**PROGRESS REPORT**

**DISCRETE MATHEMATICS**

**FINAL EXAM**

****

**Actuarial Science 2024**

Lecturer:

Dr. Edwin Setiawan Nugraha, S.Si., M.Sc.

Compiled by:

Feriza Anggraeni (021202400021)

Hana Arabella Liliana Sigalingging (021202400003)

Alexandra Mercy Christani (0212024000

**202****4**

# Members of The Group and The Jobs Distribution

1. Alexandra Mercy Christani

Alexa is responsible for coding and ensuring the application runs smoothly.

1. Feriza Anggraeni

Feriza is responsible for collecting, entering, organizing, and researching the necessary data.

1. Hana Arabella L. S.

Hana is responsible for searching, collecting, editing, and organizing the necessary data.

Table of Content

[Members of The Group and The Jobs Distribution ii](#_Toc185113414)

[1. GRAPH THEORY 1](#_Toc185113415)

[1.1 Concept of Graph Theory 1](#_Toc185113416)

[1.2 History of Graph Theory 1](#_Toc185113417)

[1.3 Background of Graph Theory 2](#_Toc185113418)

[1.4 Application in General 2](#_Toc185113419)

[1.4.1 Computer Networks 2](#_Toc185113420)

[1.4.2 Social Networks 2](#_Toc185113422)

[1.4.3 Transportation Networks 2](#_Toc185113424)

[1.4.4 Biology 3](#_Toc185113426)

[1.5 Type of Graph 3](#_Toc185113428)

[1.5.1 Uniracted Graph 3](#_Toc185113429)

[1.5.1 Weighted Graph 4](#_Toc185113431)

[1.5.2 Unweighted Graph 5](#_Toc185113432)

[1.5.3 Special Graph 5](#_Toc185113433)

[1.5.3.1 Complate Graph.](#_Toc185113434)

[1.5.3.2 Bipartite Graph 6](#_Toc185113435)

[1.5.3.3 Tree 6](#_Toc185113436)

[1.6 Graph Representation 7](#_Toc185113437)

[1.6.1 Adjacency Matrix 7](#_Toc185113440)

[1.6.2 Adjacency List 8](#_Toc185113442)

[1.7 Example 9](#_Toc185113443)

[1.7.1 Shortest Path Problem using Dijkstra's Algorithm 9](#_Toc185113444)

[1.7.2 Graph Coloring Problem using the Greedy Coloring Algorithm 10](#_Toc185113445)

[2. PYTHON AND PROGRAMMING LANGUANGE 11](#_Toc185113446)

[2.1 Programming Languange 11](#_Toc185113447)

[2.2 Python History 11](#_Toc185113448)

[2.3 Python Library 12](#_Toc185113449)

[2.4 Library for Graph 14](#_Toc185113450)

[2.5 2Aaconda 16](#_Toc185113451)

[2.6 Visual Studio Code 16](#_Toc185113452)

[2.7 Github 16](#_Toc185113453)

[2.8 Streamlit Community Cloud 17](#_Toc185113454)

[2.9 Streamlit Application 17](#_Toc185113455)

[2.10 How to Create Streamlit 18](#_Toc185113456)

[3. APPENDIX 19](#_Toc185113457)

[4. REFERENCES 22](#_Toc185113458)

[22](#_Toc185113459)

[22](#_Toc185113460)

[23](#_Toc185113461)

[5. REFERENCES 24](#_Toc185113462)

## GRAPH THEORY

Graph theory is a fundamental branch of mathematics that studies the relationships between objects. Represented through vertices and edges, it provides a powerful tool for solving complex problems in fields such as computer networks, transportation, and social media analysis. This paper introduces the basics of graph theory, exploring its definitions, types, and real-world applications.

### Concept of Graph Theory

Graph theory is a branch of mathematics that focuses on the study of graphs, which consist of nodes and edges that connect them. Graphs are used to represent various types of structures and relationships, ranging from computer networks to social networks.

### History of Graph Theory

1. The paper written by Leonhard Euler on the seven Bridges of
2. Königsberg and published in 1736 is regarded as the first paper in the
3. history of graph theory. This paper, as well as the one written by
4. Vandermonde on the knight problem, carried on with the analysis
5. situsinitiated by Leibniz. Euler’s formula relating the number of
6. edges, vertices, and faces of a convex polyhedron was studied and
7. The paper written by Leonhard Euler on the seven Bridges of
8. Königsberg and published in 1736 is regarded as the first paper in the
9. history of graph theory. This paper, as well as the one written by
10. Vandermonde on the knight problem, carried on with the analysis
11. situsinitiated by Leibniz. Euler’s formula relating the number of
12. edges, vertices, and faces of a convex polyhedron was studied and
13. The paper written by Leonhard Euler on the seven Bridges of
14. Königsberg and published in 1736 is regarded as the first paper in the
15. history of graph theory. This paper, as well as the one written by
16. Vandermonde on the knight problem, carried on with the analysis
17. situsinitiated by Leibniz. Euler’s formula relating the number of
18. edges, vertices, and faces of a convex polyhedron was studied and
19. The paper written by Leonhard Euler on the seven Bridges of
20. Königsberg and published in 1736 is regarded as the first paper in the
21. history of graph theory. This paper, as well as the one written by
22. Vandermonde on the knight problem, carried on with the analysis
23. situsinitiated by Leibniz. Euler’s formula relating the number of
24. edges, vertices, and faces of a convex polyhedron was studied and
25. The paper written by Leonhard Euler on the seven Bridges of
26. Königsberg and published in 1736 is regarded as the first paper in the
27. history of graph theory. This paper, as well as the one written by
28. Vandermonde on the knight problem, carried on with the analysis
29. situsinitiated by Leibniz. Euler’s formula relating the number of
30. edges, vertices, and faces of a convex polyhedron was studied and
31. The paper written by Leonhard Euler on the seven Bridges of
32. Königsberg and published in 1736 is regarded as the first paper in the
33. history of graph theory. This paper, as well as the one written by
34. Vandermonde on the knight problem, carried on with the analysis
35. situsinitiated by Leibniz. Euler’s formula relating the number of
36. edges, vertices, and faces of a convex polyhedron was studied and
37. The paper written by Leonhard Euler on the seven Bridges of
38. Königsberg and published in 1736 is regarded as the first paper in the
39. history of graph theory. This paper, as well as the one written by
40. Vandermonde on the knight problem, carried on with the analysis
41. situsinitiated by Leibniz. Euler’s formula relating the number of
42. edges, vertices, and faces of a convex polyhedron was studied and
43. The paper written by Leonhard Euler on the seven Bridges of
44. Königsberg and published in 1736 is regarded as the first paper in the
45. history of graph theory. This paper, as well as the one written by
46. Vandermonde on the knight problem, carried on with the analysis
47. situsinitiated by Leibniz. Euler’s formula relating the number of
48. edges, vertices, and faces of a convex polyhedron was studied and
49. The paper written by Leonhard Euler on the seven Bridges of
50. Königsberg and published in 1736 is regarded as the first paper in the
51. history of graph theory. This paper, as well as the one written by
52. Vandermonde on the knight problem, carried on with the analysis
53. situsinitiated by Leibniz. Euler’s formula relating the number of
54. edges, vertices, and faces of a convex polyhedron was studied and

