A Mini-Project Report on

"UI/UX CONTRIBUTION TO PUBLIC LABS AND CREATION OF GRAPH LIBRARY"

By

Manaswee Koli (Roll No. 5019128)
Aryan Koul (Roll No. 5019129)
Mithun Kuthully (Roll No. 5019131)
Jignesh Mathure (Roll No. 5019137)

Guided by:

Prof. Archana Shirke

Department of Information Technology
Fr. Conceicao Rodrigues Institute of Technology
Sector 9A, Vashi, Navi Mumbai – 400703

University of Mumbai
2022-2023

CERTIFICATE

This is to certify that the mini-project entitled

"UI/UX CONTRIBUTION TO PUBLIC LABS AND CREATION OF GRAPH LIBRARY"

Submitted By

Manaswee Koli
Aryan Koul
Mithun Kuthully
Jignesh Mathure

In partial fulfilment of degree of **B.E**. in **Information Technology** for term work of the mini-project is approved.

External Examiner **Internal Examiner**

_____ _____

Internal Guide

_____

Head of the Department

_____

Date: - **College Seal**

Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Manaswee Koli, 5019128

Aryan Koul, 5019129

Mithun Kuthully, 5019131

Jignesh Mathure, 5019137

Date:

Abstract

Contribution to open-source projects is very much in trend today, as it helps to contribute to public repositories, learn new things, get valid knowledge, and most importantly, it helps to increase network by interacting with fellow contributors. So, our project aims at contributing to open source in two ways, one by helping a repository made for beginners, and second by hosting a library on Python Package Index. This implementation has helped us in understanding how open source programs exactly work, and also submit contributions to them.

INDEX

List Of Figures

Chapter 1: Introduction

1.1 ABOUT OPEN SOURCE

Open-source software is computer software that is released under a license in which the copyright holder grants users the rights to use, study, change, and distribute the software and its source code to anyone and for any purpose. Open-source software may be developed in a collaborative public manner.

Some software has source code that only the person, team, or organization who created it and maintains exclusive control over who can modify it. People call this kind of software "proprietary" or "closed source" software.

Only the original authors of proprietary software can legally copy, inspect, and alter that software. And in order to use proprietary software, computer users must agree (usually by signing a license displayed the first time they run this software) that they will not do anything with the software that the software's authors have not expressly permitted. Microsoft Office and Adobe Photoshop are examples of proprietary software.

Open source software is different. Its authors make its source code available to others who would like to view that code, copy it, learn from it, alter it, or share it. LibreOffice and the GNU Image Manipulation Program are examples of open-source software.

As they do with proprietary software, users must accept the terms of a license when they use open source software—but the legal terms of open source licenses differ dramatically from those of proprietary licenses.

1.2 MOTIVATION

The first contribution was used to learn about open source projects, and how exactly we can contribute to them. The second contribution is published with the intention of providing programmers a library with the help of which they can visualize a graph by plotting it on the screen.

1.3 PROBLEM STATEMENT

Provide two open source contributions, the first being in a public repository on GitHub i.e. The website of Public Labs was not able to provide unique ids for comments table row and the second in the form of developing a library for graphs which is a non-linear data structure and plotting the graph, along with the adjacency matrix of the graph and hosting it on pypi.org.

1.4 SCOPE

The first contribution will fix bugs present in one of the Public Lab applications, and the second contribution will provide programmers a library for plotting graphs.

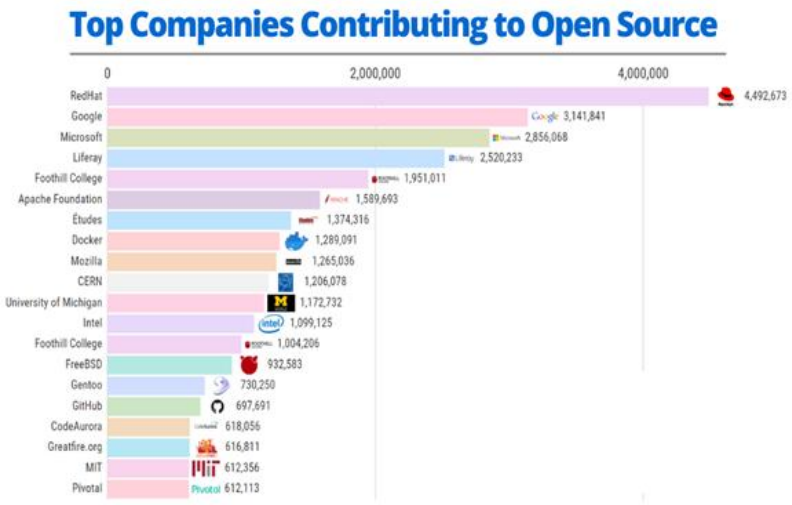


Fig 1.4.1 Open Source Contributions by corporations

Chapter 2: Literature Survey

2.1 THEORY

2.1.1. PUBLIC LAB

Public Lab is a community for DIY environmental investigation. It especially welcomes contributions from people and groups underrepresented in free and open source software. It has a welcoming "workflow" that supports you in getting started, which eventually helps you reach out to welcome someone else in turn. The Public Laboratory for Open Technology and Science (Public Lab) is a non-profit organization that facilitates collaborative, open source environmental research in a model known as Community Science It supports communities facing environmental justice issues in a do it yourself approach to environmental monitoring and advocacy Public Lab grew out of a grassroots effort to take aerial photograph of the BP Oil Spill in the Gulf of Mexico in 2010 Since then, they have launched a range of projects, including an open source spectrometer, multi-spectral camera, and low-cost microscope [1]. Public Lab's community develops open-source hardware, software, and other open methodologies to democratize environmental monitoring. Recognizing that cost, complexity, and lack of access can prevent communities from playing an active role in documenting environmental problems, the community publishes plans and guides for Do It Yourself monitoring projects.

2.1.2. GRAPHS

A graph is a non-linear data structure consisting of vertices (V) and edges (E). The most commonly used representations of a graph are adjacency matrix (a 2D array of size V x V where V is the number of vertices in a graph) and adjacency list (an array of lists represents the list of vertices adjacent to each vertex)

2.1.2.1 Types of Graphs

- **Undirected Graph**

  A graph in which edges do not have any direction. That is the nodes are unordered pairs in the definition of every edge.

- **Directed Graph**

  A graph in which the edge has direction. That is the nodes are ordered pairs in the definition of every edge.

