

# JasmineGraph UI

JasmineGraph Version : v0.1.1

Date : 27/12/2024

## 1. Introduction

JasmineGraph is an open-source distributed graph database server designed for efficient storage, processing, and querying of graph data across distributed environments. It supports advanced graph analytics and is tailored for handling large-scale graph datasets.

The JasmineGraph UI serves as a web-based interface for interacting with the JasmineGraph server, simplifying complex operations by providing an intuitive and user-friendly experience. With the UI, users can:

- Upload and manage graph datasets.
- Run queries and visualize results.
- Monitor system performance and usage metrics.
- Configure settings and manage access permissions.

This documentation aims to guide users through the setup and usage of the JasmineGraph UI, detailing its features, functionality, and user workflows. Whether you are a first-time user or an experienced developer, this document will help you maximize the potential of the JasmineGraph UI.

## 2. Getting Started

### Prerequisites

- **System Requirements:**
  - Any modern machine with Docker installed.
  - Minimum: 2 CPU cores, 4GB RAM, and 10GB disk space.
  - Recommended: 4 CPU cores, 8GB RAM, and 20GB disk space.
- **Browser Compatibility:**
  - JasmineGraph UI runs on any modern browser, such as Google Chrome, Mozilla Firefox, Microsoft Edge, or Safari.
- **Dependencies:**
  - Docker and Docker Compose must be installed on your system.
- **Backend Requirement:**
  - A running instance of the JasmineGraph server is required to use the UI. Connection details will be configured during the setup process.

## Installation and Setup

### 1. Clone the Repository:

Clone the JasmineGraph UI repository to your local machine:

```
git clone <repository-url>
```

```
cd <repository-folder>
```

JasmineGraphUI repository is located at  
<https://github.com/jasminegraph/jasminegraph-ui>

```
git clone https://github.com/jasminegraph/jasminegraph-ui.git
```

### 2. Run the Docker Containers:

Use the `docker-compose.yml` file to start the services:

```
docker-compose up -d
```

- This will spin up the following services:
  - **Frontend:** Accessible at <http://localhost:3000>.
  - **Backend:** Runs on port 8080 and connects to MongoDB.
  - **MongoDB:** Stores application data persistently using the `./mongodb-data` folder.

Note that in some cases we need to first delete two existing docker images *jasminegraph-frontend* and *jasminegraph-backend* before running *docker-compose up* command.

### 3. Verify the Setup:

- **Check Backend:**

Send a GET request to the `/ping` endpoint:

```
curl http://localhost:8080/ping
```

If the backend is working, it will respond with:

```
{ "message": "pong" }
```

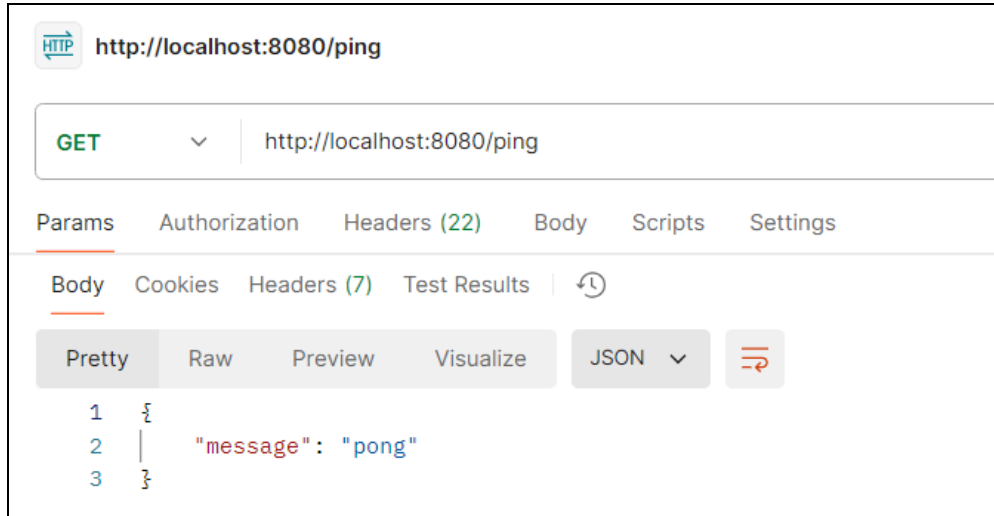


Figure 1: Checking the liveness of the JasmineGraphUI using Postman

- **Check Frontend:**  
Open <http://localhost:3000> in your browser. You should see the login page of the JasmineGraph UI.