The German city of Königsberg (present-day Kaliningrad, Russia) is situated on the Pregolya river. The geographical layout is composed of**four** main bodies of land connected by a total of **seven** bridges. The question posed to Euler was straightforward: was it was possible to take a walk through the town in such a way as to cross over every bridge once, and only once. Euler, recognizing that the relevant constraints were the four bodies of land & the seven bridges, drew out the first known visual representation of a modern graph. A modern graph, as seen in bottom-right image C, is represented by a set of points, known as **v**ertices or nodes, that connected by a set of connecting lines known as **e**dges.

By first attempting to draw paths in the graph above, then later experimenting with multiple theoretical graphs with alternating number of vertices & edges, he eventually extrapolated a general rule: In order to be able to walk in an Euler path a graph can have none or two odd number of nodes? From there, the branch of math known as graph theory lay dormant for decades. In modern times, however, it’s application is finally exploding.

### Background of Graph Theory

Graph theory began as an obscure subfield of mathematics, but it has since turned into an incredibly useful tool for understanding the modern world. It essentially is a streamlined method for dealing with abstract objects and the links between them. This area of study is usually contained within the broader field of combinatorics, but it has many unique aspects that make it useful. As the world becomes more and more connected, and as data becomes more readily available, graph theory has become an essential framework for making sense of it.

### **Application in General**

Graph theory has many practical applications, such as:

#### Computer Networks

#### Each device or node in the network can be represented as a vertex, while the connections between them, such as cables or wireless links, are represented as edges. Using graph-based models, network engineers can study connectivity, detect vulnerabilities, optimize data flow, and ensure the resilience of the network. Applications include routing protocols, load balancing, and network topology design.

#### Social Networks

#### In social network analysis, graph theory is used to model and understand the relationships between individuals or groups. Here, vertices represent individuals, and edges represent interactions, such as friendships, collaborations, or communication. derstanding and modeling relationships between individuals.

#### Transportation Networks

#### Transportation systems, such as roadways, railways, and air routes, can be modeled as graphs to solve problems related to route optimization and traffic management. In such graphs, locations (e.g., cities or intersections) are represented as vertices, and the paths or roads between them are edges.

#### Biology

#### In biology, graph theory is applied in areas such as bioinformatics to analyze molecular structures, protein interactions, and genetic relationships. For example, metabolic networks and gene regulatory networks are represented as graphs to identify key molecules or genes.

Graphs help model complex real-world problems in a more structured form and enable the application of algorithms to solve those problems.

### Type of Graph

Graphs come in many forms, each suited to different types of problems and applications. The structure and properties of a graph can vary depending on how its vertices and edges are defined or connected. By classifying graphs into specific types, we can better understand their behavior and choose the right approach for solving problems in fields such as computer science, logistics, or social networks. There are two basic components in graph, nodes and edges. Nodes are poimts in a graph and they represent entities such as computers in a network or people in a social network. Edges are line that connect two nodes and they can be directed or undirected. Edges marked with an narrow, indicating a one-way relationship.

Below are some of the types of graphs along with their characteristics:

1.5.1 Uniracted Graph  
Lines do not have direction. For example, friendships on social networks can be considered undirected because if A is friends with B, then B is also friends with A. Example:

A B

C D

The nodes are A, B, C, D. The edges are (A-B), (A-C), (B-D), and (C-D).

##### Lines have direction. An example is a web link, where one webpage might link to another page, but not the other way around. Directed graphs are used in various applications such as flowcharts, computer networks, and so on.

Example :

A B C

|

D E

#### Weighted Graph

Each line has a weight that indicates a certain value, such as distance or cost. An example of use is a transportation network, where the weight can represent the distance between two locations. Example:

4

A ---------- B

| / |

| 5 /3 | 2

| / |

| / |

C -----------D

* Nodes: A, B, C, D
* Weighted Edges:
* A to B with weight 4
* A to C with weight 5
* B to D with weight 2
* C to D with weight 7
* B to C with weight 3
* Weights on the lines indicate specific values, such as distance, cost, or capacity.

In this context, weights can represent, for example, the distance between two cities, the cost of traveling from one point to another, or the capacity of a route in a transportation network. Weighted graphs are often used in various applications such as route optimization, shortest path search algorithms (like Dijkstra's algorithm), and network analysis.

#### Unweighted Graph

Lines do not have weight. They only indicate the presence or absence of a relationship. Such graphs are often used to represent basic networks where the presence or absence of a connection is important, rather than the strength or significance of the connection. Example:

A --- B

| |

| |

C --- D

Nodes**:** A, B, C, D.

* Unweighted Edges:
* A connected with B (A - B)
* A connected with C (A - C)
* B connected with D (B - D)
* C connected with D (C - D)
* In this unweighted graph, each line only indicates the presence or absence of a relationship between the nodes, without any specific value or weight attached to the line.

#### Special Graph

##### Complete Graph Each node is directly connected to all other nodes. Example:

A ---- B

| \ / |

| \/ |

| / \ |

C ---- D

* Nodes: A, B, C, D.
* Edges: Each node is connected to all other nodes, so there is a line between every pair of nodes.

##### Bipartite Graph The nodes can be divided into two different groups where no node in one group is connected to a node in the same group.

Example :

1 3

2 . 4

* Nodes: 1, 2, 3, 4.
* Sets: The nodes can be divided into two sets: {1, 3} and {2, 4}.
* Edges: All edges connect a node from one set to a node from the other set. For instance, node 1 connects to node 2, and node 3 connects to node 4. There are no edges connecting nodes within the same set.

Bipartite graphs are particularly useful in scenarios like matching problems, where you might want to pair elements from one group with elements from another group, such as matching jobs to applicants or students to projects.

