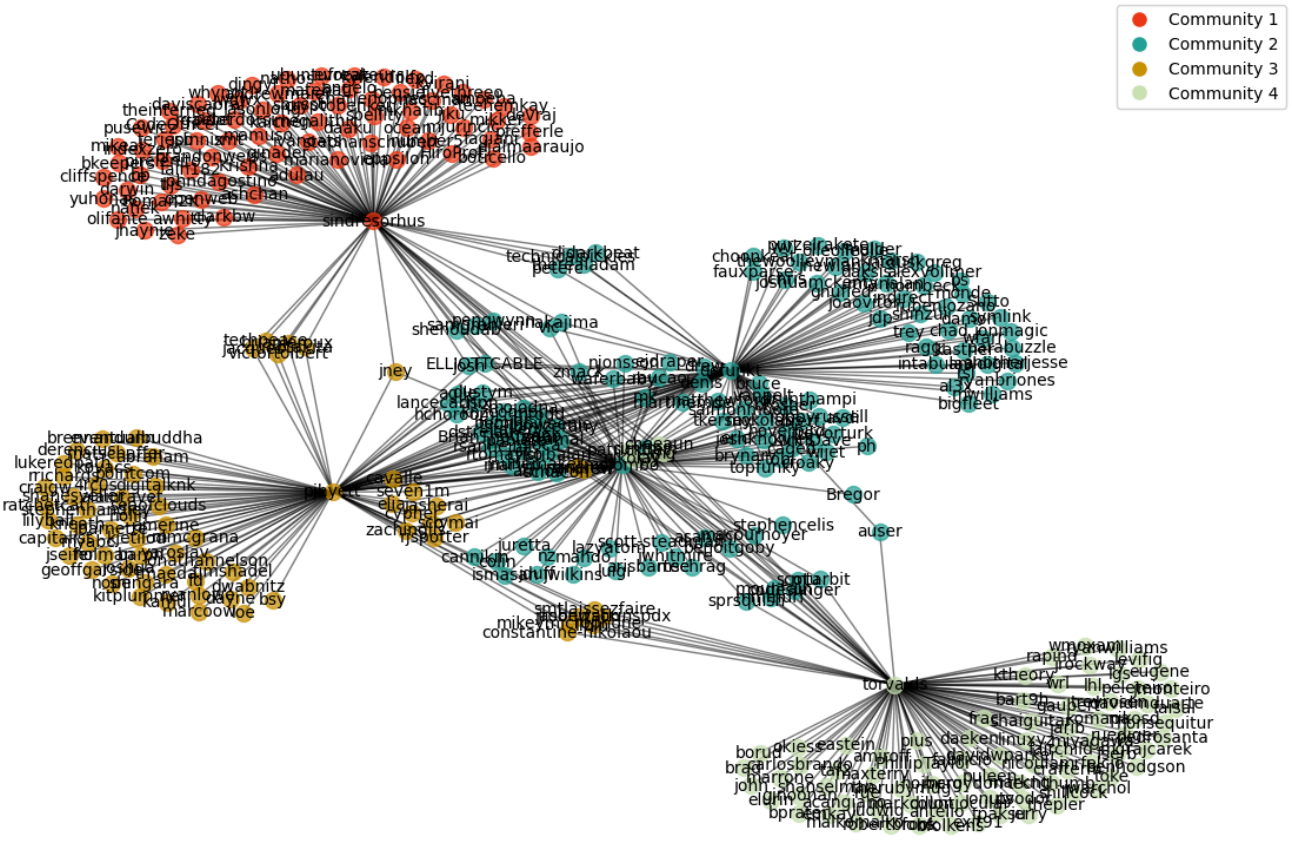
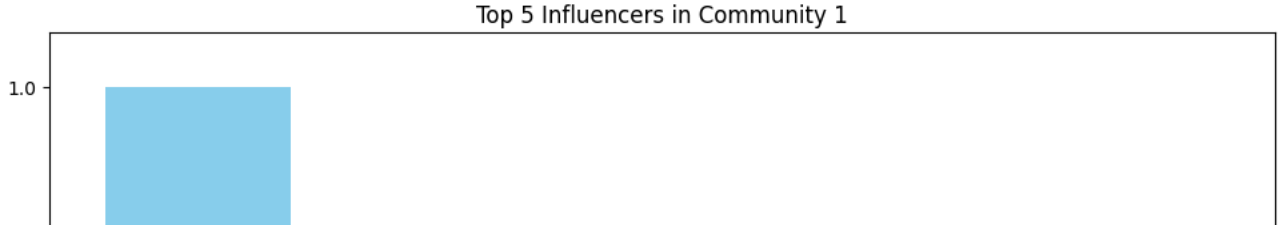
```
import requests
import pandas as pd
import csv
import networkx as nx
import matplotlib.pyplot as plt
import random

def fetch_user_data(username, access_token):
    # Fetch user data from GitHub API
    url = f"https://api.github.com/users/{username}"
    headers = {"Authorization": f"token {access_token}"}
    response = requests.get(url, headers=headers)
    if response.status_code == 200:
        return response.json()
    else:
        print(f"Failed to fetch data for {username}. Status code: {response.status_code}")
        return None

def fetch_followers(username, access_token, limit=1500):
    # Fetch followers data from GitHub API
    url = f"https://api.github.com/users/{username}/followers?per_page={limit}"
    headers = {"Authorization": f"token {access_token}"}
    response = requests.get(url, headers=headers)
    if response.status_code == 200:
        return response.json()
    else:
        print(f"Failed to fetch followers data for {username}. Status code: {response.status_code}")
        return []

def fetch_repositories(username, access_token):
    # Fetch repositories data from GitHub API
    url = f"https://api.github.com/users/{username}/repos"
    headers = {"Authorization": f"token {access_token}"}
    response = requests.get(url, headers=headers)
    if response.status_code == 200:
        return response.json()
    else:
        print(f"Failed to fetch repositories data for {username}. Status code: {response.status_code}")
        return []