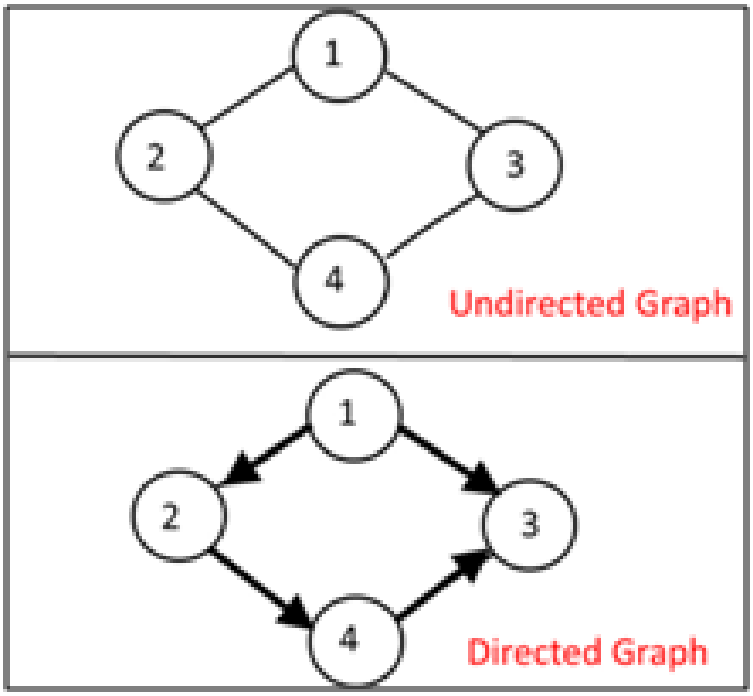


Fig. 2.1.1 Graph Types (A)

- **Connected Graph**

  The graph in which from one node we can visit any other node in the graph is known as a connected graph.

- **Disconnected Graph**

  The graph in which at least one node is not reachable from a node is known as a disconnected graph.

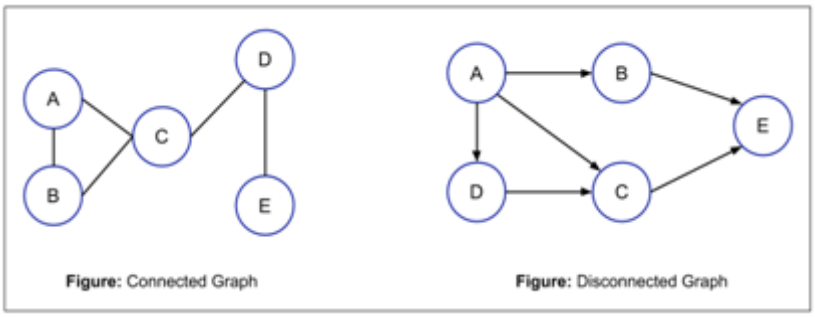


Fig. 2.1.2 Graph Types (B)

2.1.2.2 Usage of graphs:-

- Maps can be represented using graphs and then can be used by computers to provide various services like the shortest path between two cities.
- When various tasks depend on each other then this situation can be represented using a Directed Acyclic graph and we can find the order in which tasks can be performed using topological sort.
- State Transition Diagram represents what can be the legal moves from current states. In games like tic tac toe, this can be used.

2.1.2.3 COMPONENTS OF A GRAPH

- **Vertex:** A vertex is the most basic part of a graph and it is also called a node. Throughout we'll call it a note. A vertex may also have additional information and we'll call it a payload.

- **Edge:** An edge is another basic part of a graph, and it connects two vertices/ Edges may be one-way or two-way. If the edges in a graph are all one-way, the graph is a directed graph, or a digraph.

- **Weight:** Edges may be weighted to show that there is a cost to go from one vertex to another. For example in a graph of roads that connect one city to another, the weight on the edge might represent the distance between the two cities or traffic status.

- **Graph:** A graph can be represented by G where G=(V,E). V is a set of vertices and E is a set of edges. Each edge is a tuple (v,w) where w,v \in V. We can add a third component to the edge tuple to represent a weight.

- **Path:** A path in a graph is a sequence of vertices that are connected by edges. We usually define a path as w_1, w_2,..., w_n such that (w_i, w_{i+1}) in E for all 1 \le i \le n-1. The unweighted path length is the number of edges in the path (n-1). The weighted path length is the sum of the weights of all the edges in the path.

- **Cycle:** A cycle in a directed graph is a path that starts and ends at the same vertex. A graph with no cycles is called an acyclic graph. A directed graph with no cycles is called a directed acyclic graph or a DAG.

2.2 REFERRED WORK

Here are some published works we have analyzed for our project:-

- Paper [1] talks about what exactly open source is, and how licenses in open source software work. It also talks about the rules and regulations of open source software, and what are the advantages of an open source community.

- Paper [2] discusses the origin of open source, various applications of open source and how exactly this idea of a contribution based community for programmers became successful in the modern age of computers.

- Paper [3] exhibits why Public Lab was actually created, and how the contributions have actually helped communities in reality like aerial mapping, water quality monitoring and civic science practices in licensing of data.

- Paper [4] EasyGraph : EasyGraph is an open-source graph processing library. It is written in Python and supports analysis for undirected graphs and directed graphs. It covers advanced graph processing methods in structural hole spanners detection, graph embedding and several classic methods (subgraph generation, connected component discovery and isomorphic graph generation).

- Paper [5] Grafalgo - A Library of Graph Algorithms and Supporting Data Structures Grafalglo includes implementations of algorithms for a variety of classical graph optimization problems. These include the minimum spanning tree problem, the shortest path problem, the max flow problem, the min-cost flow problem, the graph matching problem and the edge colouring problem for bipartite graphs. The Grafalgo library uses index-based data structures. These are data structures in which the items of interest are represented by integers in a restricted range.

Chapter 3: Proposed System and Requirements

3.1 Proposed Solution

The project aims at contributing to open source in two ways, one by helping a repository made for beginners, and second by hosting a library on Python Package Index.

3.1.1 UI/UX Contribution to Public Labs

The website of Public Labs was not able to provide unique ids for comments table row. So we forked the repository and cloned the project into the local machine. Then we made changes in the comments.html.erb file and pushed changes to remote and made a pull request.

3.1.2 Creation of Graph Library

The library can create a graph which is a non- linear data structure using python language and also user can plot the graph by providing graph nodes and their connection with other nodes. The library is using "Matplotlib" to plot the graph. Also, the library creates the adjacency matrix of the graph.

