Final Year Project Report

By Karthik Goud Bathula

WORD COUNT	10951	TIME SUBMITTED	23-APR-2024 10:34PM
		PAPER ID	108489861

Full Stack Website Development for

Machine learning development environment for single cell sequencing data analyses

---

A Project

Presented to

The Faculty of the Graduate School

University of Missouri-Columbia

In Partial Fulfillment

Of the Requirements for the Degree

Master of Science

By

Karthik Goud Bathula

Dr. Dong Xu, Project Supervisor

MAY 2024

ACKNOWLEDGEMENTS

I am profoundly grateful to **Dr. Dong Xu**, whose consistent guidance and robust support have been fundamental throughout this project. His deep expertise, valuable advice, and unwavering commitment to scholarly excellence have significantly influenced the direction of this research and my development as a scholar. Dr. Xu's mentorship has been crucial, offering both intellectual motivation and actionable insights that greatly enhanced the quality of this work.

I also owe a great deal of gratitude to **Jiang Lei**, whose innovative concepts and reliable advice have been essential to the success of this project. Jiang's ongoing support, technical knowledge, and cooperative approach were key in addressing the complex aspects of this research. From the project's beginning to its completion, Jiang's guidance has continuously inspired and propelled the project's innovative outcomes and my growth as a researcher.

Further, I wish to express my sincere thanks to my family and friends for their unwavering support over the last two years. Their encouragement, companionship, and joyful moments have motivated me throughout this endeavor. Their faith in my abilities and my work continually reminded me of the value of dedication and enthusiasm in both scholarly and personal endeavors. I am deeply appreciative of their love and support, which were crucial in achieving this significant milestone.

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

Single-cell sequencing provides profound insights into cellular behaviors and diseases but involves complex data analysis challenges. This project develops a specialized machine learning (ML) development environment tailored for single-cell sequencing data analysis. The report details the creation of a cloud-based platform **OSCB (Open Single Cell Benchmarks)** that supports scalable workflows for processing and managing single-cell data for ML applications. Key features include a sophisticated web application built with React JS, secure authentication using JSON Web Tokens (JWT), and a Directus-based content management system for real-time updates. The platform also offers a user-friendly interface that supports single-cell analysis method development through react-json-schema, enabling detailed control over data tasks.

Further, the project introduces a workflow for user data upload and metadata enrichment, enhancing data submission with necessary metadata. This setup supports extensive benchmarking and provides an automated single-cell analysis ML pipeline that reduces complexity in data formatting, model development, and evaluation. By merging advanced web technologies with robust backend solutions, this project simplifies single-cell data analysis and sets the stage for future expansions, aiming to keep the platform at the forefront of bioinformatics research.

1. INTRODUCTION

1.1 Background

Single-cell sequencing (SCS) technologies have revolutionized our understanding of cellular diversity and function within biological tissues, enabling researchers to study the genomic profiles of individual cells in various health and disease conditions. Despite its transformative potential, analyzing SCS data involves handling vast amounts of complex data, necessitating robust computational tools that can cater to the specificity and variability of such data.

1.2 Problem Statement

Current tools for analyzing single-cell sequencing data often require considerable computational expertise and do not always seamlessly integrate with machine learning (ML) frameworks, which are essential for uncovering patterns in high-dimensional datasets. Moreover, existing platforms typically do not provide end-to-end solutions that encompass data collection, processing, analysis, and visualization in a user-friendly manner. Researchers face significant barriers due to the complexity of both the underlying biology and the computational algorithms required for effective analysis.

1.3 Project Objectives

The primary objective of this project is to develop a machine learning development environment specifically designed for single-cell sequencing data. This environment aims to:

- Simplify the process of data handling and analysis for biologists and data scientists.
- Provide an integrated suite of tools for data collection, processing, model development, and evaluation.
- Enable users to easily upload, manage, and analyze their data through a streamlined web interface.
- Foster collaborative and reproducible research through easy sharing and benchmarking capabilities.

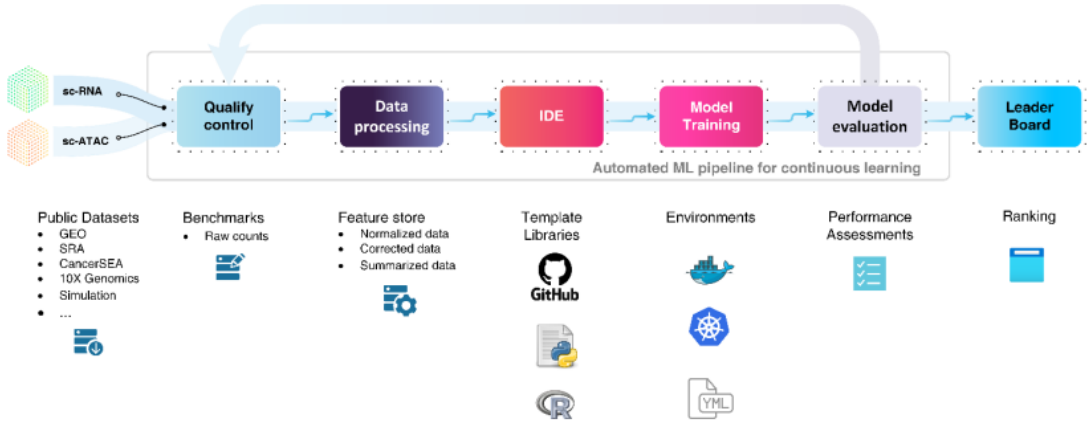


Fig 1 OSCB Workflow Diagram

1.4 Scope of the Project for Phase 1:

This project involves creating a cloud-based platform that integrates various functionalities:

- **Web Application**: Develop a responsive web application using React JS that offers a user-friendly interface for interacting with the system.
- **Authentication and Security**: Implement secure login and user management features using JSON Web Tokens (JWT) to ensure data privacy and system integrity.
- **Data Management:** Utilize Directus for content management to allow real-time updates and manage user data efficiently.
- **Integrate Tools into OSCB**: Provide access to different single cell analysis tools to analyze single-cell data.
- **User Interaction:** Facilitate extensive user interaction capabilities for uploading data, selecting analysis tools, and viewing results.
- **Benchmarking and Analysis:** Develop an automated pipeline for benchmarking datasets tailored for single-cell data.

1.5 Significance of the Project

By reducing the complexity and technical barriers associated with single-cell data analysis, this platform will make advanced bioinformatics tools more accessible to a broader range of researchers. This accessibility is expected to accelerate discoveries in genetics and disease research, ultimately contributing to personalized medicine and improved healthcare outcomes.

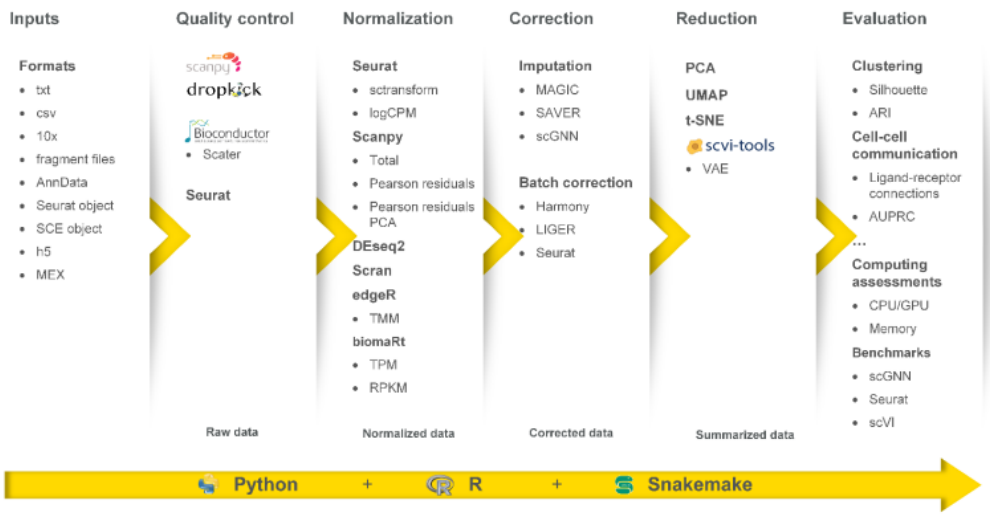


Fig 2 OSCB Tools

2. LITERATURE REVIEW

2.1 Existing Tools and Platforms in Single-Cell Sequencing Data Analysis

Single cell sequencing data analysis is a rapidly evolving field with numerous tools designed to address specific aspects of the analysis pipeline. Key platforms include:

- **Seurat**: A widely used R package for single-cell genomics data analysis, which includes functions for QC, analysis, and exploration of single-cell RNA sequencing data. Seurat supports various data scaling, feature selection, and dimensionality reduction techniques crucial for identifying and interpreting cell populations.
- **Scanpy**: A scalable toolkit for analyzing single-cell gene expression data using Python. It includes preprocessing, visualization, clustering, trajectory inference, and differential expression testing, designed to handle very large datasets efficiently.
- **Cell Ranger**: Developed by 10X Genomics, Cell Ranger is a set of command-line software tools that process raw data from single-cell sequencing experiments to generate feature-barcode matrices and perform clustering and gene expression analysis.

2.2 Limitations of Existing Tools

While these tools are powerful, they exhibit several common limitations:

- **Usability and Accessibility**: Many of the existing tools require significant computational expertise, limiting their accessibility to biologists or researchers without extensive programming backgrounds. This can hinder the broader adoption and application of single-cell technologies.
- **Integration and Workflow Management:** Often, researchers must use a patchwork of tools to manage different stages of the single-cell analysis pipeline. This lack of integration can complicate the analysis process, making it time-consuming and prone to errors.
- **Scalability and Performance:** Some tools struggle with the scale of data typically generated by modern single-cell experiments, leading to long processing times and high memory usage, which can be impractical for routine analyses.

- **Flexibility in Data Handling:** Many platforms have rigid input requirements and limited support for different data types or sources, restricting their use in environments where data heterogeneity is common.

2.3 Innovations and Improvements Proposed in this Project

This project addresses these challenges by creating a more integrated and user-friendly machine learning development environment for single-cell sequencing data analysis. The improvements include:

- **Comprehensive Web-Based Platform**: Unlike many existing tools that require local installation and manual data handling, this project develops a cloud-based platform that users can access through a web interface. This approach enhances accessibility and ease of use, particularly for users without extensive computational training.
- **Unified Workflow System:** By integrating various stages of the analysis pipeline—from data upload and preprocessing to analysis and visualization—into a single platform, this project simplifies the user workflow, reduces the potential for errors, and saves time.
- **Scalable and Efficient Data Processing:** Leveraging modern cloud technologies and scalable architectures, the platform is designed to handle large datasets more efficiently than traditional tools. This is critical for processing the voluminous data generated in single-cell sequencing experiments.
- **Enhanced User Interaction and Dynamic Content Management:** Through the integration of Directus for content management, this platform allows for real-time updates and user-driven content management without backend redeployment. This feature supports dynamic and responsive user interactions, making the platform adaptable to user needs and preferences.

3. METHODOLOGY

3.1 Web Application Development

3.1.1 Technologies and Frameworks Used

The web application for the OSCB platform is primarily developed using React JS for the front-end and Node.js for the backend. This technology stack was selected to leverage the strengths of each framework to meet the project's requirements for a robust, scalable, and user-friendly web application.

React JS:

React JS is a JavaScript library developed by Facebook, widely recognized for its efficiency in building interactive user interfaces. It operates on the principles of components and the virtual DOM, which makes it highly efficient for rendering dynamic user interfaces. React's component-based architecture allows developers to build encapsulated components that manage their own state, then compose them to make complex UIs. This modularity was crucial for developing reusable UI elements across the OSCB platform, enhancing consistency and maintainability.

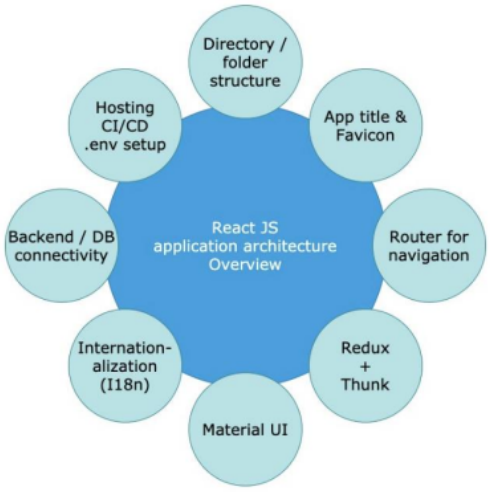


Fig 3 React JS Architecture Overview Source

Key features of React that were advantageous for this project include:

- **JSX**: React uses JSX for templating instead of regular JavaScript. JSX is a syntax extension for JavaScript that resembles HTML in appearance. It makes writing React components, handling subcomponents, and inserting JavaScript logic into markup much simpler and more intuitive.
- **Virtual DOM**: React creates a virtual DOM in memory, where it performs all the necessary manipulations before making the minimal changes required to the real DOM. This process optimizes performance, particularly important in data-intensive applications like OSCB, which handle large datasets and require frequent UI updates.
- **One-way Data Binding**: React's one-way data flow (props) ensures that changes in child components do not directly affect their parents, making the code easier to debug and understand.

Node.js:

Node.js is an open-source, cross-platform, JavaScript runtime environment that runs on the V8 engine and executes JavaScript code outside a web browser. For the OSCB platform, Node.js was chosen to handle server-side scripting—running web application server code and responding to HTTP requests.

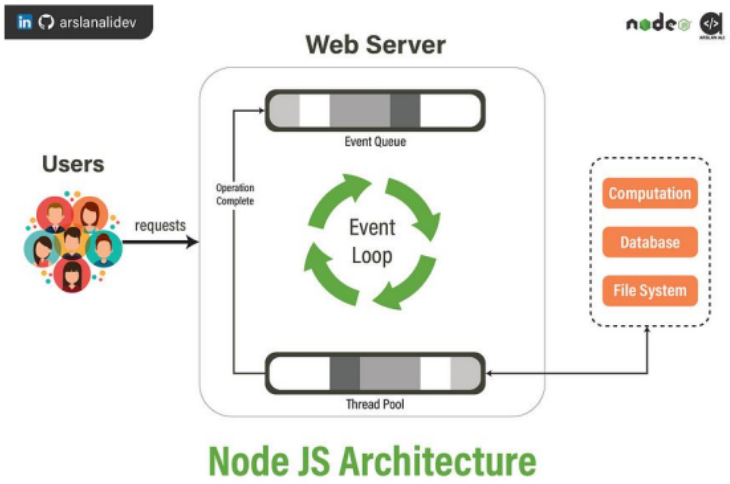


Fig 4 Node JS Architecture Overview Source

Reasons for choosing Node.js include:

- **Non-blocking I/O:** Node.js processes requests in a non-blocking manner, which makes it particularly well-suited for building applications that perform a lot of disk or network access and need to support high levels of concurrency.
- **NPM Ecosystem:** With access to the largest ecosystem of open-source libraries (npm), Node.js provides numerous packages that can be easily integrated to add functionalities, such as connecting to databases or setting up authentication systems.
- **Scalability**: Node.js utilizes a single-threaded model with event looping, which helps the server to respond in a non-blocking way, making it scalable and suitable for environments that require handling multiple requests at the same time.

3.1.2 Selection Rationale

The combination of React JS and Node.js was strategic, driven by the need for a high-performance, scalable platform capable of managing complex state management and real-time data updates, which are crucial for the OSCB platform's functionality.

- **Scalability**: Both React and Node.js are known for their performance and scalability, crucial for handling the intensive data operations required by single-cell sequencing analysis tools.
- **Ease of Development**: React's component-based architecture and Node's extensive npm packages significantly speed up development and testing. This integration simplifies building and maintaining complex applications.
- **Community Support:** Both technologies boast large communities of developers. This extensive community support means abundant resources, troubleshooting help, and continuous updates and improvements to the technologies.

Together, these technologies provide a robust foundation for developing a modern web application that is both performant and maintainable, capable of supporting the complex functionalities required by the OSCB platform.

3.2 Authentication System

Design and Implementation of JWT-Based Authentication

The OSCB platform employs a secure authentication system using JSON Web Tokens (JWT) to manage user sessions and safeguard user interactions. This section outlines the specific design choices and implementations that ensure both robust security and excellent functionality.

3.2.1 JWT Implementation Strategy:

- **Token Generation:** Upon successful user login, the backend server generates a JWT that includes encoded claims such as the username. This token is digitally signed using a secret key that is securely stored on the server.
- **Token Transmission and Storage:** The signed JWT is securely transmitted to the client's browser and stored in HttpOnly cookies to prevent access via JavaScript, enhancing security against cross-site scripting (XSS) attacks.

3.2.2 Security Measures:

- **bcrypt for Password Security:** User passwords are hashed using bcrypt before storage in the database. bcrypt provides strong salting and hashing, which protects against brute-force attacks and ensures that actual passwords are never stored or transmitted.
- **Secure Cookie Settings:** Cookies storing JWTs are set with HttpOnly and Secure flags to ensure that they are sent only over HTTPS and are not accessible via client-side scripts.

3.2.3 Managing JWT Expiry and Refresh

To optimize security while maintaining user convenience, the platform incorporates an intelligent mechanism for managing JWT expiry and refresh:

- **Expiration Policy**: JWTs have a set expiration time of 1 hour to limit the lifetime of a session and reduce the risk of unauthorized use if a token is compromised.

- **Automatic Refresh during User Activity**: The platform automatically refreshes the JWT upon each user interaction, which involves sending API requests to the server. This process resets the expiration time of the JWT, ensuring active users remain logged in without disruption.
- **Proactive Session Extension Prompt:** Five minutes before a token's expiration, the platform proactively notifies the user of the impending session timeout. Users can choose to extend their session directly from this notification, triggering a token refresh that extends the session by another hour.

1. **Implementation Details:** This functionality is implemented using client-side logic that tracks the token's expiration time and interacts with the server to refresh the token as necessary.
2. **User Experience:** This approach ensures that users are not abruptly logged out due to inactivity, particularly during extended reading or data analysis sessions, thereby enhancing the user interface experience.

3.2.4 Secure Storage and Handling of JWTs:

- **Cookies with HttpOnly and Secure Flags:** Storing JWTs in cookies marked as HttpOnly prevents access to the token via client-side JavaScript, reducing XSS risks. The Secure flag ensures cookies are transmitted only over secure HTTPS connections.
- **Cross-Origin Resource Sharing (CORS):** Proper CORS policies are implemented to restrict the sharing of resources and cookies only with trusted domains, preventing CSRF (Cross-Site Request Forgery) attacks.

3.2.5 Benefits of the Chosen Authentication Approach:

- **Enhanced Security:** The combination of JWT for session management, bcrypt for password hashing, and secure cookie settings provides robust defense mechanisms against common security threats.
- **Seamless User Experience:** Automatic and manual token refresh mechanisms are designed to support uninterrupted user sessions, thus improving the overall usability and satisfaction of the platform.

- **Scalability:** The stateless nature of JWTs allows the system to scale efficiently, as the server does not need to maintain session states, making this approach highly effective for cloud-based applications like OSCB.

3.3 Integration of Uppy File Uploader in the OSCB Application

The OSCB platform leverages Uppy, a modular and extensible file uploader, to facilitate the efficient and user-friendly upload of large-scale single-cell sequencing data files. This section provides an in-depth look at the integration and configuration of Uppy within the platform's React-based front-end and Node.js-based backend.