def extract_languages(repositories):
    # Extract programming languages used in repositories
    languages = []
    for repo in repositories:
        if repo.get('language'):
            languages.append(repo['language'])
    return languages

def clean_data(user_data):
    # Clean data by replacing missing values
    cleaned_data = {key: user_data.get(key, "Not Available") for key in ["bio", "public_repos", "followers", "login"]}
    return cleaned_data

def save_to_csv(data, filename):
    # Save data to a CSV file
    with open(filename, 'w', newline='') as file:
        writer = csv.DictWriter(file, fieldnames=data[0].keys())
        writer.writeheader()
        writer.writerows(data)

def main():
    # GitHub access token
    access_token = "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"

    # Usernames
    usernames = ["sindresorhus", "torvalds", "defunkt", "mojombo", "pjhyett"]

    all_data = []

    for username in usernames:
        # Fetch user data
        user_data = fetch_user_data(username, access_token)
        if user_data:
            user_cleaned_data = clean_data(user_data)
            all_data.append(user_cleaned_data)
```

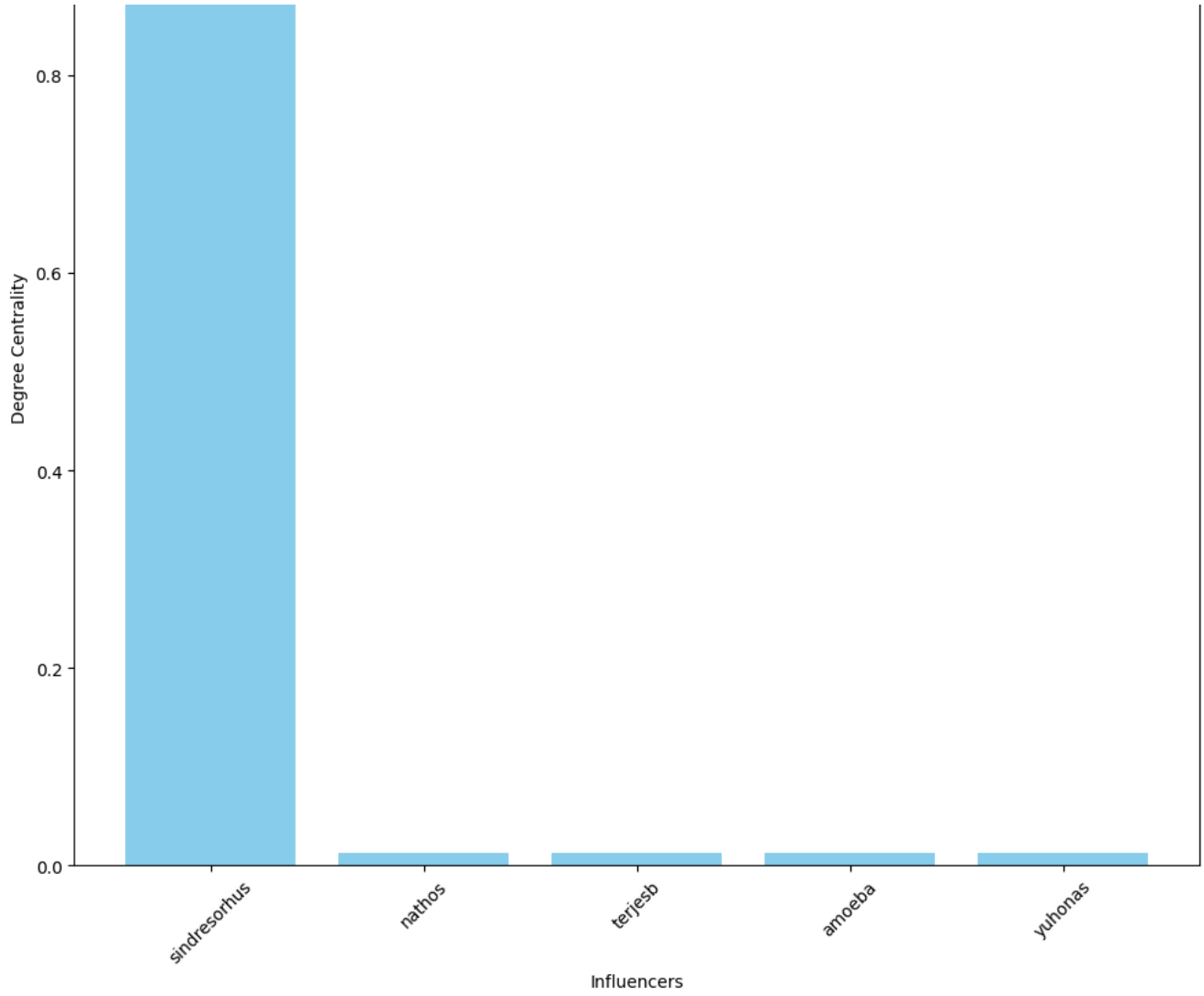
```
        # Fetch followers data
        followers_data = fetch_followers(username, access_token)
        followers_cleaned_data = [clean_data(follower) for follower in followers_data]
        all_data.extend(followers_cleaned_data)

        # Limit followers data to 1500 nodes
        if len(followers_data) > 1500:
            followers_data = followers_data[:1500]

        # Fetch repositories data