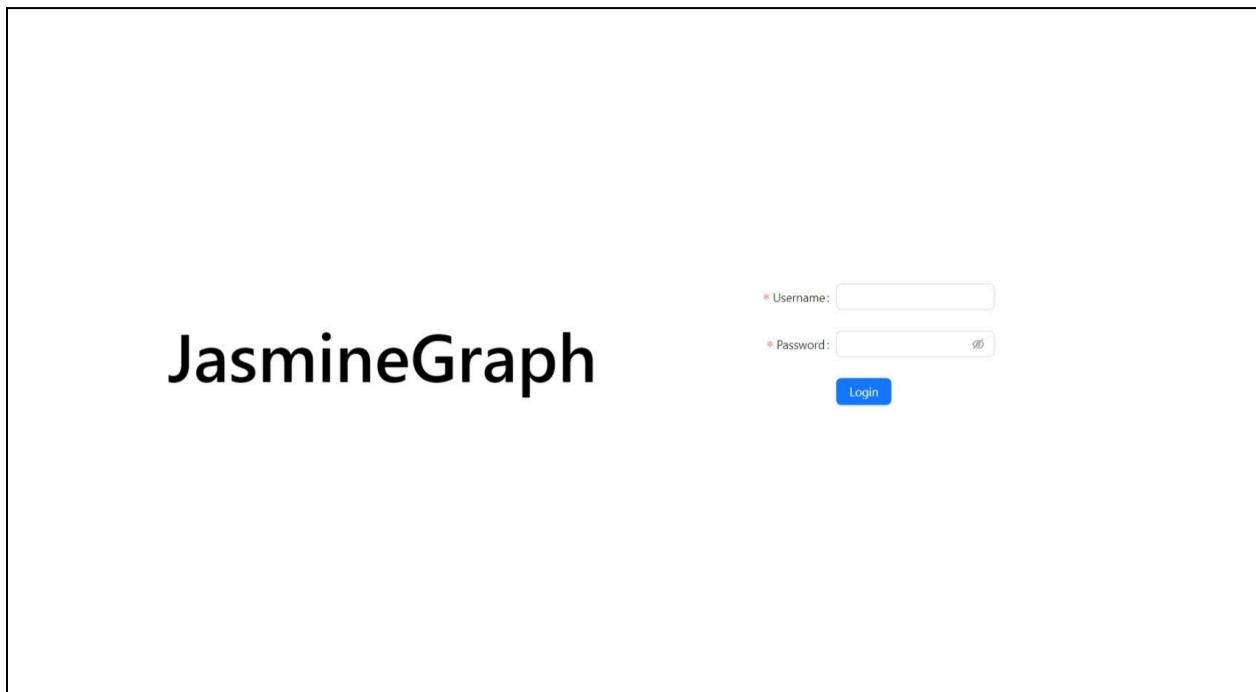


Figure 2: Logging page of the JasmineGraph UI

## Initial Configuration

After ensuring the JasmineGraph UI is running, follow these steps to configure the system for the first time:

### 1. Access the Login Page:

- Open your browser and navigate to <http://localhost:3000>.
- The application will check if an admin user is already registered in the backend. If there is no registered admin user it will indicate a message "Admin User Not Found"