##### Tree

A connected graph without cycles. This means there is a unique path between every pair of nodes. A tree is a type of graph that is acyclic and connected. This means there is exactly one path between any two nodes, and there are no cycles. Trees are used in various applications such as representing hierarchical structures, organizational charts, and file systems. Example:

**A**

**/ \**

**B C**

**/ \ \**

**D E F**

* Root Node: A is the root node.
* Child Nodes:
* B and C are children of A
* D and E are children of B
* F is a child of C.
* Lead Nodes: D, E, and F are leaf nodes (they have no children).

### Graph Representation

### To work effectively with graphs, it is important to represent them in a way that suits the problem at hand. A graph can be represented in several formats, each offering unique advantages depending on the context. The choice of representation often impacts the efficiency of algorithms and computations performed on the graph.

### Below are the most common methods for representing graphs:

#### Adjacency Matrix

#### A two-dimensional matrix that shows whether there is a line between two nodes. This representation is helpful for algorithms that need to quickly check the presence or absence of edges between nodes. For example, matrix[i][j] = 1 if there is a line from node i to node j. Example :

A ――― B  
| \ |

| \ |

| \ |

C ――― D

* Nodes: (A, B, C, and D)
* Edges: (A-B, A-C, A-D, B-D, C-D).

The adjacency matrix for this graph would be:

* Rows represent the starting nodes (A, B, C, D)
* Columns represent the ending nodes (A, B, C, D).
* A value of 1 indicates that there is an edge between the nodes, and 0 indicates there is no edge.

Another example:

* There is an edge between A and B, so the matrix has a 1 at (A, B) and (B, A).\
* There is no edge between B and C, so the matrix has a 0 at (B, C) and (C, B).

#### Adjacency List

Each node has a list of nodes connected to it. This is more storage-efficient for graphs with many nodes and few edges. Adjacency lists are useful for efficiently storing sparse graphs (graphs with fewer edges compared to the number of nodes). This representation allows for quick access to the neighboˡrs of a node, which is helpful for many graph algorithms. Example:

3

4 1

* The adjacency list for this graph would look like this:

**A: B, C, D**

**B: A, D**

**C: A, D**

**D: A, B, C**

* Each node has a list of nodes it's directly connected to. For example, node A is connected to nodes B, C, and D. Similarly, node D is connected to nodes A, B, and C.

### Example

This section explores the application of graph theory in solving two key problems: the Shortest Path Problem using Dijkstra's Algorithm and the Graph Coloring Problem using the Greedy Coloring Algorithm. Clear examples are provided to illustrate the steps and solutions for each problem.

#### Shortest Path Problem using Dijkstra's Algorithm

Problem:

There are 5 vertices: A, B, C, D, and E. The edges between these vertices and their respective weights (distances) are shown in the table below:

|  |  |
| --- | --- |
| **Edge** | **Distance** |
| A to B | 4 |
| A to C | 2 |
| B to C | 1 |
| B to D | 5 |
| C to D | 8 |
| C to E | 10 |
| D to E | 2 |

Solution:

* First step:  
  Set all distances to infinity (∞), except the starting point A, which is set to 0.

Initial distances: A = 0, B = ∞, C = ∞, D = ∞, E = ∞.

* Second step:

Look at its neighbors:

A → B = 4 → Distance to B = 4.

A → C = 2 → Distance to C = 2.

Updated distances: A = 0, B = 4, C = 2, D = ∞, E = ∞.

* Third step:

Vertex C (distance = 2). Check its neighbors:

C → B = 1 → Distance to B = 2 + 1 = 3 (better than 4).

C → D = 8 → Distance to D = 2 + 8 = 10.

C → E = 10 → Distance to E = 2 + 10 = 12.

Updated distances: A = 0, B = 3, C = 2, D = 10, E = 12.

* Fourth step:

Vertex B (distance = 3). Check its neighbors:

B → D = 5 → Distance to D = 3 + 5 = 8 (better than 10).

Updated distances: A = 0, B = 3, C = 2, D = 8, E = 12.

* Fifth step:

Vertex D (distance = 8). Check its neighbors:

D → E = 2 → Distance to E = 8 + 2 = 10 (better than 12).

Updated distances: A = 0, B = 3, C = 2, D = 8, E = 10.

* Final answer:

The shortest distance from **A to E** is **10**.

#### Graph Coloring Problem using the Greedy Coloring Algorithm

Problem:

Given a graph with 5 vertices A, B, C, D, and E. The connections (edges) between the vertices are as follows:

A is connected to B and C.B is connected to A, C, and D.

C is connected to A, B, D, and E.

D is connected to B, C, and E.

E is connected to C and D.

Color the graph such that no two connected vertices have the same color using the Greedy Coloring Algorithm.

Solution:

* First step:

Start with vertex A: Since it has no color yet, assign Color 1.

A = Color 1.

* Second step:

Vertex B is connected to A, which has Color 1, so assign Color 2.

A = Color 1, B = Color 2.

* Third step:

Vertex C is connected to A (Color 1) and B (Color 2). Assign Color 3.

A = Color 1, B = Color 2, C = Color 3.

* Fourth step:

Vertex D is connected to B (Color 2) and C (Color 3). Assign Color 1.

A = Color 1, B = Color 2, C = Color 3, D = Color 1.

* Fifth step:

Vertex E is connected to C (Color 3) and D (Color 1). Assign Color 2.

A = Color 1, B = Color 2, C = Color 3, D = Color 1, E = Color 2.

* Final answer:

The graph can be colored using 3 colors:

A = Color 1

B = Color 2

C = Color 3

D = Color 1

E = Color 2

## PYTHON AND PROGRAMMING LANGUANGE

### Programming Languange

Humans and computers can communicate through programming languages, where programmers create machine-understandable code that can carry out a variety of operations, from basic math to intricate data processing. Data processing and analysis are done using programming languages, especially Python.

The fundamental components of this language are as follows :

* Variable: A location for data storage that may alter while the application is operating.
* Statements and Expressions: A statement is a computer-executed instruction, whereas an expression is a set of values and operators that results in a new value.

A function is a block of code that can be called repeatedly within a computer and is intended to carry out a certain purpose. Programming languages are important in today's world, and by knowing them, we may develop creative solutions in a variety of disciplines. This language also contains syntax and rules that must be followed for the code to execute successfully

### Python History

Programmer Guido van Rossum developed Python at the Centrum Wiskunde & Informatica (CWI) in the Netherlands in the late 1980s. Inspired by the ABC programming language at the time, Van Rossum set out to develop a more straightforward and intelligible language.

Python 0.9.0, its initial release, was made available in February 1991. Numerous fundamental programming capabilities were already present in this version. Then, a lot of new features were added with the 2008 release of Python version 1.0. Version 2.0, which included garbage collection and Unicode compatibility, was later released in 2000. While it was not entirely compatible with earlier iterations, Python 3.0's 2008 release brought about substantial modifications to the language's grammar and structure, making it cleaner and more effective.

