) serves as the foundation for the user interface development. It works in conjunction with ‘”react-dom”‘, which helps with rendering React components into the DOM. Within the project's dependencies, ‘”axios”‘ is used for making HTTP requests to the backend server, ‘”bootstrap”‘ provides a CSS framework for responsive and styled components, and ‘”formik”‘ simplifies form management. ‘”react-router-dom”‘ facilitates routing within the React application, allowing navigation between different pages. ‘”reactstrap”‘ offers pre-designed React components to streamline UI development. Testing libraries like ‘”@testing-library/react”‘ and related packages aid in testing React components for quality assurance. Additionally, ‘”react-i18next”‘ assists with internationalization, ‘”react-toastify”‘ provides notifications, and ‘”yup”‘ is used for validation. ‘”jsonwebtoken”‘ is relevant for authentication purposes. ‘”remixicon-react”‘ and ‘”simplebarreact”‘ enhance the user interface with icons and scrollbar functionality, respectively. Finally, ‘”web-vitals”‘ is used for monitoring web application performance.

1.3.3 Python

In your RISE project Python Flask server is used, I have incorporated several libraries to enhance the server's functionality and security. The ‘Crypto.Cipher‘ library is utilized for Advanced Encryption Standard (AES) encryption to secure data transmission and storage. The ‘base64‘ library, specifically ‘b64decode‘ and ‘b64encode‘, aids in encoding and decoding data to and from Base64 format, ensuring safe data transmission. ‘decouple‘ is employed for managing configuration settings securely, while ‘Flask‘ serves as the foundation for creating web endpoints and handling HTTP requests. Finally, the ‘paramiko‘ library, featuring ‘SSHClient‘ and ‘AutoAddPolicy‘, facilitates secure SSH connections between

your server and remote hosts, enabling secure remote communication and management. I have used SSH connection between our web portal and another server (considering SEM). These libraries collectively enhance server's capabilities, ensuring data security, configuration management, and secure communication with remote hosts.

1.3.4 Database

For the RISE project, a versatile database solution was needed to handle user data, image metadata, project details, instrument data, and storage information. The choice of a NoSQL database was driven by several factors, with the primary one being its schema flexibility. In a NoSQL environment, schemas are either absent or highly adaptable, allowing for the dynamic addition of new data fields—a crucial advantage when dealing with evolving data structures and changing requirements. Additionally, NoSQL databases offer scalability, high performance, streamlined queries (no joins), support for unstructured and semi-structured data, and suitability for big data scenarios. Traditional SQL databases like PostgreSQL or MySQL were deemed less suitable for this dynamic project. MongoDB emerged as the NoSQL database of choice due to its swift data extraction and insertion capabilities, ease of setup, and initial cost-effectiveness. MongoDB stores data in JSON-like documents, excels in scalability, and effortlessly handles vast amounts of unstructured data. Within the RISE project, MongoDB played a pivotal role in storing and efficiently managing the diverse dataset. Its document-based architecture facilitated seamless data querying and filtering, enabling the extraction of valuable insights from the stored information.

1.3.5 AWS-EC2

The Python scripts, Node.js scripts, and React.js scripts were continually run on an Amazon Web Services (AWS) Elastic Compute Cloud (EC2) instance. We used three EC2 instances in this project: one for frontend and backend, one for python machine learning models, and

one for SEM simulation. AWS EC2 provides scalable computing capability in the cloud, allowing users to run virtual machines to execute applications and services. The instance was set up to execute the Python scripts indefinitely and restart automatically if they failed. EC2 also makes it simple to increase or decrease computing capacity as needed, making it suitable for managing enormous amounts of data.

1.3.6 AWS-S3

We used AWS S3 to store SEM image data and user image data. Amazon Web Services (AWS) offers a highly scalable and secure cloud storage service. It employs a flat namespace in which data is organized into ”buckets,” each with its own name. Objects (files) with unique keys can be stored in these buckets. S3 offers a RESTful API and SDKs for several computer languages, making it simple to connect with and manage your cloud data.

1.3.7 Postman

Postman is a popular API testing application that streamlines the process of testing and communicating with web-based APIs. It functions as a full platform for developing, executing, and monitoring API requests and responses. I used Postman frequently for testing the web portal APIs I created with Node.js and Python. Postman is invaluable in this context since it allows for the quick production of API requests, supports authentication, aids in the organization of test collections, and provides capabilities for automating testing and debugging, guaranteeing the reliability and functioning of APIs prior to deployment. It is a versatile tool that speeds the testing process and encourages developer collaboration, making it a valuable asset for API development and quality assurance.

1.3.8 VS-Code-Editor

Visual Studio Code (VS Code), is a popular integrated development environment (IDE), was utilized in the development of project. Here's a breakdown of how tools are used for various aspects of the project.

Python Scripts

VS Code was used for writing and editing Python scripts. Python is a popular programming language, and the IDE provides a conducive environment for Python development, including code highlighting, auto-completion, and debugging features.

Node.js APIs

In addition to Python, VS Code was used for developing Node.js APIs. Node.js is a runtime environment for server-side JavaScript, and VS Code offers excellent support for Node.js development, allowing developers to create APIs efficiently.

React.js for UI Development

It was employed for UI development using React.js. React is a JavaScript library for building user interfaces, and VS Code provides tools and extensions for React development, such as JSX syntax highlighting and code snippets.

Git Repositories

VS Code was used to manage Git repositories. Git is a version control system widely used in software development to track changes in code. VS Code has built-in Git integration, enabling developers to commit, pull, and push code changes without leaving the IDE.

Code Debugging

The code debugging was performed within VS Code. It offers a robust debugging environment with features like breakpoints, variable inspection, and step-by-step execution, making it easier to identify and fix issues in the code.

Extensions

VS Code's versatility is highlighted by the availability of extensions. The extensions for Python, MongoDB, and other libraries used in the project were installed. These extensions enhance the IDE's capabilities by providing support for specific programming languages and technologies. For example, the Python extension offers linting, code formatting, and Jupyter Notebook integration.

Integration with Other Tools

It's extensibility allows for seamless integration with other tools and technologies, such as MongoDB, a NoSQL database often used in web development.

Figure 1.2: Tools and technologies used in RISE project.

Chapter 2

Related Work and Literature Review

In this chapter, we provide some background information on the fundamental concepts behind the RISE. We then discuss the various literature works that have led to the idea and implementation of this research.

2.1 Remote Instrumentation Control

There have been prior works to improve resource efficiency of scientific instruments such as SEM via remote instrumentation. The authors in [7] introduced BRACELET, an edge cloud microservice framework designed to overcome pressing performance and security challenges while allowing access to perform remote instrumentation. Central to BRACELET is the concept of a cloudlet, positioned as an intermediary layer between scientific instruments and a cloud-based analytics service. This cloudlet plays a pivotal role in resolving issues such as network bandwidth and firewall policy limitations, which have the potential to impact the quality of user experience during remote access to scientific instruments. Similar to these efforts, another prior study [2] proposed the Remote Instrumentation and Collaboration Environment (RICE), which addresses the challenges associated with multi-

user remote access to scientific instruments as well as facilitating collaboration tasks for image analytics inspite of potential network bottlenecks.

Our work on the development of the RISE web services for image analytics builds on these notable works. We borrow the cloudlet concept from BRACELET in the web services for programmable instrument control and storage functions that are particularly close to the networks that connect to the scientific instruments. We also seek to balance the performance and security of remote instrumentation as focused in the RICE efforts. The novelty of our remote instrumentation control approach is in the chatbot-guided interface and a suite of web services that elevate the user experience in remote instrumentation, and have the ability to reduce experiment workflow durations, enhance resource utilization, improve image analytics accuracy, and diminish the dependence on specialized expertise for iterative instrument operation during a scientific discovery process

2.2 Image Analytics Automation

There have been prior works to also automate data analysis and instrument control feedback via remote instrumentation. For example, the work in 4CeeD [8] proposes an architecture for real-time data collection, curation, correlation, and coordination for reliable storage and public access involving scientific instruments such as microscopes and manufacturing tools. Similarly, the Smartscope framework in [9] streamlines cryo-electron microscopy grid screening by automating grid navigation and utilizing AI for reduced human intervention. In advanced instrumentation, an AI-AFM system [10] excels in real-time pattern recognition and feature identification, reducing the need for human intervention during scanning. Further, authors in [11] introduced the Image Analytics-Driven Direct Nucleation Control (IA-DNC), a model-free feedback control approach to monitor and analyze crystallization, with the goal of achieving stable and converged crystal shape control.

Our work has similar aims related to above works, and seeks to streamline experiment

efficiency and reduce the need for manual intervention during remote instrumentation. Our image analytics automation approach involves a chatbot-guided interface that borrows best practices from the works in [12] and [13]. Authors in [12] incorporated a chatbot-as-a-service into a user interface for managing options and generating responses for different user-instrument interactions. Authors in [13] introduced HAICO, a chatbot-based system designed for monitoring and managing distributed analytics workflows, which also provides real-time information to users. In our work, our chatbot-guided interface helps users to dynamically generate zoom, focus, and contrast settings using a reinforcement learning-based technique [6] to improve image quality in real-time. Similar to HAICO's capabilities in task automation, our chatbot-based approach helps to continually refine image quality based on user-selected options. Further, our work details how to incorporate security mechanisms in the image analytics automation to improve networked data and instrument access security.

Chapter 3

RISE Web Services And Architecture

3.1 Overview

In this section, we detail the six key components of the RISE web services architecture shown in Figure 3 featuring: (1) remote user access to experiment workspace; (2) web portal services for users to monitor workflows and interact with the chatbot-guided interface; (3) back-end services providing AI/ML-based feedback during image analytics experiments; (4) integration with local lab instrumentation environment integrated via a cloudlet; (5) storage service integrated via a cloudlet; and (6) resource access security mechanisms between SEM, storage repositories, and their respective cloudlets.

3.1.1 Researcher Remote Access

The Researcher Remote Access component enables researchers to access the web service, providing them with the capability to configure and oversee experiments using the chatbot agent. Additionally, it facilitates the evaluation of SEM commands and feedback commands, empowering researchers to select and modify SEM commands for executing new SEM experiments. Through the web services, researchers engage with the user interface

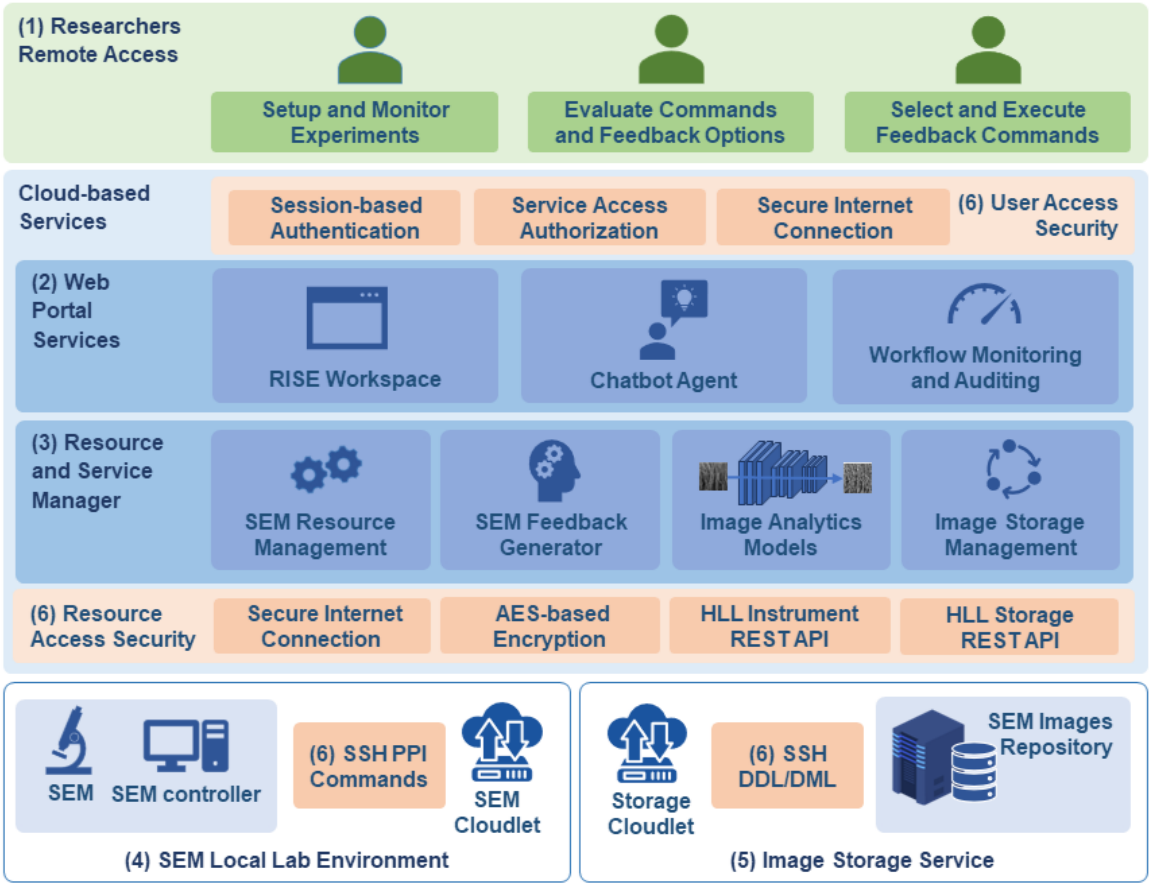

Figure 3.1: RISE web services architecture components.

to initiate actions. The progress and status of experiments are tracked in a command-line style window. Users can choose from existing, raw images or upload images from their local computers for analysis. The image analysis process prompts a reinforcement learning (RL) based chatbot guidance [6] to generate suggestions for improved experiment outcomes. Upon the researcher's execution or acceptance of these new recommendations, a fresh experiment setup is prepared. This setup can be monitored within the web service, displayed in a command-line format window.

3.1.2 Cloud-based Services

Web Components

We hosted a web application in a cloud platform for researchers to manage their image analytics projects and conduct experiments. The RISE analytical module includes the Image Analytic Workflow Management features, User Access Control functions, a Chatbot service, and the Feedback Command Interpreter that allows researchers to interact with the RISE web services. Researchers can also keep track of the progress of their experiments, check to see if any images have been produced by the SEM, preview the images that are already available, submit images for analysis, receive the results from the Image Processing Services, and save the images locally.