3.2 Gantt Chart

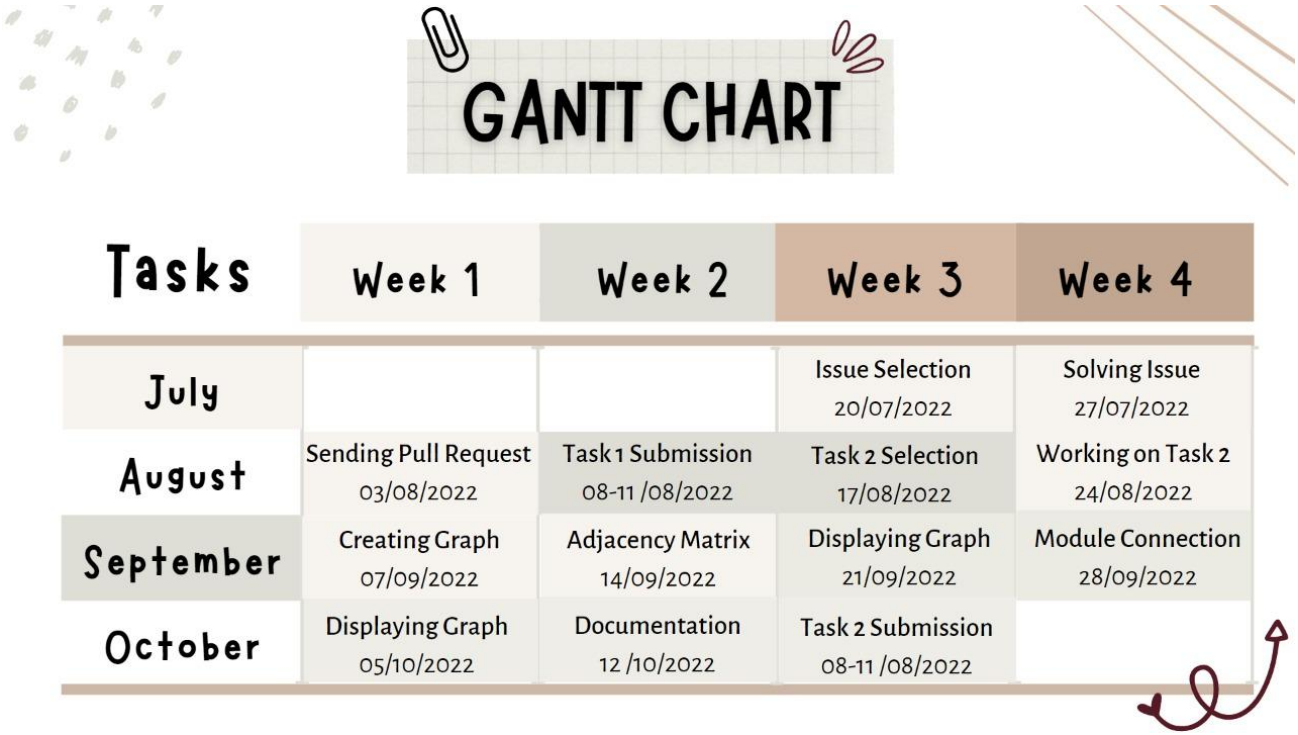


Fig. 3 Gantt Chart

3.3 Hardware and Software requirements

This section includes all the details of the minimum system requirements for the system to run smoothly.

Hardware requirements are as follows:

- Desktop/ Laptop
- AMD Ryzen 3 processor
- 4 GB of disk space

Software requirements are as follows:

- Windows OS
- Python 3.6
- Editor – VSCode

Chapter 4: Implementation Details

4.1 UI/UX Contribution to Public Labs

The first contribution is related to the field of UI/UX, where unique IDs were added to comment table rows, which helped to fix the documentation section of the code.

Steps taken:

1. Research

Contributing to open source happens at all levels, across projects. As a beginner, you could identify and fix bugs in a project. For example, the Angular project accepts issues pointing out bugs and even pull requests that fix them. Larger features will require greater community involvement, and some communities require you to earn a reputation fixing bugs before moving to feature development. Your contributions do not have to be exclusive to code. You can help a project by making comments on existing code and APIs to add context and writing documentation for the project.

The resources to help us discover and contribute to new projects are as follows:

- GitHub Explore
- Open-Source Friday
- First Timers Only

The following are steps to follow:

- Explore projects familiar to your domain
- Check for project inactivity
- Learn contribution tools
- Read the guidelines
- Ask for contribution permission.

2. Submit yours contribute to open source.

- Fork the repository
- Clone the project into your local machine
- Add your changes
- Push changes to remote
- Submit changes aka pull request
- Sooner or later maintainer will merge all your changes into the master branch of the project; unless they need changes from you. You will get a notification email once the changes have been merged.

3. Results after submitting your contribution to Open Source.

- Once a collaborator has contributed a piece of code that passes the submission guidelines the code is reviewed in detail.
- Usually, the contribution be it in code or documentation or some other type of contribution is reviewed in chunks
- At times, your submission may not get a reply even after a reasonable amount of time has gone by. In such cases, respectfully request a review or get in touch with other contributors for assistance.
- Post-review changes may be requested.
- Try to make them as soon as possible so that your contribution is integrated promptly and does not become out-of-date or forgotten.

- If your contribution is rejected, ask for feedback to understand why. When reviewers ask questions, make comments, or give feedback, be responsive and check on your work for any updates regularly.

4.2 Creation of Graph Library

The second contribution helps users to create a graph which is a non linear data structure and also user can plot the graph by providing certain inputs. The library also supports the adjacency matrix of the graph.

Steps taken:

Step 1: Write the code of library to create the graph, to plot the graph using matplotlib and create the adjacency matrix.

Step 2: Create account on pypi.org

Step 3: Host the project on your account.

Step 4: Download the Library from the PyPi Website using the “pip install” command.

Step 5: As per the format in the documentation, enter the number of nodes.

Step 6: Specify the connection between the nodes

Step 7: Assign a weight to each connection.

Step 8: Display the Graph using the library

Step 9: Create the Adjacency matrix

Chapter 5: Result

Result

5.1 UI/UX Contribution to Public Labs

- Code:

```
<% comments.each do |comment| %>
 <tr id="c<% comment.cid %>">
 <tr id="c<%= comment.cid %>">
    <td>
   <a href="/profile/<%= comment.author.username %>">
    <%=raw strip_tags(sanitize(RDiscount.new(comment.author.username).to_html)).truncate(150) %></a>
      <a href="/profile/<%= comment.author.username %>">
    <%=raw strip_tags(sanitize(RDiscount.new(comment.author.username).to_html)).truncate(150) %></a>
    </td>
   <td>
     <% if comment.comment_via == 1 %>
```