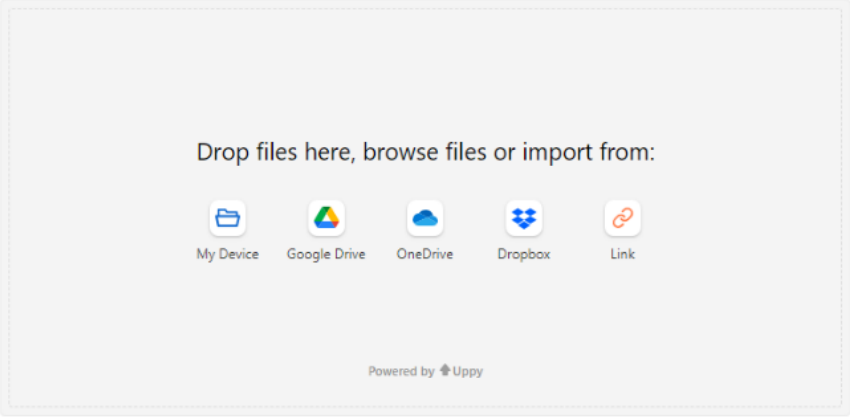


Fig 5 Uppy File Uploader Interface

3.3.1 Front-End Integration in React

The Uppy file uploader is integrated into a React component, which initializes an instance of Uppy and configures it to handle multiple file uploads, interface with cloud storage solutions, and upload files to a Node.js server endpoint.

Key Features and Configuration:

- **Modular Plugins**: Uppy is configured with plugins for Google Drive, OneDrive, Dropbox, and direct URL uploads, enhancing the flexibility for users to upload files from various sources.

- **Dynamic Resizing:** The React component adjusts the Dashboard plugin's view based on the browser window size, improving user experience across different devices.
- **Conditional Rendering:** The component conditionally renders different views based on whether the upload is for public or private datasets, driven by user selections within the application.

3.3.2 Backend Integration in Node.js

The backend, built with Node.js and Express, handles the files uploaded from the front-end. It uses multer for handling multipart/form-data, which Uppy sends.

Server Configuration and File Handling:

- **Temporary Storage:** Uploaded files are initially stored in a temporary directory, which helps in managing file validation and preprocessing before moving them to permanent storage.
- **Destination Management:** Depending on the public or private nature of the upload, files are moved to designated directories, either in a public shared space or within a user-specific directory.
- **Security and Authentication:** Files are only accepted if the user's authentication token is verified, ensuring that uploads are securely managed and tied to authenticated sessions.

3.3.3 Enhancing Security and User Experience

Security Practices:

- **HTTPS**: All communications between the front-end and back-end are secured using HTTPS, ensuring data integrity and confidentiality during file transfers.
- **JWT for Authentication:** Requests to the upload endpoint include JWTs in headers, ensuring that each file upload is performed by an authenticated and authorized user.

User Experience Considerations:

- **Real-Time Feedback:** The Uppy Dashboard provides users with real-time feedback on upload progress and status, enhancing transparency and trust in the upload process.
- **Error Handling:** The platform handles errors gracefully, providing users with actionable feedback in case of issues like network failures or file size limits.

The integration of Uppy into the OSCB platform not only streamlines the file upload process but also ensures that it is secure, efficient, and user-friendly. This setup supports the platform's goals of facilitating comprehensive data analysis while ensuring data integrity and security. Through careful configuration and robust backend support, the platform effectively manages large-scale data uploads, essential for the processing and analysis of single-cell sequencing data.

3.4 Integration of Directus headless CMS tool in the OSCB Application

For the OSCB platform, managing dynamic content effectively and enabling real-time updates without the need for backend redeployment are critical functionalities. To achieve these objectives, we integrated Directus, an open-source headless content management system (CMS), into our architecture. This section describes how Directus is used within the platform to manage content seamlessly and flexibly.

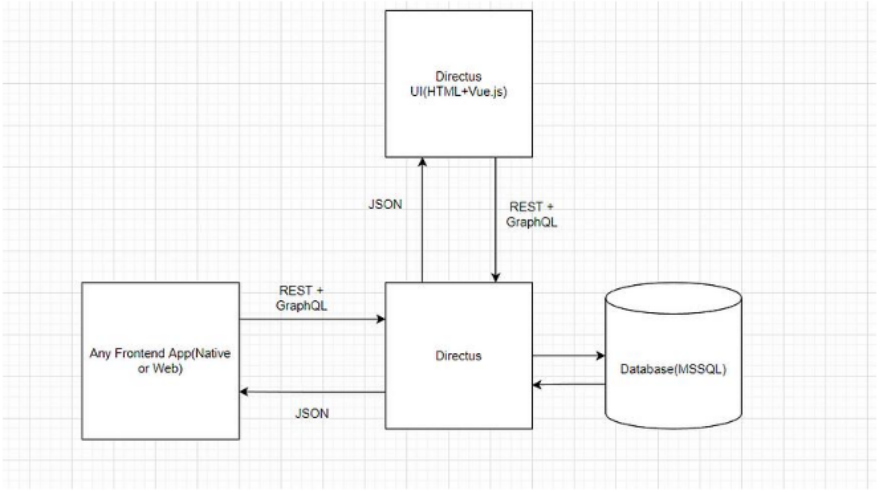


Fig 6 Directus Architecture Overview Source

3.4.1 Choosing Directus

Directus is selected for its ability to act as a headless CMS, which means it provides a backend for managing content that can be delivered via APIs to any frontend system. This flexibility is crucial for a project like OSCB, where content needs to be frequently updated and made available in real-time to users without impacting the underlying application logic or performance.

Key Benefits of Directus:

- **Flexibility and Extensibility:** Directus provides a flexible schema that can be tailored to our specific data requirements. It allows for the easy addition of new fields and customization of data types without any changes to the application code.
- **Real-Time Updates:** Changes made in the Directus dashboard are immediately available to the frontend via RESTful or GraphQL APIs, ensuring that content updates do not require redeployment of the backend.
- **Role-Based Access Control:** Directus supports fine-grained access control, which is essential for managing user permissions and ensuring that only authorized personnel can modify content.

3.4.2 Integration Process

Front-End Interaction:

- **API Consumption**: The OSCB platform's frontend dynamically fetches content managed in Directus, such as guidelines, updates, and educational materials, using Directus's API. This approach decouples the content management from the application logic, allowing non-technical team members to update content without developer intervention.
- **Real-Time Content Rendering:** Updates fetched from Directus are rendered in real-time on the platform, enhancing the user experience by providing the latest information and resources without page refreshes or downtime.

Backend Configuration:

- **Directus Setup:** We configured Directus to run on a separate server instance, connected to the same database as the OSCB platform. This setup ensures that Directus has dedicated resources while maintaining close integration with the main application database.
- **Custom Endpoints**: For specific content needs that go beyond standard CRUD operations, custom endpoints were set up within Directus. These endpoints allow for complex queries and interactions that are specific to the needs of the OSCB platform, such as aggregating data or filtering content based on user preferences.

Content Structures and Management:

- **Dynamic Pages and Sections:** Content for various sections of the OSCB platform, such as 'Get Started', 'Updates', and Team, is managed through Directus collections. Each collection is tailored to the structure of the page it serves, ensuring that content fits seamlessly into the platform's design.
- **Real-Time Content Updates:** Changes made in Directus by content managers are immediately pushed to the platform through the Directus API, ensuring that users always have access to the most current information.

Integrating Directus into the OSCB platform has significantly streamlined content management, enabling real-time updates and empowering non-developer team members to maintain and update platform content independently. This integration supports the platform's commitment to providing up-to-date and relevant content to its users, enhancing both the usability and functionality of the system.

3.5 Cloud-Based Testing Environment

Given the developmental stage of our project, we utilized test beds from a cloud lab server environment to evaluate and refine our applications. This temporary setup allowed us to conduct rigorous testing, troubleshoot issues, and validate functionalities in a controlled environment before deploying the applications to a dedicated server infrastructure.

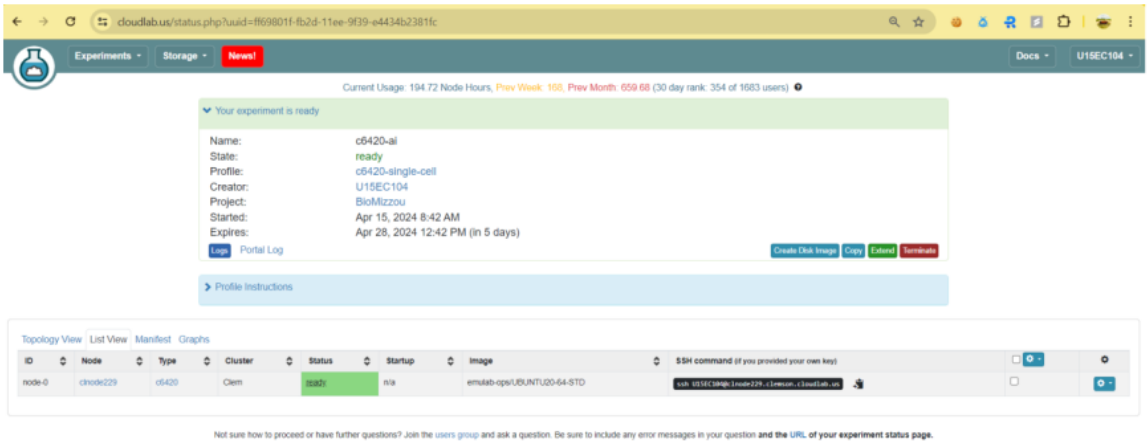


Fig 7 CloudLab Test Server

Selection of Cloud Lab Server

For the testing phase, we opted to leverage the resources provided by a cloud lab server environment. This decision was driven by the need for flexibility, scalability, and cost-effectiveness during the developmental phase. While not intended for production use, the cloud lab server served as an invaluable testing ground for our applications.

Temporary Testing Infrastructure

Using the cloud lab server, we configured temporary testing instances with sufficient compute power, storage capacity, and network connectivity to support our testing requirements. These instances were provisioned on-demand, allowing us to scale resources as needed and simulate real-world usage scenarios.

Testing Methodology

Within the cloud lab environment, we conducted comprehensive testing to evaluate various aspects of our applications, including functionality, performance, security, and compatibility.

This involved executing test cases, running performance benchmarks, and simulating user interactions to identify and address any issues or deficiencies in the software.

Future Migration Plans

While the cloud lab server served as a temporary testing environment, our long-term plan involves migrating our applications to more robust and scalable infrastructure solutions, such as Amazon EC2 or Kubernetes clusters. This transition will provide the necessary resources and capabilities to support production-level deployments and accommodate growing user demand.

Benefits and Limitations

The use of a cloud lab server for testing offered several advantages, including flexibility, rapid provisioning, and cost-effectiveness. However, it also had limitations, such as limited scalability and performance compared to dedicated server solutions. Nevertheless, it served its purpose well during the developmental phase, enabling us to iterate quickly and refine our applications.

4. DESIGN DETAILS OF OSCB ARCHITECTURE

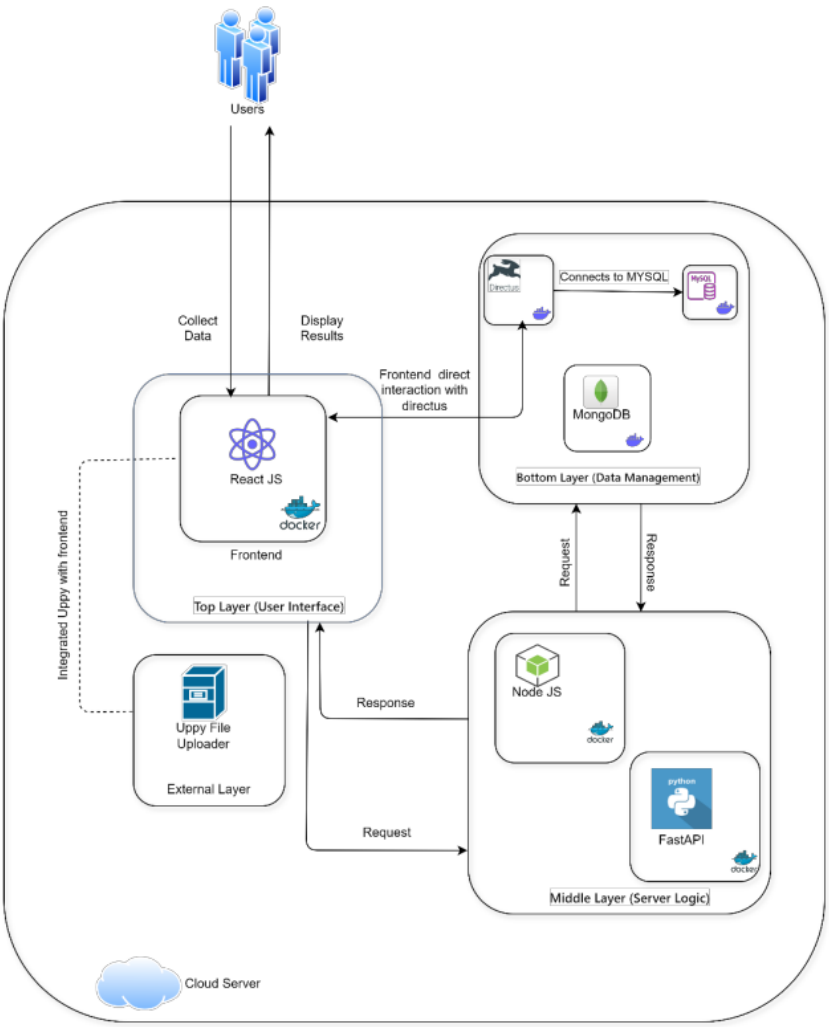


Fig. High-level architectural diagram of OSCB

4.1 User Interface Layer

- **Component-Based Framework**: React JS was chosen for its component-based architecture, which allows for a modular and scalable frontend. This framework facilitates the development of interactive user interfaces, where each component can be developed, tested, and debugged independently.

- **User Experience:** The user interface design prioritizes ease of navigation and a clean aesthetic. Key features include a responsive header with the OSCB logo, an intuitive search bar, and accessible navigation links to critical areas of the website. Each interaction is designed to be smooth and logical, from user login to data analysis workflows.
- **Docker Containerization**: The React application is containerized using Docker, which simplifies both development and deployment. Docker ensures that the application runs the same in every environment, minimizing "it works on my machine" issues and streamlining the path to production.

4.2 Server Logic Layer

- **Node.js Application:** Node.js manages user interactions through a RESTful API, ensuring smooth communication with front-end components. This choice leverages Node.js's non-blocking I/O for efficient handling of concurrent requests, essential for a data-intensive platform like OSCB.
- **FastAPI Application:** The Python FastAPI application underpins the data processing capabilities of OSCB, running analysis tools that require significant computational resources. FastAPI's asynchronous request handling and Python's rich ecosystem of scientific libraries provide the necessary performance for single-cell data analyses.

4.3 Database Management Layer

- **MySQL**: This relational database system is used for structured data management, such as user credentials and session data. MySQL's reliability and wide adoption make it a solid choice for these critical functions.
- **MongoDB**: For unstructured data such as user-generated datasets, MongoDB offers the required flexibility with its document-oriented approach. This allows the system to accommodate a variety of data types without the constraints of a fixed schema.

4.4 Directus Integration

- **Content Management:** Directus is a headless CMS that connects directly to our MySQL database. Its integration enables non-technical stakeholders to manage content through a straightforward web interface, empowering them to make changes that reflect in real-time on the OSCB platform.
- **Non-Technical User Accessibility**: Directus democratizes content management by providing a clear and intuitive interface for non-developers. This allows for immediate updates to the website's content without backend deployment, enhancing agility and responsiveness.

4.5 External Layer - Uppy Integration

- **File Management Tool:** Uppy offers a user-friendly interface for file uploads, supporting a range of sources like local storage and cloud services. Its integration into OSCB allows users to upload their datasets seamlessly, which are then processed and stored in user-specific storage within MongoDB.
- **Data Handling:** After upload, datasets undergo validation and formatting to ensure compatibility with OSCB's data processing pipelines. This transformation is crucial for enabling robust data analysis tools downstream.

4.6 Communication and Workflow

- **Request Routing:** User requests initiated at the UI layer are efficiently routed through the system. The Node.js server acts as the intermediary, processing business logic and interfacing with the databases. The FastAPI application manages heavy-duty tasks, with both layers capable of handling requests independently when suitable.
- **Backend Processes:** The server logic ensures secure user authentication, content delivery, and execution of complex data analyses. Communication between the server and databases is bi-directional, with the databases not only storing data but also informing the server logic through queries and triggers.

4.7 Containerization with Docker

- **Service Isolation**: Each OSCB service is hosted in a separate Docker container, providing isolation and scalability. This setup allows for independent service scaling and updates without affecting the entire system.
- **Cloud Deployment:** Using Docker Compose, the orchestration of multi-container applications is streamlined, facilitating deployment on cloud servers. Docker networking provides secure and efficient communication channels between containers, crucial for maintaining the integrity and performance of the OSCB platform.

5. IMPLEMENTATION AND TESTING

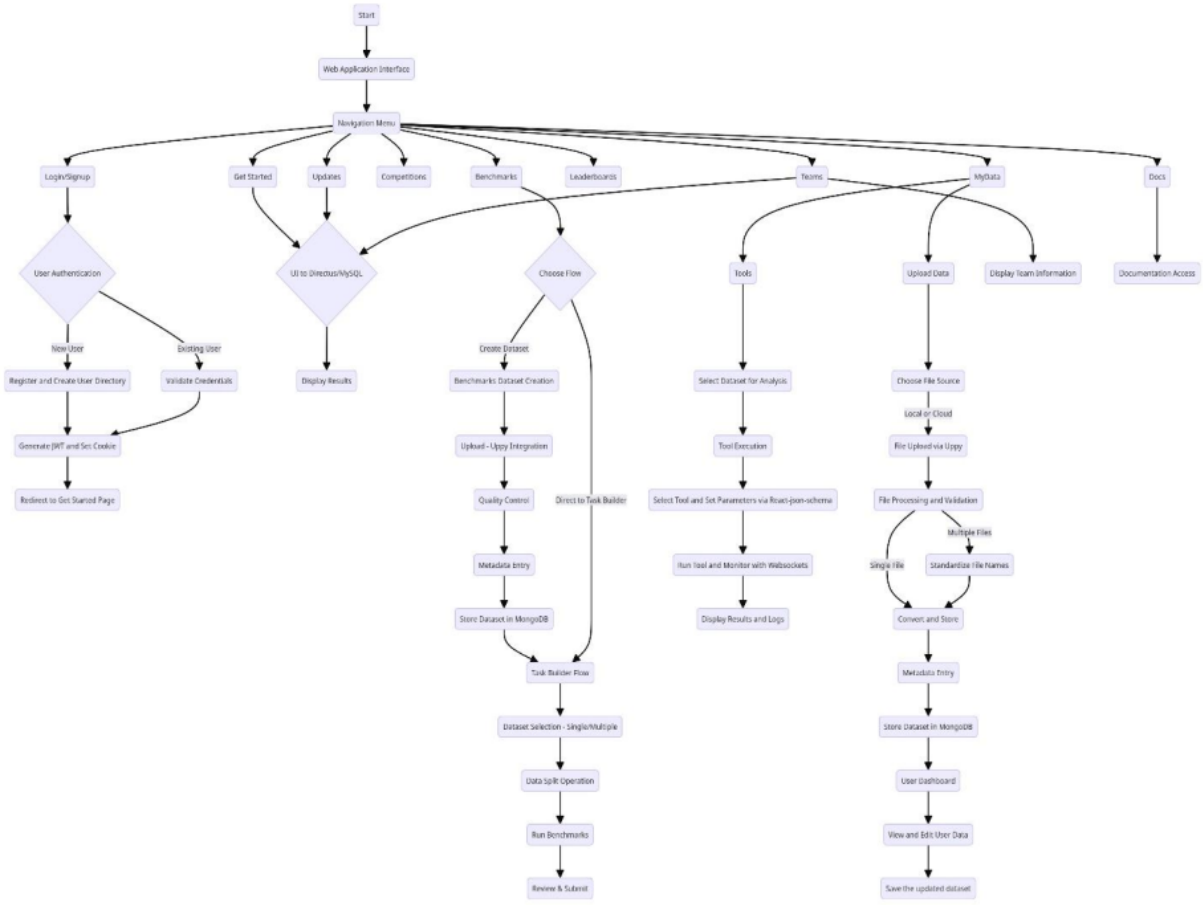


Fig 9 Complete flow of OSCB WebSite

5.1 OSCB Home page

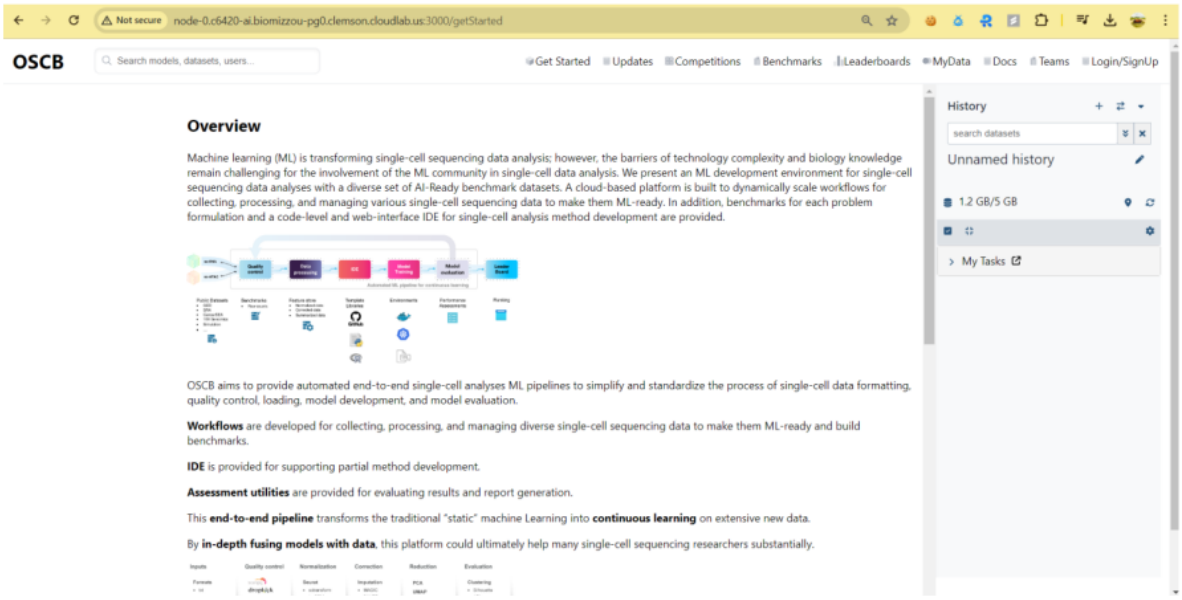


Fig 10 OSCB Home page

5.1.1 Design Philosophy

The OSCB homepage is designed with a clear focus on user-friendliness and accessibility to resources. We've put a lot of thought into creating a layout that seamlessly guides users from their first encounter with the platform to fully exploring everything it has to offer.