The technologies used to build this service are *ReactJS* and *Vite* for the front-end development, *MongoDB* as the database management system, *Postman* for API testing and validation, and *Flask* and *Node.js* on the server side. Hosting for the server is provided by an *AWS EC2* instance, while image storage and retrieval are facilitated through an *AWS S3* bucket integrated into a cloudlet instance. Additional components on the face of the portal are a Dashboard tab that provides a preview of the number of images analyzed in the week; an Instruments tab (see Figure 3.2) that shows metadata of available instruments; a Storage tab that shows different types of storage available e.g., Relational database or cloud storage, and a Projects tab (see Figure 3.3) that shows the current projects, enhancing project management and collaboration. Additionally, the *Flask* application houses our Intelligent agent, facilitating interaction with a Chatbot through RESTful APIs. We used an *x-auth-token header* to authenticate users against the web server.

Each project within the system maintains a collection of images that undergo scanning via the SEM. These SEM images exhibit distinct parameters, including resolution, volume image count, and precision, accompanied by associated metadata. At the backend, the web services are connected to the resource and service manager component where

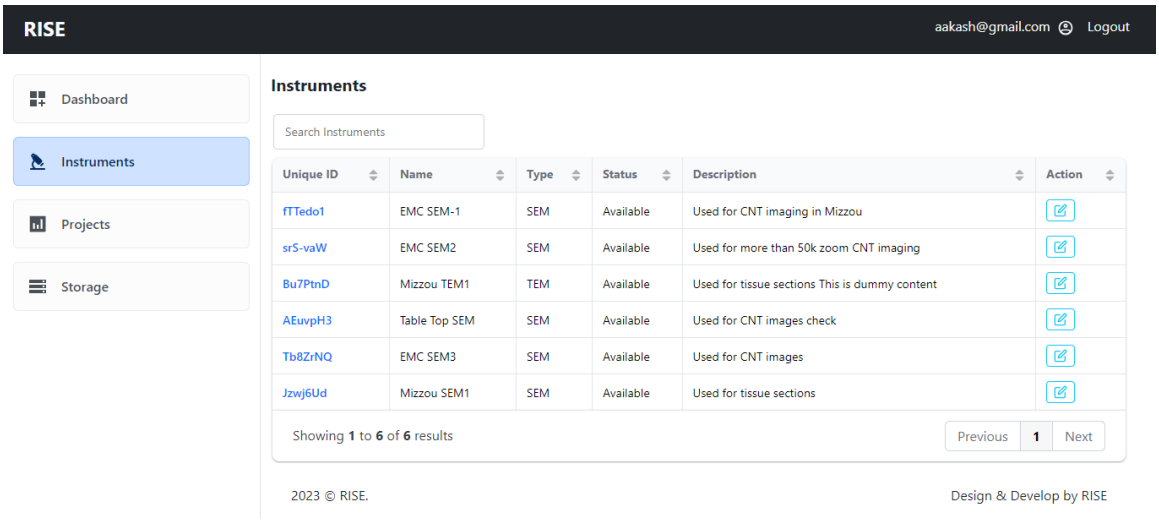

Figure 3.2: Screenshot of the Instruments tab enabled by the web services for displaying a selection of available instruments accessible to the logged-in user.

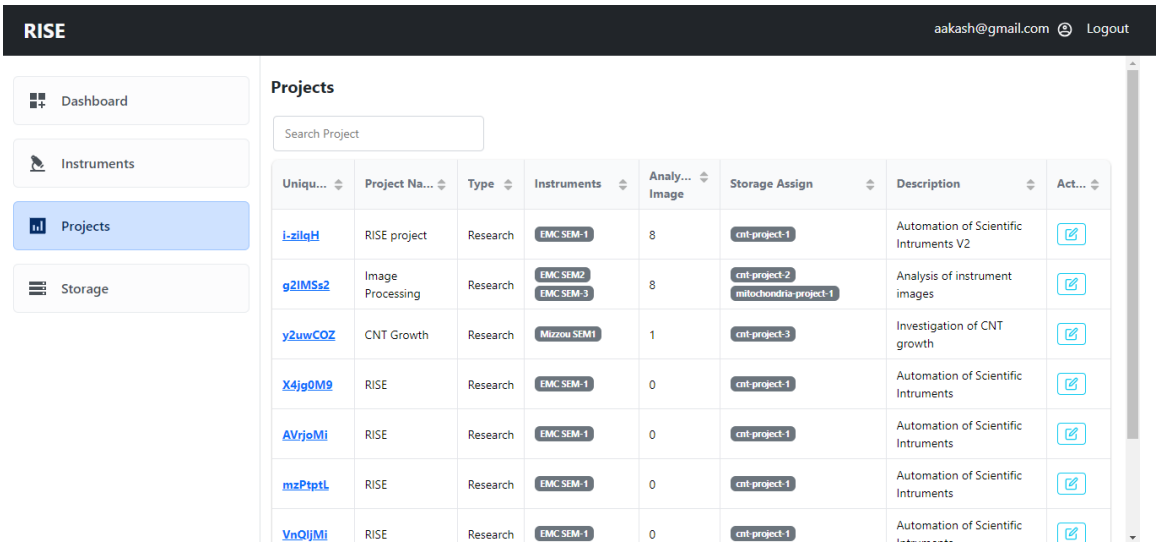

Figure 3.3: Screenshot of the Project tab in the portal, featuring comprehensive details of a chosen project. The web services showcase the project's associated images, accompanied by their respective metadata, including image name, size, storage location, SEM instrument used, and status of analysis completion.

SEM resource management, SEM feedback generator, image analytics models, and Image storage management elements persist. Furthermore, on any specific project page, users are presented with additional options including the "Manage Storage Data" and "Manage Instrument Data" buttons, which provide users with insights into the current storage and instrument choices available for the respective project. Additionally, users are offered the "Scan New Image" button, which grants them access to a dedicated interface. Within this

interface, users can upload images specific to the project, select an instrument, opt for a storage solution, and interact with a chatbot for further image refinement, as illustrated in Figure 3.4. The RISE agent allows users to improve the image quality based on the values suggested by our chatbot agent.

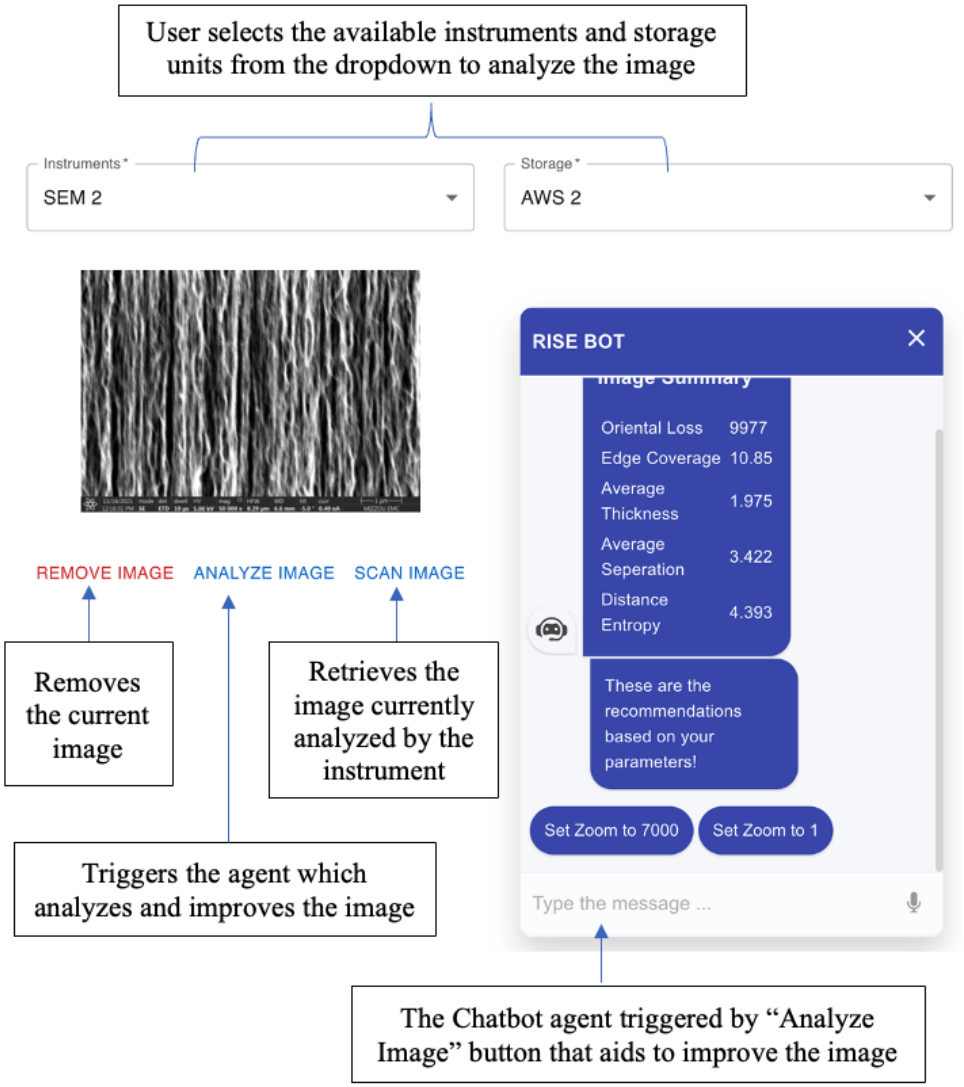

Figure 3.4: Screenshot showing the RL-based recommendations for the SEM Controller post-image analysis to optimize the parameters in image analysis.

Informed by the results of previous experiment iterations and guided by the AI-based Feedback Commands Generator, the Chatbot service extends recommendations to researchers. In accordance with the image chosen by the researcher, the intelligent agent uses reinforcement learning (RL) [6] and provides recommendations for adjustments of SEM controller

in terms of zoom, focus, and contrast. These recommendations are based on the analysis of five critical image parameters: Oriental Loss, Distance Entropy, Average Thickness, Average Separation, and Edge Coverage. Subsequently, the researcher can accept or apply the suggested values, which are conveyed to the chatbot to interact with the corresponding APIs.

Specifically, upon the researcher's selection of the recommended options, the chatbot agent proceeds to generate a new image incorporating the specified zoom, focus, and contrast values as shown in Figure 3.5. If the researcher finds the resulting image satisfactory, they have the option to retain it. However, in cases where the user desires further adjustments, they can select the "generate new image" button. In response to this action, the agent resumes its operations in the background, producing new zoom, focus, and contrast values for the researcher's consideration. This iterative process continues until the user is satisfied with the image quality.

Informed by the results of previous experiment iterations and guided by the AI-based Feedback Commands Generator, the Chatbot service extends recommendations to researchers. In accordance with the image chosen by the researcher, the intelligent agent uses reinforcement learning (RL) [6] and provides recommendations for adjustments of SEM controller in terms of zoom, focus, and contrast. These recommendations are based on the analysis of five critical image parameters: Oriental Loss, Distance Entropy, Average Thickness, Average Separation, and Edge Coverage. Subsequently, the researcher can accept or apply the suggested values, which are conveyed to the chatbot to interact with the corresponding APIs.

Specifically, upon the researcher's selection of the recommended options, the chatbot agent proceeds to generate a new image incorporating the specified zoom, focus, and contrast values as shown in Figure 3.5. If the researcher finds the resulting image satisfactory, they have the option to retain it. However, in cases where the user desires further adjustments, they can select the "generate new image" button. In response to this action, the

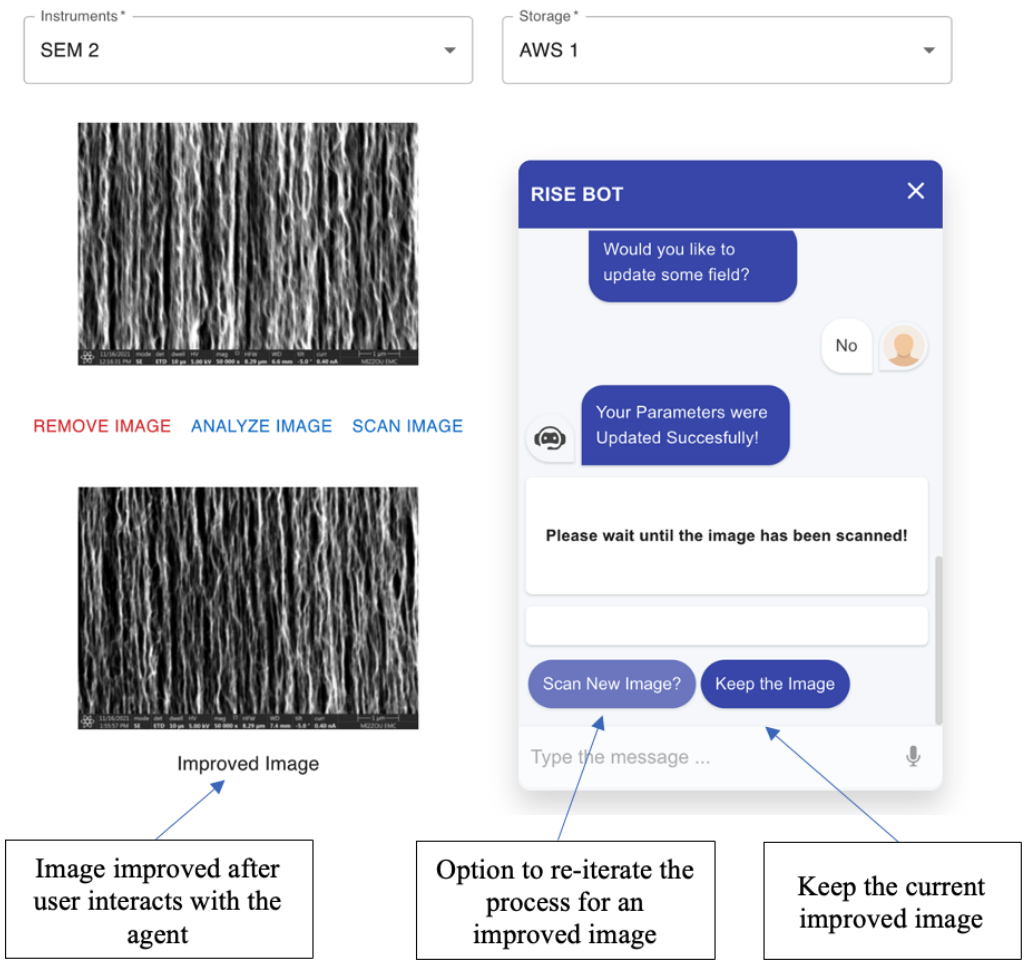

Figure 3.5: Screenshot showing enhanced image post-SEM parameter adjustments using parameters from the RL-based recommendations.

agent resumes its operations in the background, producing new zoom, focus, and contrast values for the researcher's consideration. This iterative process continues until the user is satisfied with the image quality.