Fig. 5.1.1 Code Contribution

- Contribution

Fig. 5.1.2 Code Publish Request

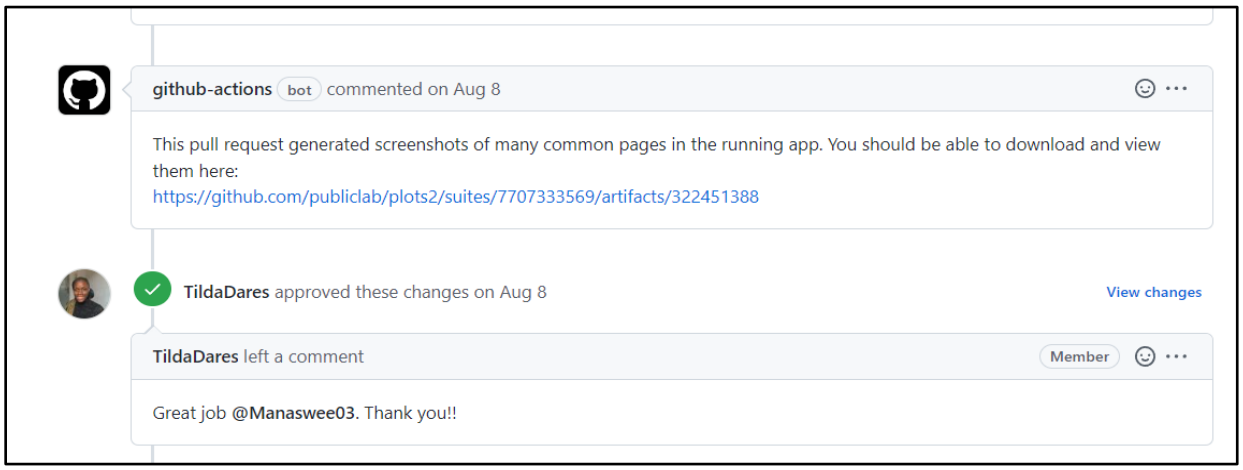


Fig. 5.1.3 Approval of Code

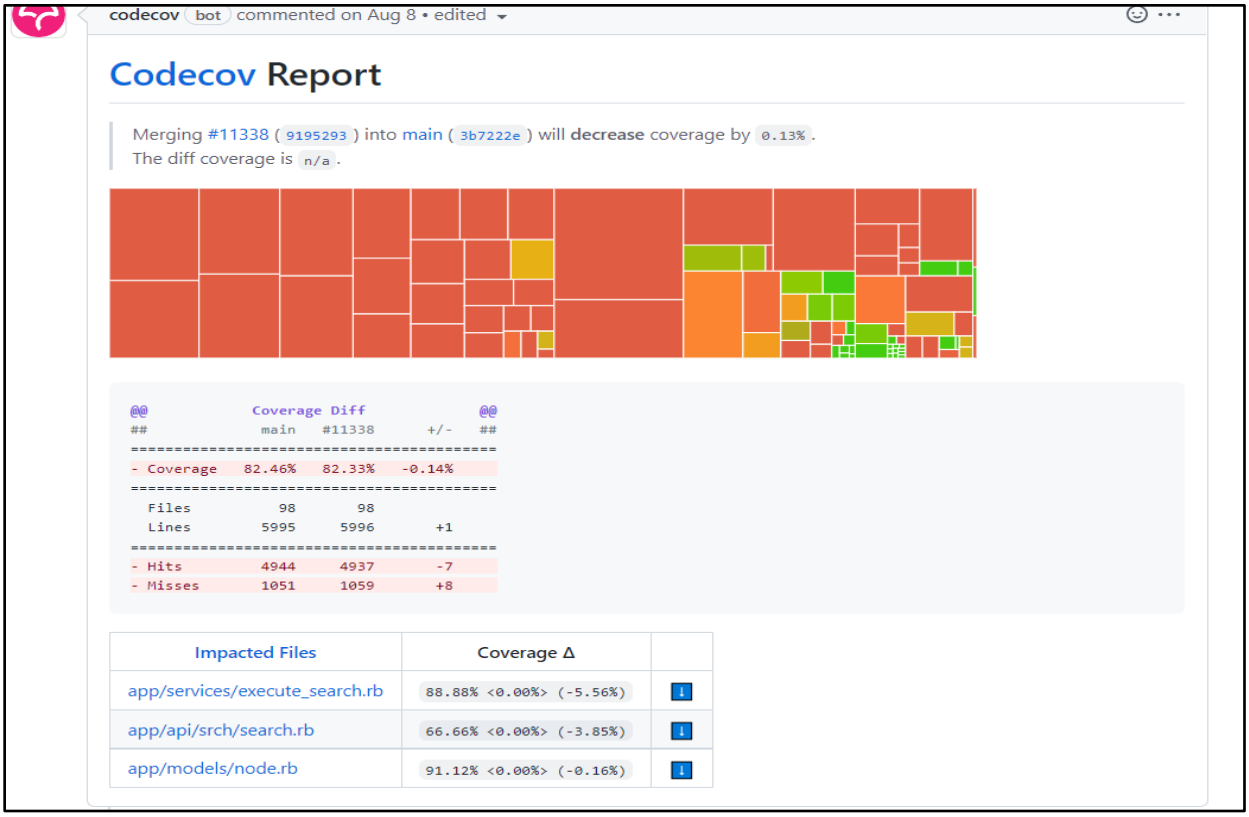


Fig. 5.1.4 Analysis of Code

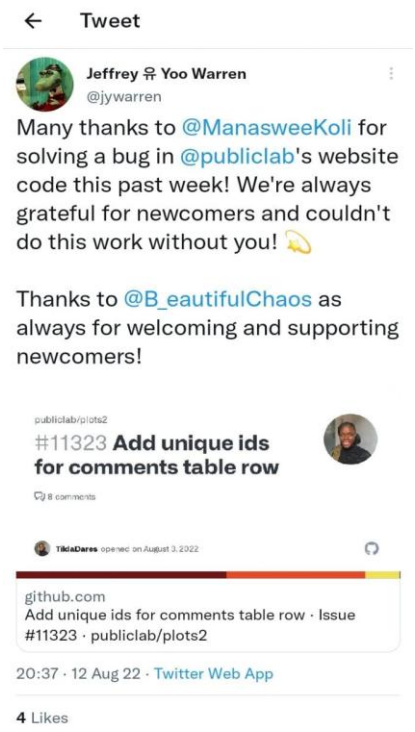


Fig. 5.1.5 Acknowledge

5.2 Creation of Graph Library

- Write the code of library to create the graph, to plot the graph using matplotlib and create the adjacency matrix.

```
#******************Graph*************************
class Graph:
    nodal1 = []
    nodal2 = []

    def __init__(self, num_of_nodes, directed=True):
        self.num_of_nodes = num_of_nodes
        self.directed = directed

        # Different representations of a graph
        self.list_of_edges = []

    # Add edge to a graph
    def add_edge(self, node1, node2, weight=1):
        self.nodal1.append(node1)
        self.nodal2.append(node2)

        # Add the edge from node1 to node2
        self.list_of_edges.append([node1, node2, weight])

        # If a graph is undirected, add the same edge,
        # but also in the opposite direction
        if not self.directed:
            self.list_of_edges.append([node1, node2, weight])

    # Print a graph representation
    def print_edge_list(self):
        num_of_edges = len(self.list_of_edges)
        print("Graph:")
        for i in range(num_of_edges):
            print("edge ", i+1, ": ", self.list_of_edges[i])


#******************Adjacency Matrix*****************************
class Adj_matrix:
    def __init__(self, num_of_nodes, directed=True):
        self.num_of_nodes = num_of_nodes
        self.directed = directed

        # Initialize the adjacency matrix
        # Create a matrix with `num_of_nodes` rows and columns
        self.adj_matrix = [[0 for column in range(num_of_nodes)]
                           for row in range(num_of_nodes)]

    def add_edge_adj_matrix(self, node1, node2, weight=1):
        self.adj_matrix[node1][node2] = weight

        if not self.directed:
            self.adj_matrix[node2][node1] = weight

    def print_adj_matrix(self):
        print("Adjaceny Matrix:")
        for i in range(0,len(self.adj_matrix)):
            print(self.adj_matrix[i])
```

- Create account on pypi.org

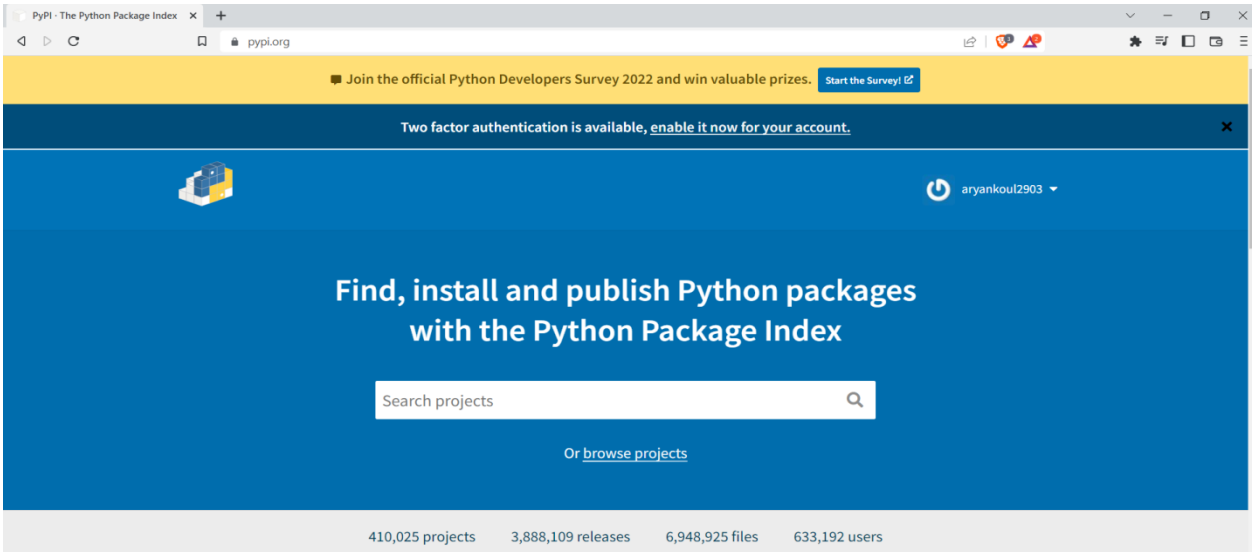


Fig. 5.2.1 Pypi Homepage

- Host the project on your account.