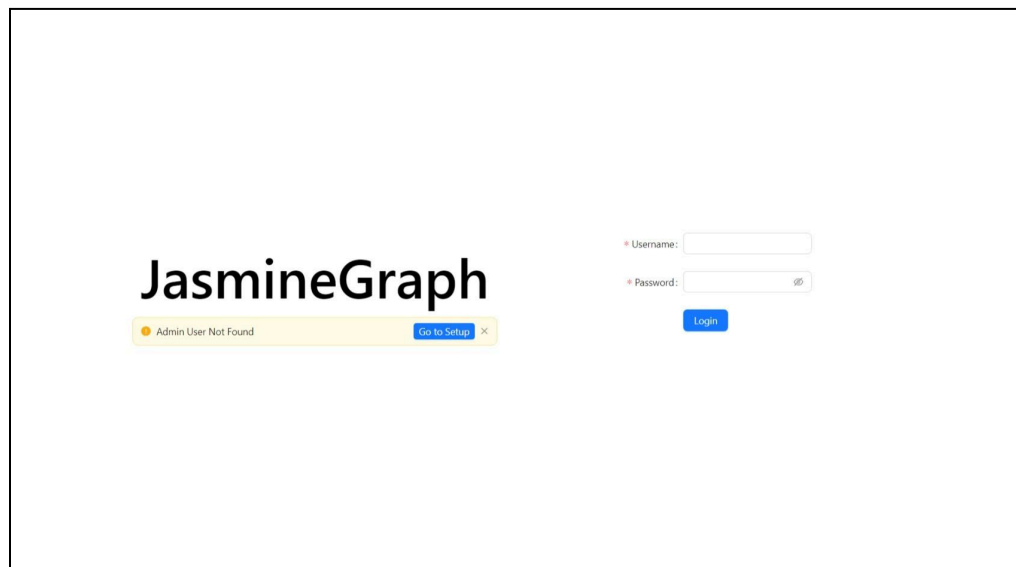
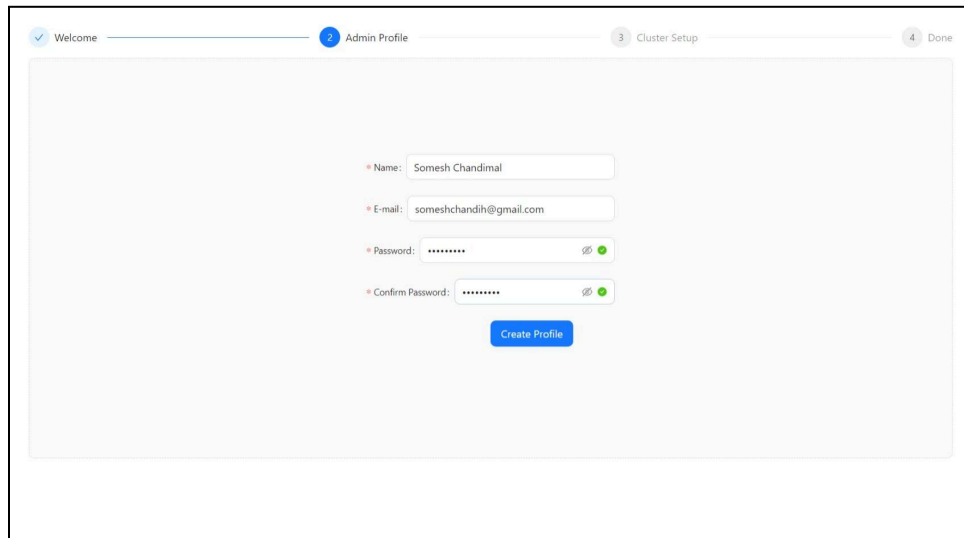


Figure 3: Accessing the JasmineGraph UI's login page for the first time

### 2. Create an Admin User (Fresh Start):

- If no admin user is found, the UI will prompt you to create one.
- Fill in the form with the following details:
  - **Username:** Desired admin username.
  - **Password:** Strong password for admin access.

- Click the **Submit** button to register the admin user.