Resource and Service Manager

Our Resource and Service Manager comprises of the SEM resource management, SEM Feedback generator, Image Analytics Module and the Image Storage Management. These components are instrumental in streamlining the management of computational resources, storage solutions, and associated services, ensuring efficient resource utilization. When the researcher chooses to analyze an image, it invokes the image analytics models at the back-

end, where we utilize a deep learning algorithm [14] known as CNTSegNet. It is a novel dual loss, orientation-guided, self-supervised deep learning network specifically designed for CNT forest segmentation in SEM images. To evaluate the quality of CNT segmentation, five unsupervised measures are computed, namely orientation loss, edge coverage, average thickness, average separation, and distance entropy, whose sample values are listed in Table 3.1. These unsupervised segmentation evaluation measures play a crucial role in improving CNT forest segmentation results.

Table 3.1: Segmentation quality measures computed for a sample CNTSegNet segmentation output.

Measurement	Value
Orientation loss	9922
Edge coverage	15.64%
Average thickness	01.9965
Average separation	2.4233
Distance Entropy	4.1677

The Chatbot Agent shown in Figure 3.4 shows the image analysis summary, which includes the results of the five image segmentation parameters. Based on the results, it recommends improved SEM controller parameters such as new zoom, contrast, and focus using the module's SEM feedback generator element. The SEM feedback generator employs RL at the backend to improve the image. The new enhanced image is shown in Figure 3.5, which is obtained after modifying the SEM controller parameters based on the chatbot recommendations.

On the other side, image storage management manages the image data as the name suggests. The user-selected image is sent from the storage cloudlet to the image analytic services for analysis, and the image metadata is simultaneously sent to the web service and stored in the image storage service.

3.1.3 SEM Local Lab Environment

By separating the SEM instrument from any external environment, a module hosted at the SEM's site safeguards it from any unauthorized access and related security issues. As a result, the SEM controller and the instrument cloudlet can coordinate for remote integration with the other RISE components. This module controls the Python Programmable Interface (PPI) API between the instrument cloudlet and the SEM controller via an SSH connection. When linked to the SEM Feedback Generator using a secure duplex connection, the HLL instrument REST API provides user commands to modify the SEM controller parameter for a new experiment. Additionally, it creates a simple channel of communication with the storage cloudlet of the image storage service component, where it sends the SEM images for storage.

3.1.4 RISE Chatbot Assistance

The proposed Chatbot service serves as a communication channel within the RISE web services, facilitating vital user interactions during experiment feedback control. Its primary purpose is to foster conversational engagement between researchers and the RISE service functionality as illustrated in Figure 3.6.

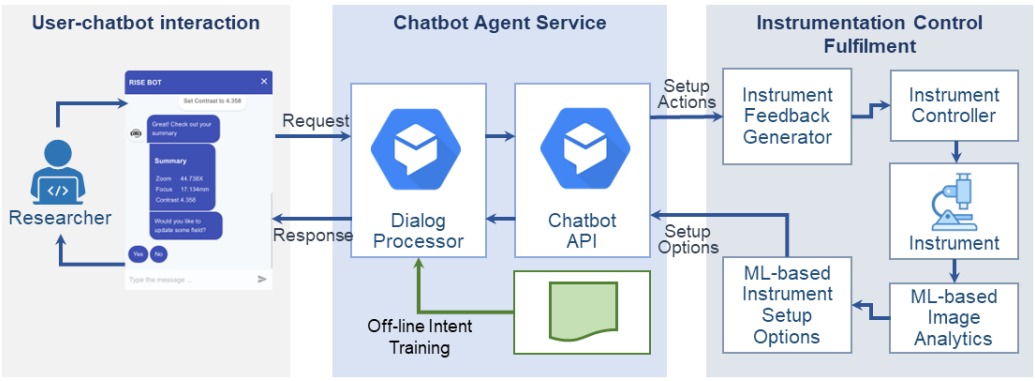

Figure 3.6: RISE Chatbot interaction for users to interact with APIs between the User and the Instrument positioned in the Instrumentation Control Fulfilment.

The Chatbot-Agent Service incorporates a Dialog Processor, a critical component responsible for handling user inquiries and formulating responses. To enhance its conver-

sational abilities, the Dialog Processor leverages intent training, allowing it to understand and interpret user intents effectively. The Instrumentation Control Fulfilment module is integrated with the chatbot, facilitating instrumentation setup and control. This module collaborates closely with the Chatbot-Agent Service to ensure efficient execution of setup actions. This is again connected to a dynamic component that generates feedback commands for instrumentation control.

3.1.5 Image Storage Service

Similar to SEM Local Environment, the Image Storage Service Module is integrated to other components of the RISE via the Storage Cloudlet. It isolates and protects the stored data from any unauthorized access and associated security risks. The storage service and storage cloudlet are both connected via secured *SSH* connections, similar to the local SEM environment. Data Definition Language (DDL) and Data Manipulation Language (DML) commands are used for communication between the instrument cloudlet and the storage service. The storage cloudlet receives the SEM image from the instrument cloudlet and stores it in the SEM images repository. To store and send images, this module communicates with an image storage management that uses a secure connection.

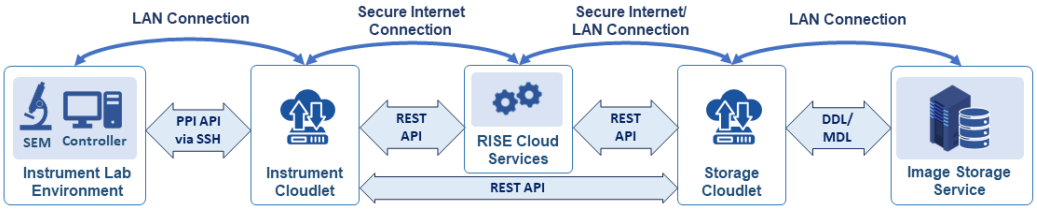

Figure 3.7: Illustration of steps involved in secure Cloudlet communications.

3.1.6 Security Components

Figure 3.7 illustrates the steps involved in the secure communication process between the cloudlets, a key aspect of the RISE web services security architecture. Our goal in the de-

sign of the security mechanisms within our RISE web services is to guarantee the confidentiality, integrity, and availability of both data and instrumentation resources. To safeguard data and maintain system integrity, the RISE uses session-based authentication, which is strengthened by the use of unique session IDs. We incorporate features such as session timeouts, IP tracking, and protection against Cross-Site Request Forgery (CSRF) attacks. Furthermore, the RISE architecture utilizes REST APIs and PPI APIs and a High-Level Language (HLL), with security enforced via the Advanced Encryption Standard (AES) and secure SSH connections. The instrument cloudlet securely converts user requests to original PPI commands and communicates with the SEM controller through SSH over a LAN Connection. SEM controller responds to the cloudlet, which sends data back to the portal via REST API over a Secure Internet Connection. Image data is directly sent from the instrument to the storage cloudlet using Secure Internet or LAN Connection for Image storage Service. Storage Cloudlet and Image Storage Service can communicate using DDL/DML commands over LAN Connection. To ensure secure image data transfer and storage, the RISE maintains encrypted communication channels and employs dedicated ports.

The authentication mechanisms we emply are essential to verify the identity of users or entities accessing a system, ensuring access control to sensitive data and resources while mitigating data breaches and security risks. The work in [15] explained various authentication methods, including Password-based, Multi-Factor (MFA), Certificate-based, Biometric, Session-based, and Token-based authentication. In our web services, we employ Session-based authentication and authorization, generating unique session IDs for authenticated users. The session IDs are stored on the server side, facilitating continued authentication and access to protected resources. This approach incorporates security features such as session timeouts, IP tracking, and anti-CSRF protection. We also incorporate resource protection mechanisms in our RISE web services. In our work, all data shared through APIs is encrypted to ensure confidentiality and privacy. The Cypher class of the

node.js crypto module supports AES-128, AES-192, and AES-256 encryption. We employ the *createCipheriv()* method with algorithm, security key, and random bytes as options to encrypt the data into ciphertext using base64 encoding, completing the process with the *final()* method. For decryption, we follow similar steps, using the same parameters and the *createDecipheriv()* method to unlock the data.

Referring to Figure 3.7, our method aligns with the principles outlined in [16], which introduces a 'Reference Roadmap' for ensuring reliability and high availability in cloud computing environments. This roadmap is divided into four key steps, addressing questions related to 'Where?', 'Which?', 'When?', and 'How?'. Our approach leverages encrypted high-level commands to establish secure API connections between the web service and a Cloudlet, ensuring data privacy and integrity. The Cloudlet, based on request data, decrypts these commands, generating real/low-level commands. In a private network configuration, the Instrument Cloudlet forms a secure SSH connection to the SEM controller, as depicted in Figure 3.7. Simultaneously, the Storage Cloudlet connects securely to private network storage, ensuring that original commands do not traverse the public network. This strategic configuration enhances system integrity, data accuracy, and availability, safeguarding these aspects exclusively for authorized users.

3.2 Database

MongoDB is a NoSQL database management system known for its flexibility, scalability, and performance advantages over traditional SQL databases. Here are some key characteristics and reasons why MongoDB is often considered a better choice for certain applications: Flexibility, Scalability, Query Performance, Aggregation Framework, No Joins, and Open Source. In this RISE project, the database structure is crucial for efficient data management. To achieve this, I designed and implemented five distinct collections, each serving a specific purpose.

user_data	
email	String
firstname	String
lastname	String
address	String
contact	String
inputZip	String
inputCountry	String
password	String
createdAt	Date
updatedAt	Date

storage_data	
bucketName	String
type	String
usage	String
imagecount	String
createdAt	Date
updatedAt	Date

Figure 3.8: `user_data` and `storage_data` schema

imageMetaData	
uniqueId	String
projectId	String
filename	String
fileType	String
filepath	String
resolution	String
status	String
size	String
storage	String
instrument	String

imageMetaData	
edgeCoverage	String
OrientationLoss	String
averageThickness	String
averageSeparation	String
distanceEntropy	String
contrast	String
focus	String
zoom	String
createdAt	Date
updatedAt	Date

Figure 3.9: imageMetaData schema

These collections are named: `imageMetaData`, `instrumentData`, `projectData`, `user_data`, and `storage_data` shown in Figure 3.8, 3.9, 3.10. The `imageMetaData` collection houses essential information about the images we use in the project. The `instrumentData` contains data relevant to the instruments employed, while `projectData` centralizes details about the projects themselves. The `user_data` collection focuses on user-related information, ensuring efficient management of permissions and access. Lastly, the `storage_data`

collection is integral for secure and organized data storage. This database structure not only streamlines data storage and retrieval but also enhances the overall project performance and data integrity, ultimately contributing to the success of the RISE project.

instrumentData	
uniqueId	String
name	String
type	String
description	String
status	String
IPaddress	String
createdAt	Date
updatedAt	Date

projectData	
uniqueId	String
type	String
description	String
projectName	String
imageAnalyzed	Number
imageScanned	Number
instruments	Array
storageAssign	Array
createdAt	Date
updatedAt	Date

Figure 3.10: instrumentData and projectData schema

3.3 Web Services REST API's

Representational State Transfer, or REST for short, is like a set of rules for how the internet talks to itself. It's a way to access web services simply and flexibly. It's a bit like sending and receiving messages. When you want to ask for something or give something to a website, you send a request in a specific way, just like typing a web address. The website then sends you back a response, which can be in various forms like web pages, pictures, or data. Nowadays, we mostly use a format called JSON for this communication, which is easy to work with and efficient for the internet. So, REST is like a universal language for web services, and it's all about sending and receiving messages using web addresses. In our project, we have used REST APIs where every API returns a JSON response. In JSON response status and msgType are sent with other response attributes, where status 1 reflects

API ran successfully without any errors and msgType is “success”. If anywhere API breaks the status is sent as 0 and msgType will be “fail” One more attribute will be sent as msg containing a short error message.

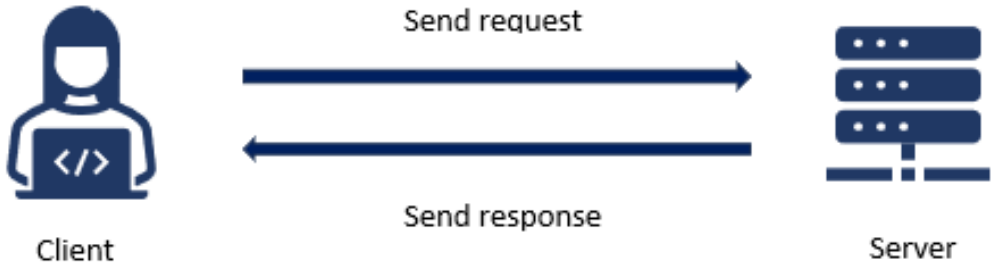

Figure 3.11: Client sending a request to server and server sending a response to the client using REST API.

3.3.1 /register

To enter the web portal user has to register and then only be logged in with the valid registered credentials. To register go to the register page and click “Register Now”. On the register page when a user enters the details and clicks on “Sign In” the API at the backend performs the logic. This POST API first checks if the email is already registered with us and data is available in the database. If it’s true it sends a response as the User email already exists. If it does not, step 2 compares the user password making sure the entered password is correct. If the password does not match it shows the error “Passwords Incorrect”. Once the user fills in the correct data, the password is encrypted using the same bcrypt library we discussed above and stored in the database with other details. The POST API payloads and response format are shown in Figure 3.12, with the Postman tool with response status 200. Figure 3.13 describes the flow of register API.