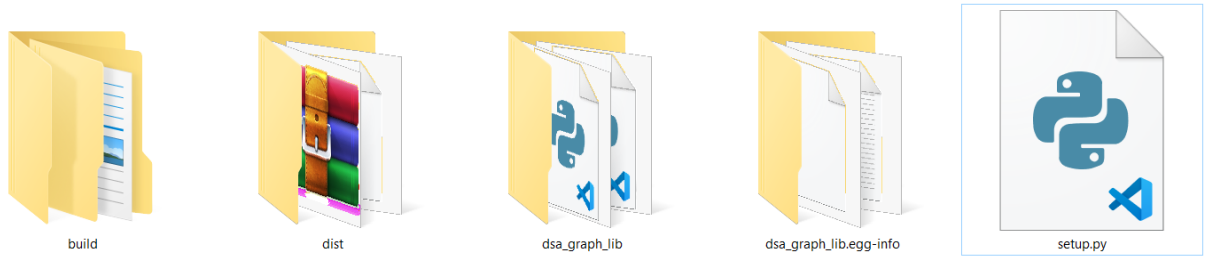
```
PS C:\Users\ARYAN KOUL\Desktop\ROSPL> pip3 install setuptools
Requirement already satisfied: setuptools in c:\users\aryan koul\appdata\local\programs\python\python38\lib\site-packages (65.5.0)
```

```
PS C:\Users\ARYAN KOUL\Desktop\ROSPL> python setup.py sdist bdist_wheel
running sdist
running egg_info
creating dsa_graph_lib.egg-info
writing dsa_graph_lib.egg-info\PKG-INFO
writing dependency_links to dsa_graph_lib.egg-info\dependency_links.txt
writing requirements to dsa_graph_lib.egg-info\requires.txt
writing top-level names to dsa_graph_lib.egg-info\top_level.txt
writing manifest file 'dsa_graph_lib.egg-info\SOURCES.txt'
reading manifest file 'dsa_graph_lib.egg-info\SOURCES.txt'
writing manifest file 'dsa_graph_lib.egg-info\SOURCES.txt'
warning: sdist: standard file not found: should have one of README, README.rst, README.txt, README.md

running check
creating dsa_graph_lib-0.0.1
creating dsa_graph_lib-0.0.1\dsa_graph_lib
creating dsa_graph_lib-0.0.1\dsa_graph_lib.egg-info
copying files to dsa_graph_lib-0.0.1...
copying setup.py -> dsa_graph_lib-0.0.1
copying dsa_graph_lib\__init__.py -> dsa_graph_lib-0.0.1\dsa_graph_lib
copying dsa_graph_lib\dsa_graph_lib.py -> dsa_graph_lib-0.0.1\dsa_graph_lib
copying dsa_graph_lib.egg-info\PKG-INFO -> dsa_graph_lib-0.0.1\dsa_graph_lib.egg-info
copying dsa_graph_lib.egg-info\SOURCES.txt -> dsa_graph_lib-0.0.1\dsa_graph_lib.egg-info
copying dsa_graph_lib.egg-info\dependency_links.txt -> dsa_graph_lib-0.0.1\dsa_graph_lib.egg-info
copying dsa_graph_lib.egg-info\requires.txt -> dsa_graph_lib-0.0.1\dsa_graph_lib.egg-info
copying dsa_graph_lib.egg-info\top_level.txt -> dsa_graph_lib-0.0.1\dsa_graph_lib.egg-info
Writing dsa_graph_lib-0.0.1\setup.cfg
creating dist
Creating tar archive
removing 'dsa_graph_lib-0.0.1' (and everything under it)
running bdist_wheel
running build
running build_py
creating build
creating build\lib
creating build\lib\dsa_graph_lib
copying dsa_graph_lib\dsa_graph_lib.py -> build\lib\dsa_graph_lib
copying dsa_graph_lib\__init__.py -> build\lib\dsa_graph_lib
C:\Users\ARYAN KOUL\AppData\Local\Programs\Python\Python38\lib\site-packages\setuptools\command\install.py:34: SetuptoolsDeprecationWarning: setup.py install i
s deprecated. Use build and pip and other standards-based tools.
```