5.1.2 Navigation and Accessibility

At the top of the home page, you'll find a navigation bar that acts as the hub for navigating the core sections of the platform. These sections include "Get Started," "Updates," "Competitions," "Benchmarks," "Leaderboards," "My Data," "Docs," "Teams," and "Login/SignUp." This navigation bar is pivotal to the user interface, created to make it easy and intuitive for users to explore the platform's wide range of features and capabilities.

5.1.3 Platform Overview and Workflow Visualization

At the heart of the home page is an 'Overview' section that highlights OSCB's mission to combine machine learning with single-cell sequencing data analysis. Here, you'll find a visual representation of the platform's workflow, illustrating the entire process from data collection to quality control, model development, and evaluation. This workflow chart isn't just an informative graphic—it also represents our dedication to making the analytical process simpler and more standardized.

5.1.4 User-Centric Features

In alignment with the goal of fostering an interactive user community, the home page includes dynamic features such as a 'History' panel. This tool is specifically designed to enhance user engagement by keeping track of their activities and datasets, thus providing a personalized experience for each user.

5.1.5 Technical Implementation

The home page is a prime example of cutting-edge web development, leveraging the full capabilities of the React JS framework. By opting for React JS, we've crafted a user interface that's not only responsive and dynamic but also capable of updating and syncing components seamlessly, all without the need for page reloads. This mirrors the real-time nature of research and collaboration in the field of single-cell biology.

5.2 User Authentication System

At the heart of any interactive web application lies a sturdy user authentication system. OSCB's authentication mechanism serves a dual purpose: safeguarding the platform's security while also tailoring a personalized experience for each user. To unlock the full range of services OSCB offers, users are prompted to either log in or sign up. However, we've made sure that certain functionalities are accessible even to visitors who haven't registered, allowing them to explore the website's features.

Presently, visitors can freely access the "Get Started" and "Teams" pages without registering. However, registration becomes mandatory for accessing user-specific pages. Additionally, users must log in to make use of any tools provided by OSCB.

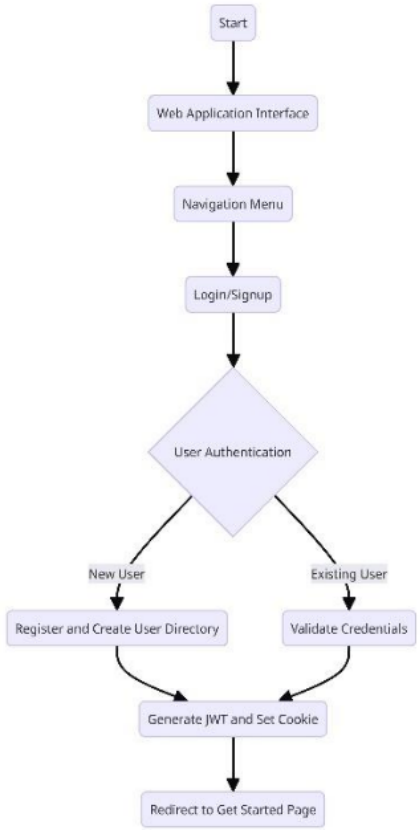


Fig 11 User Authentication System Flow

5.2.1 User Registration Process (Sign Up)

The sign-up feature serves as the entry point for newcomers to join the OSCB community. Built using the React JS framework, it includes a form where users can provide necessary information like their username, email ID, password, and confirmation password. We've taken special care to ensure that each username is unique within the OSCB system. If a new user tries to register with a username that's already taken, they'll see an error tooltip prompting them to choose a different one. This proactive approach is crucial in our user experience design, helping to prevent any

registration issues.

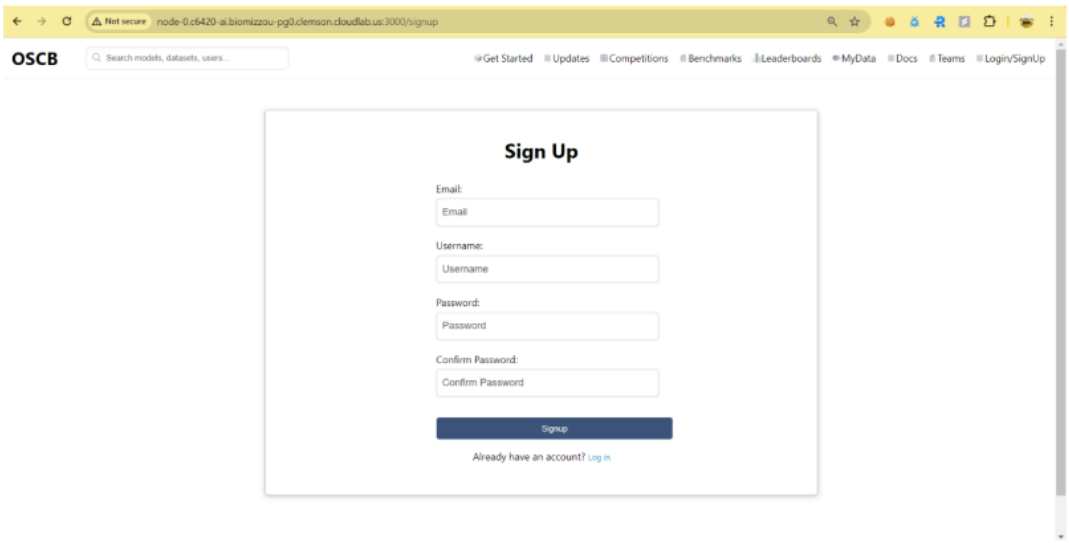


Fig 12 React Sign Up Page

We take password security seriously, and that's why our password fields undergo thorough validation. When users enter their password, they must confirm it to ensure accuracy and strengthen security right from the start. Once entered, this information is securely transmitted through an axios call to our Node.js server. Acting as a bridge between the React frontend and the MySQL database, our server ensures that user data remains safe and protected.

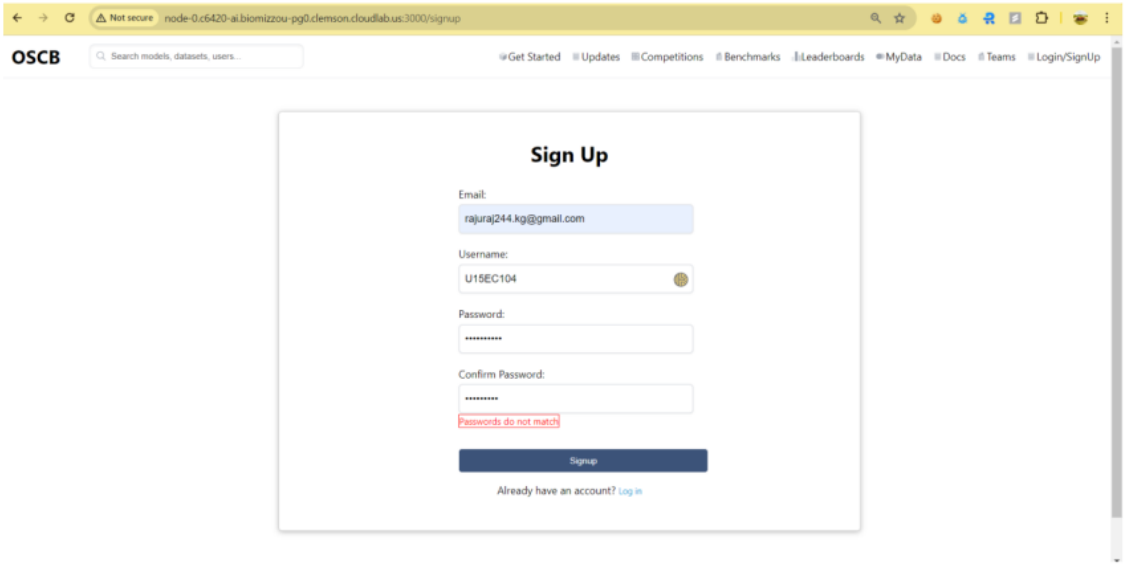


Fig 13 Sign Up Page Password Validation

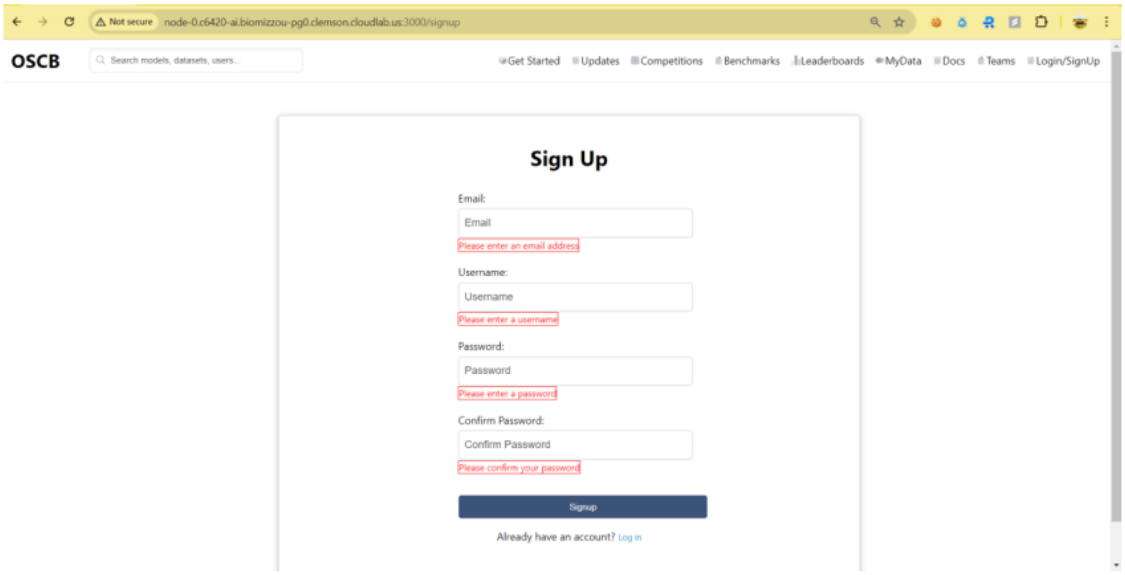


Fig 14 Sign Up Page Fields Validation

Here, the server validates the uniqueness of the username and ensures the password meets security standards. Upon successful validation, the server utilizes bcrypt to hash the password with a salt factor of 10, ensuring that user credentials are securely stored. A new table named **Users** is created to store usernames and hashed passwords, ensuring the security of user information.

Additionally, the sign-up process involves the creation of dedicated storage space for each user. OSCB utilizes a centralized storage directory located at **/usr/src/app/storage/** as the main storage repository. This is a shared docker volume created upon starting the OSCB application. This shared volume is configured to point to the path /usr/src/app/storage/, enabling seamless data access and management across different components of the OSCB ecosystem. For every registered user, a directory is created within this storage directory using the username as the directory name. For example, if a user registers with the username "john_doe," OSCB allocates storage space at /usr/src/app/storage/john_doe for their exclusive use. Each user's storage space is limited to 5GB, ensuring fair allocation of resources and optimal platform performance. This approach enables users to securely manage their data within OSCB while adhering to storage limitations.

Also, the server generates a JSON Web Token (JWT) after successful registration and stores it as a cookie, jwtToken, in the user's browser, facilitating secure communication between the client and server.

5.2.2 Login Component

For existing users, the LoginComponent simplifies the re-entry process. Users authenticate themselves by submitting their credentials, which are then verified against the secure user credentials stored within MySQL, illustrating OSCB's commitment to maintaining a secure environment.

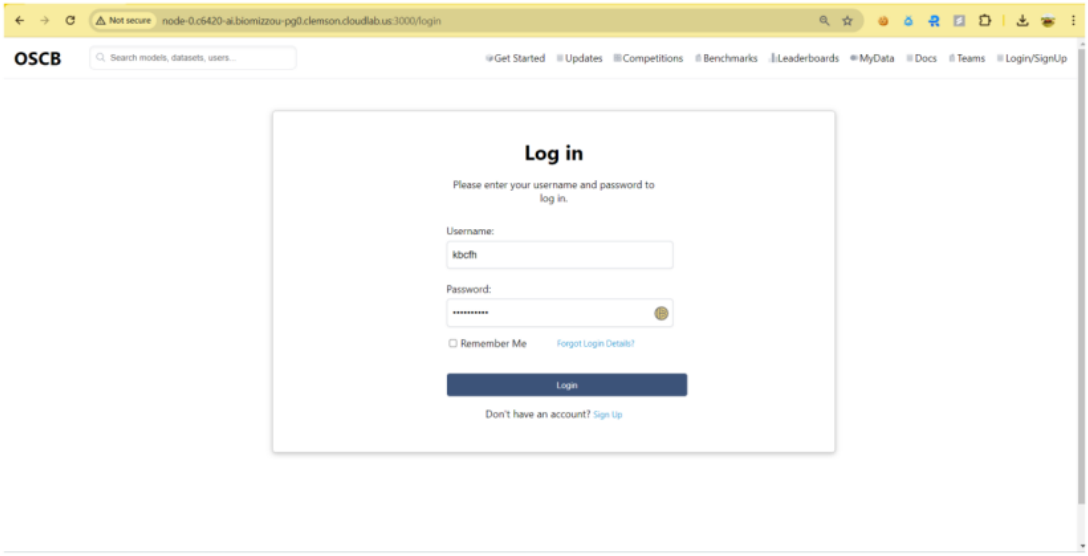


Fig 15 Login Page

Upon submission, bcrypt is used to hash the password before comparing it with the stored hash in the database Users table. If the credentials match, the server generates a JSON Web Token (JWT) and stores it as a cookie, jwtToken, in the user's browser, facilitating secure communication between the client and server.

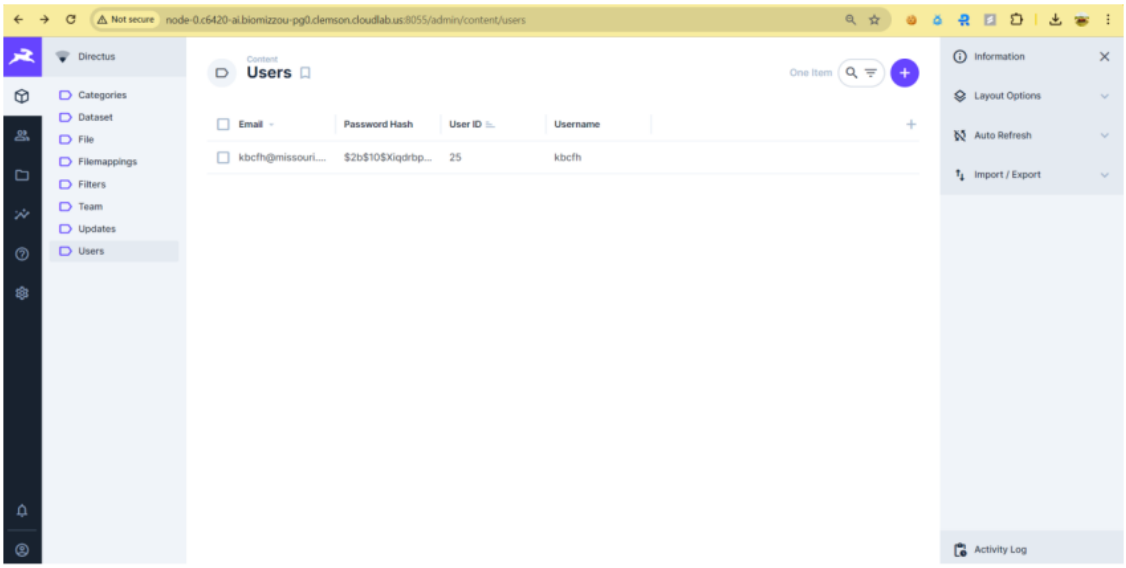


Fig 16 Directus Users Table Interface

5.2.3 JWT Token: The Authentication Token

Each user is assigned a JSON Web Token (JWT), a compact token used for secure communication between parties. This token is stored as a cookie, jwtToken, in the user's browser, with a preset expiration time set for an hour. This method ensures that user sessions are maintained with integrity, as the token includes claims that are verified upon each request to the server.

Subsequent Requests and Session Management

When a user makes a request that requires authentication, the stored JWT is included in the request headers, allowing the server to validate the session continuously. If the token is verified successfully, the server processes the request based on the user's authenticated identity, enabling secure and accurate interactions with OSCB's resources.

The successful authentication process is concluded by redirecting the user to the 'Get Started' page, symbolizing their entry into the OSCB environment.