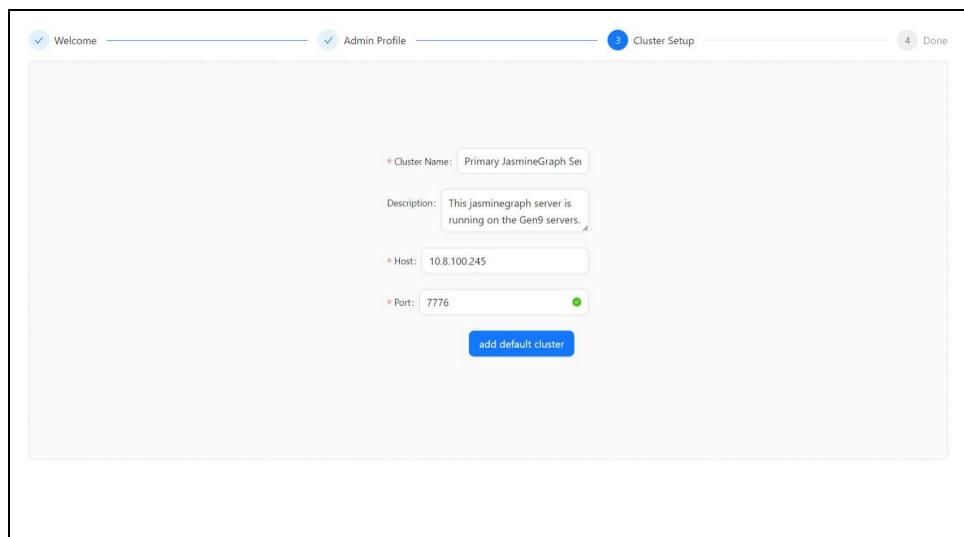


The screenshot shows a web interface with a progress bar at the top indicating four steps: 1. Welcome, 2. Admin Profile (current), 3. Cluster Setup, and 4. Done. The Admin Profile form contains the following fields: Name (Somesh Chandimal), E-mail (someschandih@gmail.com), Password (masked with dots), and Confirm Password (masked with dots). Each field has a red asterisk icon to its left. To the right of the Password and Confirm Password fields are green checkmark icons. A blue 'Create Profile' button is located at the bottom of the form.

Figure 4: Registering a new user

### 3. Configure the JasmineGraph Server:

- After registering the admin user, the system will navigate to the server configuration page.

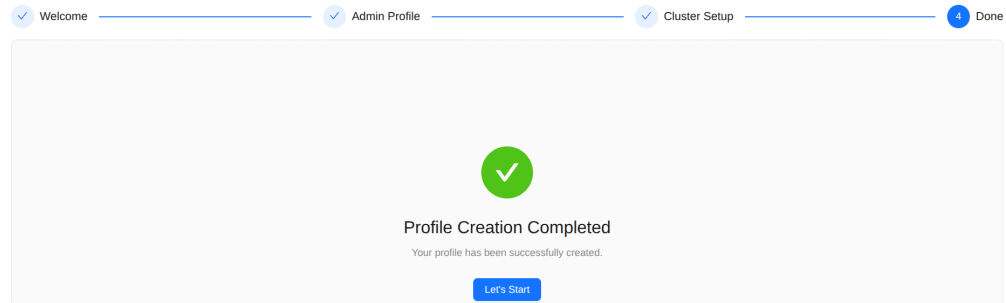


The screenshot shows the Cluster Setup page, which is the third step in the progress bar (Welcome, Admin Profile, Cluster Setup, Done). The form contains the following fields: Cluster Name (Primary JasmineGraph Set), Description (This jasminegraph server is running on the Gen9 servers), Host (10.8.100.245), and Port (7776). Each field has a red asterisk icon to its left. To the right of the Port field is a green checkmark icon. A blue 'add default cluster' button is located at the bottom of the form.

Figure 5: Setting up JasmineGraph cluster. For Docker mode Host can simply be localhost.

- Provide the following information about the JasmineGraph server:
  - **Cluster Name:** A name for the server (e.g., "Primary Server").
  - **Description:** A brief description of the server's purpose.

- **Host Address:** The IP or hostname of the JasmineGraph server. For example, with docker mode of JasmineGraph this host address should be localhost
- **Port:** The port number where the server (i.e., JasmineGraph UI frontend service) is accessible (i.e., 7776).
- Click the **add default cluster** button to save the server details.



Click on Let's Start

#### 4. Access the Home Page:

- Once the setup is complete, the UI will redirect you to the **Home Page**, where you can begin interacting with the system.
- From the Home page, click on the select button of the cluster to mark it as the "Selected Cluster"

