Social Network Analysis Platform Using Graph Database

1. Group Details

Group name: **Three Musketeers**

- **Sreejita Saha** (21CS30052)
- **Anwesha Das** (21CS30072)
- **Ratan Junior** (21CS30041)
  and the friends we made along the way

Github repository: https://github.com/ratanjr/Social-Network-Analysis-Platform

2. Objective

The Social Network Analysis Platform is a web application designed to efficiently analyse and visualise social network data sourced from the Stanford Large Network Dataset Collection. In particular, we make use of the dataset collected from the Facebook application. The platform allows users to identify patterns and run queries (described in detail later) on the database, offering valuable insights serving commercial purposes.

2.1 Social circles: Facebook Dataset

This dataset comprises 'circles' or 'friends lists' extracted from Facebook from survey participants. The dataset has node attributes (profiles), circles, and ego networks. To safeguard privacy, the Facebook data has undergone anonymization, whereby the internal Facebook identifiers for each user have been replaced with new values. Additionally, the interpretation of feature vectors within this dataset has been obfuscated

- The dataset contains **4039 nodes** and **88234 edges**
- **Average clustering coefficient: 0.6055**

3. Methodology

3.1 Backend

We have used python with the following libraries:

- **flask**: for developing a web-based interface, handling HTTP requests, rendering dynamic content, and serving static files, making it an essential element.

- **pyvis**: for interactive network visualizations, imported as

```
from pyvis.network import Network
```

  and used in functions `get_network_data()` and `filter_graph()`
- **wget** is used to download the datasets from the internet as used in `download_dataset(dataset_url, save_directory)` in `databsase.py` while **neo4j** is used for implementing the graph database more on which follows this section.

3.2 Neo4j

Following are the key uses of Neo4j database used in `graph.py` and `database.py`:

1. **Establishing Connection:** The `GraphDatabase.driver` function is used to establish a connection to the Neo4j database. This function requires the URI of the database and authentication credentials (username and password).
2. **Executing Cypher Queries:** Cypher queries are executed on the Neo4j database to retrieve data. The `session.run()` method is used to execute Cypher queries within a session context. The result of the query is obtained, and relevant data is extracted from the result for further processing.
3. **Querying Nodes and Relationships:** Various endpoints in the Flask application are designed to execute Cypher queries to retrieve nodes, relationships, and their properties from the Neo4j database. For example, in the `search()` function, a Cypher query is executed to find nodes and their relationships based on a search term provided by the user.
4. **Data Retrieval for Visualization:** In functions like `get_network_data()`, data is retrieved from the Neo4j database to construct a network visualization. Cypher queries are used to fetch nodes and relationships, and the retrieved data is processed and formatted before being used to create the visualization.
5. **Handling Transactions:** The `with driver.session()` context manager is used to handle transactions with the Neo4j database. This ensures that resources are properly managed, and transactions are committed or rolled back as needed.

3.3 Frontend

The frontend of the project has been implemented using a combination of **HTML, CSS, JavaScript**, and the **PyVis** library for network visualization.

- HTML templates are used to structure the different pages of the web application, providing layout and placeholders for dynamic content.
- CSS styling enhances the visual appearance and layout of the web pages, while JavaScript adds interactivity and dynamic behavior, handling user input, form validation, and asynchronous requests to the backend using AJAX.

- The PyVis library is utilized for network visualization, creating dynamic and interactive visualizations of the social network data.
- Flask's `render_template` function is used to dynamically render HTML templates with data passed from the backend, allowing content retrieved from the Neo4j database or processed in the backend to be rendered dynamically in the frontend.
- Forms implemented using HTML `<form>` elements enable users to input data or perform actions, with JavaScript and Flask routes handling form submission and processing user input.

Overall, the frontend provides an intuitive and interactive user interface for interacting with social network data and visualizations, facilitating data exploration and analysis.