3.3.2 /login

The login API is the very first API which is used to login to the web portal. This API takes two parameters in request data which is email and password. The internal logic

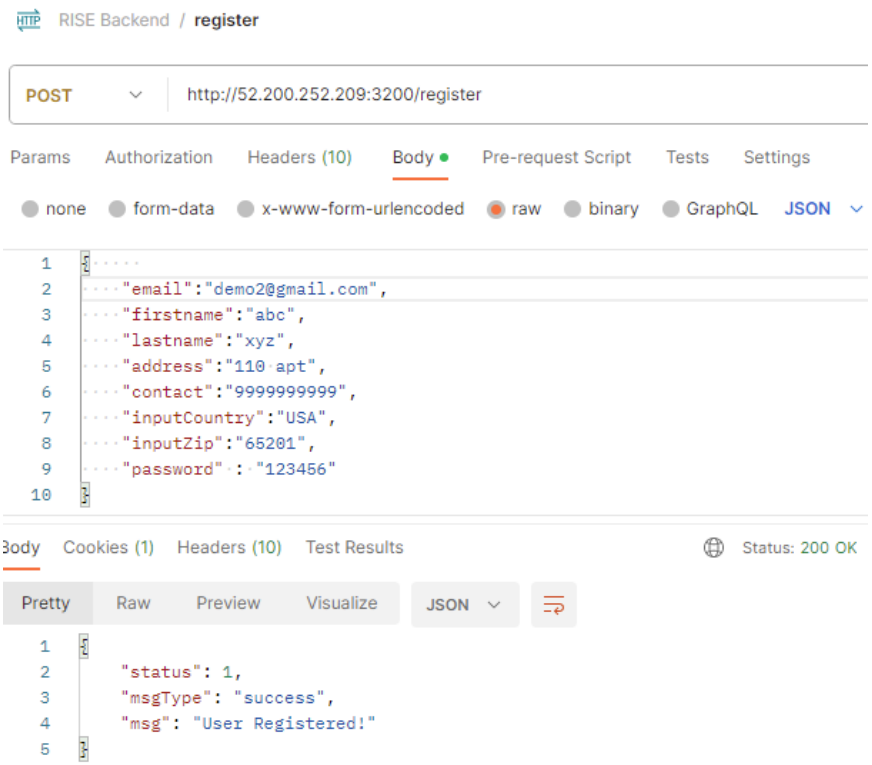

Figure 3.12: register API response and request JSON data.

authenticates the user based on the data provided. At step 1 it checks in the database if the user is registered with us or not. If the user is not registered it sends the response user is not registered but if it exists in database, it verifies the password stored in database in the encrypted format using the bcrypt library. The ‘bcrypt‘ library in Node.js provides a secure way to hash passwords by converting them into irreversible, fixed-length strings using a computationally intensive hash function. It incorporates a random ”salt” to enhance security against attacks, and its adjustable work factor slows down hash generation to deter brute-force attacks. To hash a password, ‘bcrypt.hash()‘ is used, while ‘bcrypt.compare()‘ compares a plain-text password to a stored hash. If the given password is verified, the user is redirected to the portal’s dashboard and a session is created for 24h with a unique session ID. The POST API payloads and response format are shown below in Figure 3.15 and the flowchart is shown in Figure 3.14.

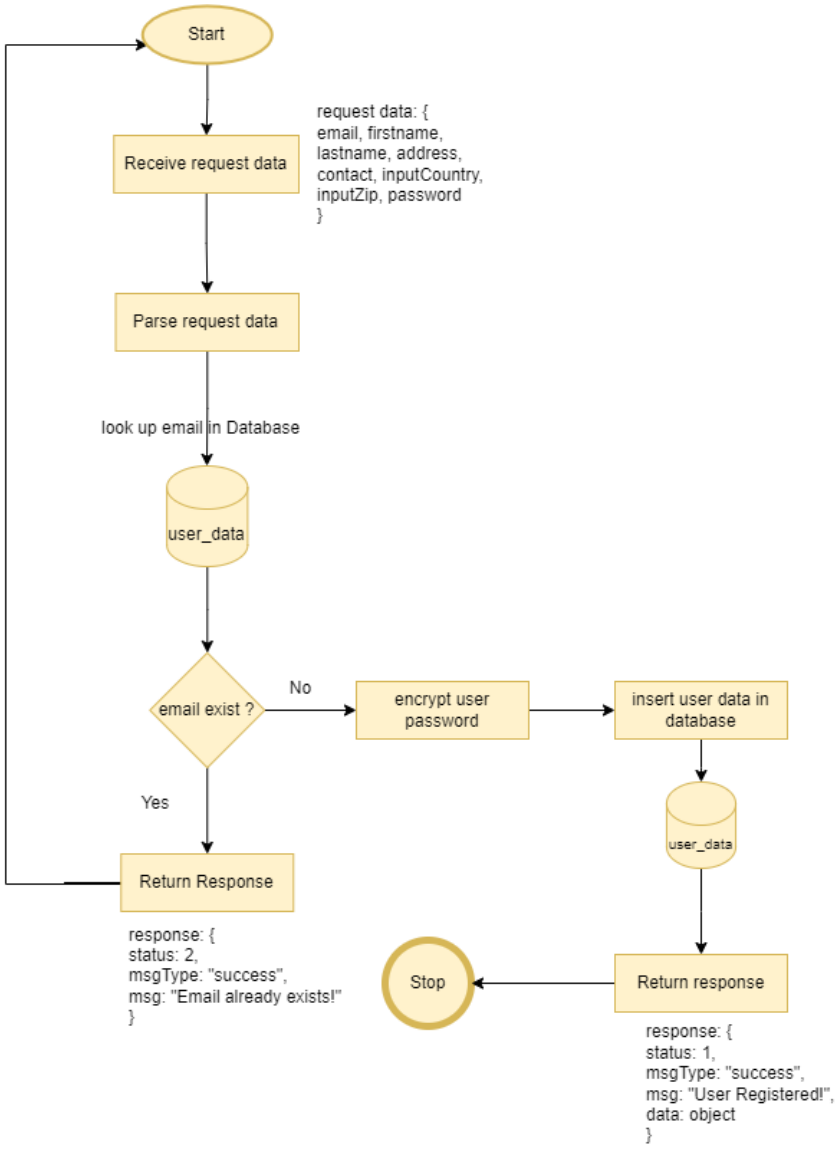

Figure 3.13: register API response and request JSON data.

3.3.3 /dashboard

This is GET API that sends data from last week's each day about how many images were uploaded by researcher's personal computer and how many images were analyzed by the ML model. The API connects to the MongoDB databases's collection `user_data` and fetches the count of analyzed images and uploaded images by logged in user. Api status and responses can be seen in the Figure Figure 3.16.

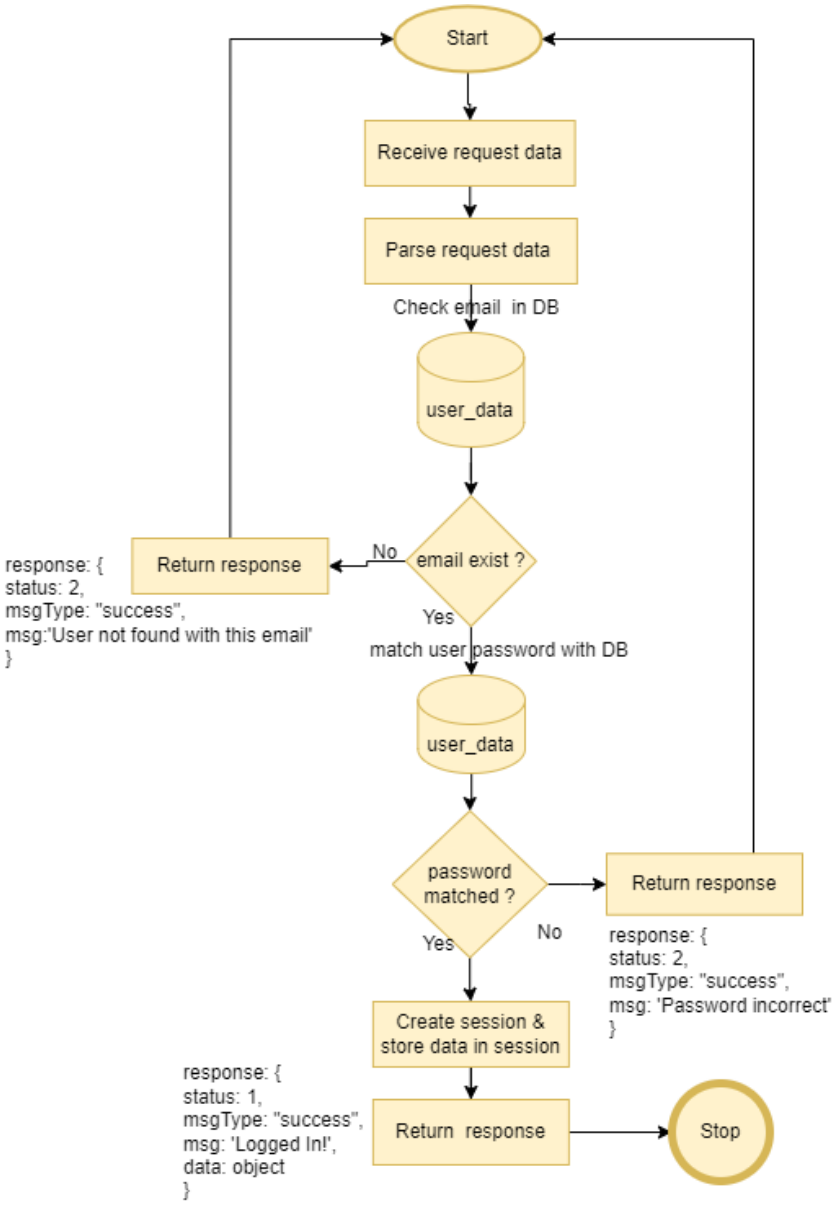

Figure 3.14: Login API flow chart showing all validations

3.3.4 /allProjects

This is GET API it fetches all the available projects from the database that are assigned to the researcher with the details of the project such as project name, unique ID, project type, project description and storage, and instruments assigned to the project. See the image below for the API response from the backend EC2 server. In response, it can be seen their status is 1 which means no errors and msgtype is “success” saying API ran successfully,

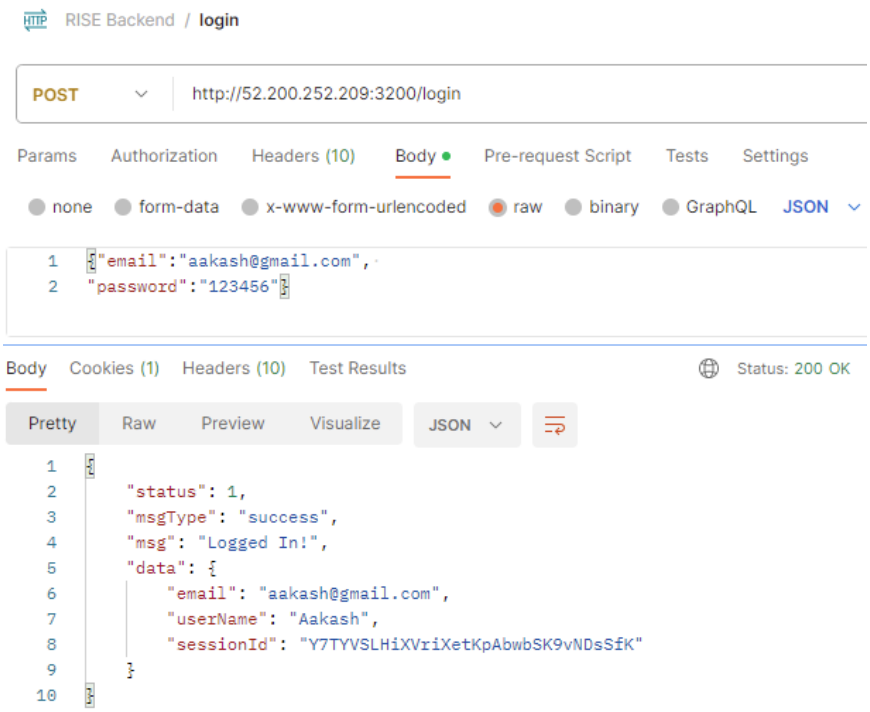

Figure 3.15: login POST API payloads and response for valid credentials

and data contains the project data as an array of objects. Figure 3.17 shows API payloads and responses.

3.3.5 /updateProject

In the project tab project name, project type, and project description can be changed and updated using this API. This POST Rest Api takes project's uniqueId, type, description, projectName as requested data. It updates the database collection of project data in Mongo DB. Payloads of Api and status code can seen in the below Figure 3.18.

3.3.6 /getAllImages

It is very crucial in the RISE system to know what image data is available under each project. This POST API is called to show all the image data along with image metadata present in a project. The project's unique Id is the only payload sent from the client side

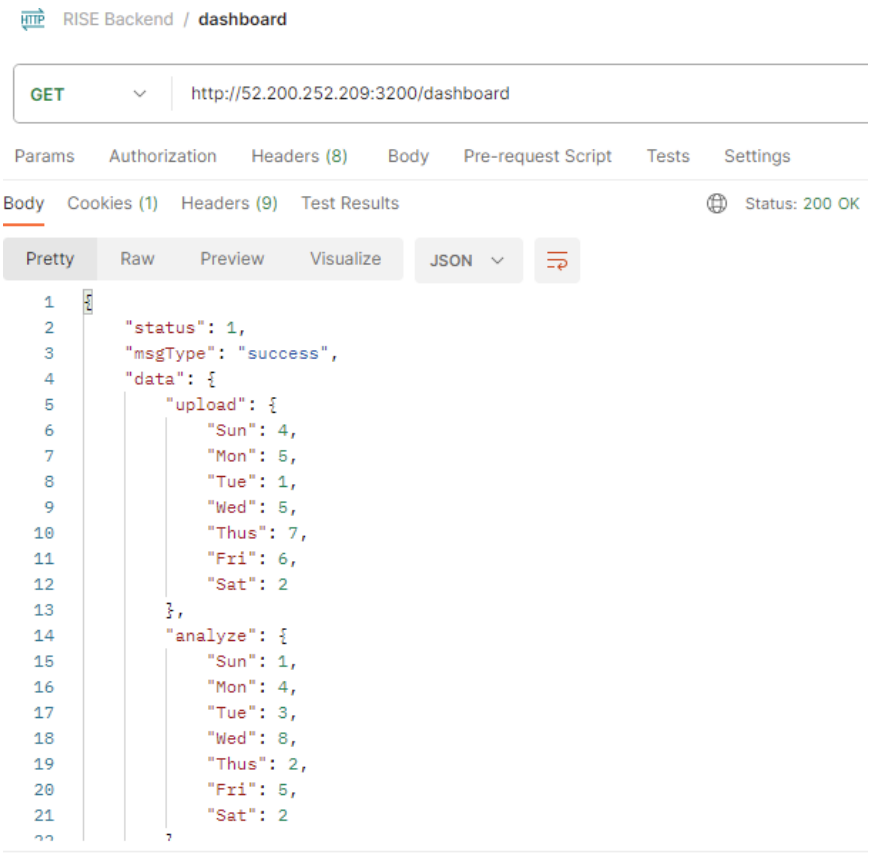

Figure 3.16: : Dashboard API showing the response of last week's image upload and image analyzed data by researchers.

and server response data is an array of objects containing all the image metadata. The API fetches data from MongoDB. API payloads and responses can be seen below in Figure Figure 3.19.