Fig. 5.2.2 Code Command Line

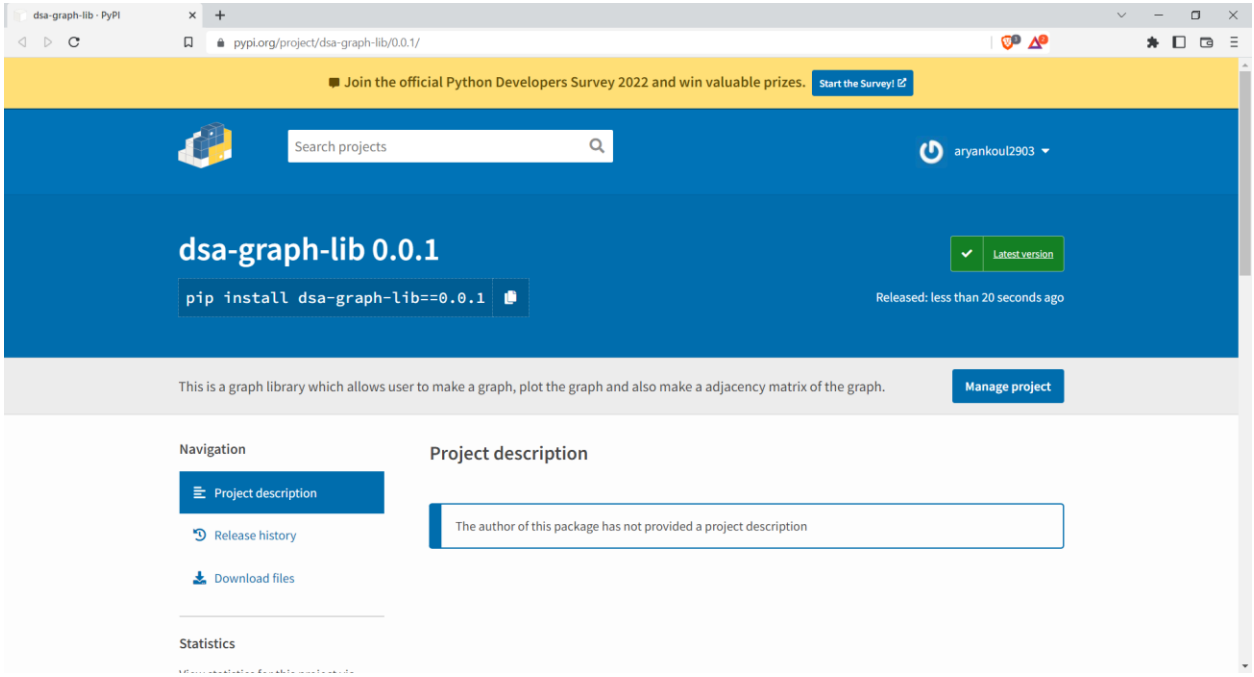
```
  warnings.warn(
installing to build\bdist.win-amd64\wheel
running install
running install_lib
creating build\bdist.win-amd64
creating build\bdist.win-amd64\wheel
creating build\bdist.win-amd64\wheel\dsa_graph_lib
copying build\lib\dsa_graph_lib\dsa_graph_lib.py -> build\bdist.win-amd64\wheel\.\dsa_graph_lib
copying build\lib\dsa_graph_lib\__init__.py -> build\bdist.win-amd64\wheel\.\dsa_graph_lib
running install_egg_info
Copying dsa_graph_lib.egg-info to build\bdist.win-amd64\wheel\.\dsa_graph_lib-0.0.1-py3.8.egg-info
running install_scripts
creating build\bdist.win-amd64\wheel\dsa_graph_lib-0.0.1.dist-info\WHEEL
creating 'dist\dsa_graph_lib-0.0.1-py3-none-any.whl' and adding 'build\bdist.win-amd64\wheel' to it
adding 'dsa_graph_lib/__init__.py'
adding 'dsa_graph_lib/dsa_graph_lib.py'
adding 'dsa_graph_lib-0.0.1.dist-info/METADATA'
adding 'dsa_graph_lib-0.0.1.dist-info/WHEEL'
adding 'dsa_graph_lib-0.0.1.dist-info/top_level.txt'
adding 'dsa_graph_lib-0.0.1.dist-info/RECORD'
removing build\bdist.win-amd64\wheel
```




```
PS C:\Users\ARYAN KOUL\Desktop\ROSPL> pip3 install twine
Collecting twine
  Downloading twine-4.0.1-py3-none-any.whl (36 kB)
Collecting requests-toolbelt!=0.9.0,>=0.8.0
  Downloading requests_toolbelt-0.10.1-py2.py3-none-any.whl (54 kB)
     ---------------------------------------- 54.5/54.5 kB 1.4 MB/s eta 0:00:00
Collecting importlib-metadata>=3.6
  Downloading importlib_metadata-5.0.0-py3-none-any.whl (21 kB)
Collecting rfc3986>=1.4.0
  Downloading rfc3986-2.0.0-py2.py3-none-any.whl (31 kB)
Collecting pkginfo>=1.8.1
  Downloading pkginfo-1.8.3-py2.py3-none-any.whl (26 kB)
Collecting readme-renderer>=35.0
  Downloading readme_renderer-37.2-py3-none-any.whl (14 kB)
Collecting keyring>=15.1
  Downloading keyring-23.9.3-py3-none-any.whl (35 kB)
Requirement already satisfied: urllib3>=1.26.0 in c:\users\aryan koul\appdata\local\programs\python\python38\lib\site-packages (from twine) (1.26.6)
Requirement already satisfied: requests>=2.20 in c:\users\aryan koul\appdata\local\programs\python\python38\lib\site-packages (from twine) (2.27.1)
Collecting rich>=12.0.0
  Downloading rich-12.6.0-py3-none-any.whl (237 kB)
     ---------------------------------------- 237.5/237.5 kB 2.1 MB/s eta 0:00:00
Requirement already satisfied: zipp>=0.5 in c:\users\aryan koul\appdata\local\programs\python\python38\lib\site-packages (from importlib-metadata>=3.6->twine
(3.5.0)
Collecting pywin32-ctypes!=0.1.0,!=0.1.1
  Downloading pywin32_ctypes-0.2.0-py2.py3-none-any.whl (28 kB)
Collecting jaraco.classes
Collecting bleach>=2.1.0
  Downloading bleach-5.0.1-py3-none-any.whl (160 kB)
     ---------------------------------------- 160.9/160.9 kB 1.1 MB/s eta 0:00:00
Requirement already satisfied: idna<4,>=2.5 in c:\users\aryan koul\appdata\local\programs\python\python38\lib\site-packages (from requests>=2.20->twine) (2.1
Requirement already satisfied: charset-normalizer~=2.0.0 in c:\users\aryan koul\appdata\local\programs\python\python38\lib\site-packages (from requests>=2.20
twine) (2.0.12)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\aryan koul\appdata\local\programs\python\python38\lib\site-packages (from requests>=2.20->twine
(2021.5.30)
Collecting typing-extensions<5.0,>=4.0.0
  Downloading typing_extensions-4.4.0-py3-none-any.whl (26 kB)
  Using cached webencodings-0.5.1-py2.py3-none-any.whl (11 kB)
```

```
PS C:\Users\ARYAN KOUL\Desktop\ROSPL> twine upload dist/*
Uploading distributions to https://upload.pypi.org/legacy/
Enter your username: aryankoul2903
Enter your password:
Uploading dsa_graph_lib-0.0.1-py3-none-any.whl
100% ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6.5/6.5 kB • 00:01 • ?
Uploading dsa_graph_lib-0.0.1.tar.gz
100% ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 5.9/5.9 kB • 00:00 • ?

View at:
https://pypi.org/project/dsa-graph-lib/0.0.1/
PS C:\Users\ARYAN KOUL\Desktop\ROSPL>
```