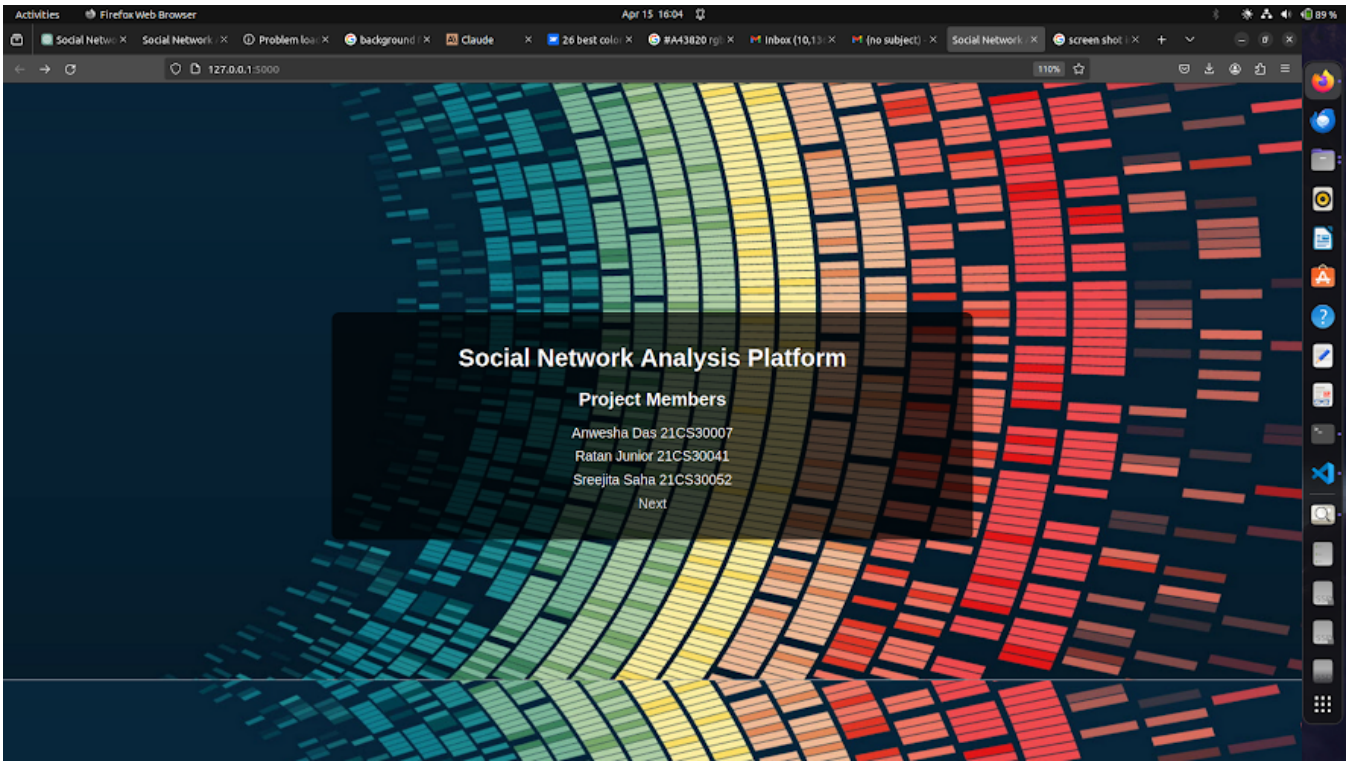
3.4 Project Structure

The project including the Github repository is structured as follows:

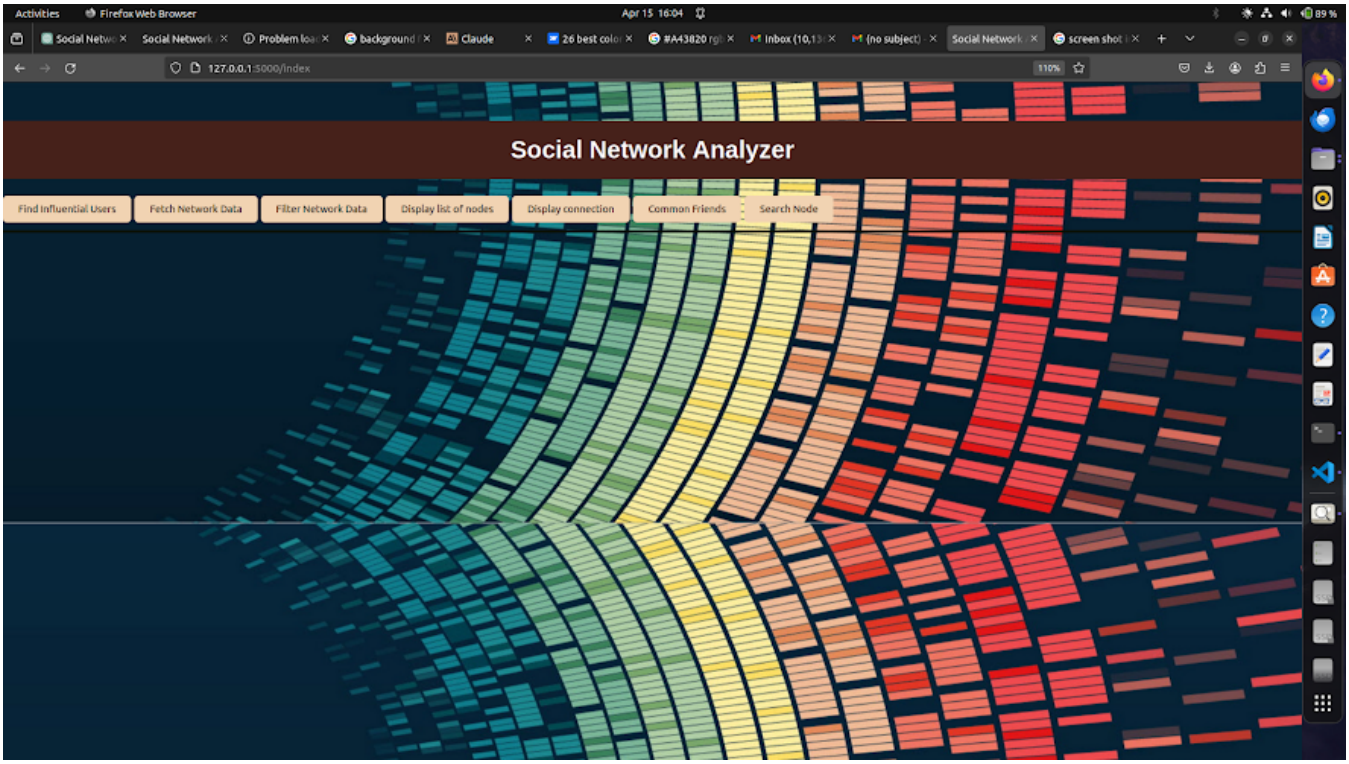
- `./datasets` contains the essential `facebook_combined.txt` containing edges from all ego-nets combined.
- `./templates` contains the HTML templates used for rendering the frontend of the application, includes files such as `index2.html`, `query1.html`, `query2.html`, `search_form.html`, `common_form.html`, etc.
- `graph.py` provides a function to download datasets from the Stanford Large Network Dataset Collection. The `download_dataset()` function takes two parameters: `dataset_url` (the URL of the dataset to be downloaded) and `save_directory` (the directory where the downloaded dataset will be saved).
- `database.py` initializes a Flask application for building a web-based platform for social network analysis and visualization. It imports necessary modules such as Flask, Neo4j, Pandas, and PyVis. The Flask application is configured with routes to handle different functionalities