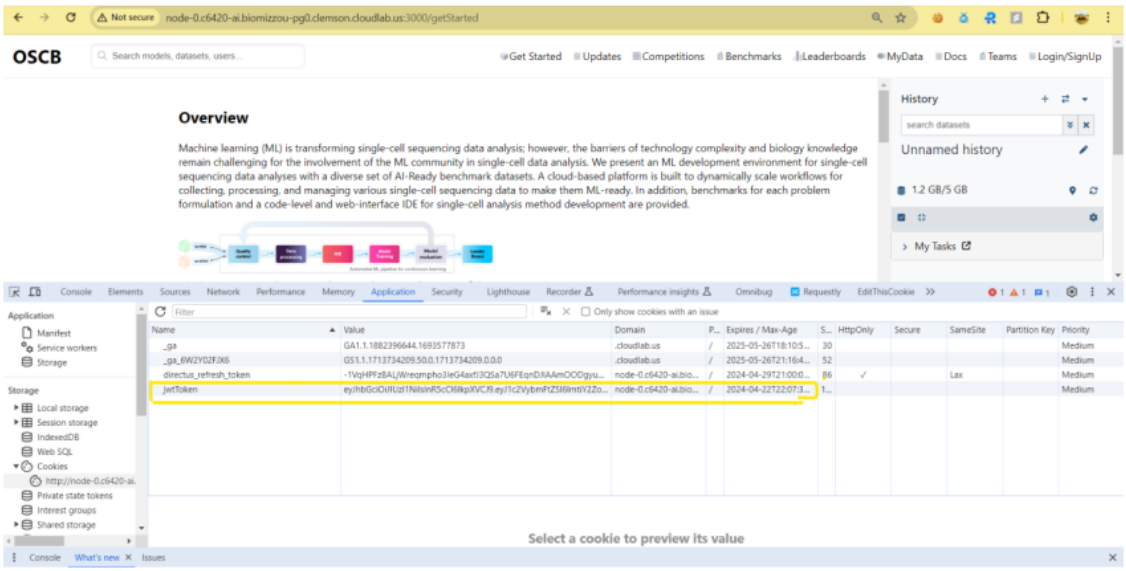


Fig 17 JWT Token Cookie Storage

5.3 Content Management and Real-Time Updates

Integration of Directus as a CMS

In the world of web development, efficiently managing content is crucial to keeping a platform agile and responsive. At OSCB, we've made a significant change by introducing Directus as our content management system (CMS). This move has had a transformative effect, allowing us to separate content management from the complexities of business logic and application code. As a result, we can now update content in real-time without requiring technical intervention, giving us greater flexibility and control over our platform.

Directus: Bridging Gaps between Technical and Non-Technical Domains

Directus is known for its user-friendly interface, making it easy for non-technical users to interact with MySQL database tables seamlessly. This interaction goes beyond simple data entry and includes file management directly within the database, showcasing the system's versatility. At OSCB, we've configured Directus to integrate seamlessly with MySQL, laying a solid foundation for dynamic content management.

Content Deployment on OSCB

Leveraging Directus APIs, OSCB renders several crucial pages that serve as the face of the platform:

5.3.1 GetStarted Page

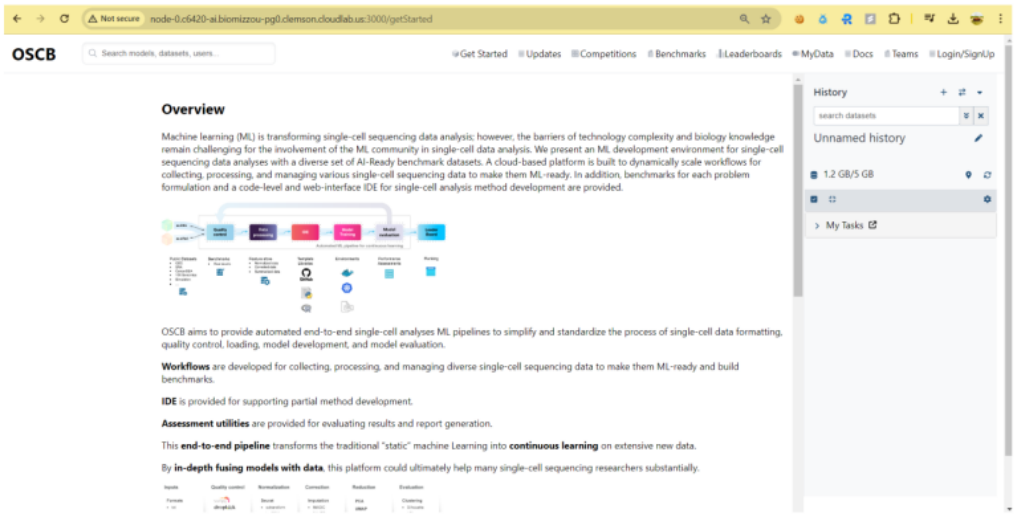


Fig 18 Get Started Page

The GetStarted page serves as the introductory hub for users, offering insights into the purpose and functionalities of the OSCB website. It acts as a starting point, providing answers to common questions and guiding users on how to navigate the platform effectively.

Implemented through Directus, the GetStarted page leverages the flexibility of its APIs to manage content seamlessly. A dedicated MySQL table named **filemappings** stores essential information about files, with filenames serving as keys for retrieval. Within this table, a markdown file named "getstarted" encapsulates the content for the GetStarted page.

Upon accessing the GetStarted page, the React application routes the request to the corresponding component. Utilizing Directus APIs, the application fetches the markdown file

dynamically. Subsequently, the react-markdown npm package converts the markdown content into HTML, facilitating its display to users.

Directus API Endpoint -

http://<HostName>:<Port>/items/filemappings?filter[filename]=getstarted"

A notable advantage of this approach is the agility it offers in content management. Non-technical users, such as business stakeholders, can effortlessly update the contents of the GetStarted page in near real-time. By simply replacing the file in the filemappings table, subsequent requests to Directus will retrieve the latest content, eliminating the need for application redeployment. This decoupling of content management from core business logic empowers stakeholders with enhanced flexibility and responsiveness in content administration.

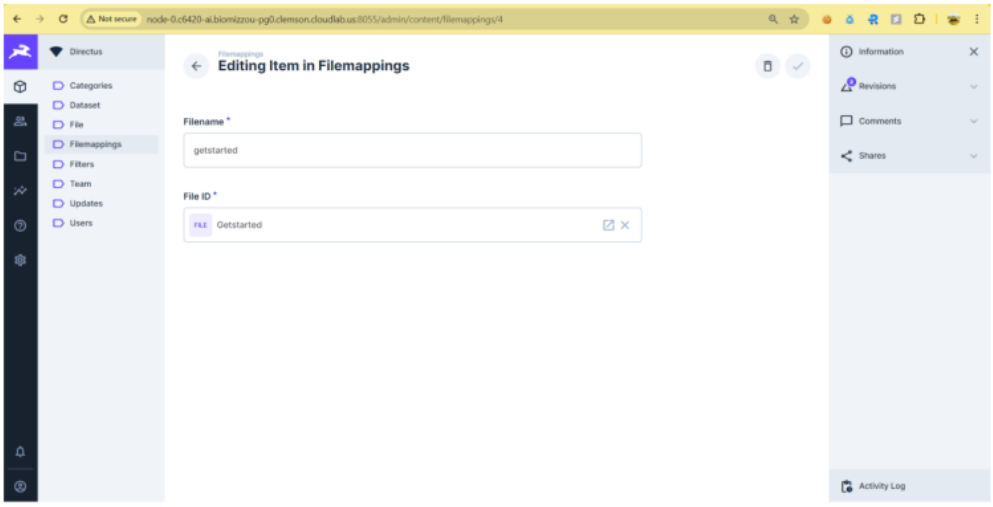


Fig 19 Directus filemappings Table Interface for Get Started Page

5.3.2 Updates Page

The 'updates' table in Directus to include detailed information such as titles, publication dates, and descriptions. This setup ensures that the newest updates are prominently featured, keeping users engaged with the freshest content available.

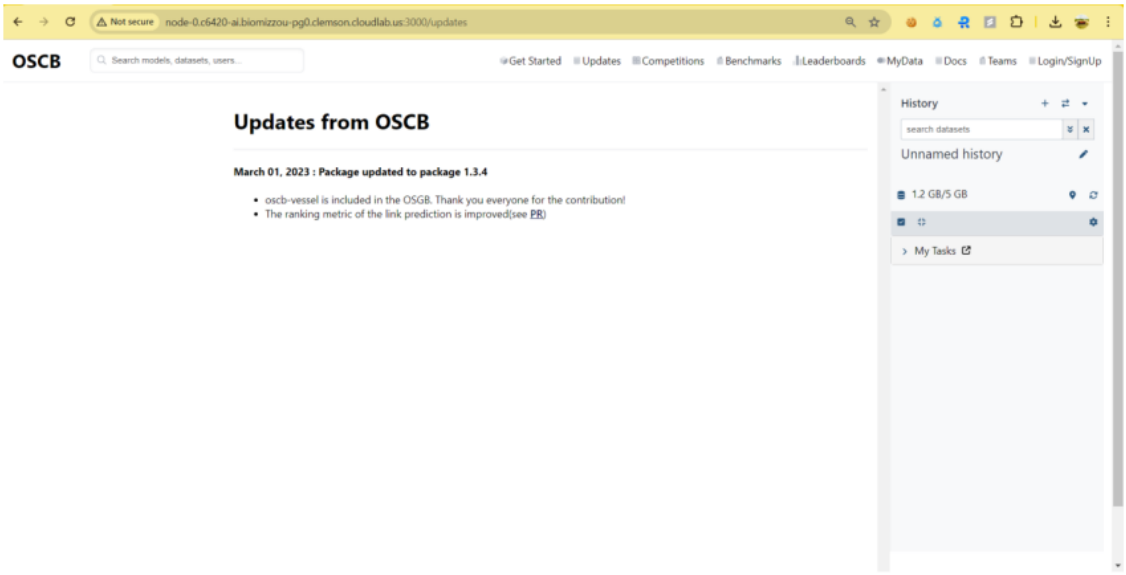


Fig 20 Updates Page

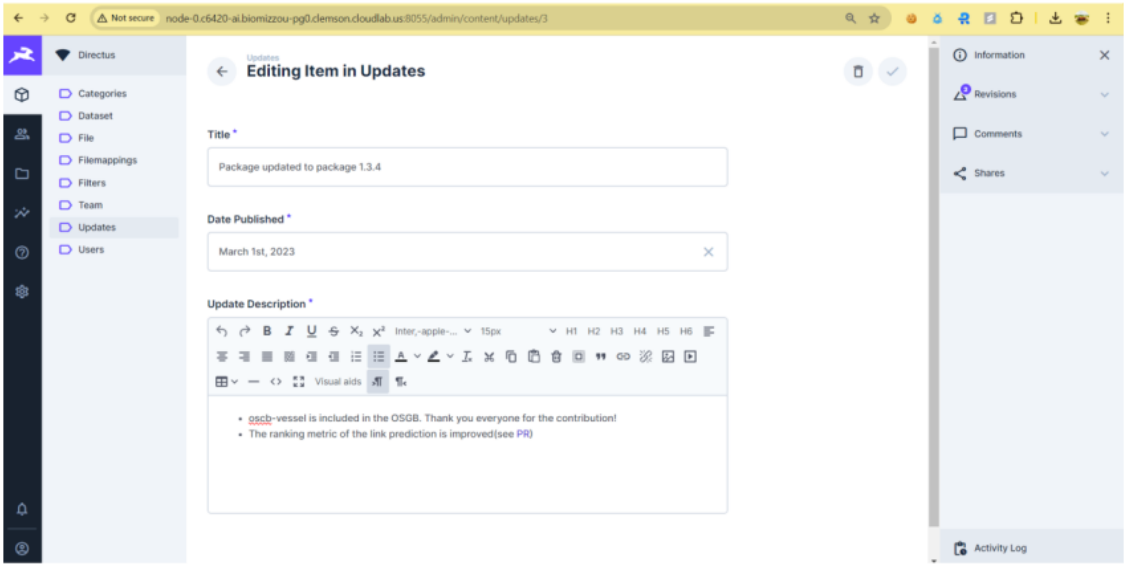


Fig 21 Directus Updates Table Interface for Updates Page

5.3.3 Teams Component:

To promote transparency and community engagement, we've structured the 'Team' table in Directus to showcase the details of OSCB's core developers and steering committee. This table is

carefully designed to display team members in a predetermined order, ensuring that each member receives appropriate recognition and highlighting their contributions to the project. By providing this information, we aim to enrich the user's understanding of the project's leadership and the diverse talents driving its success.

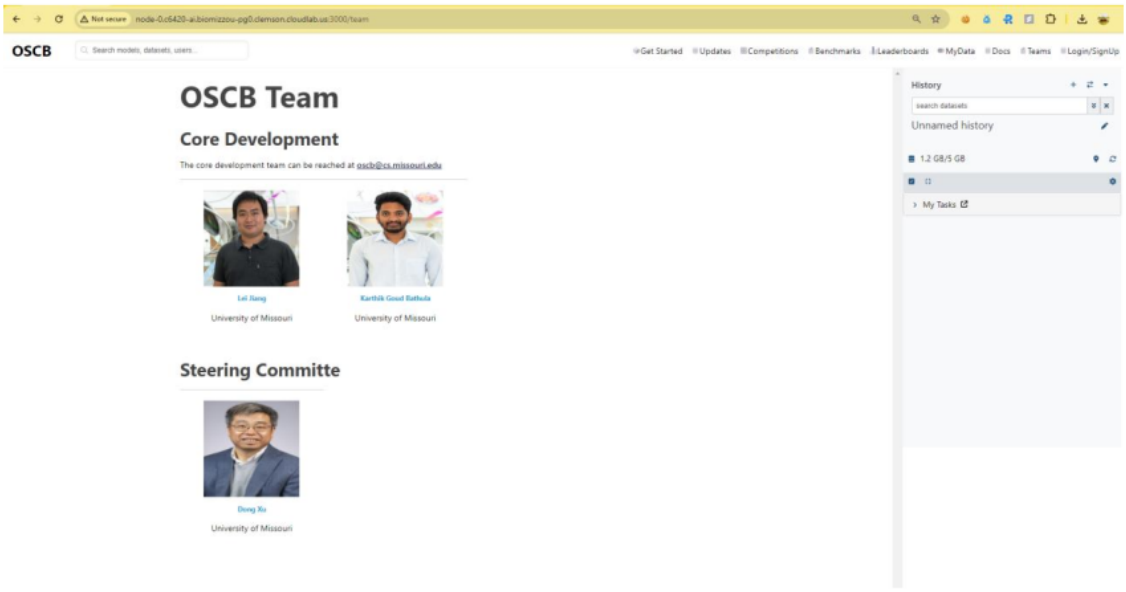


Fig 22 Teams Page

By employing Directus, OSCB has placed the power of content management in the hands of those who may not be versed in coding, yet are integral to the platform's narrative and evolution. This strategic implementation speaks to the adaptability and user-centric focus of OSCB, allowing for a dynamic and engaging platform that serves its community with both stability and innovation.

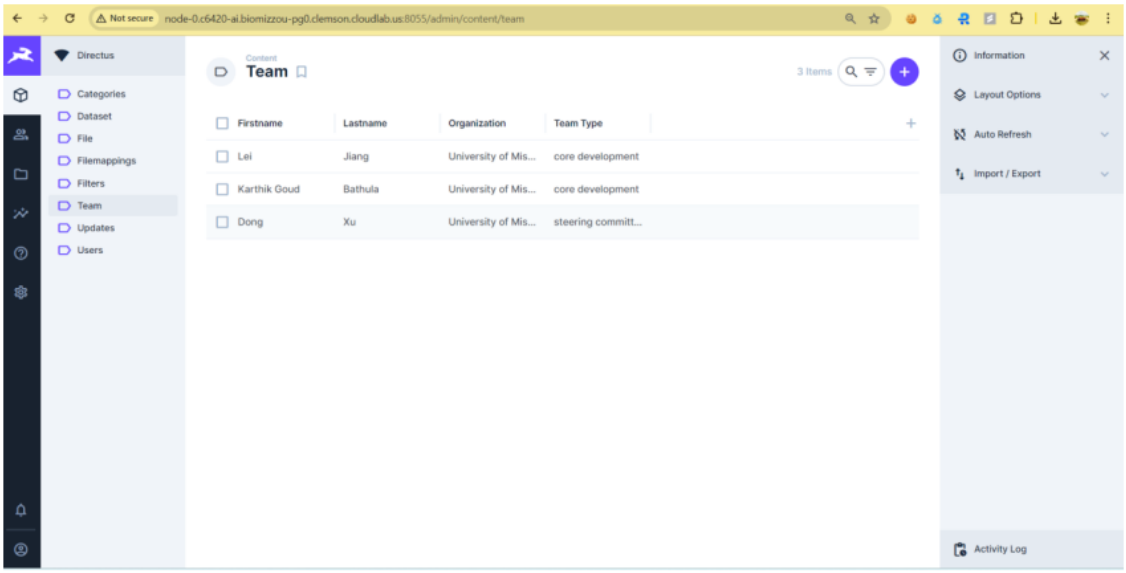


Fig 23 Directus Team Table Interface for Team Page

5.4 Facilitating User Engagement through Dataset Uploads

5.4.1 Optimizing the Data Upload Workflow

A key aspect of the OSCB platform is its data upload workflow, which is carefully designed to gather and process AI-Single-cell datasets submitted by users. Understanding the significance of each dataset in advancing scientific inquiry, OSCB provides 5GB of personal storage for each user, encouraging active participation and contribution. This initiative serves as a gateway for users to explore and utilize the diverse range of analytical tools and workflows offered by OSCB, empowering them to further their research endeavors.

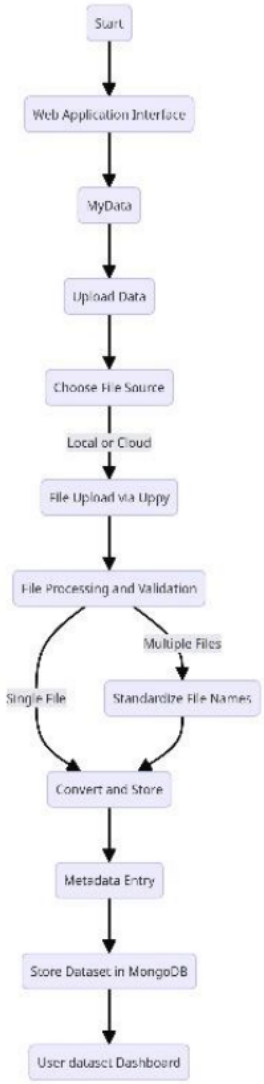


Fig 24 Workflow for Dataset Uploads

5.4.2 Data Upload Step

5.4.2.1 Implementing Uppy for File Management

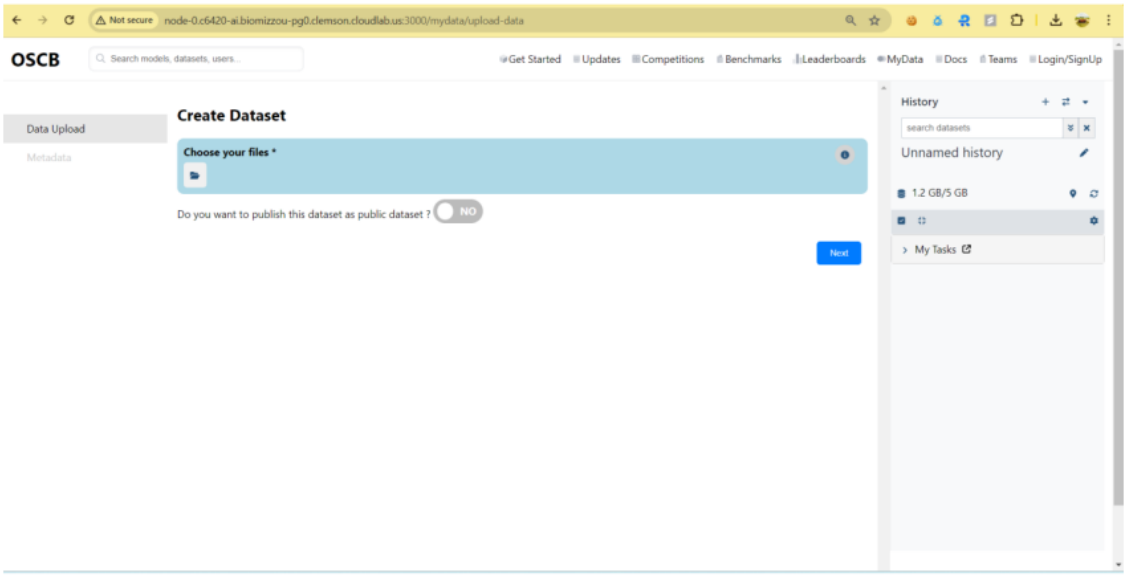


Fig 25 Step 1 of Datasets upload process

To kickstart the upload process, we've leveraged the power of Uppy, a reliable file management tool, to enable users to upload datasets from various sources. Its integration into OSCB's framework ensures users can effortlessly transfer files from their local systems, Google Drive, and Dropbox, highlighting the platform's flexibility. This initial step is divided into two separate processes. For single-file datasets, the extension of the file determines the method of data parsing. Meanwhile, multi-file datasets require standardized naming conventions for proper file identification and seamless integration into OSCB's processing pipeline.