In 2001, Guido founded the Python Software Foundation (PSF) to promote the open source development and dissemination of this language. Since its introduction, Python has grown to become one of the most widely used programming languages worldwide, with applications in a wide range of domains, including data analysis, web development, artificial intelligence, and much more. - **Buku "Menguasai Python: Membuka Masa Depan Pemrograman"** oleh Pri Anton Subardio In this progress, we use 2 Python libraries, there’s numpy and matplotlib.

### Python Library

Python library is a collection of pre-compiled code that can be used to complete various tasks in programming. This library serves to simplify the development process by providing ready-to-use functions, so programmers do not need to write code from scratch for each required function. This library is also often equipped with documentation, configuration data, and classes that facilitate its us.

Here are the most commonly used Python libraries:

1. NumPy

This is a library for numerical computation that allows users to work with multidimensional arrays and perform complex mathematical operations such as linear algebra and Fourier transforms. NumPy is essential in the fields of data science and numerical analysis.

1. Pandas

This library is designed for data manipulation and analysis. Pandas provides flexible and expressive data structures, as well as features like handling missing data and complex indexing, making it very useful for data analysis.

1. Matplotlib

Used for data visualization, Matplotlib allows users to easily create static graphs, animations, and interactive visualizations. It is very useful in data presentation and visual analysis.

In this project we use Numpy and Mathpolib for:

1. Mathematical Operations with Trigonometric Function
   * + 1. **Sinus (Sin)**

A function that produces values between -1 and 1, useful for calculating waves and oscillations.

* + - 1. **Cosinus (Cos)**

Similar to sine but shifted 90 degrees, often used for rotation calculation.

* + - 1. **Tangent (Tan)**

The ratio of sin/cos, its value can be very large (infinite),

useful for calculating slopes

1. Creating an Array with Linspace

* Function to create a sequence of numbers with equal spacing
* Requires a starting value, an ending value, and the desired number of points
* Creating the x-axis in a graph
* Dividing intervals evenly
  + - * Generating orderly data for simulations

1. Manipulation of Numerical Data

* Basic operations (add, subtract, multiply, divide)
* Statistic (average, median, standard deviation)
* Data transformation (normalization, scalling)

1. Creating visualization/plots
2. Setting subplot layout
3. Setting properties

### Library for Graph

Libraries for Graph Analysis in Streamlit:

1. NetworkX

This Library is pure Python for complex network analysis, focuses on computational graph algorithms, and used for creating, manipulating, and studying complex networks

Key Features:

* Graph generation methods
* Network analysis algorithms
* Support for multiple graph types (directed, undirected, weighted)
* Centrality measures computation
* Path finding algorithms

1. Plotly

This library use for Interactive visualization library, Web-based, supports multiple programming languages, and for creates interactive, publication-quality graphs

Key Features:

* Dynamic, zoomable visualizations
* Multiple chart types including network graphs
* Browser-based interactions
* Easy web integration
* Supports 3D visualizations

1. Graphviz

This is help for graph visualization software, provides a way to represent structural information as diagrams, and make originally developed at AT&T Bell Laboratories

Key Features:

* utomatic graph layout algorithms
* Support for various graph formats
* Command-line and programming interfaces
* Generates visual representations of graphs and networks

1. PyViz (HoloViews)

Library to make high-level visualization, can help to simplifies complex visualization tasks, and works with multiple backend libraries

* Declarative API for data visualization
* Supports multiple plotting libraries
* Flexible data handling
* Simplified complex visualization creation
* Works well with scientific computing ecosystems

### 2Aaconda

Anaconda's Conda integration simplifies package administration by allowing users to install, uninstall, and manage packages without worrying about version conflicts. This platform includes analytical tools including NumPy, Pandas, Matplotlib, SciPy, and Scikit-learn, which are required for data science and machine learning. Anaconda also integrates with popular IDEs like Jupyter Notebook, Spyder, and Navigator to make development and debugging easier. In addition, Anaconda works with Windows, macOS, and Linux.

In this Progress, we already install Anaconda

### Visual Studio Code

Microsoft created the source code editor known as Visual Studio Code (VS Code). VS Code is intended to be a quick and lightweight editor that focuses on application development using a variety of programming languages, in contrast to Visual Studio, which is a more extensive and sophisticated Integrated Development Environment (IDE). The multiplatform code editor Visual Studio Code (VS Code) is compatible with a number of programming languages, including C#, Python, and JavaScript. The integrated debugging tools make application testing easier, and the IntelliSense function offers code completion and automatic suggestions. Additionally, VS Code features a Live Share tool for in-the-moment developer collaboration and an extension marketplace for adding functionality.

We Already have Visual Studio Code for this project.

### Github

GitHub is a web-based platform for storing, managing, and sharing software project source code that uses the Git version control system. This platform includes repositories for storing code and related files, as well as collaboration tools like pull requests, which allow users to propose changes and get feedback from other team members. GitHub also provides issue tracking services to manage bugs, feature requests, and tasks, as well as support for open-source project development, allowing other developers to contribute to and learn from existing code. GitHub is vital in modern software development and developer collaboration.

In this progress, we already create GitHub account

### Streamlit Community Cloud

Streamlit Community Cloud is a free platform that allows you to deploy, manage, and share your Streamlit applications effortlessly. By integrating directly with your GitHub repositories.

Key Features:

* One-Click Deployment: Deploy your app in under a minute by selecting a repository, branch, and file, then clicking "Deploy."
* Automatic Updates: Your app updates instantly with each git push, ensuring users always access the latest version.
* Secure Data Connections: Connect to various data sources using secure protocols, maintaining data integrity and security.
* Access Control: Authenticate viewers with per-app allow-lists to restrict access as needed.
* App Management: Manage all your apps in one place, facilitating collaboration and oversight.

### Streamlit Application

Streamlit The application we created consists of 3 menus.