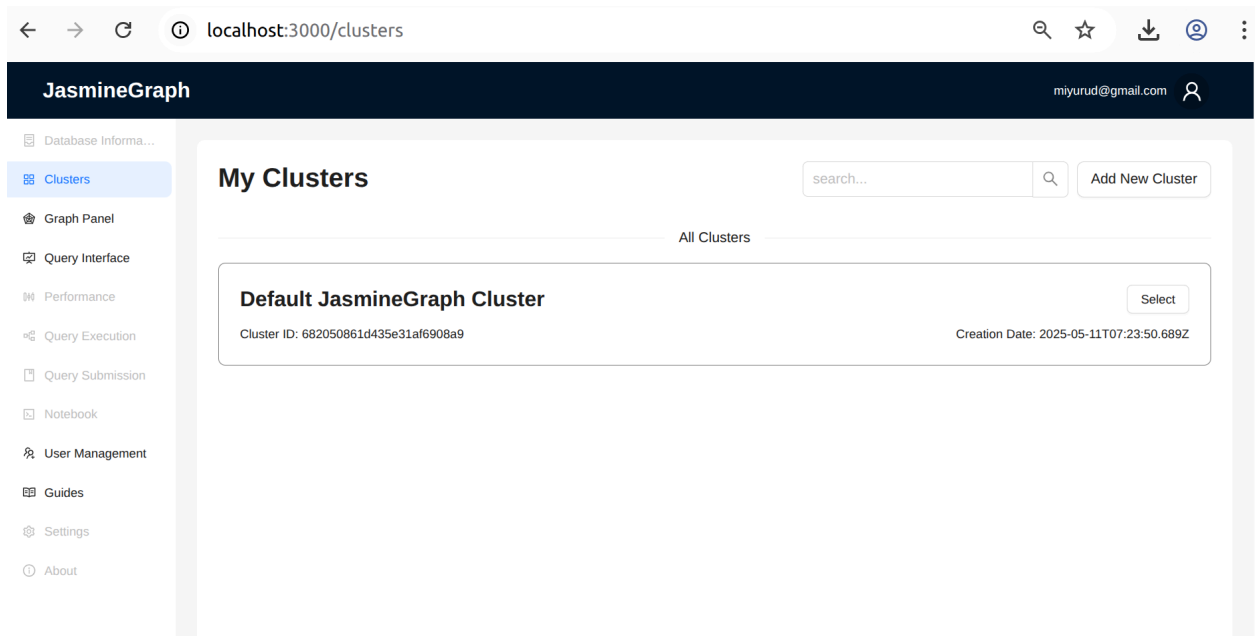


Figure 6: Home Page immediately after clicking the "Let's Start" button. Need to click on the "Select button" on the cluster to make it "Selected Cluster".

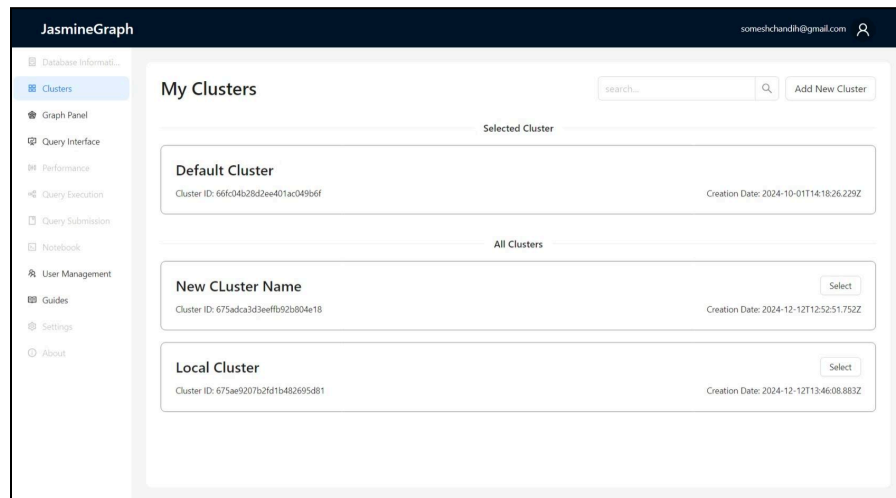


Figure 6: Home Page of JasmineGraph UI with multiple clusters. Note that there is only one selected cluster at a time.

## 3. Features and Pages

### Clusters Page:

The **Clusters** page allows users to manage and monitor the JasmineGraph clusters. The Clusters page comprises a list of JasmineGraph clusters. By default the list has one element, which corresponds to the cluster configured during the initial setup. Once we click on one of the clusters, it will go into a more descriptive page. The page includes five tabs for cluster management. These tabs are as follows,

#### 1. Cluster Overview:

- Displays details about the selected cluster, such as:
  - **Cluster Name:** Editable text field for naming or describing the cluster.
  - **Cluster ID:** Unique identifier for the cluster.