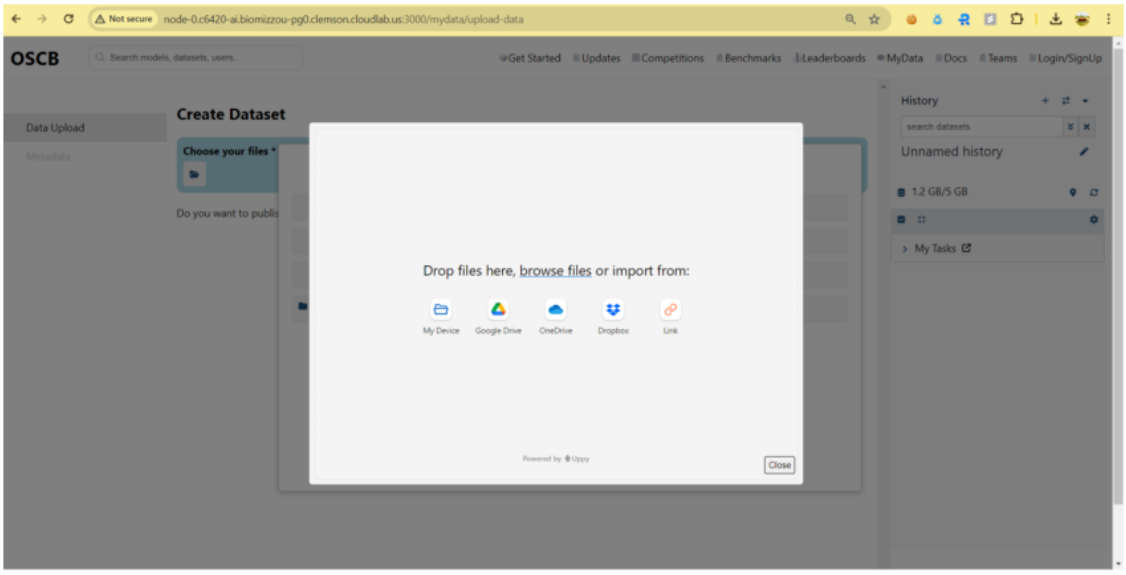


Fig 26 Uppy Integration with Upload Step

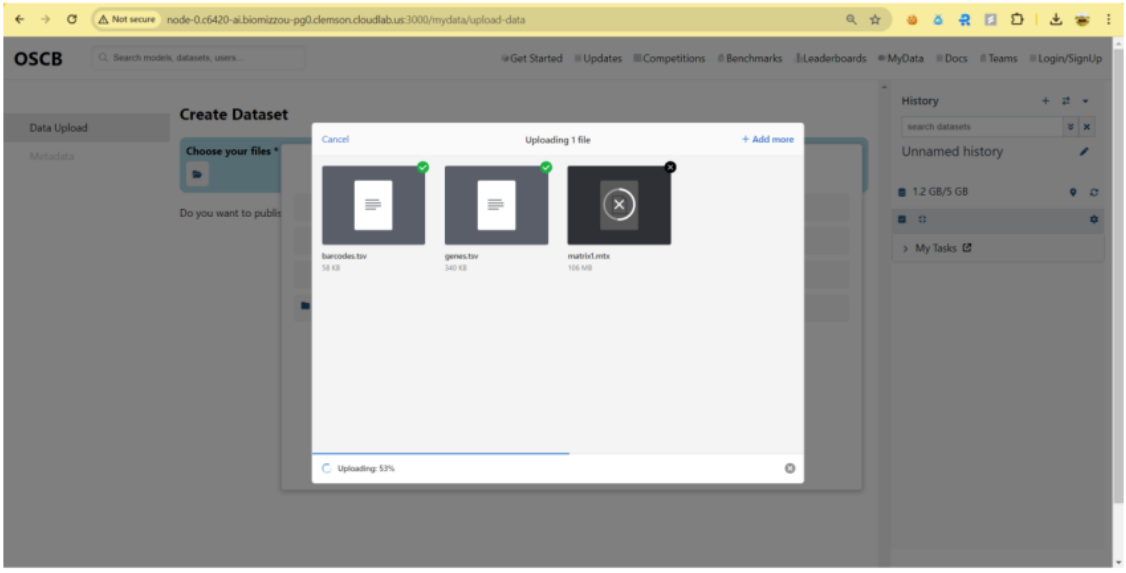


Fig 27 Uppy multiple file uploads

5.4.2.2 Ensuring Consistency Across Datasets

Single File Upload: For single-file uploads, the focus shifts from filename relevance to file extension significance. OSCB's backend processing logic relies on file extensions to determine

data processing methods. Supported single file formats encompass ".tsv", ".csv", ".txt.gz", ".txt", ".h5ad", "rds", "h5seurat", "tsv.gz", "mtx.gz", "h5", "xlsx", "hdf5", "gz", "Robj", "zip", "rar", "tar", "tar.bz2", "tar.xz".

Multiple File Upload:

When users upload multiple files, OSCB employs a custom logic to prompt for standardization if the filenames do not match expected backend processing protocols. This is a critical function, ensuring that datasets can proceed smoothly through the platform's analysis tools. A dropdown list provides users with standard naming options, aligning with the backend requirements.

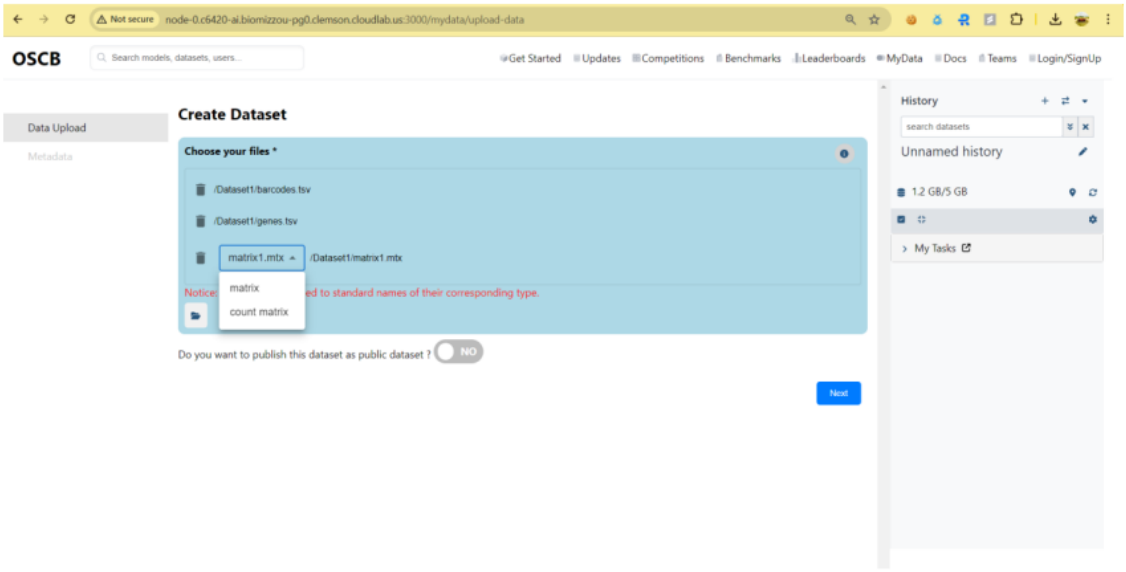


Fig 28 Functionality for maintaining standard filenames

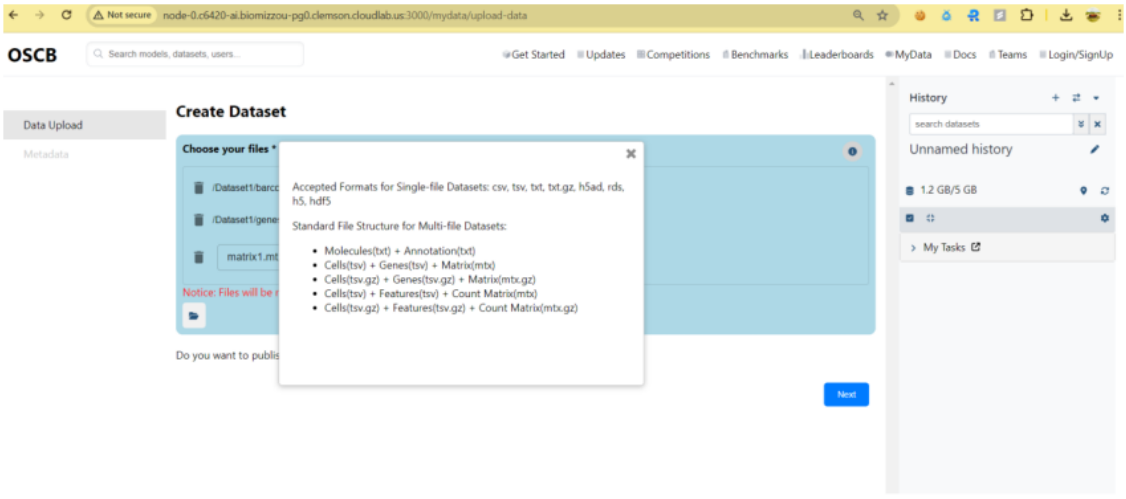


Fig 29 Accepted File Formats Tooltip

5.4.2.3 Dataset Selection:

From Public Datasets: OSCB extends users the option to select datasets from the publicDatasets folder, housing datasets shared by fellow users within the platform. This feature enhances data accessibility and collaboration among users.

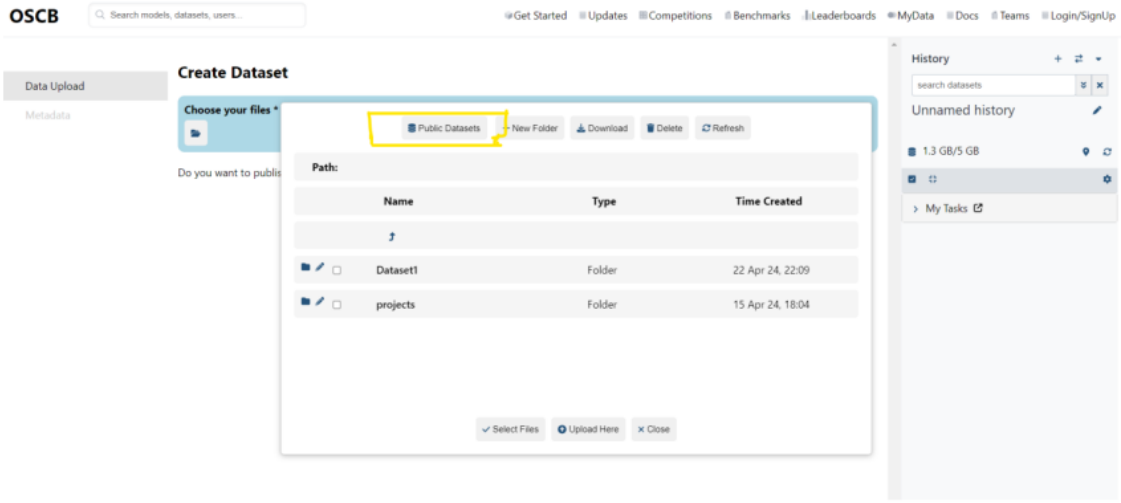


Fig 30 Dataset Uploads from Public Directory(Shared)

5.4.2.4 Facilitating Dataset Sharing

One standout feature of this upload interface is the option for users to share their datasets with the wider OSCB community. With a simple toggle switch, users can choose to make their datasets publicly available, fostering a collaborative atmosphere within the platform. This functionality reflects OSCB's commitment to advancing science through shared knowledge, empowering users to both contribute to and benefit from a collective pool of resources.

5.4.2.5 Data Conversion

After the user uploads a file and clicks the "Next" button, OSCB ensures that the dataset is converted to either AnnData (.h5ad) or Seurat (.h5seurat) format to maintain consistency for downstream analysis across the platform. This conversion process is facilitated by making a backend API call and passing relevant input parameters. The backend API handles the conversion of the dataset to the specified formats.

During this process, two checks are performed:

- If the uploaded file is identified as a Seurat file (.h5seurat, .Robj, or .rds), OSCB prompts the user to select the default assay from a list of available assays. This step is crucial because Seurat datasets organize data into "assays," each representing different omics data types (e.g., gene expression, chromatin accessibility). Selecting the default assay ensures that downstream analyses are performed on the appropriate data type, enhancing clarity and accuracy in the analysis pipeline. By empowering users to choose the default assay, OSCB promotes control, transparency, and prevents errors or unintended consequences that may arise from analyzing the wrong data type. This selection process encourages users to align their analysis goals with the nature of their data effectively.
- For any other file format, OSCB proceeds with the same backend API call to convert the dataset to an AnnData object. The resulting paths are then stored in the react state and moved to the next step of the workflow. This streamlined process ensures that datasets are converted and prepared for downstream analysis seamlessly, regardless of their original format.

5.4.3 Metadata Step

The second step of the upload process is the meticulous curation of metadata. Users are presented with a form designed to capture detailed information about their datasets, including the biological species involved, authorship details, and the date of dataset publication. This metadata is not merely additional information but serves as a scaffold for data discoverability and reuse within the research community.

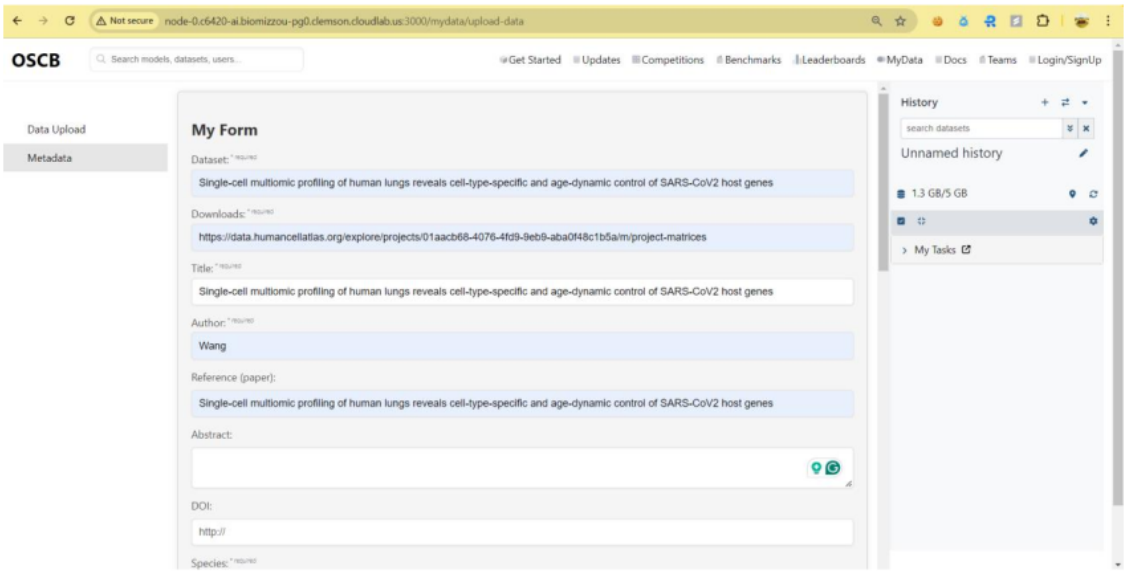


Fig 31 Step 2 of Datasets upload process – Metadata

5.4.3.1 Validating User Input

To maintain the integrity of data submissions, OSCB employs error tooltips to prompt users to complete all required metadata fields. This validation process underscores the platform's commitment to data quality and comprehensive documentation.

5.4.4 Storing and Finalizing Datasets

Upon successful completion of both upload and metadata collection steps, users initiate dataset submission, triggering an API call to store dataset details comprehensively. This includes

metadata information and file data, meticulously recorded within OSCB's MongoDB user-datasets collection. By centralizing dataset management within MongoDB, OSCB ensures efficient data storage, retrieval, and accessibility, empowering users with seamless access to contributed datasets for collaborative research endeavors.

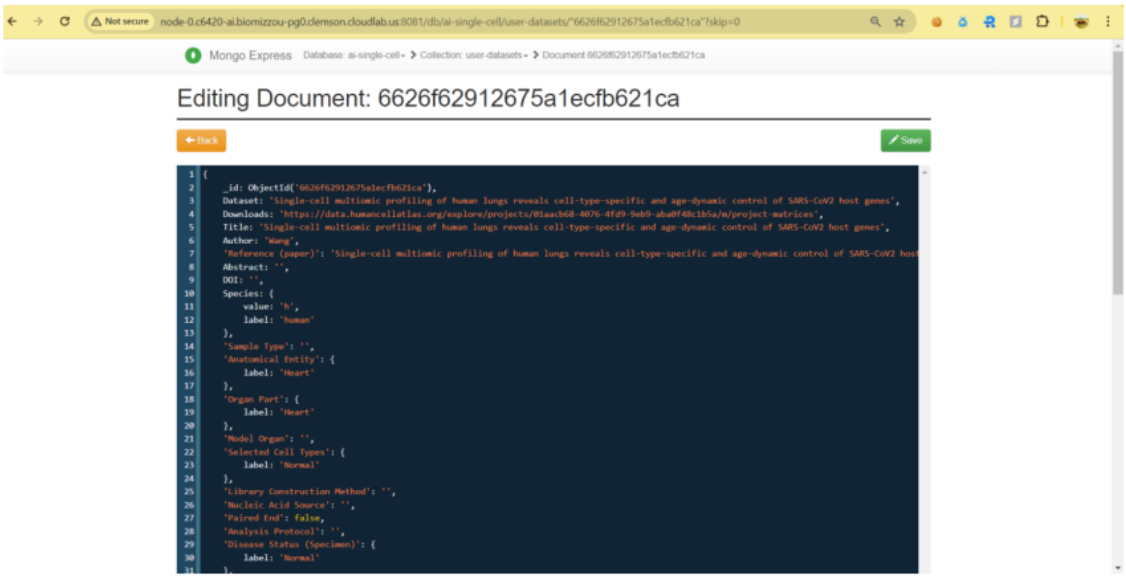


Fig 32 MongoDB user-datasets collection storing user created datasets

5.5 Tool Access

The OSCB platform's Tools page serves as a cornerstone for user interaction with datasets. Its design revolves around the objective of displaying a suite of analytical tools, allowing users to run an array of analyses on their own datasets, be they personal, shared, or benchmark datasets. The layout is conceived to provide an intuitive and efficient user experience through three main components, detailed below.

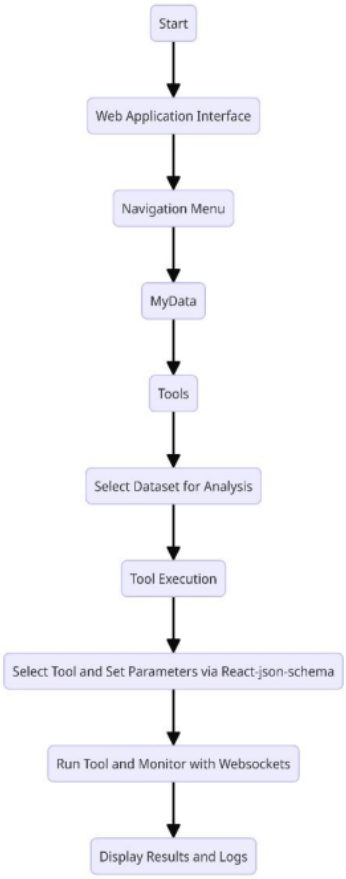


Fig 33 Workflow for Tools Execution Process

5.5.1 Navigational Ease with the Leftnav Component

Central to the user's navigation is the Leftnav component, which presents the available tools categorized under labels such as quality control, normalization, integration, evaluation, and imputation. The implementation mirrors the functionality of facets and filters found on e-commerce sites, providing a familiar interface to users.