* + - 1. In the first menu, which is the profile, the page displays the title of this project, which is 'East Java City Application'. Below it, there is information about the team members, and there is a special block that can navigate to each member's page according to their name.
      2. The next menu is graphics, this page shows a rough sketch before the map creation is refined. there are interconnected blocks with the names of cities in East Java province.
      3. The last menu is the map. This page is an extension of the previous graphic. showing a map of East Java province. the page can be zoomed in and adjusted with touch. the visual of this map is created based on the commonly used colour grading, and there are also connection lines linking one city to another

This is the link to visit ourstreamlit :

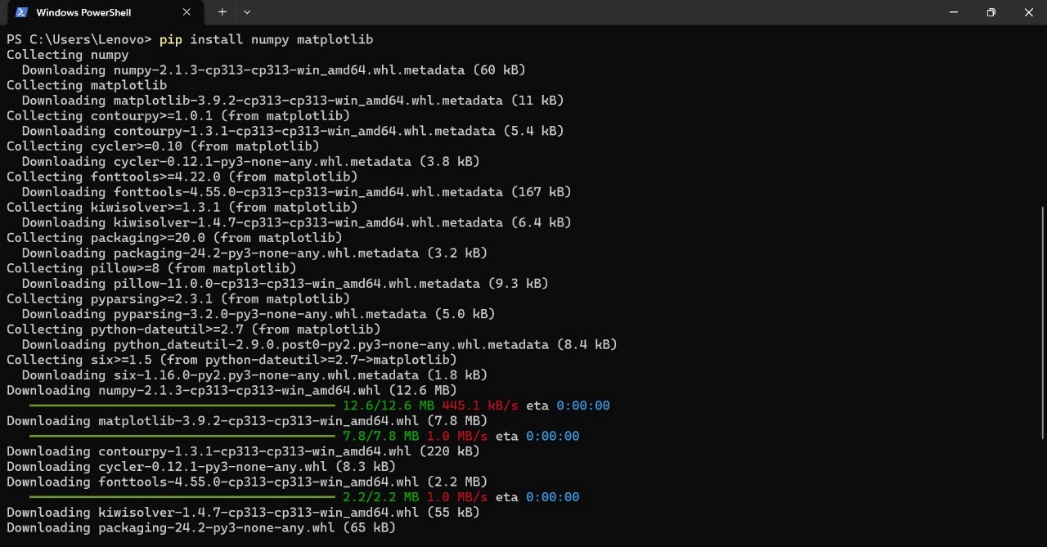
<https://anjay-slebew-dspqpevdfnwqs68pcyahnz.streamlit.app/>

### How to Create Streamlit

In the progress, make Streamlit its quiet simple, here’s the tutorial for create streamlit that you can watch on Youtube :

* Video one : <https://youtu.be/c1n5iCMzr9E?si=QdQccCUpwFCxCfwu>
* Video two : <https://youtu.be/HU_kd-1uIkQ?si=s04EJ6uJa8e02aGk>