3.3.7 /deleteImage

As the name suggests deletion is performed in this API. Researchers can delete the image data from the storage. This POST API deletes the image from the S3 storage. After deletion is performed it updates the imageMetaData collection of mongoDb. API responses and payload can be seen in Figure 3.20

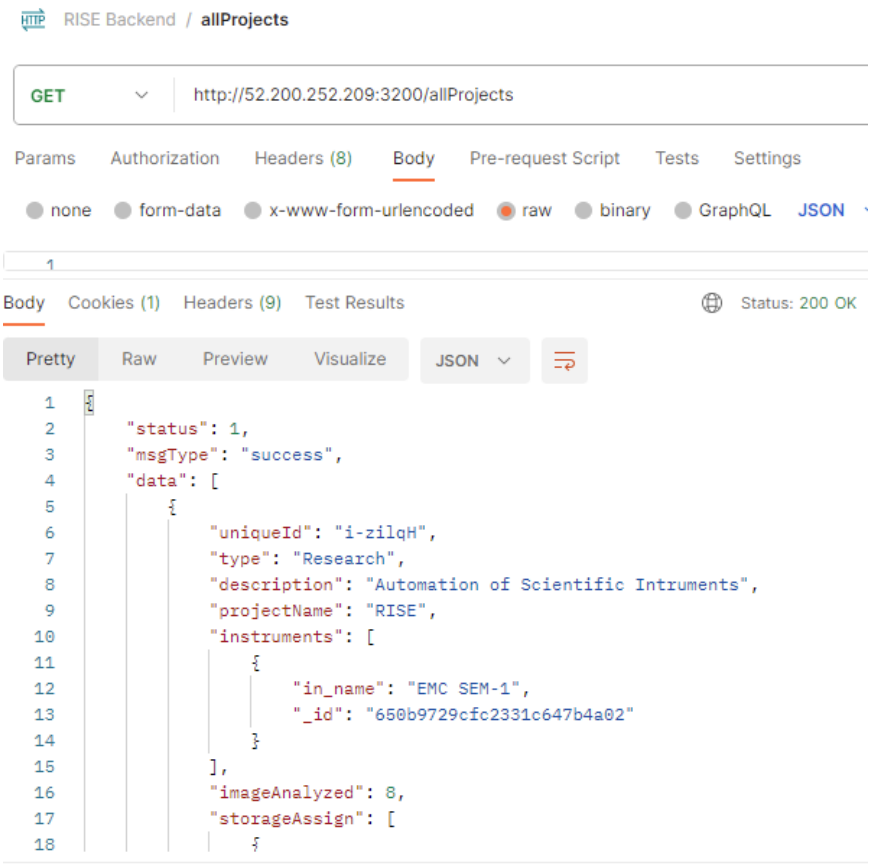

Figure 3.17: allProjects API response and request JSON data

3.3.8 /showImage

This API is used to show the image content in the web portal. It connects to the S3 bucket and finds the image based on the given request data. In response, it sends the image data in base64 format. API testing is shown in Figure 3.21

3.3.9 /analyzeImage

This is the most critical API of RISE web services, this allows researchers to connect to the ML/AL model for image analysis and provides the image analysis summary. It also provides new recommendations for SEM controllers to get better experiment results. It takes the requested data as an image filename, unique ID, and storage name where it is stored. Encrypts this data and sent to flask server where ML model is used to analyze

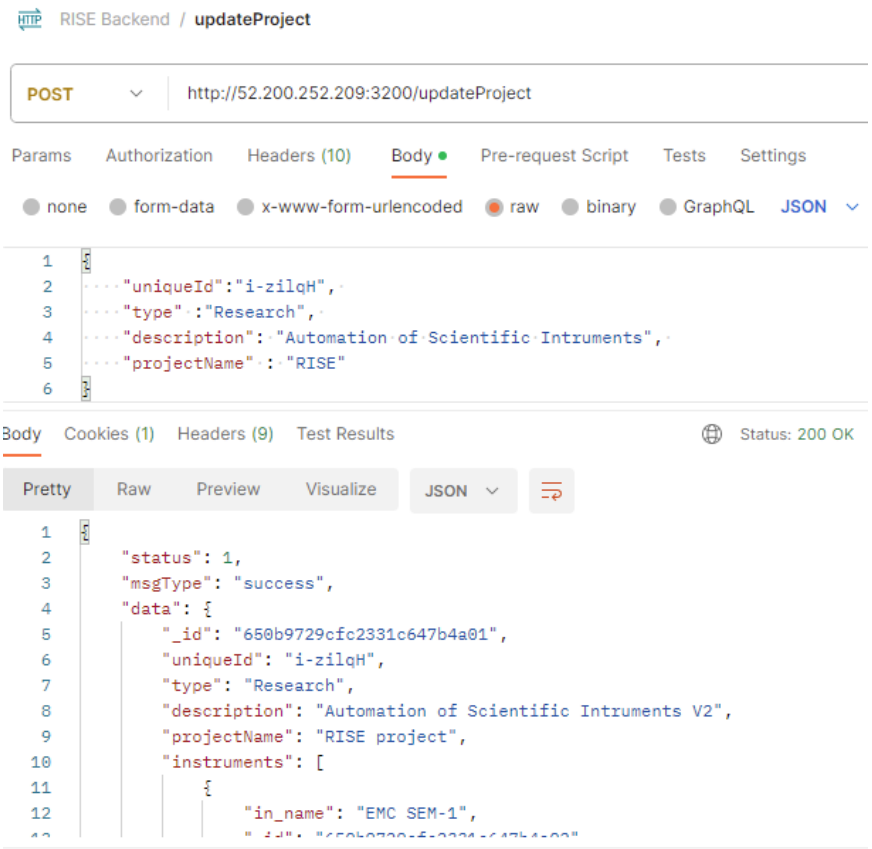

Figure 3.18: updateProject API status 200 with payloads and response.

the image. The Flask server decrypts this data performs analysis for the fetched S3 image and returns the encrypted data to web API. The response is decrypted and imageMetaData collection is updated with the new analysis results and other details. After updating the response is sent to the client. API testing is shown in Figure 3.23 and the flow of API is described in Figure 3.22

3.3.10 /scanButton

To scan a new image from SEM, the user needs to click on the Scan New Image inside the specific project. Clicking the button will trigger this POST API that takes projectId as request data and provides the storage and instrument assigned to the project as the response. This connects to the MongoDB to fetch data. This data is shown in another tab to select instrument and storage before scanning a new image from SEM. The API testing can be

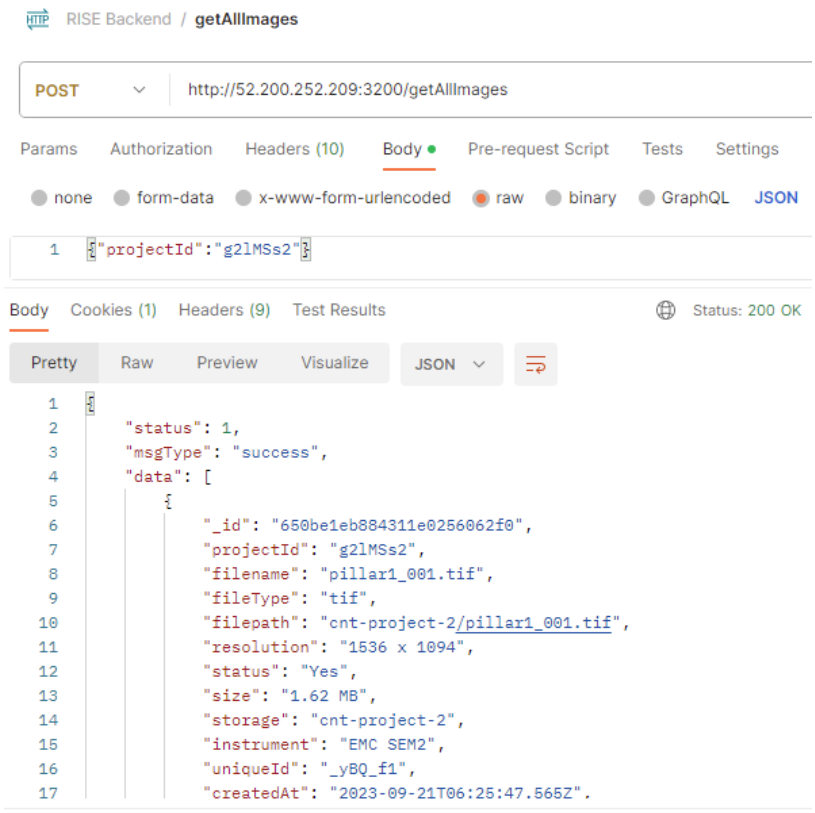

Figure 3.19: All image metadata is sent as response JSON available for a project.

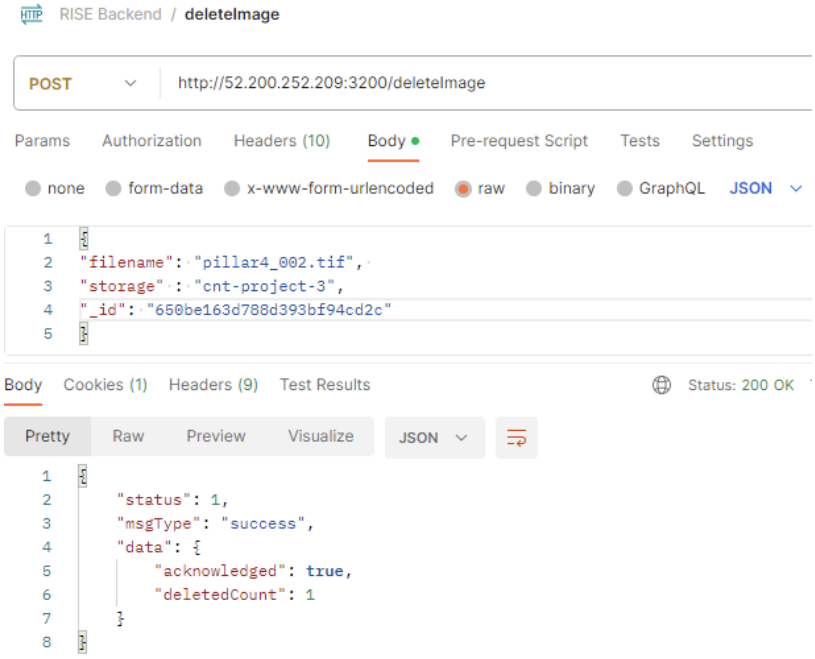

Figure 3.20: deleteImage API payloads and response where data is an object showing deleteCount as 1

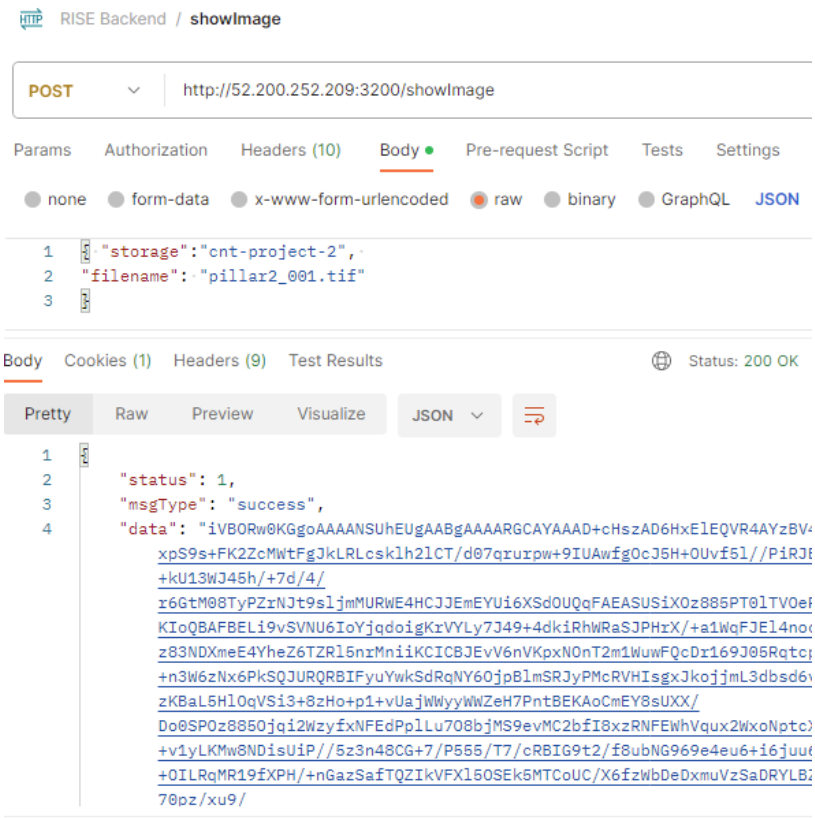

Figure 3.21: showImage API response of image coming from S3

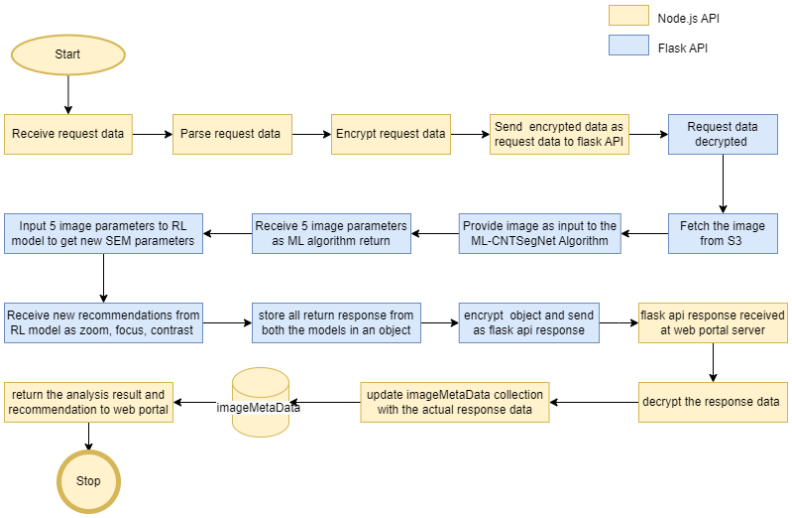

Figure 3.22: analyzeImage post API flowchart.

seen in Figure 3.24 using Postman.