- **JasmineGraph Version:** The version of JasmineGraph running in the cluster.
- **Platform:** (Currently not specified in the screenshot).

### Tabs for Cluster Management:

- **Cluster Details:** Provides a detailed view of the cluster, including its nodes.
- **Access Management:** Configures user or role-based access to the cluster.
- **Instance:** Manages instances related to the cluster.
- **Deployment:** Handles cluster deployment settings.
- **Logs:** Access and view logs for troubleshooting and monitoring.

## 2. Cluster Details:

Cluster details tab provides a cluster overview at the top. Then it provides master nodes and worker nodes details (i.e., node information).

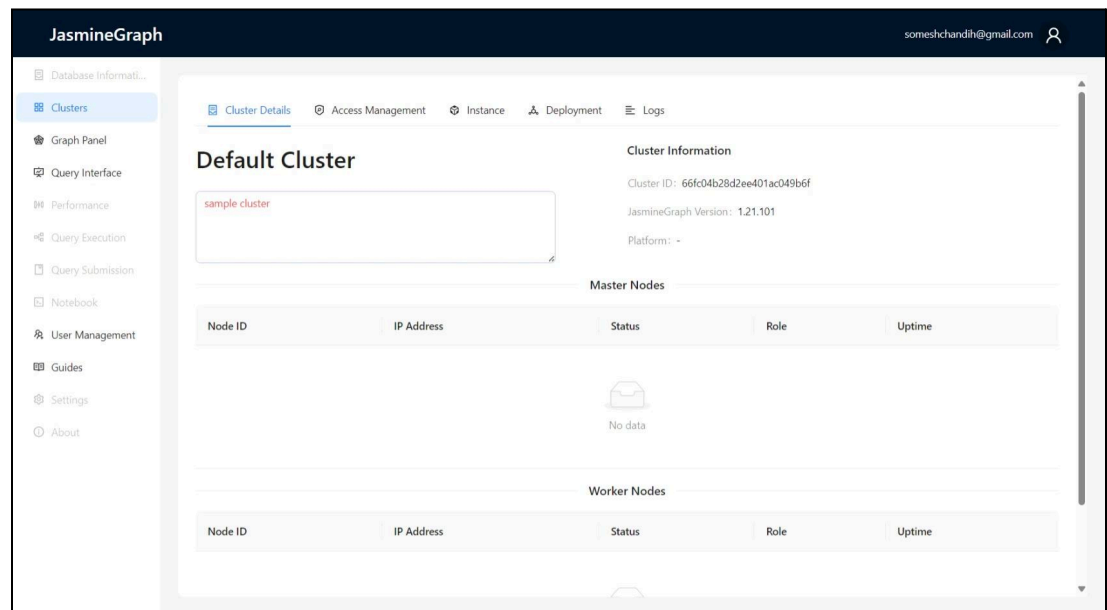


Figure 7: Cluster Details Tab

## 3. Node Information:

- Lists **Master Nodes** and **Worker Nodes** associated with the cluster.
- Columns include:
  - **Node ID:** Identifier for the node.
  - **IP Address:** IP Address of the node.
  - **Status:** Current status of the node (e.g., active or inactive).
  - **Role:** The role of the node (e.g., master or worker).
  - **Uptime:** How long the node has been running.



## Graph Panel Page:

The **Graph Panel Page** allows users to upload, visualize, and manage graph data. With the integration of **Kafka** and **Hadoop**, the page offers enhanced capabilities for uploading large-scale graph datasets and analyzing them efficiently.

The **Graph Panel** consists of two main tabs:

### 1. Upload Tab

The **Upload Tab** allows users to upload graph data in various ways, including through **Kafka** and **Hadoop** for more scalable, distributed data handling. The features included in this tab are:

- **Upload Graph:**
  - **File Selection:** Users can select graph files from their local system or specify a file path. Supported file formats may include `.csv`, `.json`, `.xml`, and other graph-related formats.
  - **Upload via Kafka:**
    - Users can send graph data through **Kafka**, which streams data in real-time to the graph system.
  - **Upload via Hadoop:**
    - Users can upload graph data stored in Hadoop's **HDFS (Hadoop Distributed File System)**.