## APPENDIX

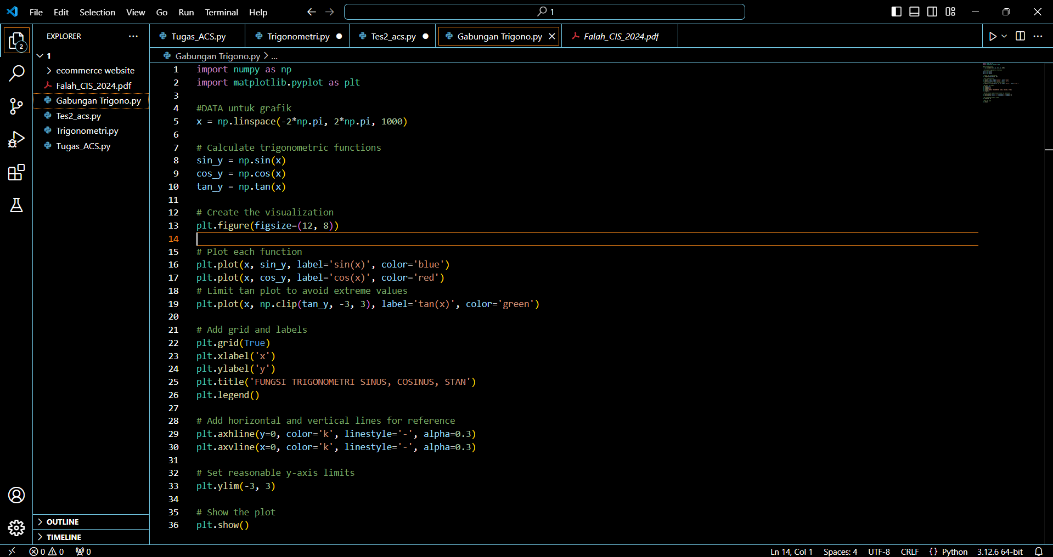
****

Pict Visualization Trigonometri Function, Installed Numpy and Matplotlib

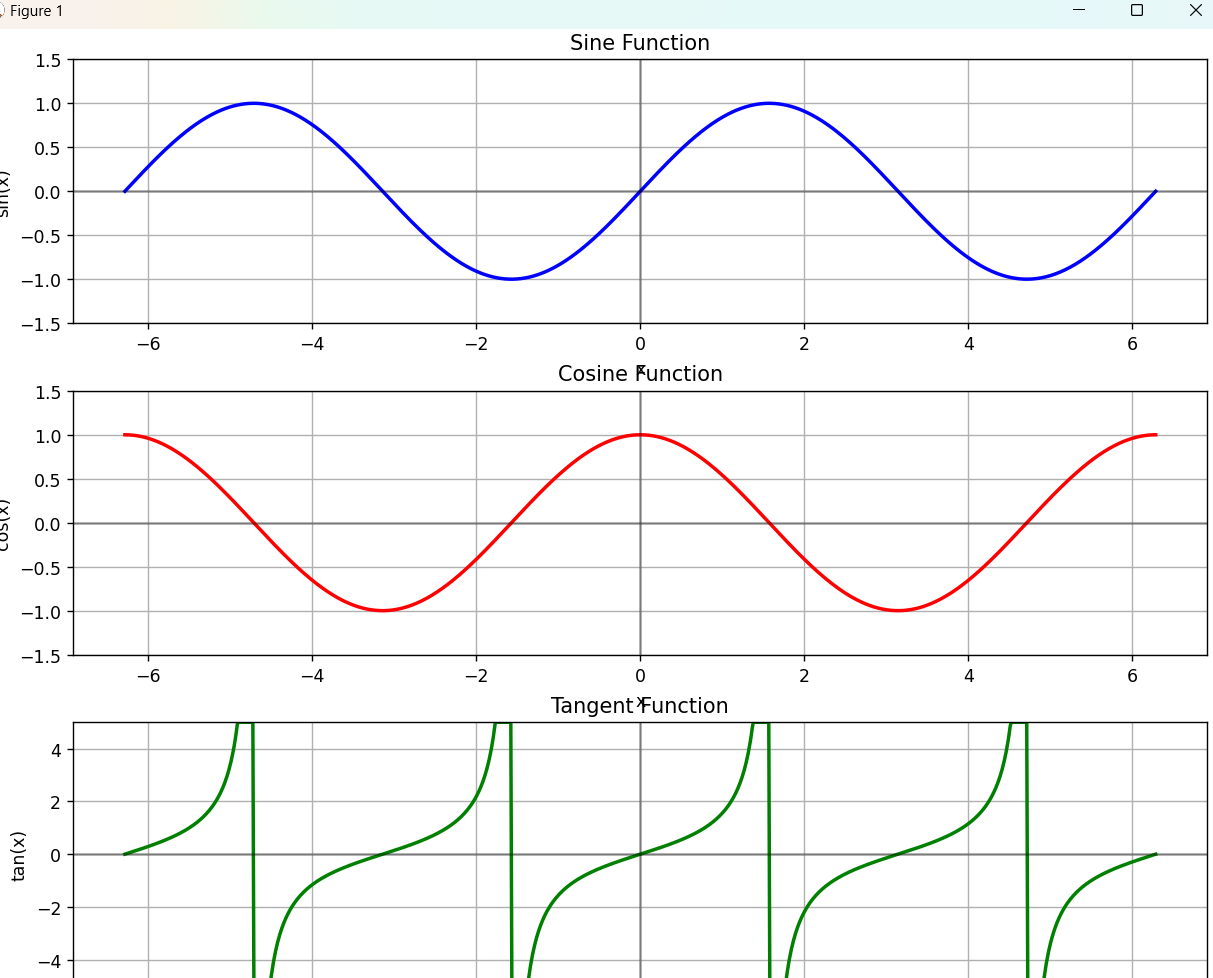
**A screen shot of a computer

Description automatically generated**

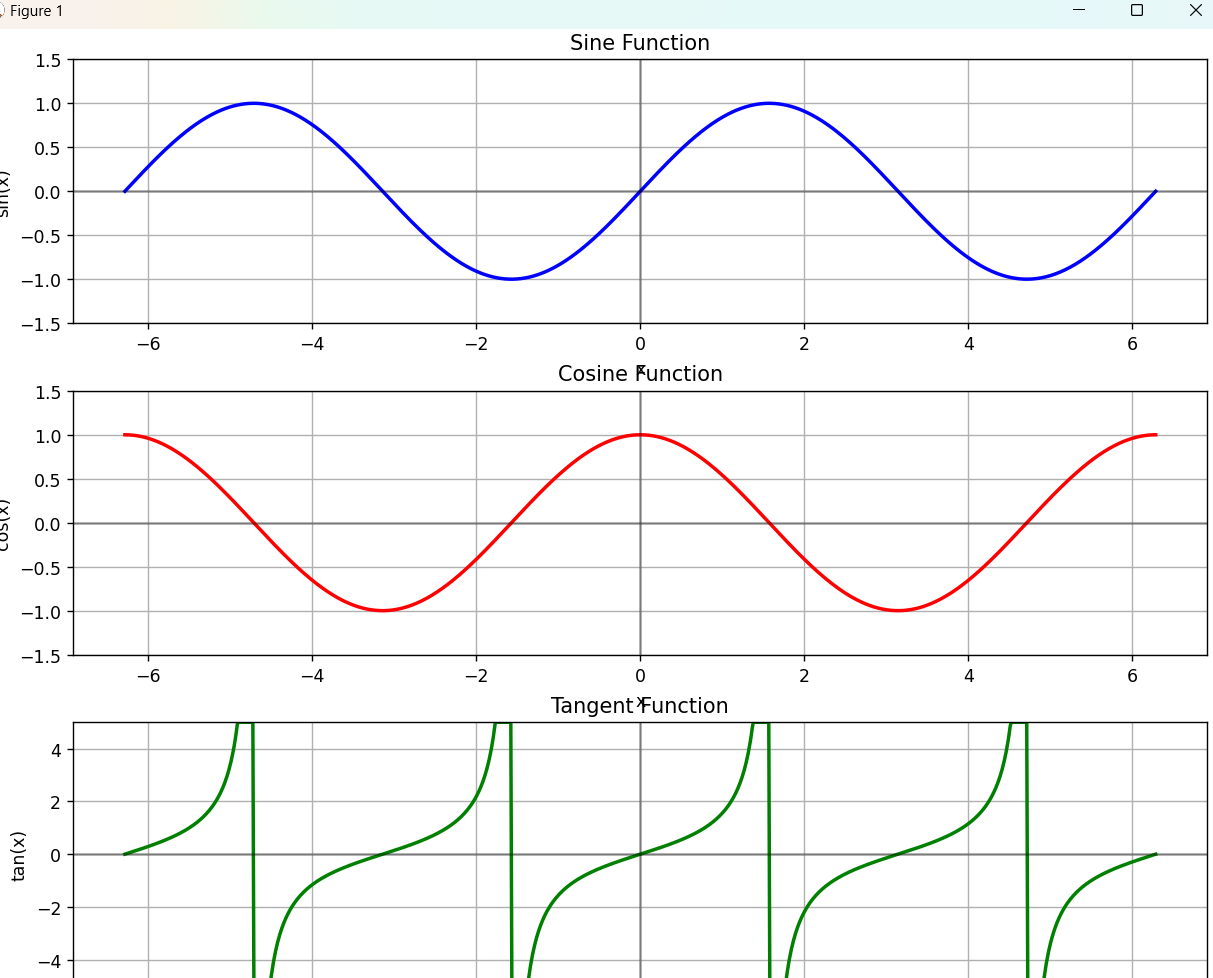
Pict Code for Sin, Cos, and Tan Function