- Download the Library from the PyPi Website using the “pip install” command.

```
PS C:\Users\ARYAN KOUL> pip install dsa-graph-lib
Collecting dsa-graph-lib
  Using cached dsa_graph_lib-0.0.2-py3-none-any.whl (2.8 kB)
Requirement already satisfied: matplotlib in c:\users\aryan koul\appdata\local\programs\python\python38\lib\site-packages (from dsa-graph-lib) (3.4.1)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\aryan koul\appdata\local\programs\python\python38\lib\site-packages (from matplotlib->dsa-graph-lib) (2.4.7)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\aryan koul\appdata\local\programs\python\python38\lib\site-packages (from matplotlib->dsa-graph-lib) (2.8.1)
Requirement already satisfied: pillow>=6.2.0 in c:\users\aryan koul\appdata\local\programs\python\python38\lib\site-packages (from matplotlib->dsa-graph-lib) (8.3.0)
Requirement already satisfied: numpy>=1.16 in c:\users\aryan koul\appdata\local\programs\python\python38\lib\site-packages (from matplotlib->dsa-graph-lib) (1.20.2)
Requirement already satisfied: cycler>=0.10 in c:\users\aryan koul\appdata\local\programs\python\python38\lib\site-packages (from matplotlib->dsa-graph-lib) (0.10.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\aryan koul\appdata\local\programs\python\python38\lib\site-packages (from matplotlib->dsa-graph-lib) (1.3.1)
Requirement already satisfied: six in c:\users\aryan koul\appdata\local\programs\python\python38\lib\site-packages (from cycler>=0.10->matplotlib->dsa-graph-lib) (1.16.0)
Installing collected packages: dsa-graph-lib
Successfully installed dsa-graph-lib-0.0.2
PS C:\Users\ARYAN KOUL>
```

- As per the format in the documentation, enter the number of nodes.

- Specify the connection between the nodes

- Assign a weight to each connection.

```
Graph:
edge  1 :  [0, 1, 25]
edge  2 :  [1, 2, 5]
edge  3 :  [2, 3, 3]
edge  4 :  [3, 4, 1]
edge  5 :  [4, 0, 15]
```

Fig. 5.2.3 Assignment in Graph

- Display the Graph using the library

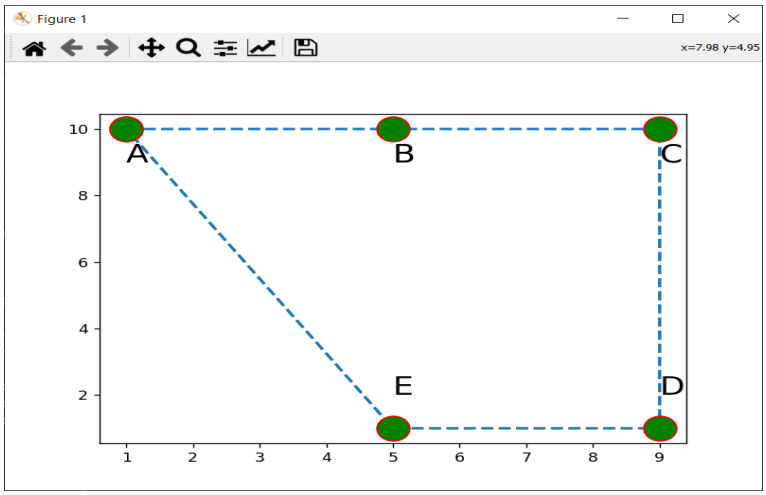


Fig. 5.2.4 Graph Visualization

- Create the Adjacency matrix

```
Adjaceny Matrix:
[0, 25, 0, 0, 0]
[0, 0, 5, 0, 0]
[0, 0, 0, 3, 0]
[0, 0, 0, 0, 1]
[15, 0, 0, 0, 0]
```

Fig. 5.2.5 Adjacency Matrix

Chapter 6: Conclusion

Conclusion

We have contributed to two open-source projects, one being the UI/UX contribution on Public Lab, and second being the hosting of Graph Library on PyPi.

We are able to provide unique ids to the website of Public Labs for comments table row. We forked the repository and cloned the project into the local machine. Then we made changes in the comments.html.erb file and pushed changes to remote and made a pull request. Our pull request was successfully merged into the master branch of the repository.