### 2. Graph Tab

The **Graph Tab** allows users to view and analyze the graph data once it's uploaded. Features include:

- **Graph Visualization:**
  - **Graph View:** Displays the graph in a visual format with nodes and edges.
  - **Zoom and Pan:** Users can zoom in/out or pan across the graph to explore different sections.
  - **Node Highlighting:** Clicking on nodes or edges highlights specific parts of the graph for better analysis.
- **Graph Details:**
  - **Node Information:** Detailed information about individual nodes, such as node ID, properties, and relationships.
  - **Edge Information:** Details about edges, including edge ID, source and target nodes, and edge properties.
  - **Graph Structure:** Overview of the graph structure, including total nodes, edges, and other relevant statistics.

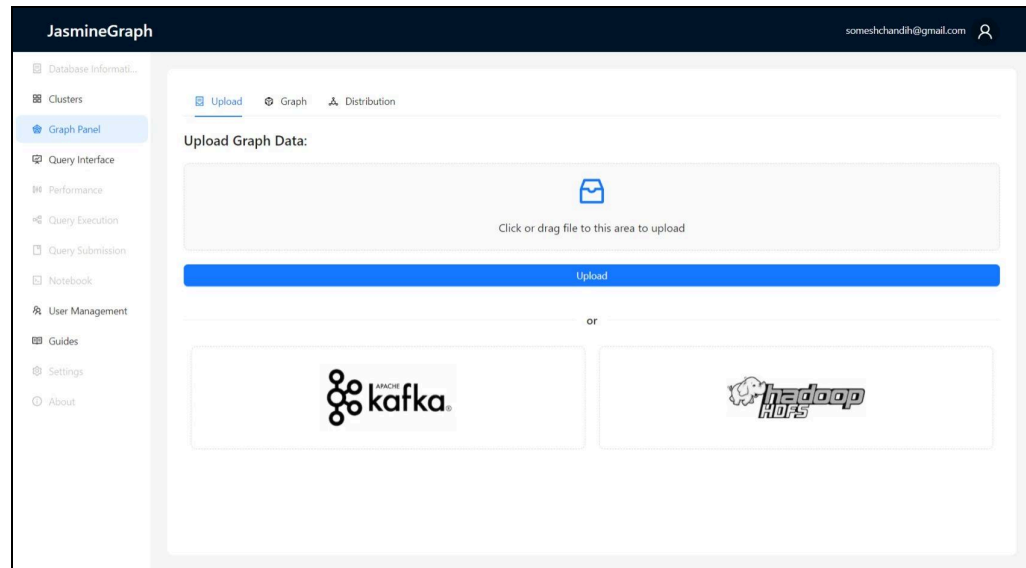


Figure 8: Graph upload panel

## Query Interface Page:

The **Query Interface Page** allows users to write and execute queries against the connected JasmineGraph server, providing real-time results. The page includes two main tabs: **Enter and See Query Result** and **Graph Properties**, enabling users to query graph data and run analytics.

The **Query interface** consists of two main tabs:

### 1. Query Tab

This tab is where users can write and execute queries to interact with the graph data. The features include:

- **Write Queries:**
  - **Query Input Box:** A text area where users can write graph queries, typically using a query language **Cypher**

### 2. Graph Properties Tab

The **Graph Properties Tab** provides tools for running advanced analytics on the graph data. These properties help users understand the structure and characteristics of the graph through various algorithms, such as **Triangle Count** and **Page Rank**.

- **Triangle Count:**
  - **Triangle Detection:** This algorithm detects triangles within the graph, which are subgraphs consisting of three nodes that are mutually connected.

## User Management Page

The **User Management Page** displayed in the screenshot is designed for managing users and their roles within the JasmineGraph application. It provides role-based access control to ensure only authorized users can perform specific actions