4. Results and Demonstration

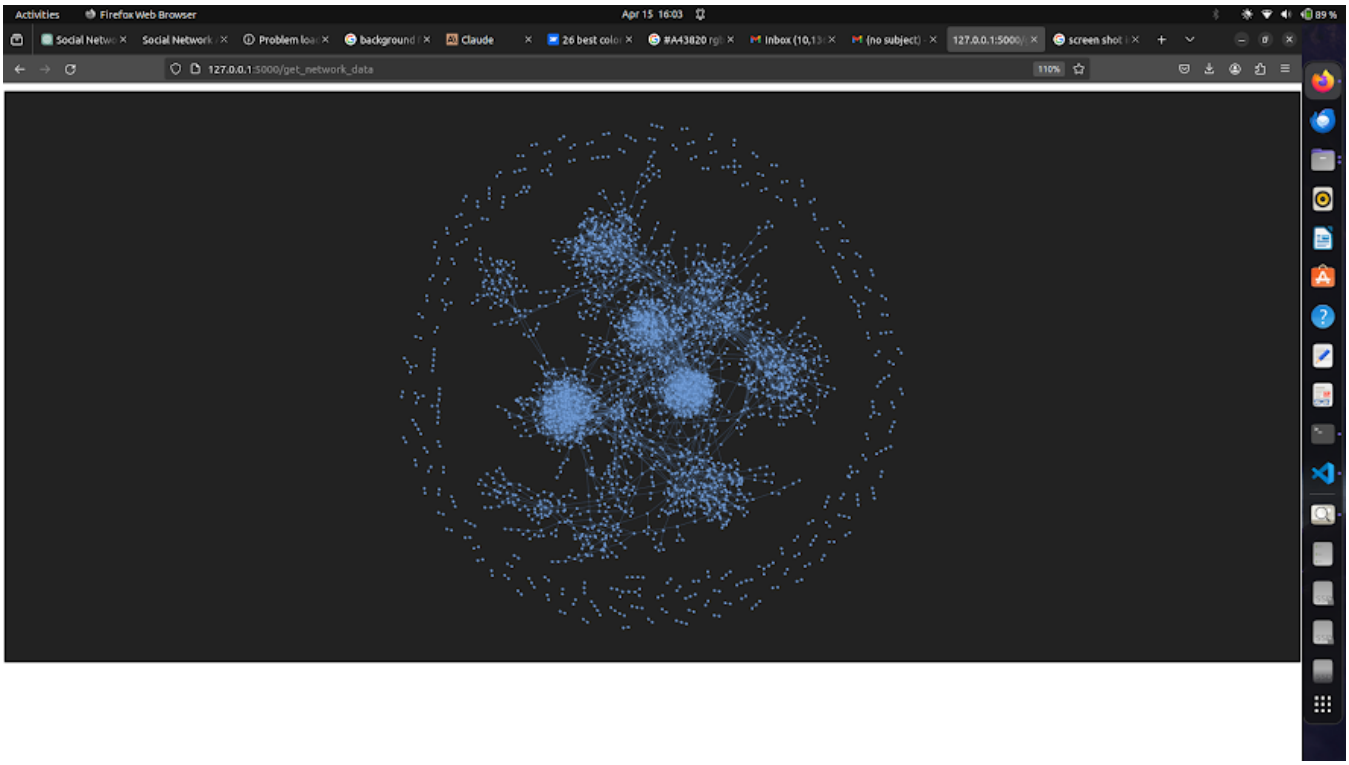
4.1 Home Page



4.2 Feature Page

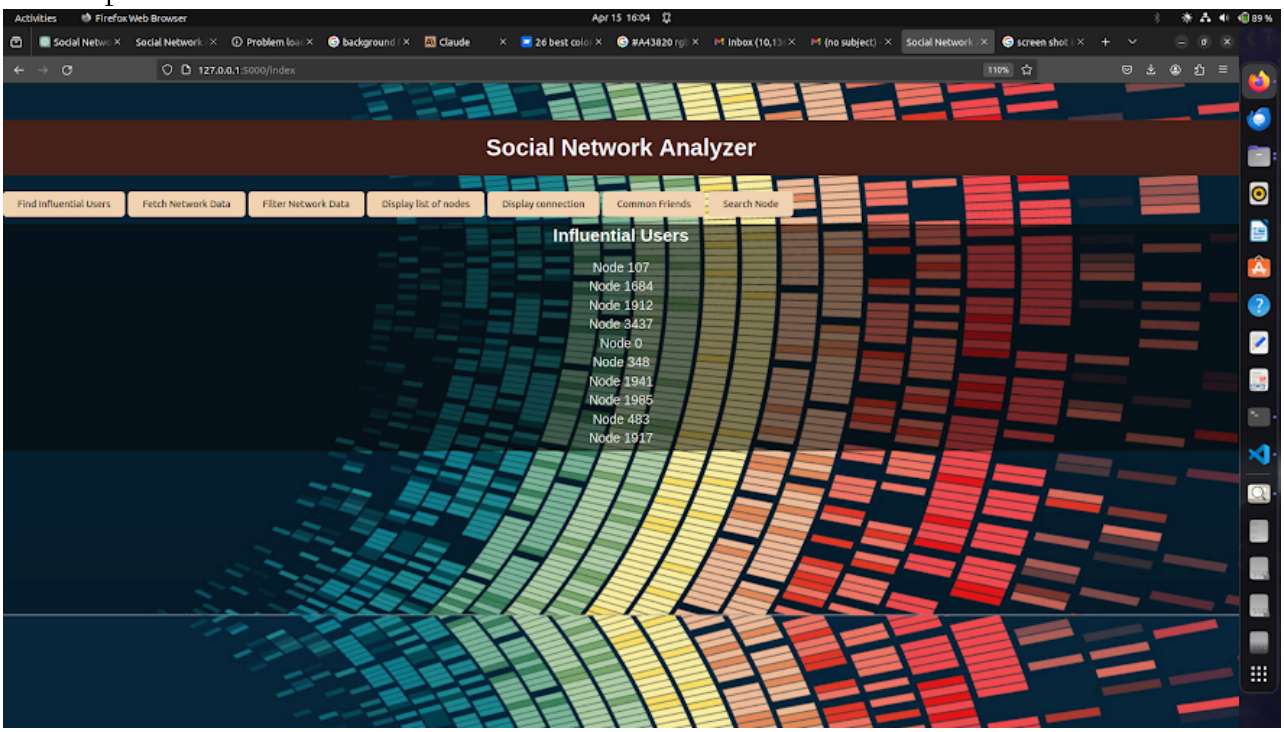