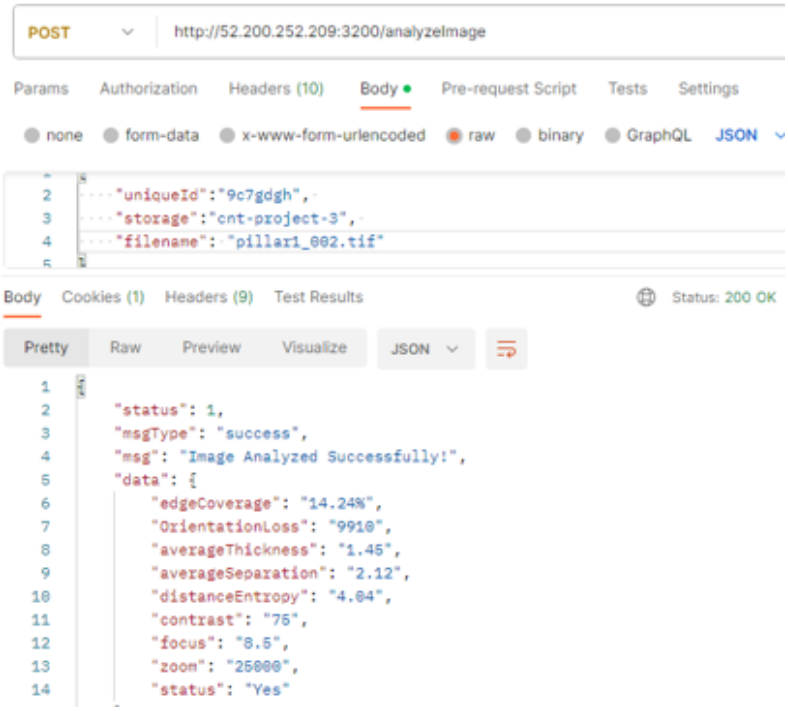

Figure 3.23: Image Analysis and new recommendation are shown in response of analyzeImage response.

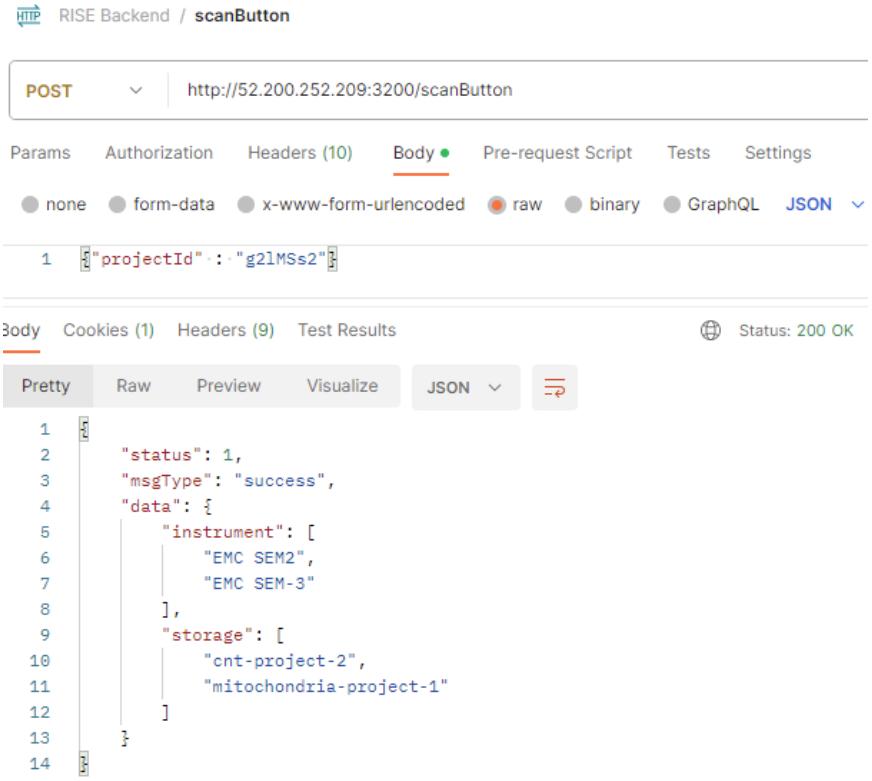

Figure 3.24: Response data of scanButton API with projectId as payload

3.3.11 /upload

This Node.js API, accessed via a POST request to "/upload," uploads images to an S3 bucket and stores relevant details in MongoDB. The uploaded image is read as a buffer from req.file.buffer. Its size in bytes and content type (mime-type) are obtained as well. The image size is then converted from bytes to megabytes (MB). It extracts project, storage, and instrument info from the request. The image's size is calculated and uploaded to S3. The code retrieves existing storage usage data from the MongoDB database based on the storage parameter. It calculates the updated usage by adding the size of the newly uploaded image and updates the storage usage and image count in the database. Metadata-like resolution is read using the "sharp" library. The image data is structured and added to MongoDB. The API responds with a success status and the inserted document. API response can be seen in Figure 3.25

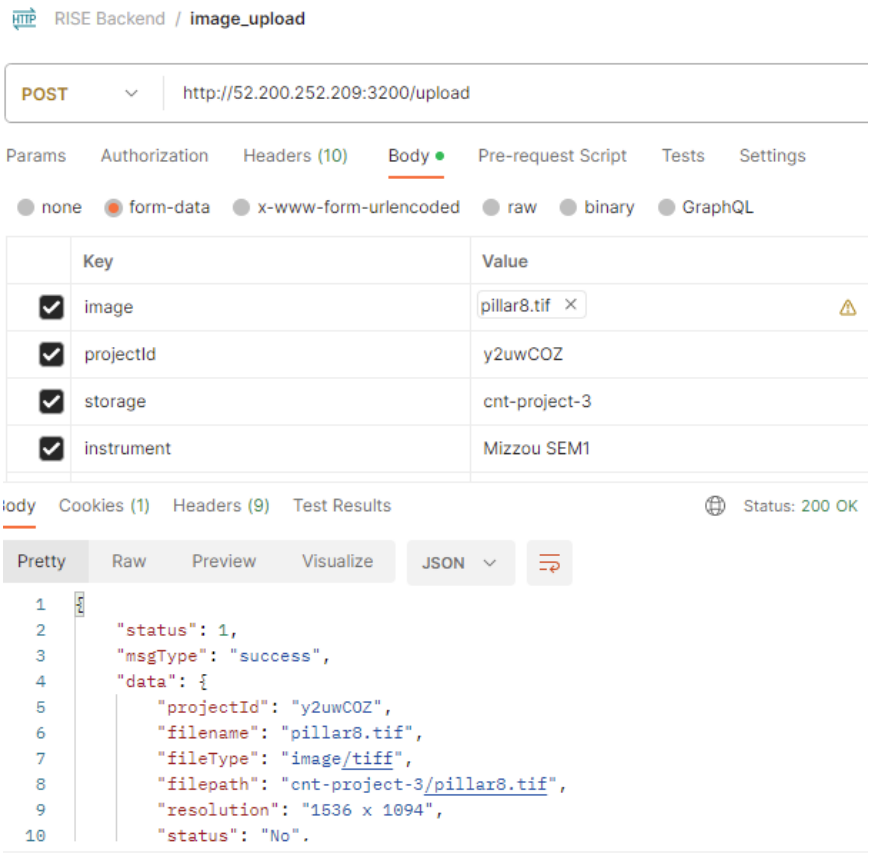

Figure 3.25: upload Api showing success response and uploading an image using multer library.

3.3.12 /semScan

This API processes instrument scans, sending encrypted data to instrument cloudlet where it connects to PPI API using SSH connection for new SEM image scan with the given new parameters (zoom, focus, contrast). PPI API returns new image data. The cloudlet image stored in the S3 sends an encrypted response. This response is decrypted at the portal server and fetches the newly uploaded S3 image. It then calculates image metadata, updates storage information, and stores the image data in a MongoDB database. Finally, it responds with success and the image data in base64 format or an error message in case of issues. API response and API flow can be seen in Figure 3.26 and Figure 3.27

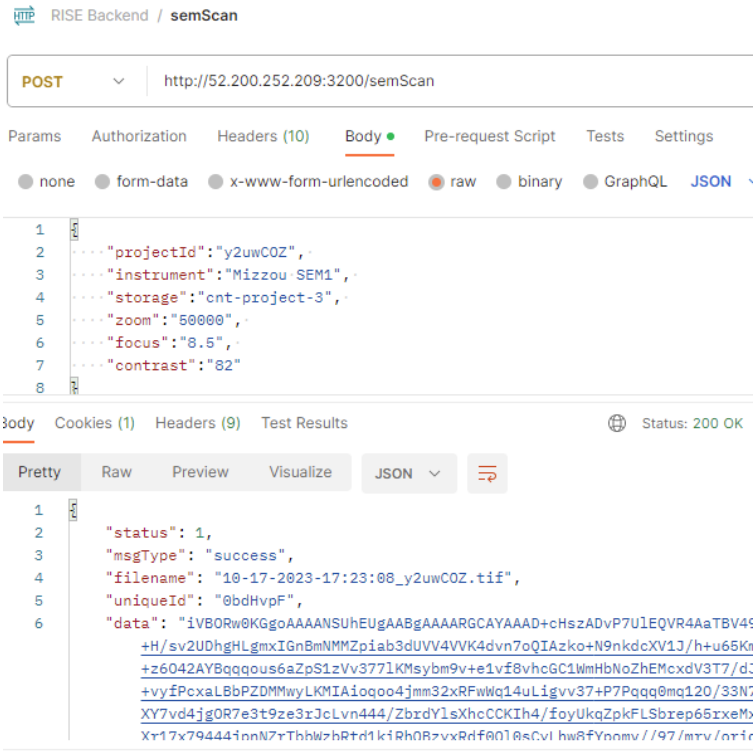

Figure 3.26: New Sem scan image data as response from semScan API

3.3.13 /getStorage

This is GET API it fetches the storage information from the mongoDb database collection and as a response, it sends all the array of objects. API response can be seen in Figure 3.28

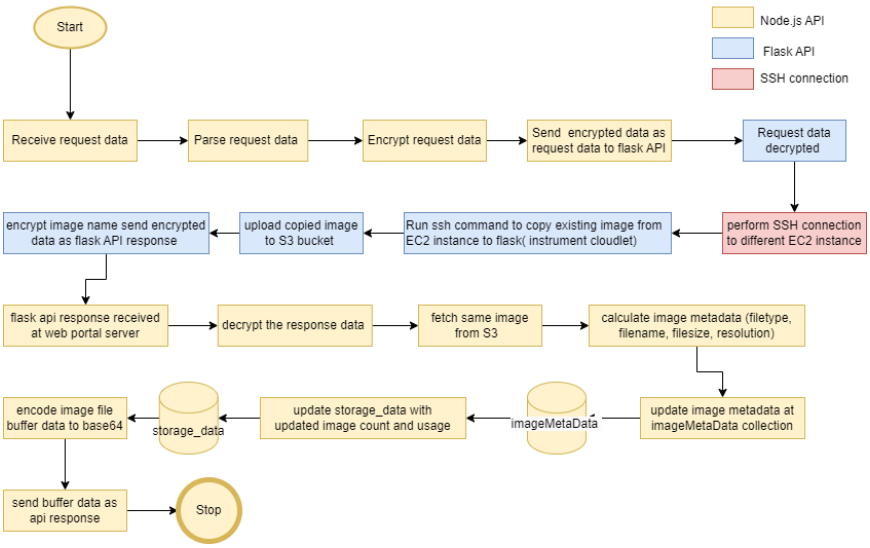

Figure 3.27: semScan POST API flowchart

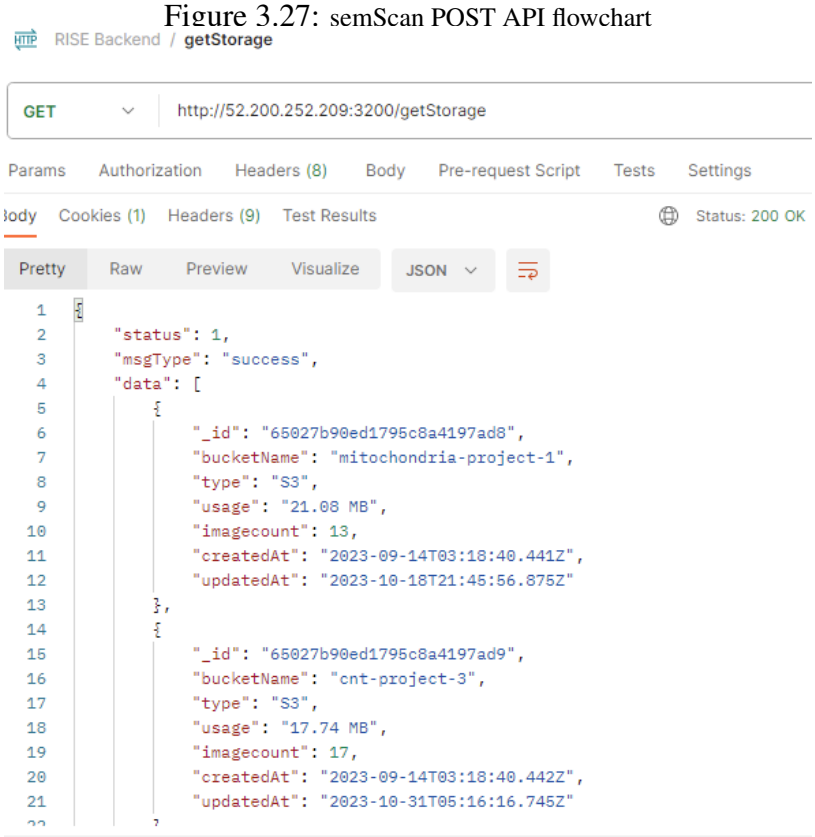

Figure 3.28: All the available storage data is sent as a response from getStorage API