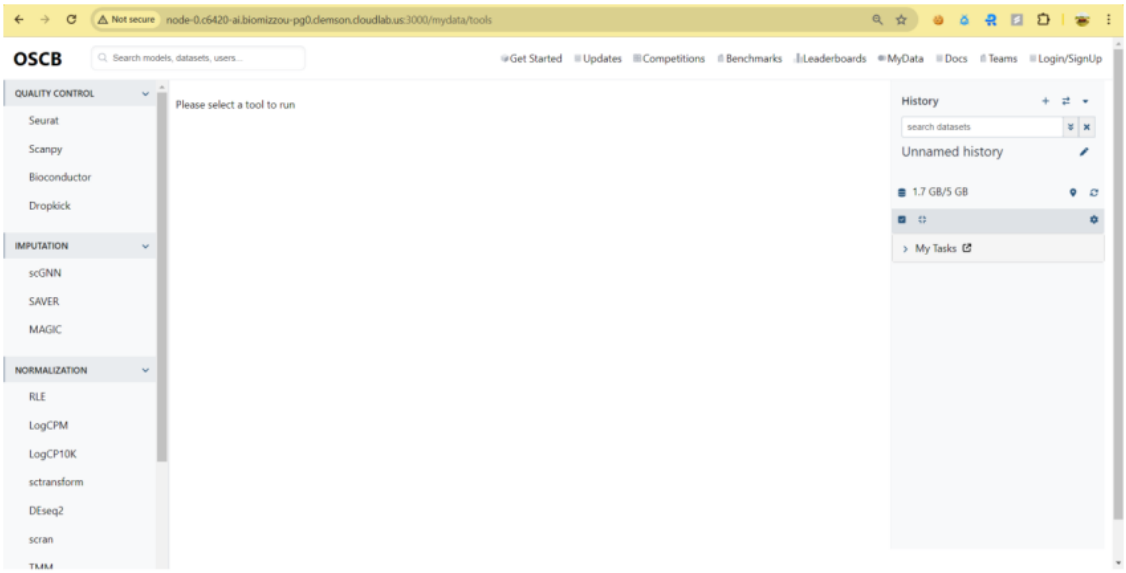


Fig 34 Tools Page With LeftNavigation

In the implementation, I've set up two tables in the MySQL database: "category" and "filters". The "category" table contains predefined categories such as quality control, normalization, integration, evaluation, and imputation. These categories remain static and are not subject to change dynamically. Each category has multiple filters associated with it, and these filter values are stored in the "filters" table. To establish a relationship between the two tables, I've implemented a foreign key constraint, where the "Id" field from the "category" table corresponds to the "categoryId" field in the "filters" table.

To interact with this database setup, I've developed a new API endpoint in Node.js. This API connects to the MySQL database and executes queries to fetch data from both the "category" and "filters" tables. The response from these tables is structured as facets and filters. Each facet represents a category along with its associated filters. This structured data is then sent to the frontend for rendering.

In the frontend implementation, the received facet data is rendered using the leftnav component on the tools page. By default, the first three facet categories are expanded, displaying their respective filters, while the remaining categories are collapsed. This design choice optimizes the

user experience by providing easy access to commonly used categories and filters while minimizing clutter and allowing users to expand additional categories as needed. Overall, this implementation ensures efficient navigation and usability within the OSCB platform's tools section.

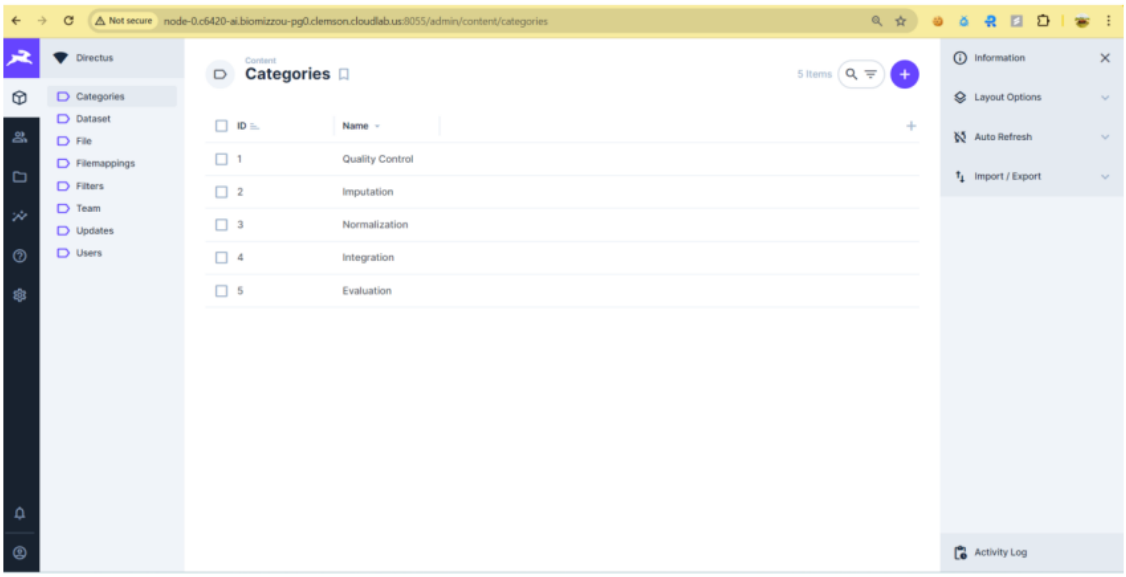


Fig 35 Directus Categories Table Interface

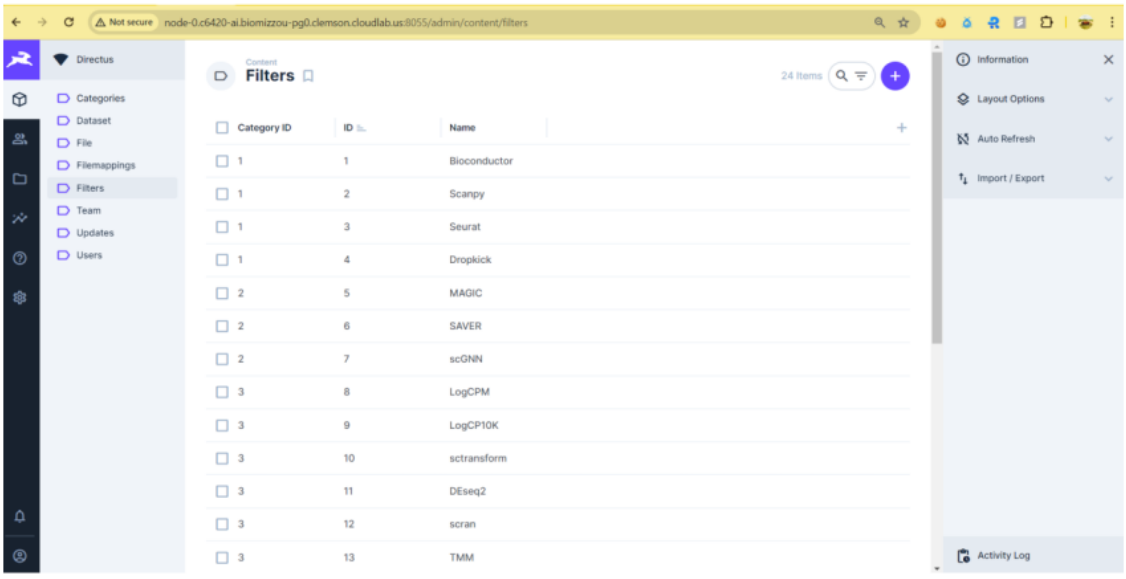


Fig 36 Directus Filters Table Interface for Tools Page

5.5.2 Main Content Component

The main content component of our application comprises two key sections: Dataset selection and Tool-specific parameters. To capture tool-specific parameters, we've leveraged react-json-schema, an intriguing package for rendering forms dynamically within React applications.

React-json-schema simplifies form development by allowing us to define form structures using JSON schemas. We've established schemas for each tool along with corresponding UI schemas. These schemas are imported dynamically based on the user's tool selection, ensuring the form aligns with the selected tool's requirements. One notable advantage is its ability to swiftly generate aesthetically pleasing forms while ensuring data structure integrity and performing validations as per the defined schema.

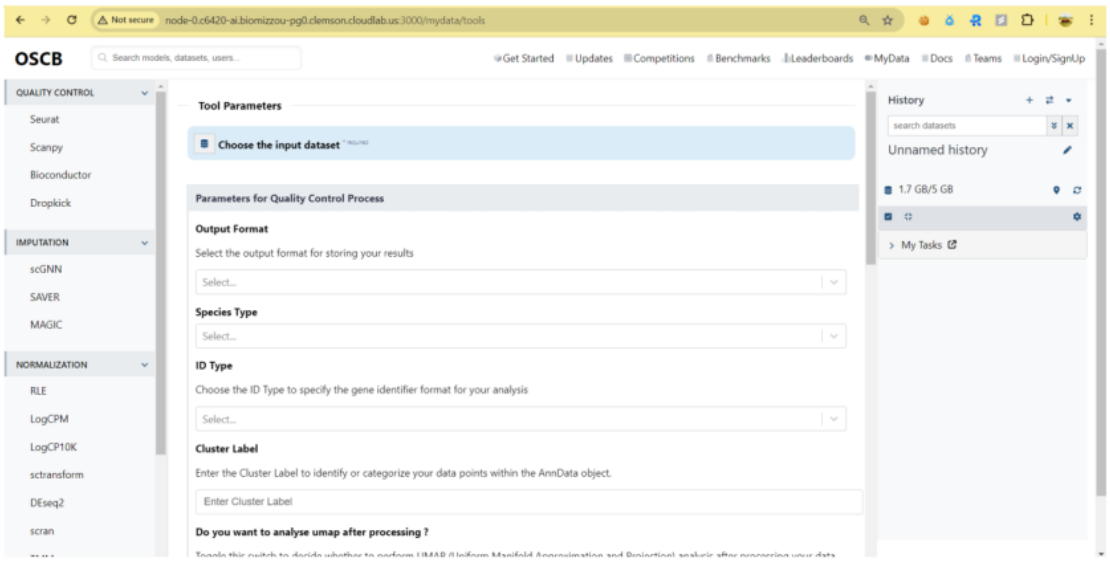


Fig 37 Tools Main Component Form Using react-json-schema

Let's take the scanpy tool from the quality control category as an example. The react-json-schema for scanpy defines the form's structure, including parameters like clustering method and resolution. Meanwhile, the UI schema may specify custom UI widgets for enhanced user experience.

5.5.2.1 react-json-schema for the scanpy tool:

```
{
  "type": "object",
  "properties": {
    "parameters": {
      "type":"object",
      "title": "Parameters for Quality Control Process",
      "properties": {
        "output_format": {
          "type": "string",
          "title": "Output Format",
          "description": "Select the output format for storing your results"
         },
        "species": {
          "type": "string",
          "title": "Species Type"
         },
        "idtype": {
           "type": "string",
           "title": "ID Type",
           "description": "Choose the ID Type to specify the gene identifier
format for your analysis"
         },
         "cluster_label": {
          "type": "string",
          "title": "Cluster Label",
          "description": "Enter the Cluster Label to identify or categorize your
data points within the AnnData object."
        },
        "show_umap": {
          "type": "boolean",
          "title": "Do you want to analyse umap after processing ?",
          "default": true,
          "format": "switch",
          "description": "Toggle this switch to decide whether to perform UMAP
(Uniform Manifold Approximation and Projection) analysis after processing your
data."
        },
        "show_error": {
          "type": "boolean",
          "title": "Do you want to see the errors of the task execution ?",
          "default": true,
          "format": "switch",
          "description": "Errors during task execution are shown by default,
aiding in troubleshooting and refining your analysis. Disable if preferred."
        },
```

5.5.2.2 UI-schema for the scanpy tool:

```
export const uiSchema = {
  "parameters": {
    "classNames": "category",
      "output_format": {
        "classNames": "sub-category",
        "ui:widget": "SelectComponent",
        'ui:options': {
          clearable: true ,
          placeholder: "Select the Output Format",
          creatable: false,
          searchable: true,
          opts:["AnnData", "SingleCellExperiment", "Seurat", "CSV"]
        }
      },
      "species": {
        "classNames": "sub-category",
        "ui:widget": "SelectComponent",
        'ui:options': {
          clearable: true ,
          placeholder: "Select the Species type",
          creatable: false,
          searchable: true,
          opts:["human", "mouse"]
        }
      },
      "idtype": {
        "classNames": "sub-category",
        "ui:widget": "SelectComponent",
        'ui:options': {
          clearable: true ,
          placeholder: "Select the ID type",
          creatable: false,
          searchable: true,
          opts:["SYMBOL", "ENSEMBL", "ENTREZID", "REFSEQ"]
        }
      },
      "cluster_label": {
        "classNames": "sub-category",
        "ui:widget": "ClusterLabelInput"
      },
      "show_umap": {
        "classNames": "sub-category",
        "ui:widget": "toggle"
      },
      "show_error": {
```

5.5.2.3 Schema Explanation

JSON Schema:

- The JSON schema outlines the structure of the form, specifying the properties and their respective types.
- It includes parameters such as Output Format, Species Type, ID Type, Cluster Label, and various quality control parameters.
- Each parameter is defined with its data type, title, description, default value, and additional options where applicable.
- For example, "output_format" is defined as a string with options for selecting the output format, while "show_umap" and "show_error" are boolean switches to control visibility.

UI Schema:

- The UI schema complements the JSON schema by defining the appearance and behavior of the form elements.
- It specifies custom widgets, classNames for styling, and options for select components.
- Each parameter in the JSON schema is mapped to its corresponding UI representation in the UI schema.
- For instance, "output_format" is configured with a custom SelectComponent widget, allowing users to choose from predefined options with search functionality.
- Similarly, "show_umap" and "show_error" utilize toggle switches for user-friendly interaction.

Explanation:

- This form is designed to capture essential parameters required for conducting quality control processes within our application.
- Users can select output formats, species types, and ID types relevant to their datasets, ensuring compatibility with downstream analyses.
- Quality control parameters such as min_genes, max_genes, and n_neighbors are provided with range sliders for precise input.
- Custom UI widgets and configurations enhance user experience by offering intuitive controls and clear visual cues.
- Overall, this form facilitates seamless parameter input, ensuring users can tailor their quality control processes according to their specific requirements.

Upon form submission, react-json-schema generates JSON output, which can be directly utilized by backend APIs. This seamless integration streamlines data transmission and processing.

5.5.2.4 Dataset Selection

Moving on to Dataset selection, this component is pivotal as users must choose a dataset to utilize any tool. Users can select datasets from their own storage, shared datasets, or benchmarks. We've implemented three checkboxes—Private, Public, and Shared—allowing users to filter datasets based on their preferences.

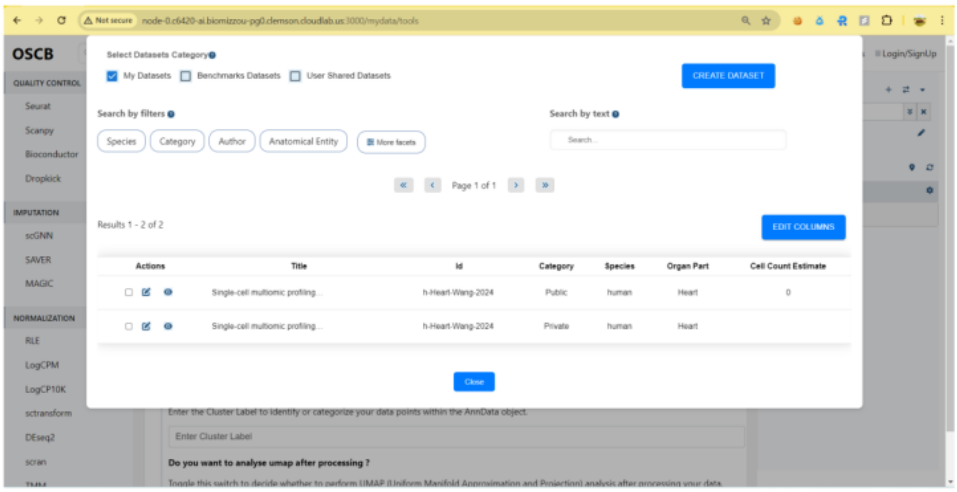


Fig 38 Dataset Selection Interface

To Select the dataset, user must click on the icon beside the label **Choose the input Dataset** to get the expansion panel as shown in the above figure.

This functionality closely resembles a mini e-commerce platform, complete with search, pagination, and filtering capabilities. To support this, I have developed a backend API that interacts with MongoDB, querying datasets based on user selections. Noteworthy collections in our MongoDB include user-datasets for user-created datasets and datasets for admin-created datasets. Shared datasets are stored within the user-datasets collection with a designated category flag.

The backend API receives flags and filters from the frontend, facilitating dataset retrieval based on user-defined criteria. The API returns JSON results, grouped with pagination and facets components, offering users a structured dataset browsing experience.

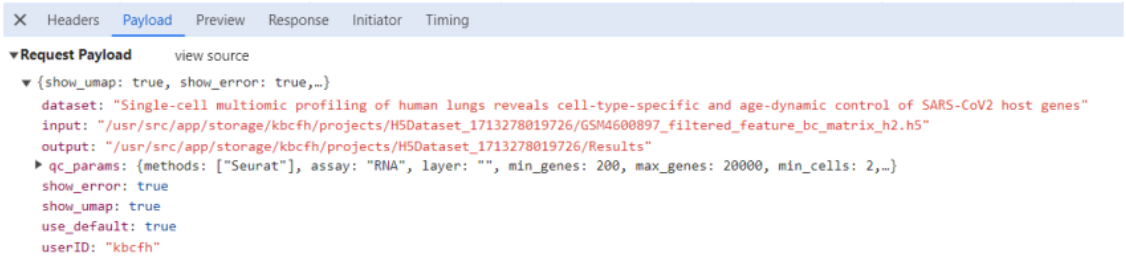
Note: Users are typically restricted to selecting a single dataset for tool execution, except during integration processes where multiple datasets may be necessary to run specific tools.

5.5.2.5 Executing Tools with FastAPI and Celery

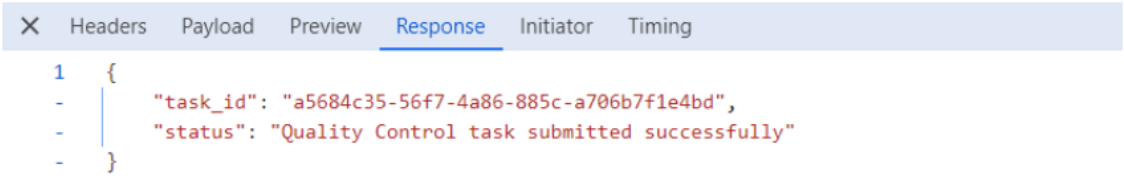
Upon dataset and parameter selection, the backend API, written in Python using FastAPI, takes over. The API request kicks off an analysis task using Celery, a distributed task queue system adept at handling concurrent requests across worker nodes. Each task is assigned a unique 'task_id,' becoming a reference for subsequent status checks, live logs, and results retrieval.

API Details -

Request Payload –



Response Payload –



5.5.2.6 Post-Execution: Tracking and Results

Once you've sent your task, you'll be taken to a page called taskDetails. Here, you'll find all sorts of info about your task, like which dataset you used, what stage your task is at (like waiting, working, or done), and even a live update of what's happening. Once your task is done, you'll see the results right there too.