4.3 Visualize Network and Sub-Networks



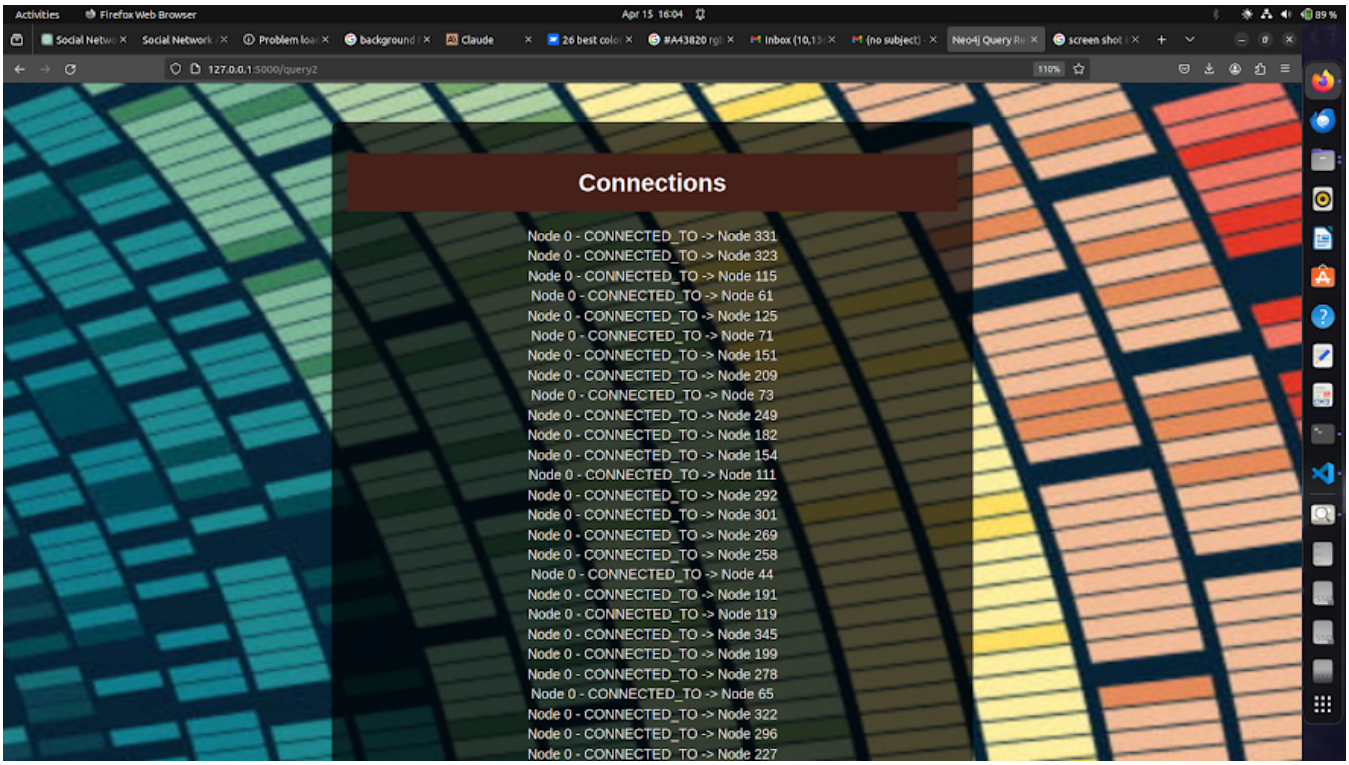
4.4 Influential Users