3.3.14 /getinstruments

To fetch all the instruments assigned to the researcher this API is used. This is a GET API that connects to the MongoDB collection called instrumentData. The collection stores the

information of instruments such as name, type, description, and status of instruments. In response to client requests it returns all the instrument's data. API response can be seen in Figure 3.28

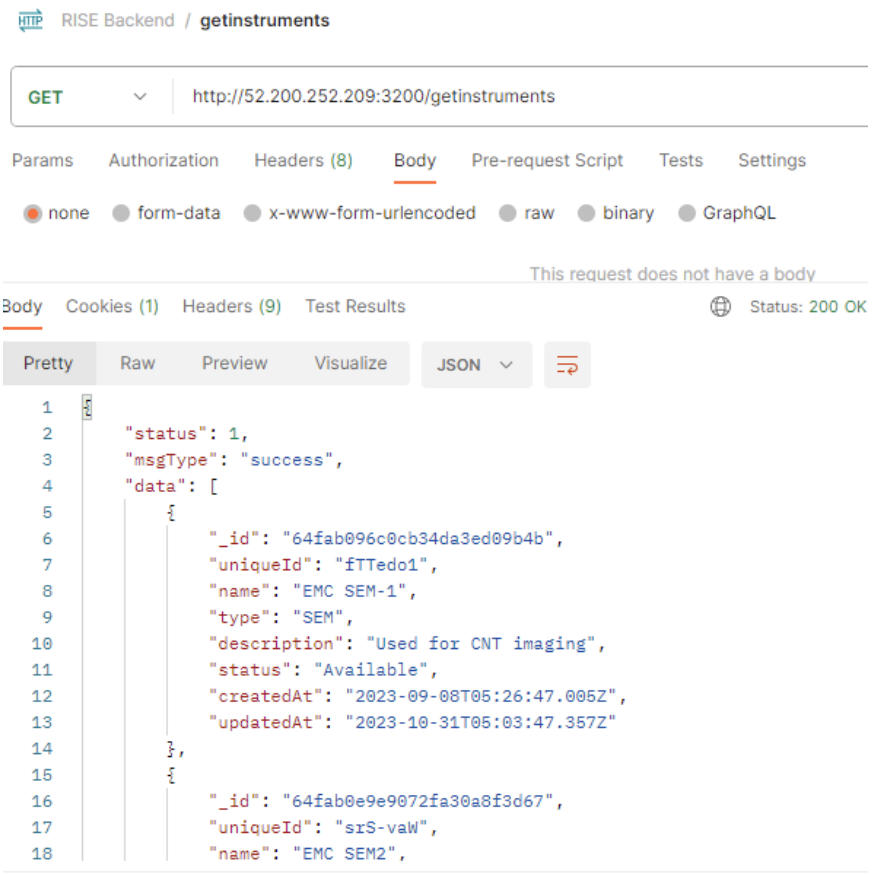

Figure 3.29: Instrument data shown as the response of getinstruments API

3.3.15 /updateInstruments

Instrument's description and name can be updated by the researcher using this POST API. It takes `_id`, name, and description as request data connects to the mongoDb and updates the instrument details. API response can be seen in Figure 3.30

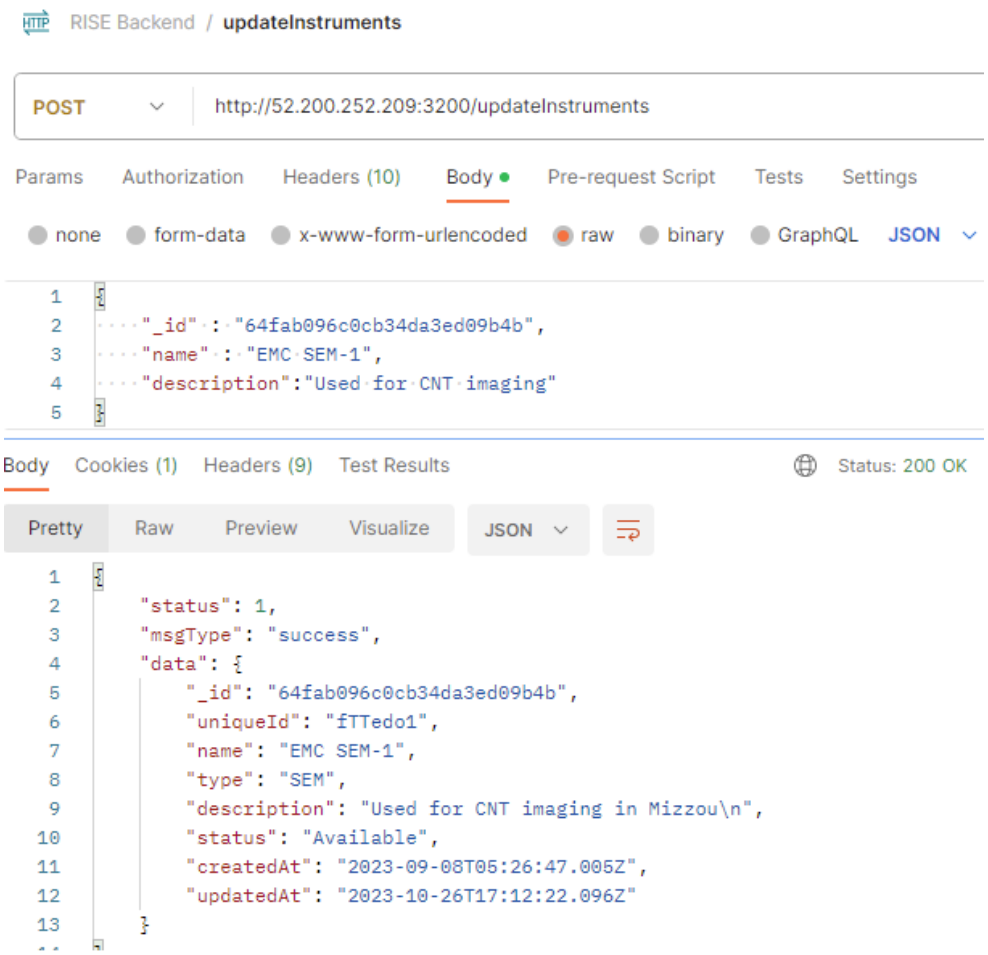

Figure 3.30: Instrument data shown as the response of getinstruments API

Chapter 4

RISE Web Services Evaluation

In this section, we describe our deployment of the RISE system on a cloud platform testbed and a usability study for a CNT image segmentation use case.

4.0.1 RISE Web Services Testbed Implementation

Figure 4.1 shows the RISE testbed-related cloud infrastructure on which we implement and evaluate our RISE web services. This testbed combines various elements, including a Phenom SEM instrument in a research laboratory at the University of Missouri (MU), the MU Lewis HPC cluster for AI-ML model training, and AWS for cloud-based services. The SEM instrument communicates with the cloud-hosted RISE web services through the SEM Cloudlet, acting as an intermediary. It translates commands between RISE web services and the SEM instrument using HLL commands to configure instruments and collect scanned images. Images are transmitted via a REST API to an *AWS S3* bucket, serving as the cloud-based image repository. We streamlined image storage by implementing an interface in our image storage cloudlet, optimizing storage and retrieval. The CNTSEgNet [14] model was developed and trained at the MU Lewis HPC cluster. Training CNT images are stored locally at the HPC facility. After training, the model is transferred to a cloud-

based compute instance for assembling CNT segmentation analysis. The RL-based SEM feedback control model is trained in a local machine and then transferred to the testbed to analyze the five resultant parameters from the CNT segmentation model and generate the most suitable values for zoom, focus, and contrast to set the SEM configuration to improve the quality of the image on a next iteration of the experiment.

The RISE web services run on two *AWS EC2* virtual machine instances. The first instance hosts the ML models to analyze new CNT images from the S3 image repository by performing segmentation using CNTSegNet. It also hosts the RL-based model that generates suggestions for the zoom, focus, and contrast settings to enhance image scan quality. The second *AWS EC2* instance hosts the web service and chatbot services. They provide the interface for researchers to define SEM parameters for zoom, focus, and contrast with the aid of the chatbot. Researchers can then confirm the final settings to assemble the feedback SEM commands to be sent to the instrument via a REST API. Upon completion, the new scanned CNT image is stored in the *AWS S3* bucket, triggering notifications to the web services. Researchers can review the images and iterate with the chatbot assistant to steer new adjustments on the SEM as needed.

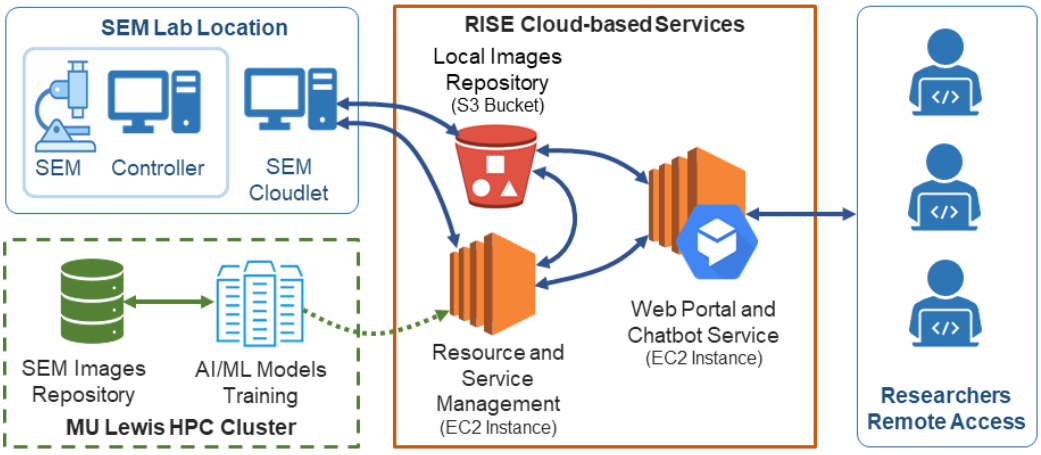

Figure 4.1: RISE testbed infrastructure showing the remote lab location of the SEM, related controller, and Cloudlet with Internet access; the off-line AI/ML training environment hosted by the MU Lewis HP Cluster; the cloud-based RISE services including a host for the ML analytic models, local image repository, and host for the web service and chatbot service.

4.0.2 Usability Study

To evaluate the effectiveness of our RISE web services approach, we conducted a usability study by following a methodology outlined in [17]. To gather insights on our RISE web services, we administered a user survey comprising three distinct questions, described in the subsequent subsections. A total of 7 researchers participated in this survey, who engaged in activities related to SEM instrumentation operation for CNT synthesis experimentation, and CNT image collection and analysis. The questions below were formulated and answered as described in the examples for each item.

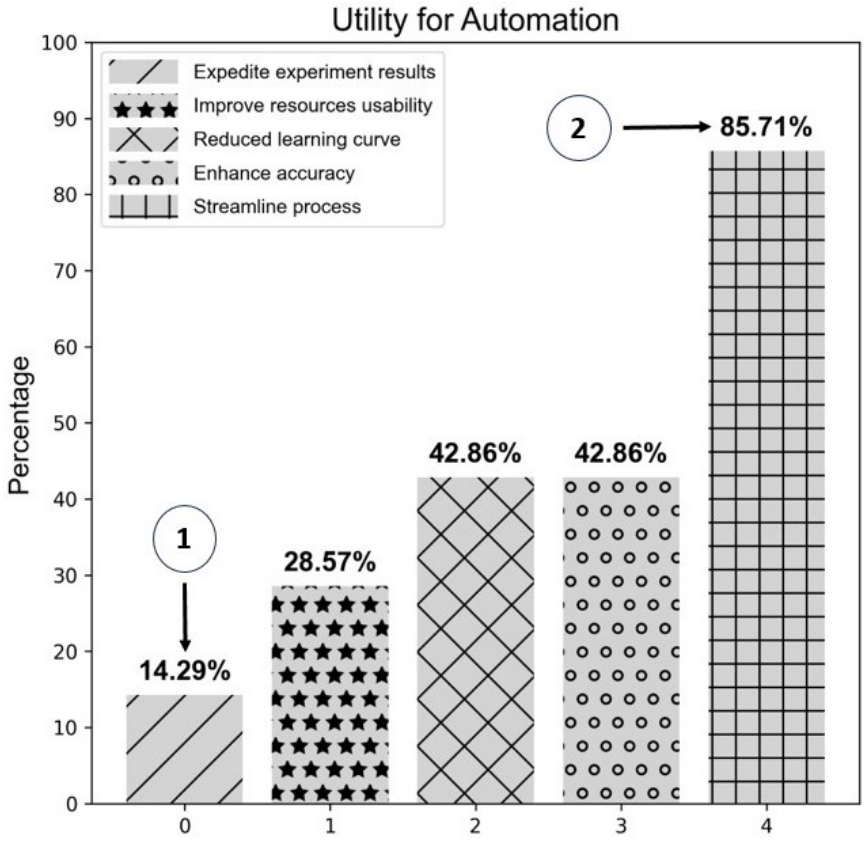

Figure 4.2: Results for the five distinct aspects of research utility for automation showing "Expedite experiment results" being a relatively minor feature (denoted as 1), while "Streamline process" emerges as the major feature (denoted as 2).