        repositories_data = fetch_repositories(username, access_token)
        languages = extract_languages(repositories_data)

        # Add programming languages to cleaned data
        user_cleaned_data['languages'] = languages

# Save data to CSV
save_to_csv(all_data, "github_data.csv")

# Community detection
# Example using NetworkX
G = nx.Graph()
for username in usernames:
    followers_data = fetch_followers(username, access_token)
    for follower in followers_data:
        G.add_edge(username, follower['login'])

communities = list(nx.algorithms.community.label_propagation.label_propagation_communities(G))

# Generate unique colors for each community
community_colors = {}
for i, community in enumerate(communities):
    community_colors[i] = (random.random(), random.random(), random.random())

# Plotting communities
plt.figure(figsize=(15, 10))
pos = nx.spring_layout(G)

for i, community in enumerate(communities):
    nx.draw_networkx_nodes(G, pos, nodelist=list(community), node_color=community_colors[i], node_size=100, alpha=0.7)

nx.draw_networkx_edges(G, pos, alpha=0.5)
nx.draw_networkx_labels(G, pos, font_size=10)

# Legend
legend_handles = [plt.Line2D([0], [0], marker='o', color='w', markerfacecolor=community_colors[i], markersize=10, label=f"Community {i+1}
plt.legend(handles=legend_handles, loc="best")

plt.title("GitHub Network - Communities")
plt.axis('off')
plt.show()

# Print communities
print("Communities:")
for i, community in enumerate(communities):
    print(f"Community {i+1}: {community}")

