Task

Stack Requirements:

- Node.js v22/ ExpressJS
- Angular v19 /Angular Material
- MongoDB
- AG Grid

Initial/Setup:

- Create a backend Node.js/ExpressJS Server
- Server should be running on Port 3000 Create a folder:
  - controllers/
  - routes/
  - helpers/
  - models/
- app.js file should contain all the initializations
- Create an Angular App with Angular Material
- The app should be configured to run on port 4200

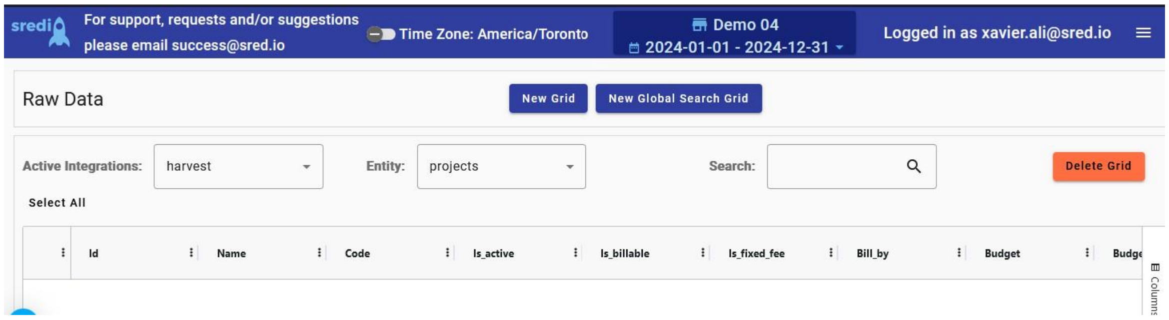
Objective:

a) Display the data:

- At this point, GitHub integration should be working and we should be able to fetch GitHub Data: repos, commits, pull requests, issues, issue history.

- Within your testing organization, make sure you have atleast 3 repos and each repo should have atleast: 2000 commits, 2000 pulls, 500 issues. You can import public open-source libraries into your test repo as well.
- We will store the above data into their separate collection in the database.
- We need the ability to view all the github data. Create a dynamic AG Grid that can display all the fields into an AG Grid. If the object is nested, for example, if we have author: {name: “ABC”, id: “1”}, we can display them in the column as author_name, author_id. Each field should be a separate column.
- We should be able to display all the collections from the GitHub mongodb database in the Entities dropdown.
- When the collection is selected, we need to fetch the data.
- As we are dealing with thousands of records, we need to ensure pagination is implemented.

Active Integrations Dropdown: Github
Entity Dropdown: List of Collections in the GitHub database.
Search: Ability to search keyword in the AG Grid.
AG Grid Table will display all the fields from the collection. We need to get all the fields dynamically from the selected collection and display them in the AG Grid.



Active Integrations: harvest
Entity: projects

b) Search

- When searching for a ticket, it is difficult to find which user completed the ticket
- Add a hyperlink at the beginning of the search results that opens a new tab with an AG-Grid
    - The hyperlink is called: Find User
    - Ensure that there is a space between Find User and other text within the search result
- Within the AG-Grid, we will need to display:
    - Ticket ID
    - User
        - Which user completed the Ticket
        - Name displayed in the Timesheets page
    - Date
        - Same date retrieved in the Timesheet search
    - Summary, Description, Message
        - Name of ticket (Ticket Summary)
- All the columns should support filters and search.
- We also need to implement a global search, which will search results across all the collections.
- **Custom Filters**: Allow users to define custom filters, such as selecting a specific date range for commits or filtering pull requests by status, allowing for a more personalized data view.
- **Faceted Search**: Implement a faceted search system where users can filter results by multiple facets (e.g., "user," "repo," "status," "date range") and dynamically update the AG Grid based on selected filters.

c) Building a Relationship Builder with MongoDB Collections

We need to create and visualize relationships between data from multiple MongoDB collections: commits, pull requests, issues, issue history (changelogs) and repos. These collections should have a common identifier (e.g., repo_id, commit_hash, etc.), allowing the system to combine and display related data together.

- **Backend (Node.js / ExpressJS):**
  - Implement an aggregation pipeline using MongoDB to join data from the commits, pull requests, and issues collections using a common identifier (e.g., repo_id).
  - Create an API that fetches a repository's data along with its related commits, pull requests, and issues.
  - Ensure efficient pagination and filtering capabilities for large datasets.
- **Frontend (Angular / AG Grid):**
  - Dynamic Data Display: Create an Angular application that fetches and displays data from the backend.
  - Use AG Grid to display the related data in a table format. Each row should display fields from the commits, pull requests, and issues collections (e.g., commit hash, PR status, issue title).
  - Expandable Rows: Implement expandable rows in AG Grid to show detailed data when a user clicks on a row.
  - Dynamic Columns: Automatically generate AG Grid columns based on the fields in the fetched data (e.g., commit message, PR status, issue description).
  - Search and Filters: Implement a search bar that allows users to search across all fields, and allow filtering of results by specific columns (e.g., commit author, PR status).
- **Additional Features:**
  - Pagination: Handle large datasets by implementing pagination within AG Grid and the backend API.
  - Relationship Exploration: Allow users to click on a commit, PR, or issue to view more details and explore relationships with other data.

d) Frontend Enhancements

- **Advanced AG Grid Features**: Add features like **drag-and-drop columns**, **custom cell rendering** (e.g., showing avatars for GitHub users or repositories), and **row grouping** for hierarchical data.
- **Responsive Design**: Ensure the frontend is mobile-friendly. Implement a **responsive layout** that adapts to various screen sizes and ensures that the AG Grid remains usable across devices.
- **Error Handling and User Feedback**: Add advanced error handling on both the frontend and backend, such as displaying proper loading indicators, error messages when the GitHub API is unavailable, or when large datasets fail to load.

Some additional requirements:
- Utilize the maximum real estate
- All columns should support filters
- Each field in the collection should be a separate field and a separate column.
- Search should apply search in all the columns.
- Implement Pagination.

Testing Tips:

- Create a GitHub account for testing purposes.
- Import Open source/Public Repos.
- Connect the testing account to make sure the authentication is working properly.
- AG Grid filters are working properly. All the fields/columns are displayed correctly.
- Ensure that there is a hyperlink for every result within the Search Tickets bar in the Timesheets page
- Ensure that when the Find User hyperlink is clicked that a new tab opens with an AG Grid
- Ensure that the AG-Grid displays what user worked on the ticket

(Version: 2.0)