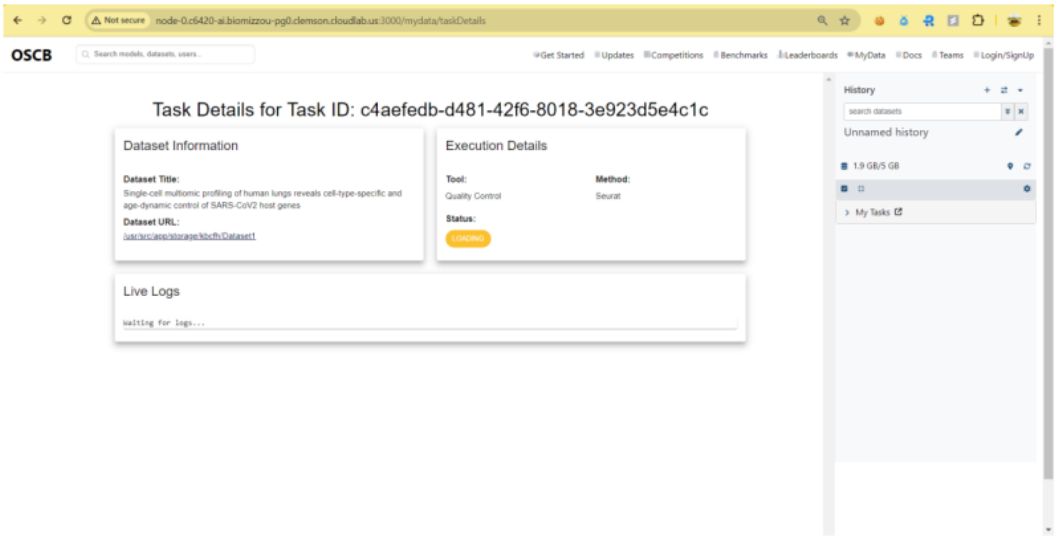


Fig 39 Task Results Page

We also keep a record of all your tasks so you can check back anytime. This way, you can see what you've done before and how it turned out. The taskDetails page is made to be easy to use, so you can find what you need quickly and understand what's going on with your tasks. It's like a control center for all your single-cell data analysis jobs, helping you stay on top of things and get the most out of your analyses.

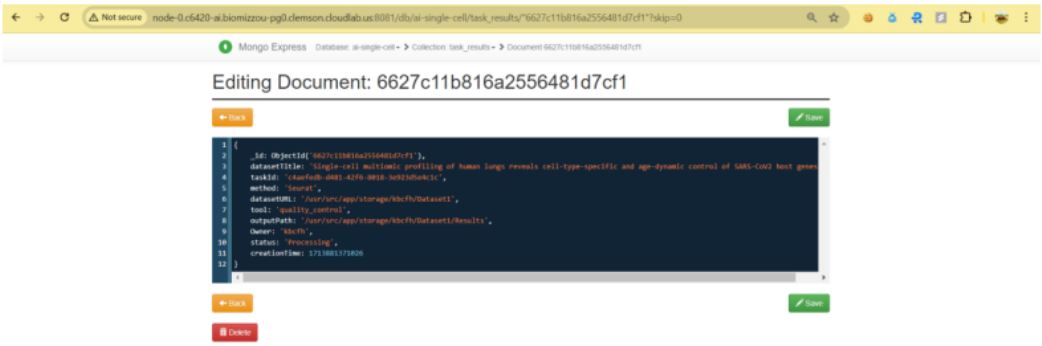


Fig 40 MongoDB task_results collection to store intermediate task information

5.6 Benchmark Datasets Creation Flow

At OSCB, our platform isn't just a collection of analysis tools—it's also a hub for generating and sharing benchmark datasets. These datasets play a crucial role as standard references for scientists to validate and compare different analytical methods. This process is reserved exclusively for admin users and is divided into two main parts: the Benchmark Dataset Creation and the Task Builder Flow.

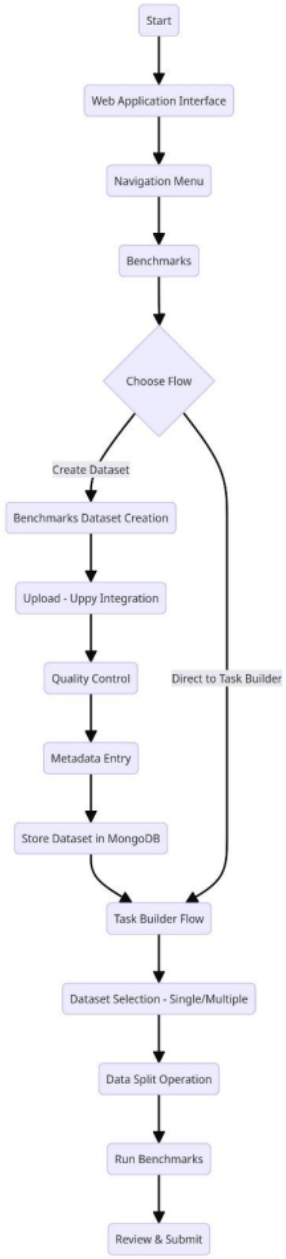


Fig 41 Workflow for Benchmarks Dataset Creation

1. Benchmark Dataset Creation

The creation of benchmark datasets is a structured journey through three fundamental steps: Upload, Quality Control, and Metadata.

5.6.1 Upload Phase

The journey begins with the upload phase. Here, the integration of the Uppy file management tool is pivotal, enabling the admin to upload datasets from a variety of sources with ease. The OSCB's backend logic comes into play when handling multiple files. It ensures filenames align with the backend processing logic through a dropdown menu that prompts the admin to assign standard names to each uploaded file.

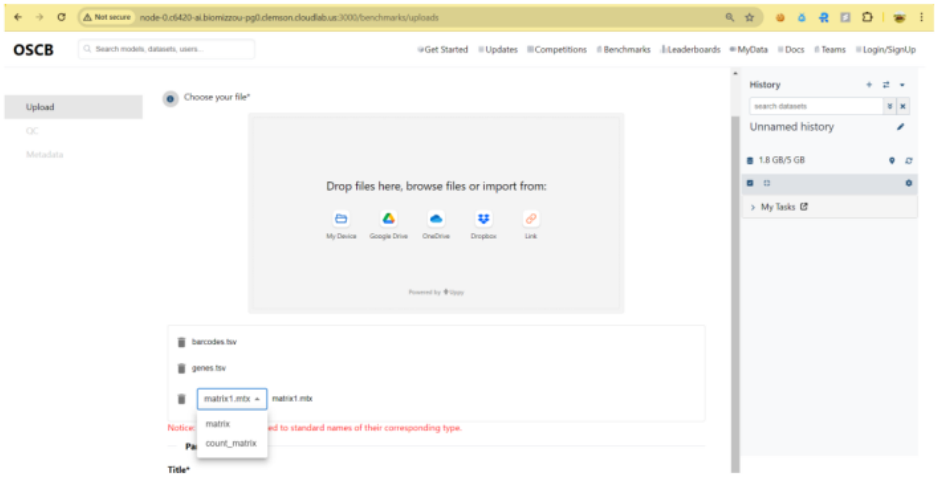


Fig 42 First step of Benchmarks Process - upload

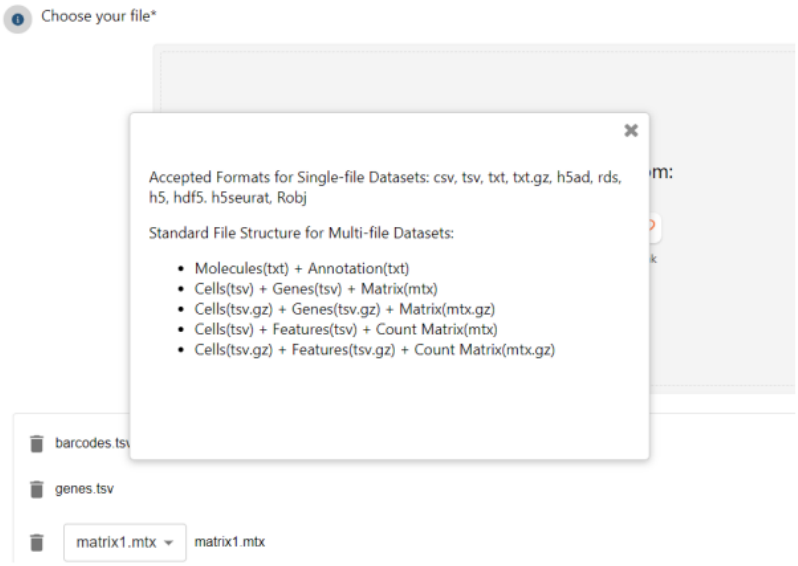


Fig 43 Benchmarks Accepted File Formats Tooltip.

5.6.2 Quality Control Phase:

Once the dataset is uploaded and moves to the quality control step, we offer users a set of quality control parameters to fine-tune the process. By default, these parameters are set to standard values, and users have the option to customize them based on their dataset. When users adjust the parameters, the "use default" switch turns off. If they prefer to stick with the standard settings, they can simply switch it back on.

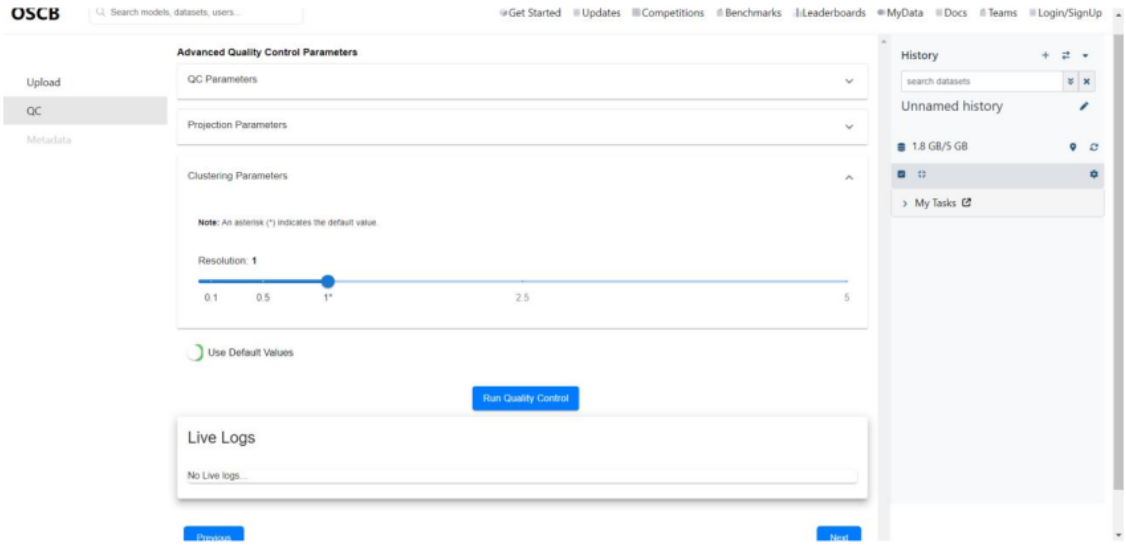
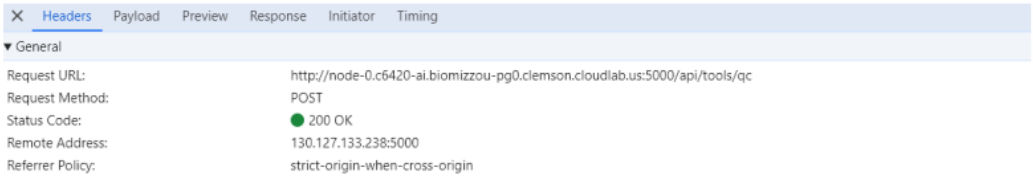


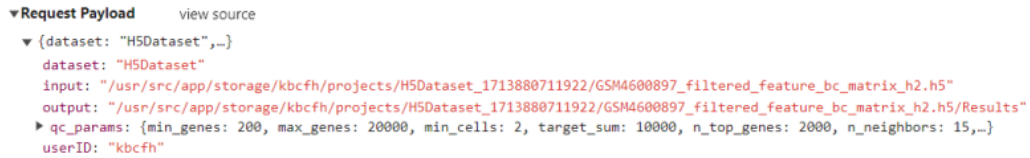
Fig 44 Second step of Benchmarks Process

When users click on the run quality control button after adjusting the parameters, several actions occur simultaneously. First, a backend API call is made to execute the quality control process. Depending on the file type, either a Seurat or Scanpy quality control process is executed. This API generates a task ID using a celery task distributed queue system and returns an output JSON with the task ID. This ID can be used to check the task status, retrieve live logs, and obtain the quality control results once the task is completed.

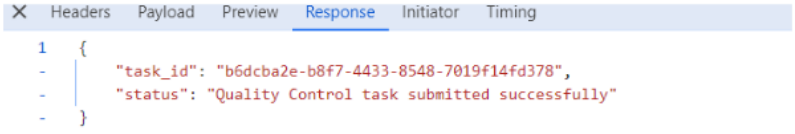
API Endpoint:



Request Payload:



Response Payload:



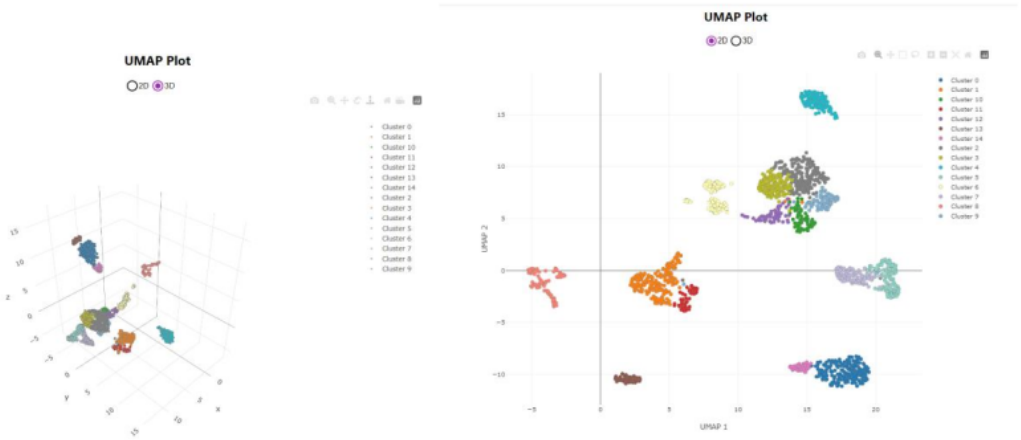
Secondly, two WebSocket connections are established based on the received task ID. One connection is dedicated to updating the task status in real-time. When the task is complete, the WebSocket sends the updated status information to the frontend, allowing the page to reflect the task results accordingly.

The other WebSocket connection is for streaming live logs associated with the task ID. This allows users to monitor the backend processes as they wait for the results. This feature enhances user experience by providing transparency and visibility into the analysis process.

Fig 45 Live Logs

Once the quality control results are generated, the next step is to collect metadata about the dataset. This ensures that users have comprehensive information about their data, facilitating further analysis and interpretation.

Benchmarks Results:



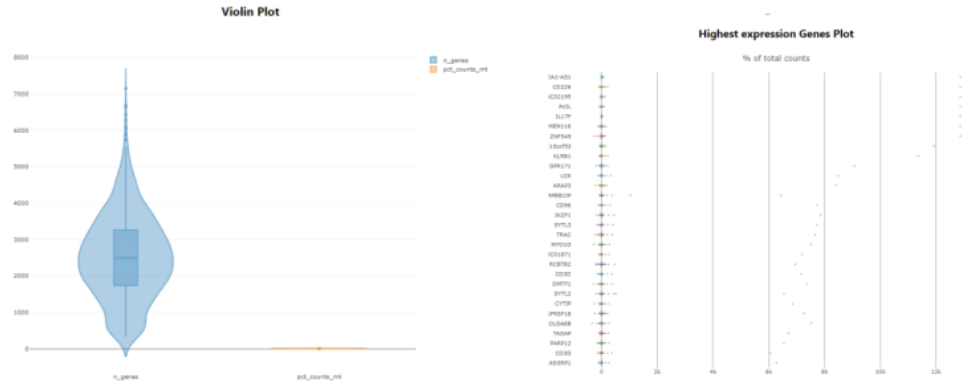


Fig 46 Benchmarks Results

5.6.3 Metadata Collection:

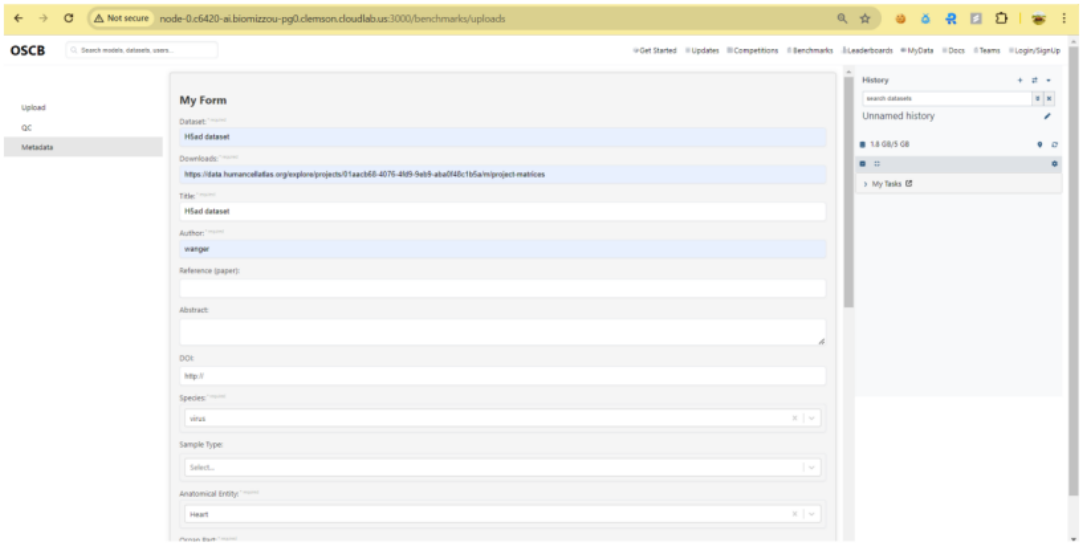


Fig 47 Third step of Benchmarks Process

I designed a form where users can input details about their datasets, such as the species it's related to, who created it, and when it was published. Some fields are mandatory, so if users forget to fill them, little reminders pop up to help them out.

Once users finish filling out the form and hit "submit," their dataset is created and stored in our MongoDB database in **datasets** collection. To handle this, I developed a specialized API using Node.js. It takes all the dataset information from the fsorm and neatly organizes it in MongoDB.

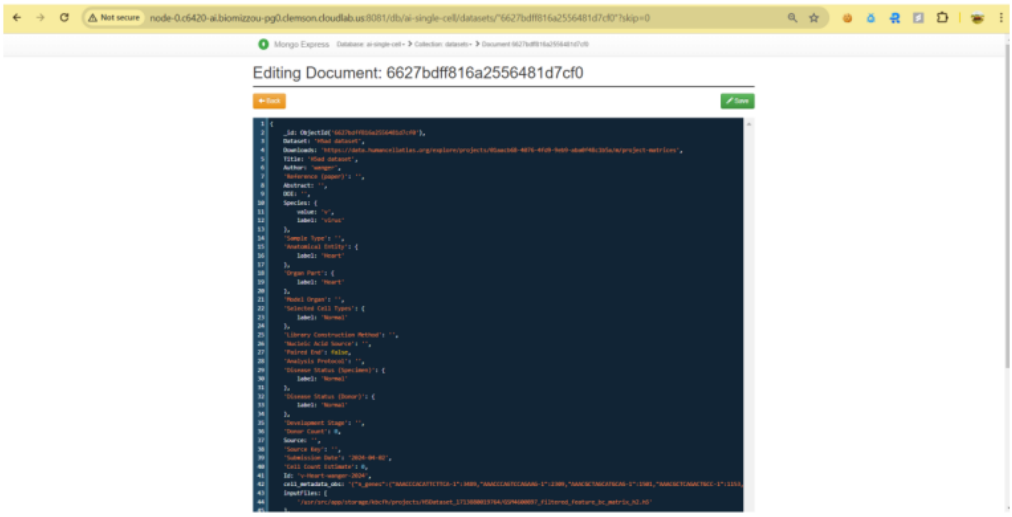


Fig 48 MongoDB datasets collection to store benchmarks datasets

Additionally, any benchmark datasets created by our admin team are also stored in the same database. This way, everything is neatly organized, making it easy for users to access and manage their datasets whenever they need to.