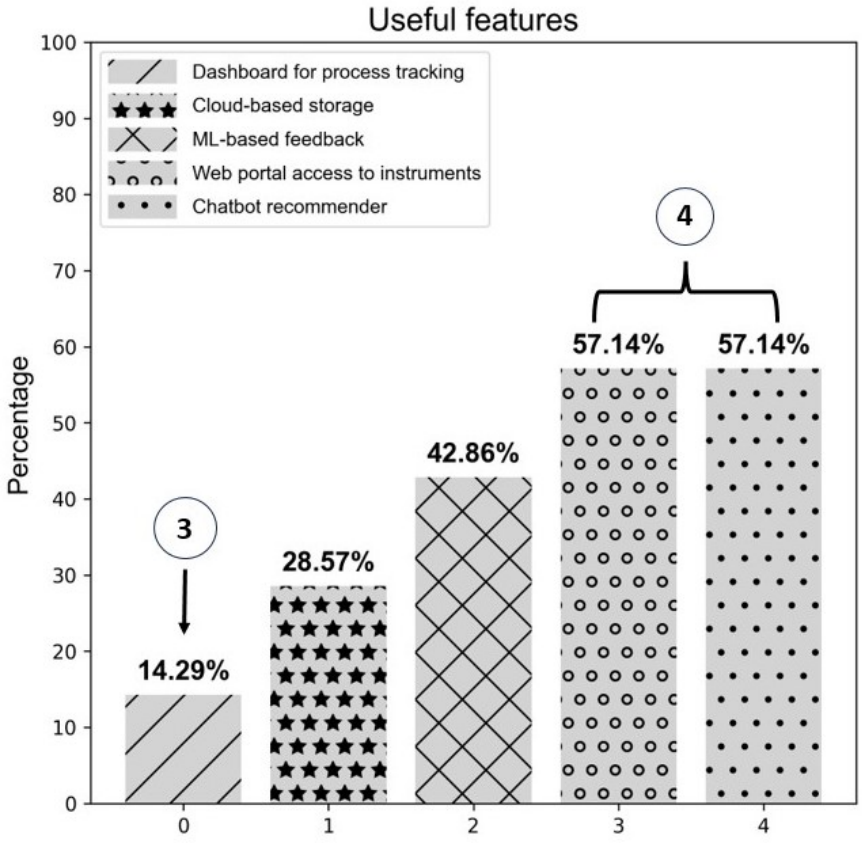

Figure 4.3: Results for the useful features for user experience across five distinct aspects showing ”Dashboard for process tracking” is considered a minor feature (denoted as 3), while both ”Chatbot recommender” and ”web service access to instruments” emerge as major features (denoted as 4).

Utility for Automation evaluation

In response to the question *“Did the tool demonstrate how you can achieve insights on image analytics using automation and guided interface compared to manual image analytics process? If so, how?”, users had the following comments“* (comments are not syntactically modified in any way):

[1] *”Yes, the tool demonstrated how automation can improve image analytics by using a chatbot to recommend parameter adjustments based on image metrics, streamlining the process and enhancing accuracy compared to manual methods.”*

[2] *”Yes it can. When a local image is uploaded, the chatbot analyzes the image and recommends zoom, focus and contrast iteratively until an image has been produced.”*

[4] *”Yes, demonstrates how to access, scan new image and send for analysis on cloud*

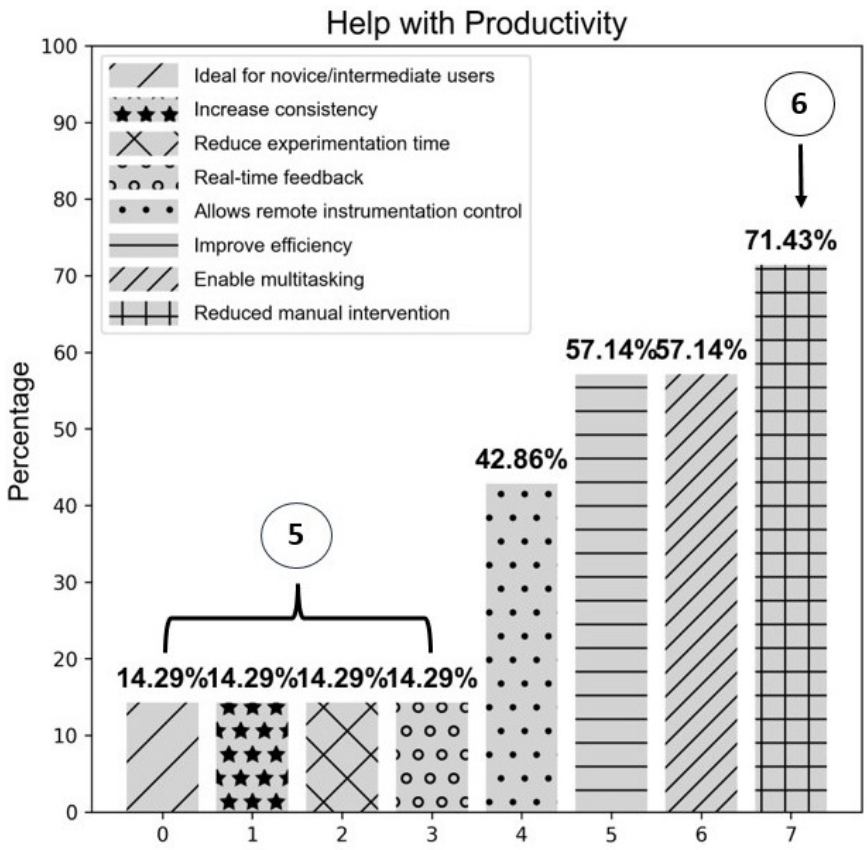

Figure 4.4: Results for the impact on productivity across eight distinct aspects showing four features, namely "ideal for novice intermediate users," "Increase consistency," "Reduce experimentation time," and "Real-time feedback," are considered relatively minor (denoted as 5); Conversely, the feature "Reduced manual intervention" is deemed a major asset (denoted as 6).

resources automatically."

Productivity Assistance evaluation

In response to the question *"Did the tool identify any opportunities to improve productivity compared to methods used previously? If so, what were they?", users had the following comments"* (comments are not syntactically modified in any way):

[1] *"The tool offers productivity benefits such as remote instrument control, real-time feedback, and automated tasks, reducing manual intervention, enabling multitasking, and improving efficiency in research processes."*

[2] *"The tool offers productivity benefits such as remote instrument control, real-time feedback, and automated tasks, reducing manual intervention, enabling multitasking, and*

improving efficiency in research processes."

[4] *"Yes, earlier methods would involve several manual steps for data storage, transfer and launching analysis. The tool cuts down many of these steps and so can increase productivity significantly."*

Useful Features evaluation

In response to the question *"Were there any features within the tool that stood out as particularly useful? ", users had the following comments* (comments are not syntactically modified in any way):

[1] *"Key features include the chatbot for real-time recommendations, remote instrument control via a web service, machine learning for optimization, cloud-based storage, and a dashboard for tracking progress, enhancing the overall research experience."*

[2] *"Remote Access and Real-time feedback particularly stood out. Automating the whole pipeline increases efficiency".*

[4] *"Recommendations provided and interactions with chatbot to do so are good functionalities."*

4.0.3 Results discussion

Figures 4.2, 4.3 and 4.4 present the synthesized survey responses to each question in three separate bar plots. These plots depict the primary and minor elements of the online site based on survey responses. In Figure 4.2, responses to the question Utility for Automation evaluation are presented in five distinct aspects of "Enhance accuracy", "Expedite experiment results", "Improve resource usability", "Reduced learning curve", "Streamline process" where "Expedite experiment results" is considered as a minor feature, around 14.29% support (denoted as 1), while "Streamline process" as the predominant feature, commanding approximately 85.71% support (denoted as 2).

In Figure 4.3, responses to the question Useful Features evaluation are represented in five distinct aspects known as ”Chatbot recommender”, ”web service access to instruments”, ”Dashboard for process tracking”, ”ML-based feedback”, and ”Cloud-based storage”. ”Dashboard for process tracking” is considered a minor feature, securing 14.19% support (denoted as 3), while both ”Chatbot recommender” and ”web service access to instruments” emerge as major features, each receiving approximately 57.14% support (denoted as 4).

Figure 4.4 assesses the RISE web services impact on productivity across eight distinct aspects ”Allows remote instrumentation control”, ”Real-time feedback”, ”Reduced manual intervention”, ”Enable multitasking”, ”Improves efficiency”, ”Increase consistency”, ”Ideal for novice/intermediate users”, and ”Reduce experimentation time”. Four features, namely ”ideal for novice intermediate users,” ”Increase consistency,” ”Reduce experimentation time,” and ”Real-time feedback,” are considered relatively minor, each obtaining 14.29% support (denoted as 5). Conversely, the feature ”Reduced manual intervention” is deemed a major asset, commanding an impressive 71.43% support (denoted as 6).

Chapter 5

Conclusion And Future Work

In this thesis, we presented a novel suite of RISE web services underlying a chatbot-guided interface for automated image analytics that allows for feedback control of remote instrumentation in the discovery process e.g., material science. We showed how our approach successfully addresses both performance improvement and data/instrument security during image analytics tasks, providing remote access to scientific instruments with seamless and secure communication. Further, we detailed a usability study for measuring user experience of RISE web services in terms of utility for automation, productivity assistance, and useful features.

The study results revealed that our RISE approach effectively streamlines processes but may not significantly expedite experiment results. Researchers found remote access to instruments via the web portal and chatbot-guided interface to be the most valuable features of the RISE system. Reducing manual intervention in instrument setup was deemed crucial for improving productivity by all the users. These findings demonstrate that our RISE web services provide a promising solution for remote instrumentation, offering both remote access to instruments and guided assistance to users in reducing manual effort and enhancing productivity for image analytics.

Future work could involve extending our RISE web services to other types of scientific

instrument resources and customizing the various options to suit other scientific domain applications e.g., biomedical sciences and plant sciences.

Bibliography

[1] Taher Hajilounezhad, Damola M Ajiboye, and Matthew R Maschmann. Evaluating the forces generated during carbon nanotube forest growth and self-assembly. *Materialia*, 7:100371, 2019.

[2] Prasad Calyam, Abdul Kalash, Ramya Gopalan, Sowmya Gopalan, and Ashok Krishnamurthy. Rice: A reliable and efficient remote instrumentation collaboration environment. *Advances in Multimedia*, 2008, 2008.

[3] Nguyen Phuoc Nguyen, Ilker Ersoy, Jacob Gotberg, Filiz Bunyak, and Tommi A White. Drpnet: automated particle picking in cryo-electron micrographs using deep regression. *BMC bioinformatics*, 22(1):1–28, 2021.

[4] Donglai Wei, Zudi Lin, Daniel Franco-Barranco, Nils Wendt, Xingyu Liu, Wenjie Yin, Xin Huang, Aarush Gupta, Won-Dong Jang, Xueying Wang, et al. Mitoem dataset: large-scale 3d mitochondria instance segmentation from em images. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 66–76. Springer, 2020.

[5] Taher Hajilounezhad, Rina Bao, Kannappan Palaniappan, Filiz Bunyak, Prasad Calyam, and Matthew R Maschmann. Predicting carbon nanotube forest attributes and mechanical properties using simulated images and deep learning. *npj Computational Materials*, 7(1):134, 2021.

[6] Ashish Pandey, Ramakrishna Surya, Matthew Maschmann, and Prasad Calyam. Reinforcement learning based carbon nanotube growth automation. In *2021 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*, pages 1–10. IEEE, 2021.

[7] Phuong Nguyen, Tarek Elgamal, Steven Konstanty, Todd Nicholson, Stuart Turner, Patrick Su, Klara Nahrstedt, Timothy Spila, Roy H Campbell, John Dallesasse, et al. Bracelet: Edge-cloud microservice infrastructure for aging scientific instruments. In *2019 International Conference on Computing, Networking and Communications (ICNC)*, pages 692–696. IEEE, 2019.

[8] Phuong Nguyen, Steven Konstanty, Todd Nicholson, Thomas O'brien, Aaron Schwartz-Duval, Timothy Spila, Klara Nahrstedt, Roy H Campbell, Indranil Gupta, Michael Chan, et al. 4ceed: Real-time data acquisition and analysis framework for material-related cyber-physical environments. In *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, pages 11–20. IEEE, 2017.

[9] Jonathan Bouvette, Qinwen Huang, Alberto Bartesaghi, and Mario J Borgnia. Smartscope: Ai-driven grid navigation for high-throughput cryo-em. In *2021 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*, pages 1–6. IEEE, 2021.

[10] Boyuan Huang, Zhenghao Li, and Jiangyu Li. An artificial intelligence atomic force microscope enabled by machine learning. *Nanoscale*, 10(45):21320–21326, 2018.

[11] Ákos Borsos, Botond Szilagyi, Paul Serban Agachi, and Zoltán K Nagy. Real-time image processing based online feedback control system for cooling batch crystallization. *Organic Process Research & Development*, 21(4):511–519, 2017.

[12] Marcos Baez, Florian Daniel, Fabio Casati, and Boualem Benatallah. Chatbot integration in few patterns. *IEEE Internet Computing*, 25(3):52–59, 2021.

[13] Alexander Neumann, Sascha Welten, Julia Kunz, Ralf Klamma, and Stefan Decker. Haico: A monitoring and transaction chatbot for distributed analytics workflows. In *2023 IEEE International Conference on Web Services (ICWS)*, pages 707–709, 2023.

[14] Nguyen P Nguyen, Ramakrishna Surya, Matthew Maschmann, Prasad Calyam, Kannappan Palaniappan, and Filiz Bunyak. Self-supervised orientation-guided deep network for segmentation of carbon nanotubes in sem imagery. In *Computer Vision–ECCV 2022 Workshops: Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part VIII*, pages 412–428. Springer, 2023.

[15] Syed Zulkarnain Syed Idrus, Estelle Cherrier, Christophe Rosenberger, and Jean-Jacques Schwartzmann. A Review on Authentication Methods. *Australian Journal of Basic and Applied Sciences*, 7(5):95–107, March 2013.

[16] Mohammad Reza Mesbahi, Amir Masoud Rahmani, and Mehdi Hosseinzadeh. Reliability and high availability in cloud computing environments: a reference roadmap. *Human-centric Computing and Information Sciences*, 8:1–31, 2018.

[17] Mayank Kejriwal and Pedro Szekely. mydig: Personalized illicit domain-specific knowledge discovery with no programming. *Future Internet*, 11(3):59, 2019.