- Lists top 10 influential users based on users with most friends

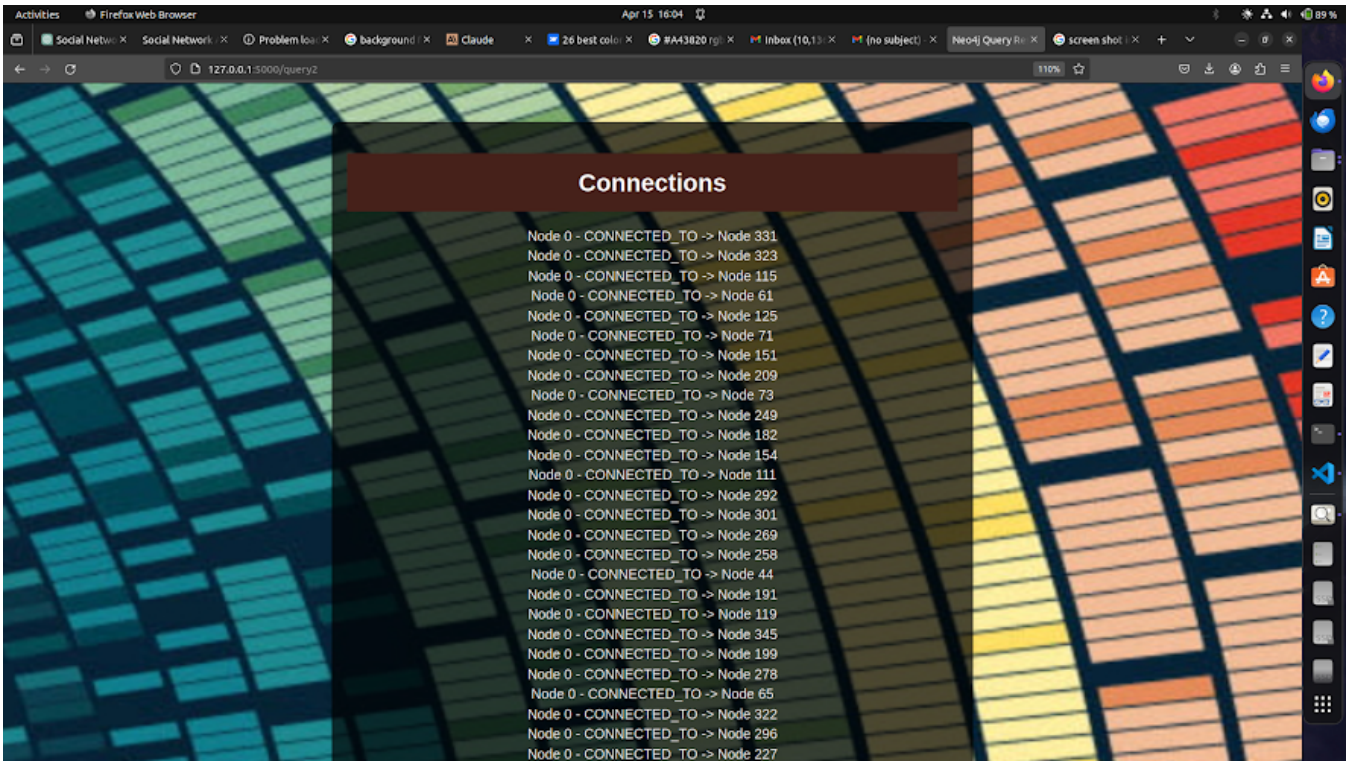


4.5 List

Connections:

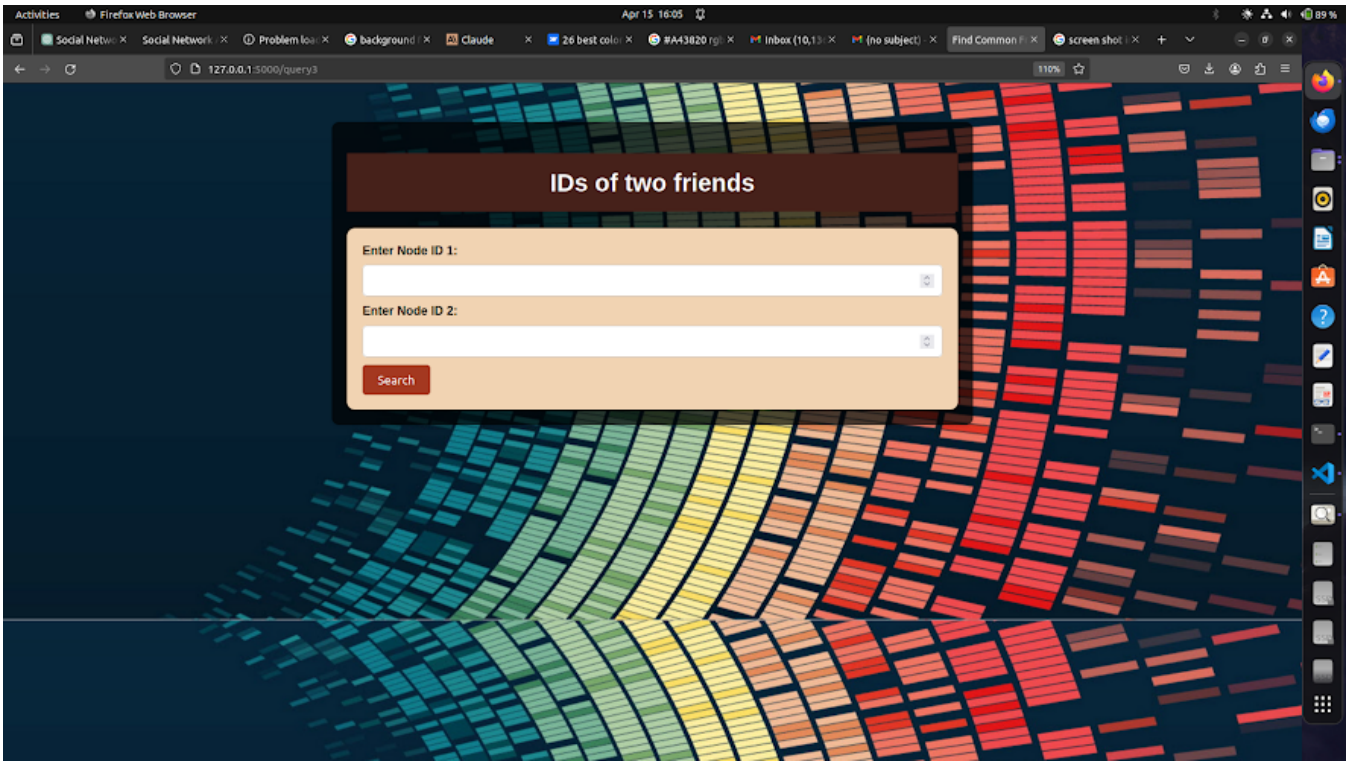


Nodes:

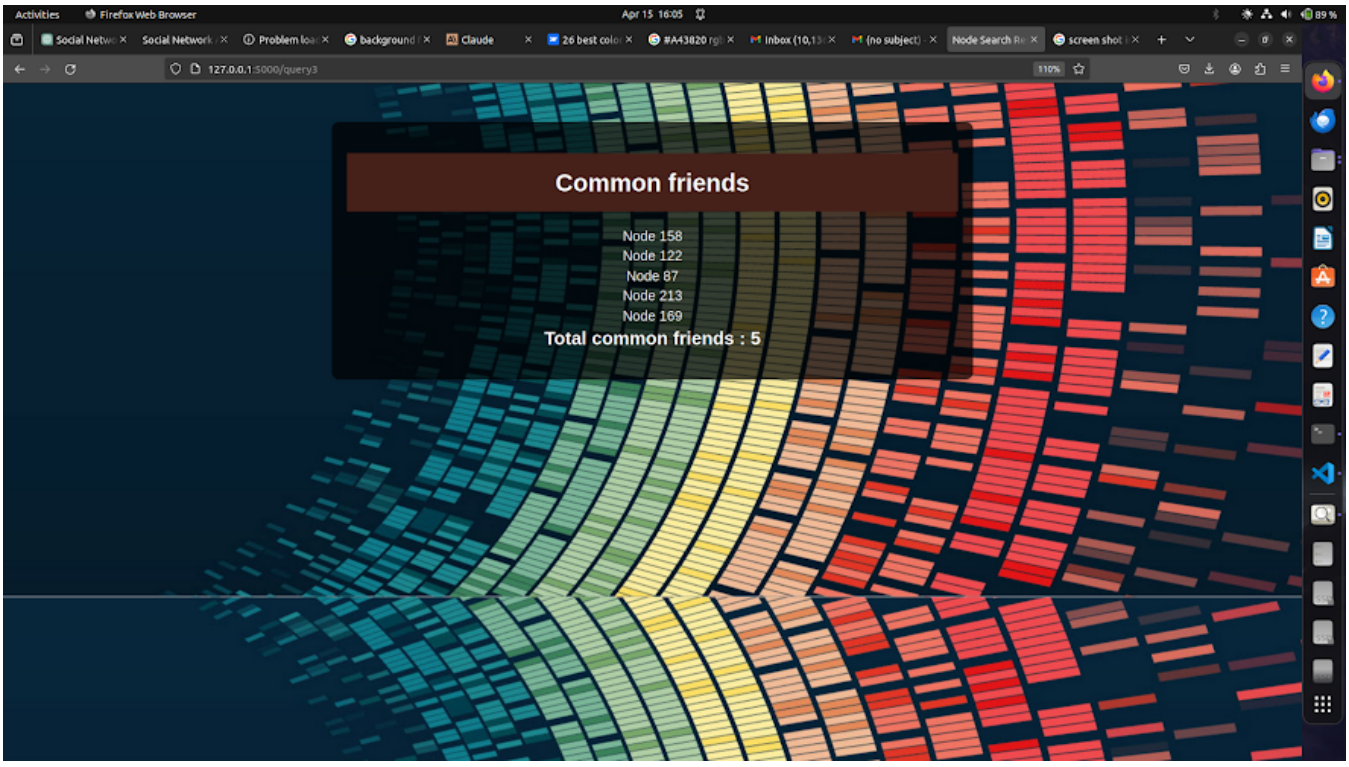


4.6 Find Mutual Friends For Two IDs

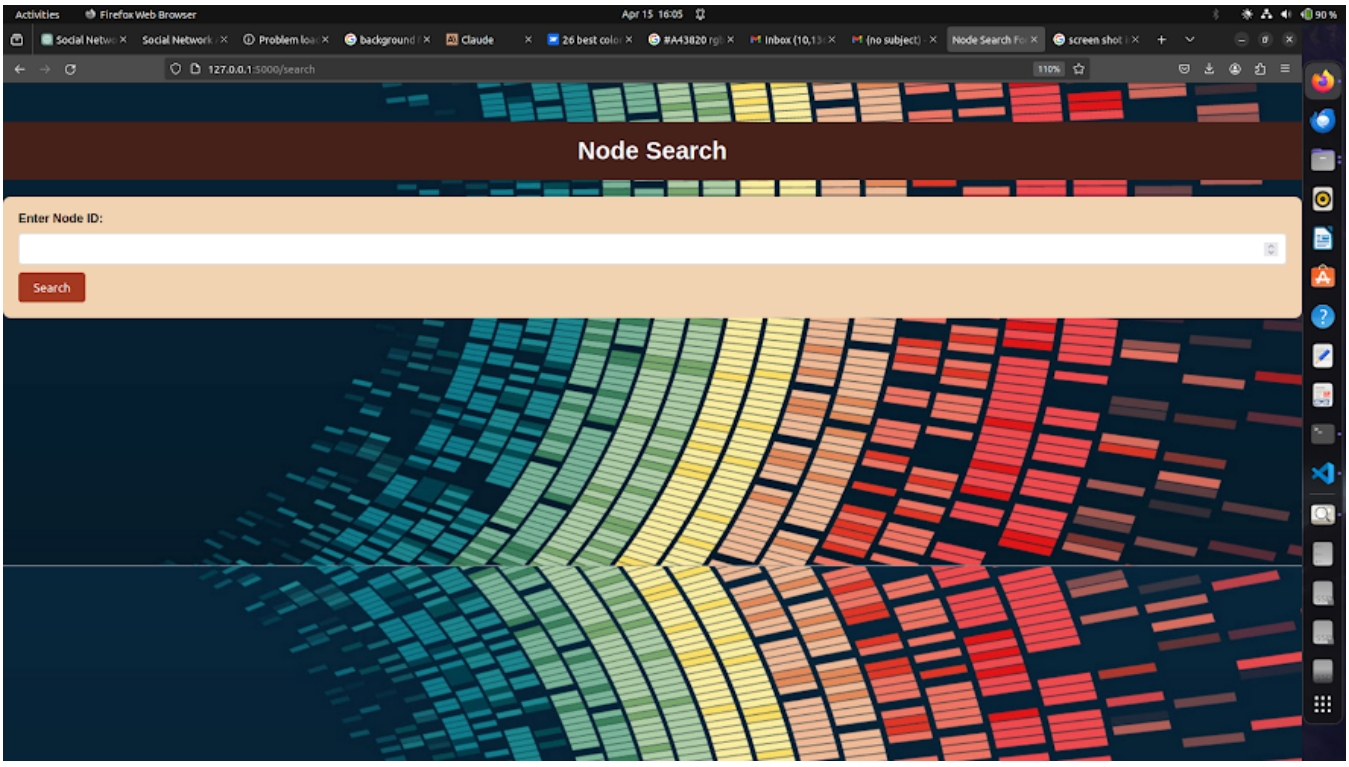
Query



Results



4.7 Get Results For ID



5. References

1. For the dataset: https://snap.stanford.edu/data/ego-Facebook.html
2. Documentation for neo4j: https://neo4j.com/docs/
3. Intro to Graph Databases by neo4j: https://www.youtube.com/playlist?list=PL9Hl4pk2FsvWM9GWaguRhlCQ-pa-ERd4U
4. For data visualisation: https://towardsdatascience.com/pyvis-visualize-interactive-network-graphs-in-python-77e059791f01
5.