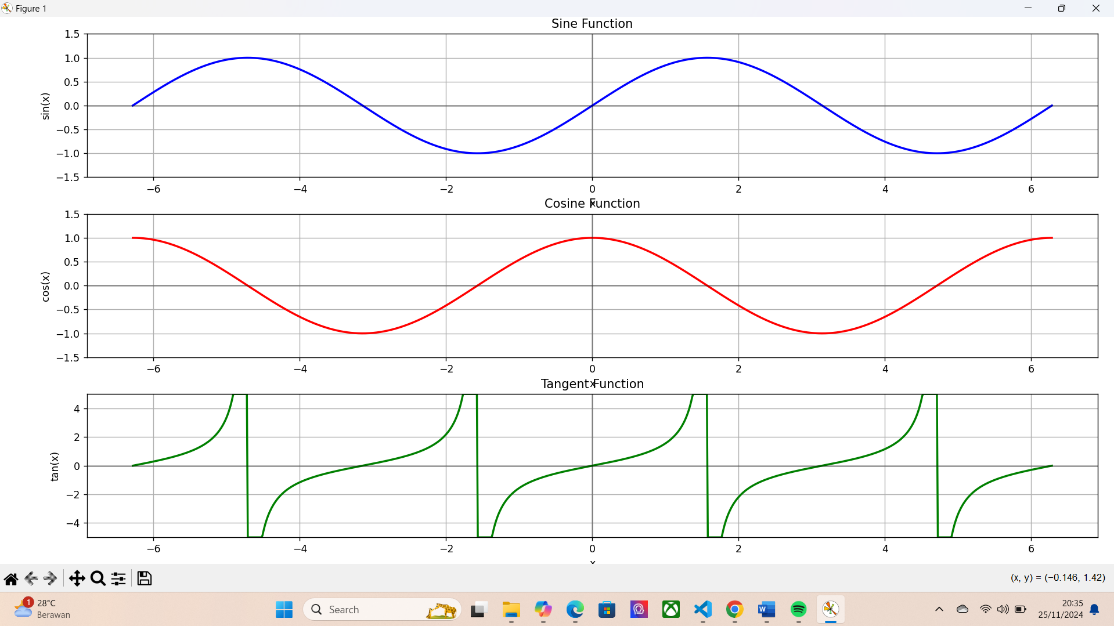
Pict Code for Combination of Sin, Cos, and Tan Function