Our package is developed and uploaded successfully on PyPi.org. It can be installed on any python supported device using the pip command. The command to install is “pip install dsa-graph-lib”. Our package contains Graphs, Graph plotting and Adjacency Matrix. This package can be used for creating the graph, plotting the graph and creating the adjacency matrix of the graph.

Chapter 7: References

References

[1] Perens, Bruce. "The open source definition." Open sources: voices from the open source revolution 1 (1999): 171-188.

[2] Weber, Steven. The success of open source. Harvard University Press, 2004.

[3] Rey-Mazón, Pablo, et al. "Public lab: Community-based approaches to urban and environmental health and justice." Science and Engineering Ethics 24.3 (2018): 971-997.

[4] Zheng Li, Min Gao, EasyGraph, 2020

[5] Jonathan Turner, Grafalgo - A Library of Graph Algorithms and Supporting Data Structures (revised), 2016

[6] Public Lab: a DIY environmental science community. Retrieved October 28, 2022, from https://publiclab.org/

[7] PyPI. Retrieved October 28, 2022, from https://pypi.org/

[8] GitHub. Retrieved October 28, 2022, from https://github.com/

[9] Finding ways to contribute to open source on GitHub. GitHub Retrieved October 28, 2022, from shorturl.at/hAG07

[10] Twitter. Retrieved October 28, 2022, from https://en.wikipedia.org/wiki/Twitter

Chapter 8: Appendix A

Appendix: Sample Code

Graph Package:

```
from os import environ

environ["QT_DEVICE_PIXEL_RATIO"] = "0"
environ["QT_AUTO_SCREEN_SCALE_FACTOR"] = "1"
environ["QT_SCREEN_SCALE_FACTORS"] = "1"
environ["QT_SCALE_FACTOR"] = "1"


#******************Graph*************************
class Graph:
    def __init__(self, num_of_nodes, directed=True):
        self.num_of_nodes = num_of_nodes
        self.directed = directed

        # Different representations of a graph
        self.list_of_edges = []

    # Add edge to a graph
    def add_edge(self, node1, node2, weight=1):
        # Add the edge from node1 to node2
        self.list_of_edges.append([node1, node2, weight])

        # If a graph is undirected, add the same edge,
        # but also in the opposite direction
        if not self.directed:
            self.list_of_edges.append([node1, node2, weight])

            # Print a graph representation
    def print_edge_list(self):
        num_of_edges = len(self.list_of_edges)
        print("Graph:")
        for i in range(num_of_edges):
            print("edge ", i+1, ": ", self.list_of_edges[i])

#******************Graph Plotting*******************************
import matplotlib.pyplot as plt
```

```
class GraphPlot:
    def plotGraph(self, n):
        e = n/2
        f = n//2

        if(e == f):
            l = f

        else:
            l = int(e) + 1

        x = 1
        y = 0
        node1 = []
        node2 = []
        x_values = []
        y_values = []

        for i in range(0, l):
            point1 = [x, 10]
            node1.append(point1)
            if(x>y):
                y = x
            x += 4

        for i in range(l, n):
            point2 = [y, 1]
            node2.append(point2)
            y -= 4
        node2.append([1,10])

        f_node = node1 + node2

        for i in range(0, n+1):
            x_values.append(f_node[i][0])
            y_values.append(f_node[i][1])

        text = []
        for i in range(65, 65+n):
```

```
            text.append(chr(i))

        for i in range(0,len(x_values)-1):
            if(y_values[i]==10):
                plt.text(x_values[i], y_values[i]-1, text[i], fontsize = 20)
            else:
                plt.text(x_values[i], y_values[i]+1, text[i], fontsize = 20)

        plt.plot(x_values, y_values, marker='o', markersize=20, markeredgecolor="red",
markerfacecolor="green", linestyle="dashed", linewidth=2)
        plt.show()

#********************Adjacency Matrix*****************************
class Adj_matrix:
    def __init__(self, num_of_nodes, directed=True):
        self.num_of_nodes = num_of_nodes
        self.directed = directed

        # Initialize the adjacency matrix
        # Create a matrix with `num_of_nodes` rows and columns
        self.adj_matrix = [[0 for column in range(num_of_nodes)]
                           for row in range(num_of_nodes)]

    def add_edge_adj_matrix(self, node1, node2, weight=1):
        self.adj_matrix[node1][node2] = weight

        if not self.directed:
            self.adj_matrix[node2][node1] = weight

    def print_adj_matrix(self):
        print("Adjaceny Matrix:")
        for i in range(0,len(self.adj_matrix)):
            print(self.adj_matrix[i])
```

Testing the Package:

```
from dsa_graph_lib import *

print()
```

```
n = int(input("Enter the number of nodes you want to enter: "))

#Graph
graph = Graph(n)

graph.add_edge(0, 1, 25)
graph.add_edge(1, 2, 5)
graph.add_edge(2, 3, 3)
graph.add_edge(3, 4, 1)
graph.add_edge(4, 0, 15)

print()
graph.print_edge_list()

#Plotting the graph
pltgraph = GraphPlot()
pltgraph.plotGraph(n)

# #Adjacency Matrix
adj_matrix = Adj_matrix(n)

adj_matrix.add_edge_adj_matrix(0, 1, 25)
adj_matrix.add_edge_adj_matrix(1, 2, 5)
adj_matrix.add_edge_adj_matrix(2, 3, 3)
adj_matrix.add_edge_adj_matrix(3, 4, 1)
adj_matrix.add_edge_adj_matrix(4, 0, 15)

print()
adj_matrix.print_adj_matrix()
```

Chapter 9: Acknowledgement

ACKNOWLEDGMENT

Constant Motivation, guidance and inspiration have always played a key role in success of any project. We are also highly indebted to Fr. Conceicao Rodrigues Institute of Technology, Vashi and our beloved **principal Dr.S.M Khot** for providing necessary tools, information regarding the subject also for their support in completing the project. Also, we express our deepest gratitude to our guide for their constant support, motivation, supervision and guidance. We would also like to extend our gratitude to our **HOD Dr. Vaishali Bodade** for her constant support and supervision.

For us this project was a learning experience as this was a whole new thing for us. We, would like to extend special thanks to our guide **Prof. Archana Shirke** without whom we wouldn't have selected this topic for our project. She motivated us to go ahead with this topic as our project as it is something out of the blue. Also, we would like to express our sincere heartfelt gratitude to all the staff members of IT Engineering Department who helped us directly or indirectly during this course of work. Lastly, we would like to thank our parents and siblings who remained an important source of both inspiration and support.

Yours sincerely,

MANASWEE KOLI (5019128)

ARYAN KOUL (5019129)

MITHUN KUTHULLY (5019131)

JIGNESH MATHURE (5019137)