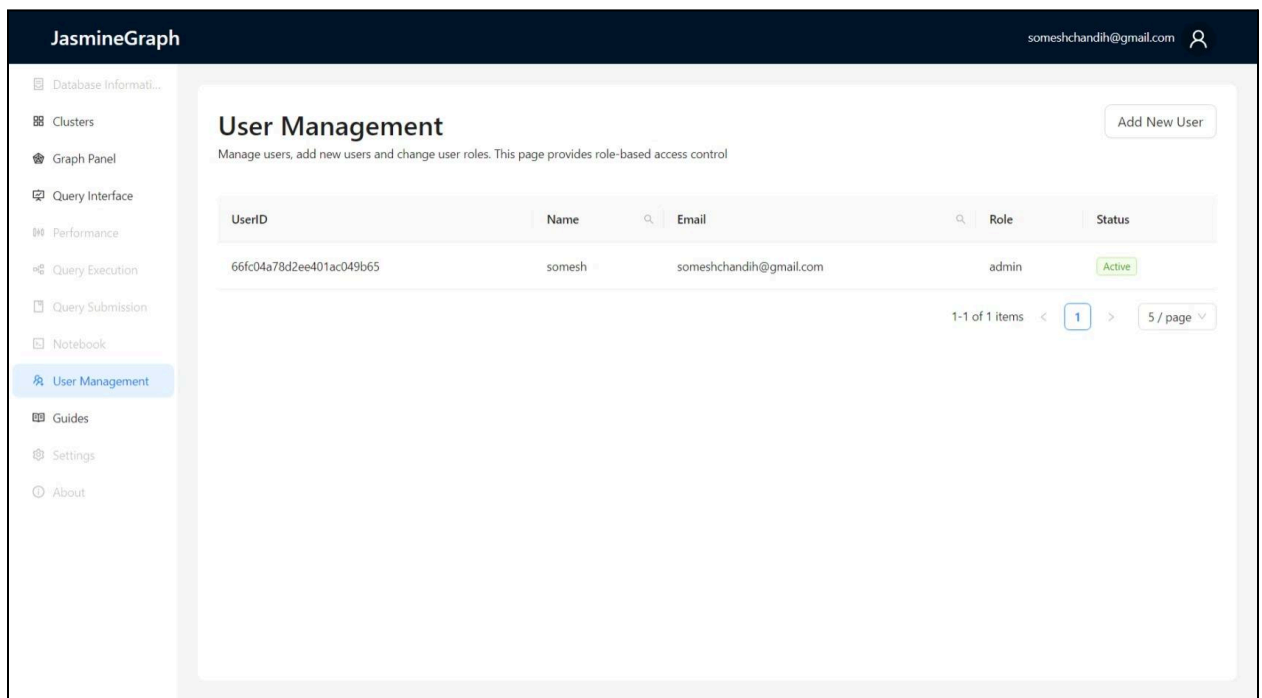


Figure 9: User management page

### Key Features of the User Management Page:

#### 1. User Table Overview

The page contains a table that lists all users in the system, along with their details:

- **User ID:** A unique identifier for each user in the system.
- **Name:** The display name of the user.
- **Email:** The email address associated with the user account.
- **Role:** The role assigned to the user, which determines their permissions within the application.
  - **Admin:** This role grants full access, including the ability to add or manage users, graphs, and settings.
  - **Viewer:** This role provides limited access, allowing users to view data and results but restricts administrative actions.
- **Status:** Indicates whether the user account is currently active.
  - **Active:** The user can log in and access the system.
  - **Inactive:** The user is temporarily restricted from accessing the system.

## 2. Search and Pagination

- **Search:** Allows administrators to quickly locate users by filtering based on the name or email.
- **Pagination:** Provides the ability to navigate through a large list of users by specifying how many users are displayed per page (e.g., 5 items per page).

## 3. Add New User Button

- **Add New User:** This button opens a form where an administrator can:
  - Add a new user by entering their details (name, email, and role).
  - Assign the appropriate role to the user during the creation process.

## 4. Role-Based Access Control

- **Admin:**
  - Can manage all aspects of the system, including adding new users, updating roles, and managing graph configurations.
- **Viewer:**
  - Limited to viewing analytics, running queries, and exploring graph data but cannot make system-wide changes.

## Guides Page:

The **Guides** section in the JasmineGraph system is designed to help users understand the platform's interface and functionality. It provides comprehensive, step-by-step instructions for using various features, ensuring that both new and experienced users can navigate the system effectively.

# Appendix

## Appendix A

### **Build the Docker Images:**

Use the provided `build.sh` script to build the frontend and backend Docker images:

```
chmod +x build-docker.sh
```

```
./build-docker.sh
```

This will create two Docker images:

- `jasminegraph-frontend` for the Next.js frontend.
- `jasminegraph-backend` for the Express backend.