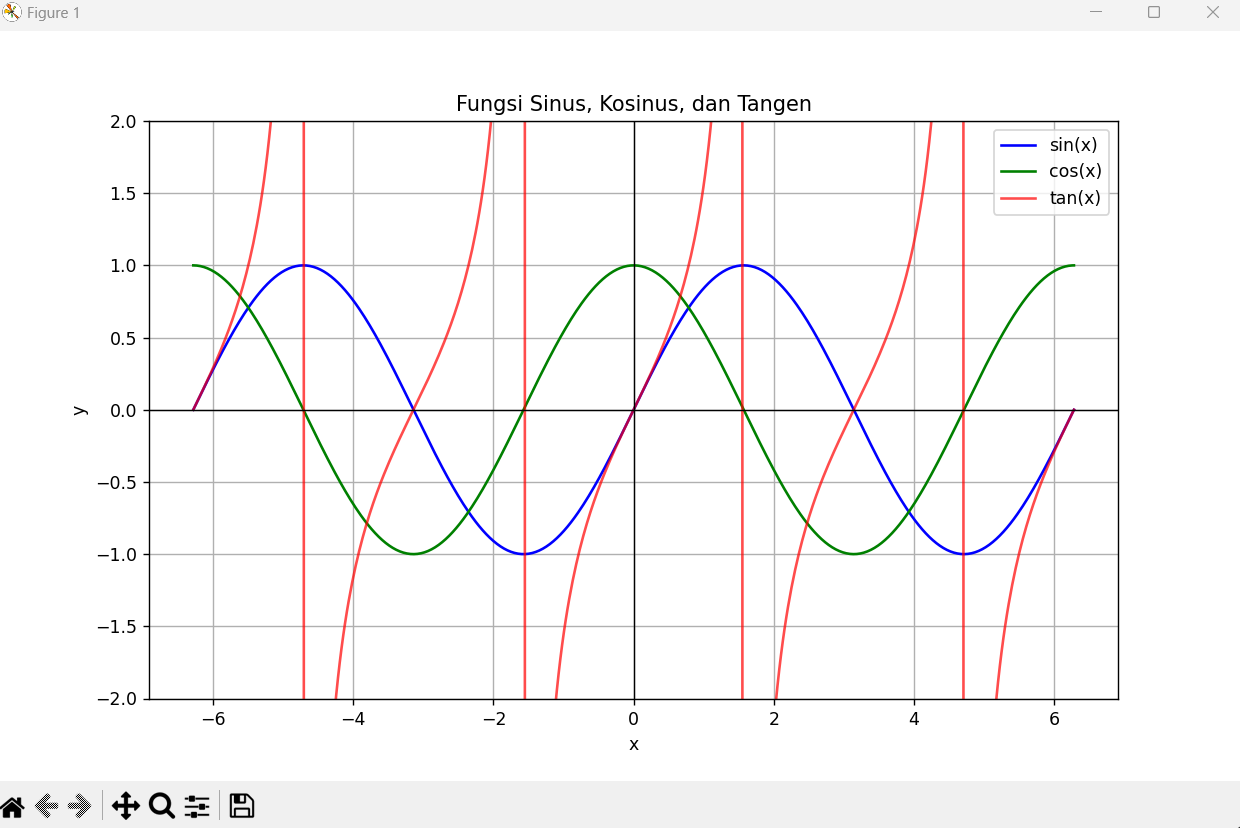
Pict The Result of Sin Function



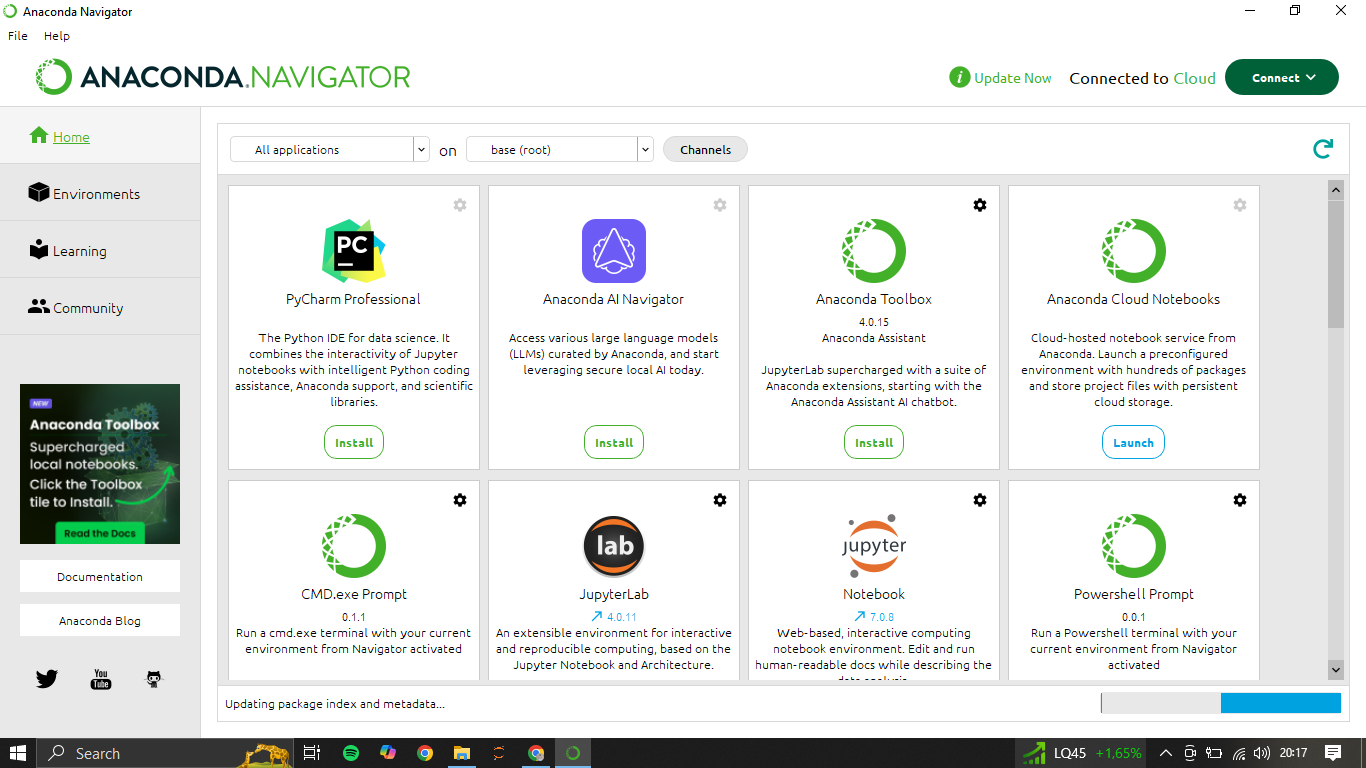
Pict The Result of Cos Function



Pict The Result of Tan Function



Pict The Result of The Combination of Sin, Cos, and Tan Function

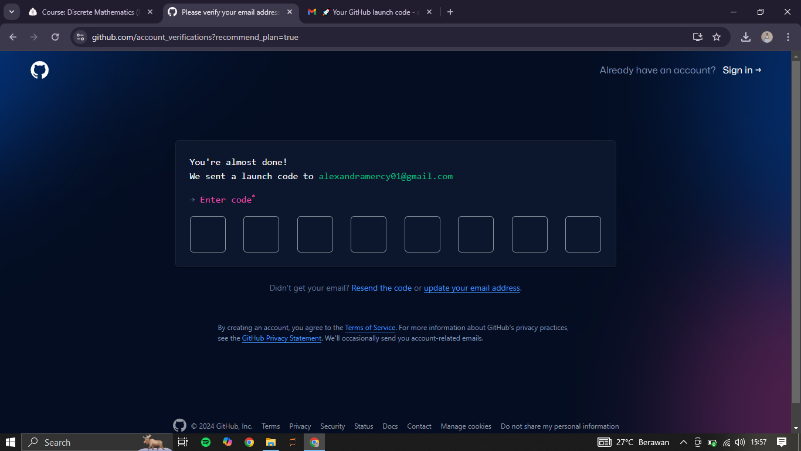
****

Pict Installed Anaconda

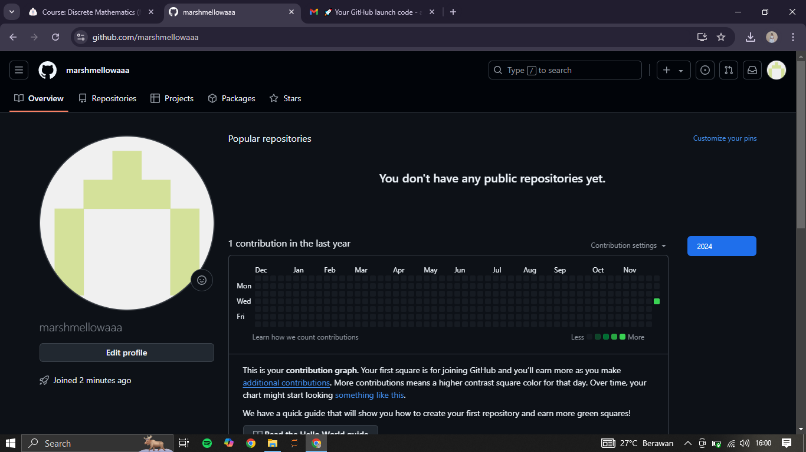
**A screenshot of a computer

Description automatically generated**

Pict Installed Studio Visual Code

****

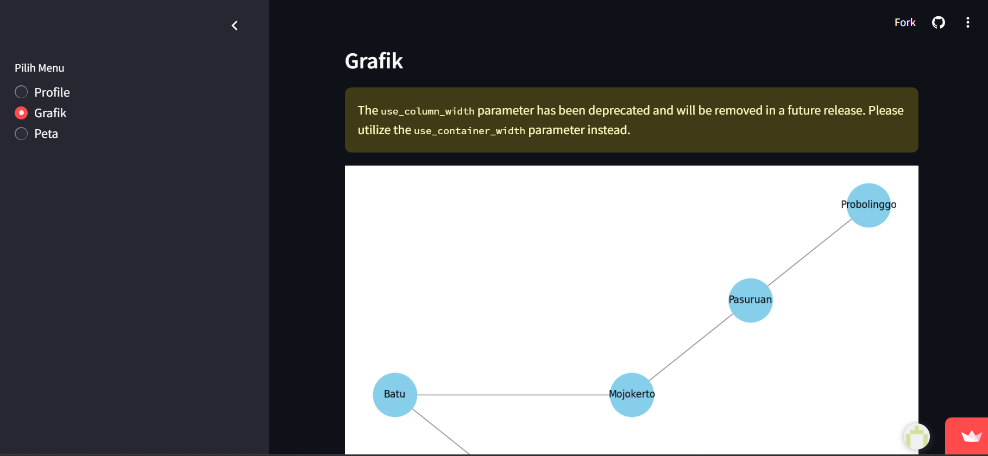
Pict Creating GitHub Account

****

Pict Registering GitHub

## REFERENCES

Pict 12 The First Menu of The Streamlit



Pict 13 The Second Menu of The Streamlit

# 

Pict 14 The Third Menu of The Streamlit

# 

## REFERENCES

Subardio Pri Anton. 2023. Menguasai Python: Membuka Masa Depan Pemrograman SIP Publishing. Indonesia

Chacon. Strain Ben. 2014 Pro Git The Expert Voice.

Maulid Reyvan. 2023. "Mengenal Streamlit, Tools Favorit Data Scientist" in <https://dqlab.id/mengenal-streamlit-tools-favorit-data-scientist>

Calculus Beyond. 2018. "Graph Theory,History & Overviewm Part I What Is Graph Theory & Why Is It Relevant Today? " in https://www.setzeus.com/community-blog-posts/graph-theory-history-overview