5.7 Task Builder Flow

The Task Builder flow is a sophisticated framework tailored for OSCB admin users, empowering them to leverage benchmark datasets to create and deploy analytical tasks across the platform. This workflow plays a pivotal role in providing comprehensive data analysis capabilities to all OSCB users, enriching the platform's functionality and nurturing a vibrant research ecosystem.

Divided into three strategic phases—Task Builder, Benchmarks, and Review—the Task Builder flow is meticulously crafted to streamline the task creation process, ensuring efficiency and effectiveness at every step.

5.7.1 Task Builder: Dataset Selection

Once a user creates a benchmark dataset, they are automatically directed to the Task Builder Flow, or they can manually navigate to it. The Task Builder Flow consists of three main steps:

- Task Builder
- Benchmarks
- Review

In the Task Builder step, users start by selecting datasets for their tasks. They have the option to choose a single dataset or multiple datasets. The available datasets, stored in the MongoDB **datasets** collection, are displayed in an expansion panel. The interface resembles an e-commerce website, offering search, pagination, and filtering options for easy dataset selection.

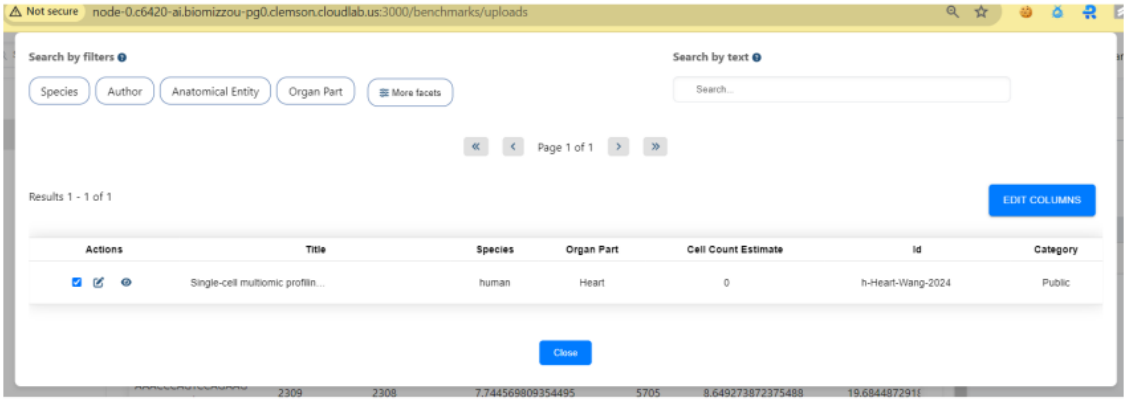


Fig 49 Dataset Selection Interface for Task builder

Once a dataset is selected, users need to specify the task they want to perform. While several tasks are available, currently only clustering is supported. However, future plans include integrating and supporting additional tasks like imputation, marker gene identification, and more.

For each dataset, users specify a task, a label, and data split parameters. Data split fractions determine how the dataset is divided into training, testing, and validation sets. To perform data split, a new API is used, which takes the input file path and fractions and returns a zipped folder containing the split files.

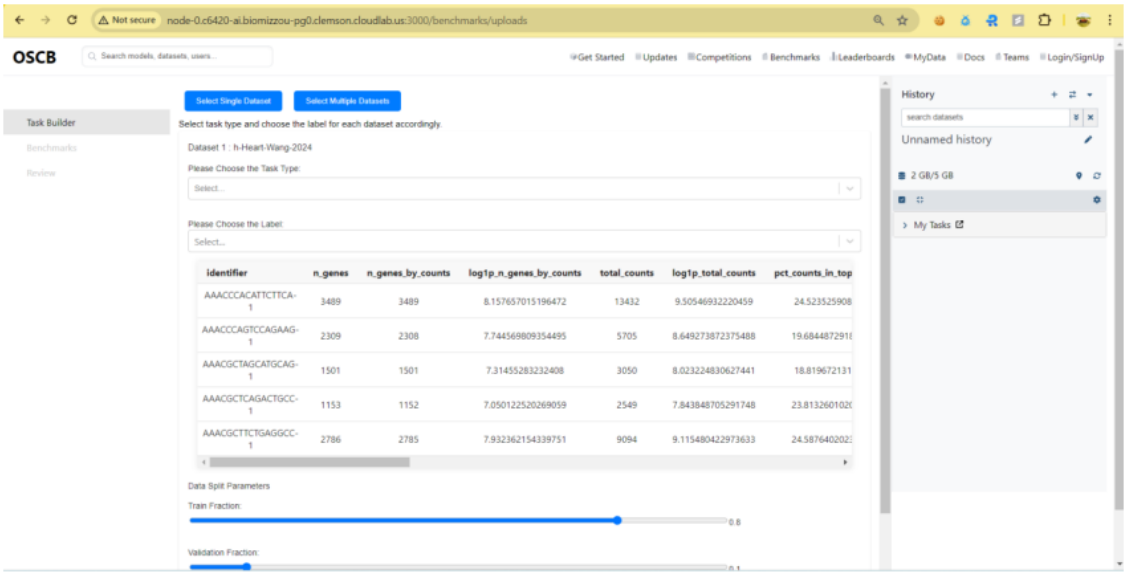


Fig 50 First step of task builder flow

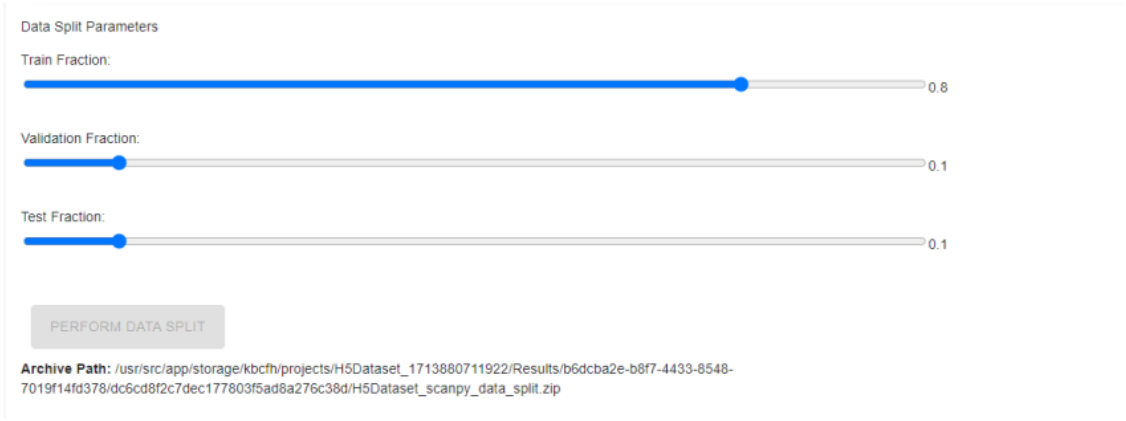


Fig 51 Data split interface

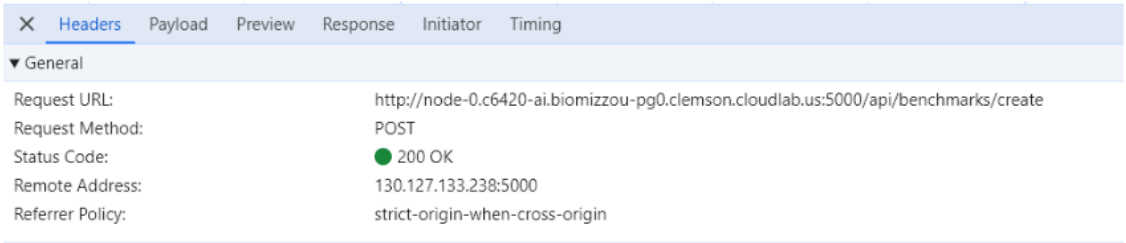
In summary, dataset selection and data split are essential steps to proceed to the next phase of task building.

5.7.2 Benchmarks: Parameter Setting and Task Initialization

State Management with React: OSCB leverages React's state management to keep track of all user inputs and settings throughout the task creation process, ensuring a seamless user experience and data integrity.

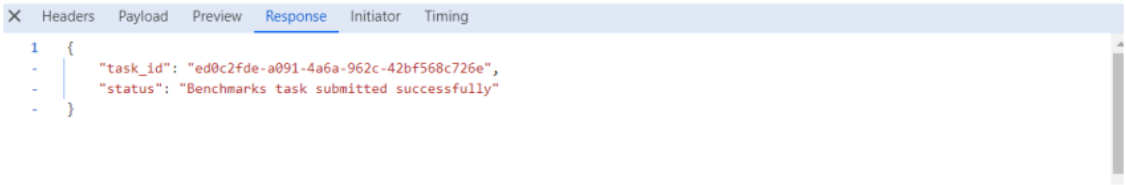
API Integration and Task Execution: A dedicated API call is made to initiate the analysis process. This call includes details such as the dataset ID, the selected analysis type, and user credentials, packaged into a request that triggers the backend processing using Celery. This task management framework efficiently handles concurrent operations, distributing tasks across available resources to optimize performance.

API Endpoint –



Request Payload –

Response Payload –



5.7.3 Review and Finalization

The final Review phase is where the admin users can oversee the tasks they've created, with a comprehensive view of the dataset parameters and preliminary results. This phase is crucial for quality control and ensures that all tasks meet the platform's standards before they are finalized.

Live Updates and Websocket Communication: To enhance transparency and keep the admin informed, OSCB employs websocket connections that provide live updates and logs of the

ongoing tasks. This real-time feedback mechanism is integral for monitoring the analysis process and making necessary adjustments.

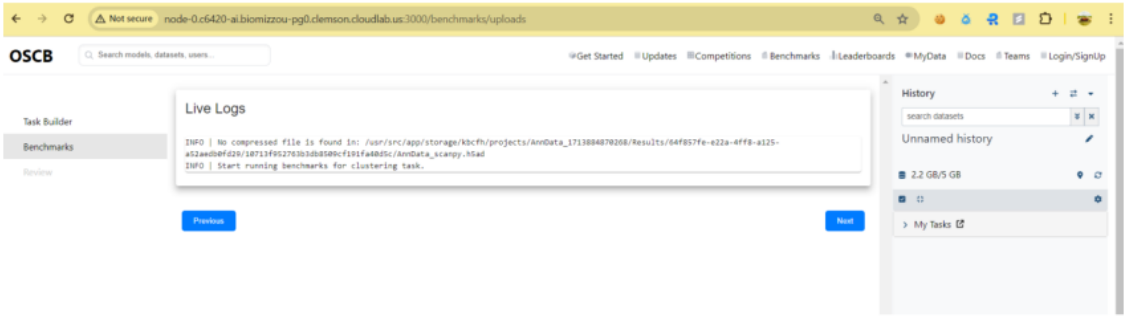


Fig 52 live Logs for Benchmarks API

Submission and Archiving: Upon satisfactory review, the tasks are submitted for final processing. The results are then stored in the 'tasks' collection within MongoDB, ensuring they are accessible for future reference and analysis.

Backward Compatibility and User Experience: Notably, the Task Builder flow is designed with backward compatibility, allowing users to revisit and modify earlier steps without losing data or progress. This feature underscores OSCB's commitment to a user-centric approach, facilitating ease of use and flexibility.

6. CONCLUSION

The development of the Open Single Cell Benchmarks (OSCB) platform marks a significant advancement in biological research, blending state-of-the-art technology with the intricate requirements of single-cell analysis. This report has outlined a set of functionalities aimed at bolstering the platform's resilience and ensuring it serves as a comprehensive, user-centric ecosystem that fosters scientific discovery.

Central to OSCB's design philosophy is a focus on user experience and security, starting with a robust authentication system that protects user data while providing seamless access to the platform's resources. By leveraging technologies like React, Node.js, and Python, and utilizing MongoDB for data management, we demonstrate our commitment to employing advanced technological solutions to enhance usability and efficiency.

Moreover, the incorporation of Directus as a content management system (CMS) empowers both technical and non-technical users to dynamically manage content in real-time, reducing reliance on developers and enabling immediate updates to keep the platform's content current and relevant.

The data management capabilities of OSCB are particularly noteworthy. Through sophisticated data upload features, users not only contribute to but also actively participate in managing and refining their data. This not only improves the quality of data stored within OSCB but also enhances overall research outcomes.

Additionally, OSCB's innovative benchmark and task builder workflows showcase its potential as a collaborative hub for the scientific community. These functionalities allow users to create, manage, and utilize benchmark datasets, facilitating a shared environment for scientific exploration that is scalable and impactful.

7. FUTURE WORK

- **Expanding Benchmark Support**: Currently, our platform supports clustering benchmark tasks. However, we aim to broaden our support to include other benchmark types, such as imputation, marker gene identification, and more. This expansion will provide users with a wider range of analytical options.
- **User Task State Management**: We plan to implement functionality to maintain user states within a particular task. This means that if a user leaves a process midway and returns later, they should be able to continue from where they left off. This feature will enhance user convenience and streamline their workflow.
- **Enhanced User Experience:** We are committed to improving the overall user experience by refining the website's styles and adding more user-friendly functionalities. These enhancements will make the platform more intuitive and enjoyable for users to navigate and utilize.
- **Additional Website Components**: In addition to the existing functionalities, we intend to introduce new components such as leaderboards and a documentation page. The leaderboards will showcase user performance and achievements, fostering a sense of competition and community. The documentation page will provide users with comprehensive resources and guidance on using the platform effectively.
- **Streamlined Results Presentation:** We aim to optimize the presentation of analytical results to users. This involves finding efficient ways to convey complex data and insights in a clear and digestible format. By streamlining the results delivery process, we ensure that users can easily interpret and utilize the findings from their analyses.

8 REFERENCES

[1] Lei Jiang, Yuexu Jiang, Cankun Wang, Clement Essien, Juexin Wang, Anjun Ma, Qin Ma, Dong Xu. "Machine learning development environment for single-cell sequencing data analyses", 2022 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), 2022

[2] W. Huber, V. J. Carey, R. Gentleman, S. Anders, M. Carlson, B. S. Carvalho, H. C. Bravo, S. Davis, L. Gatto, T. Girke et al., "Orchestrating high-throughput genomic analysis with bioconductor," Nature methods, vol. 12, no. 2, pp. 115–121, 2015.

[3] T. Stuart, A. Butler, P. Hoffman, C. Hafemeister, E. Papalexi, W. M. Mauck III, Y. Hao, M. Stoeckius, P. Smibert, and R. Satija, "Comprehensive integration of single-cell data," Cell, vol. 177, no. 7, pp. 1888–1902, 2019.

[4] F. A. Wolf, P. Angerer, and F. J. Theis, "Scanpy: large-scale single-cell gene expression data analysis," Genome biology, vol. 19, no. 1, pp. 1–5, 2018.

[5] D. Van Dijk, R. Sharma, J. Nainys, K. Yim, P. Kathail, A. J. Carr, C. Burdziak, K. R. Moon, C. L. Chaffer, D. Pattabiraman et al., "Recovering gene interactions from single-cell data using data diffusion," Cell, vol. 174, no. 3, pp. 716–729, 2018

[6] React Documentation. Available: https://react.dev/reference/react

[7] Node JS Documentation. https://nodejs.org/docs/latest/api/

[8] MongoDB Documentation. https://www.mongodb.com/docs/manual/

[9] Directus Documentation. Available: https://docs.directus.io/

[10] MySQL Documentation. Available: MySQL

[11] JSON Web Token (JWT). https://jwt.io/introduction

[12] Celery Documentation. Available: https://docs.celeryq.dev/en/stable/

[13] WebSocket API https://websockets.readthedocs.io/en/stable/

[14] Python https://docs.python.org/3.12/tutorial/index.html

[15] FastAPI https://fastapi.tiangolo.com/

[16] OpenAI. (2022). ChatGPT. https://openai.com/chatgpt

Final Year Project Report

ORIGINALITY REPORT

7%
SIMILARITY INDEX

PRIMARY SOURCES

1	global.oup.com Internet	239 words — 2%
2	Lei Jiang, Yuexu Jiang, Cankun Wang, Clement Essien, Juexin Wang, Anjun Ma, Qin Ma, Dong Xu. "Machine learning development environment for single-cell sequencing data analyses", 2022 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), 2022 Crossref	172 words — 1%
3	www.proceedings.com Internet	47 words — < 1%
4	www.biorxiv.org Internet	39 words — < 1%
5	Yadagiri Shiva Sai Sashank, Ankit Agrawal, Ritika Bhatia, Ashutosh Bhatia, Kamlesh Tiwari. "Chapter 19 D-insta: A Decentralized Image Sharing Platform", Springer Science and Business Media LLC, 2023 Crossref	28 words — < 1%
6	dslsrv1.rnet.missouri.edu Internet	28 words — < 1%
7	ieeebibm.org Internet	26 words — < 1%

8	elitex.systems Internet	21 words — < 1%
9	digitalcollection.utem.edu.my Internet	20 words — < 1%
10	www.theknowledgeacademy.com Internet	17 words — < 1%
11	Dylan Molho, Jiayuan Ding, Wenzhuo Tang, Zhaoheng Li et al. "Deep Learning in Single-Cell Analysis", ACM Transactions on Intelligent Systems and Technology, 2024 Crossref	14 words — < 1%
12	vdocument.in Internet	13 words — < 1%
13	forum.mapcreator.here.com Internet	12 words — < 1%
14	core.ac.uk Internet	11 words — < 1%
15	digitalarchive.boun.edu.tr Internet	11 words — < 1%
16	docs.ris.wustl.edu Internet	11 words — < 1%
17	scholarscompass.vcu.edu Internet	11 words — < 1%
18	oroshaza.focus-aha.eu Internet	9 words — < 1%
	repository.up.ac.za	

19	Internet	9 words — < 1%
20	www.frontiersin.org Internet	9 words — < 1%
21	www.hindawi.com Internet	9 words — < 1%
22	Ashwinikumar Kulkarni, Ashley G. Anderson, Devin P. Merullo, Genevieve Konopka. "Beyond bulk: a review of single cell transcriptomics methodologies and applications", Current Opinion in Biotechnology, 2019 Crossref	8 words — < 1%
23	Cao, Yingxin. "Deep Representation Learning for Single-Cell Sequencing Data Analysis", University of California, Irvine, 2023 ProQuest	8 words — < 1%
24	David G. Aragones, Miguel Palomino-Segura, Jon Sicilia, Georgiana Crainiciuc et al. "Variable selection for nonlinear dimensionality reduction of biological datasets through bootstrapping of correlation networks", Computers in Biology and Medicine, 2024 Crossref	8 words — < 1%
25	Iacopo Colonnelli, Barbara Cantalupo, Ivan Merelli, Marco Aldinucci. "StreamFlow: Cross-Breeding Cloud With HPC", IEEE Transactions on Emerging Topics in Computing, 2021 Crossref	8 words — < 1%
26	dokumen.pub Internet	8 words — < 1%
27	ftp5.gwdg.de Internet	

		8 words — < 1%
28	lew.ro Internet	8 words — < 1%
29	proceedings.mlr.press Internet	8 words — < 1%
30	www.getapp.ae Internet	8 words — < 1%
31	www.ir.juit.ac.in:8080 Internet	8 words — < 1%
32	www.vskills.in Internet	8 words — < 1%
33	Li, Wei. "Statistical Methods for Bulk and Single-cell RNA Sequencing Data.", University of California, Los Angeles, 2019 ProQuest	7 words — < 1%
34	Shah, Shrut. "Enhancement to Text Alignment Visualization", California State University, Sacramento, 2024 ProQuest	7 words — < 1%
35	Babeș-Bolyai University Publications	6 words — < 1%

EXCLUDE QUOTES	OFF	EXCLUDE SOURCES	OFF
EXCLUDE BIBLIOGRAPHY	OFF	EXCLUDE MATCHES	OFF