# Influencer identification for each community
for i, community in enumerate(communities):
    subgraph = G.subgraph(list(community))
    degree_centrality = nx.degree_centrality(subgraph)
    influencers = sorted(degree_centrality, key=degree_centrality.get, reverse=True)[:5]
    print(f"\nTop 5 Influencers in Community {i+1}:")
    for j, influencer in enumerate(influencers):
        print(f"{j+1}. {influencer} - Degree Centrality: {degree_centrality[influencer]}")

# Plotting influencers for each community
for i, community in enumerate(communities):
    subgraph = G.subgraph(list(community))
    degree_centrality = nx.degree_centrality(subgraph)
    influencers = sorted(degree_centrality, key=degree_centrality.get, reverse=True)[:5]
    plt.figure(figsize=(10, 10))
    plt.bar(range(5), [degree_centrality[influencer] for influencer in influencers], color='skyblue')
    plt.xlabel('Influencers')
    plt.ylabel('Degree Centrality')
    plt.title(f'Top 5 Influencers in Community {i+1}')
    plt.xticks(range(5), influencers, rotation=45)
    plt.tight_layout()
```

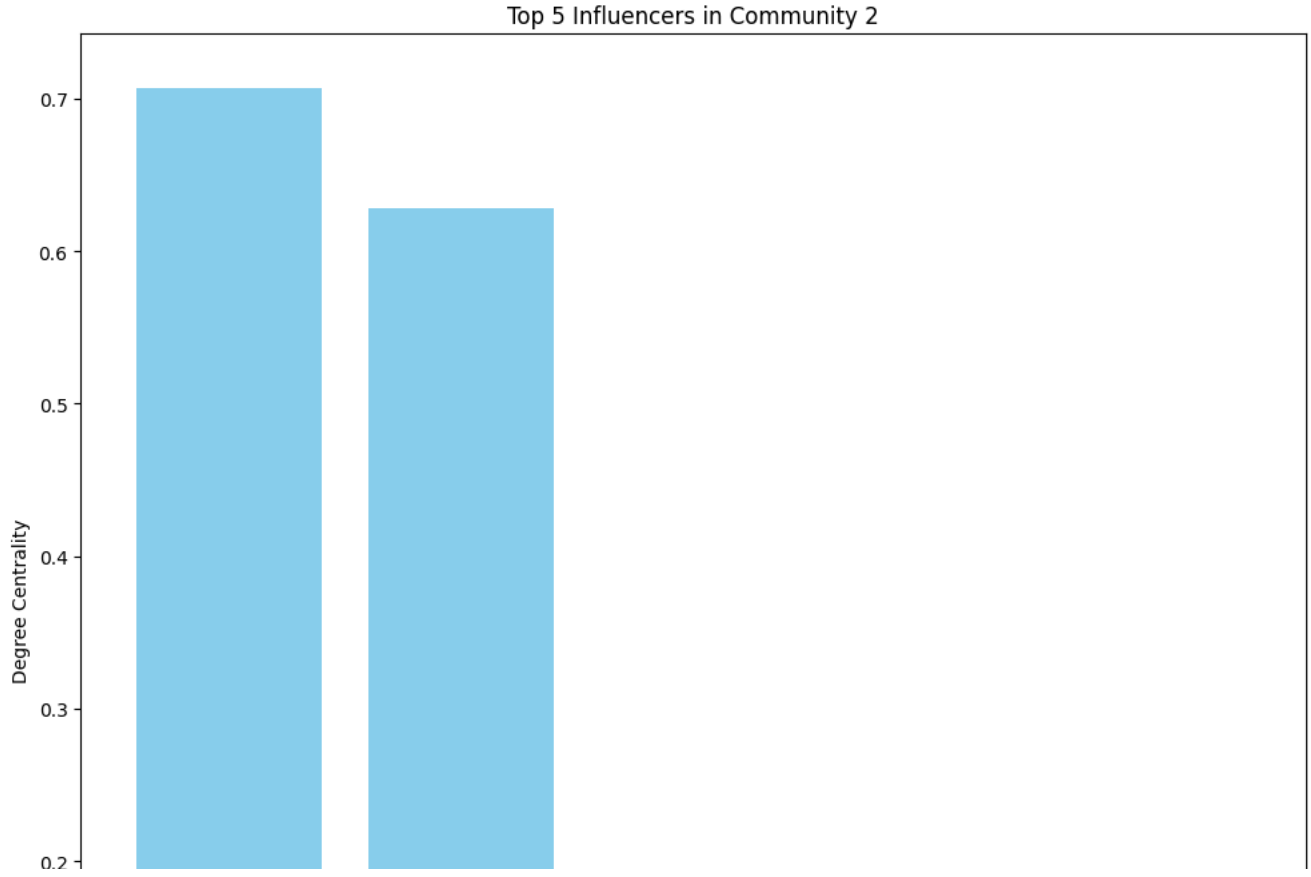
```
            plt.tight_layout()
            plt.show()

            # Influencer identification
        # Example using degree centrality
        degree_centrality = nx.degree_centrality(G)
        influencers = sorted(degree_centrality, key=degree_centrality.get, reverse=True)[:5]
        print("\nTop 5 Influencers:")
        for i, influencer in enumerate(influencers):
            print(f"{i+1}. {influencer} - Degree Centrality: {degree_centrality[influencer]}")

        # Plotting influencers
        plt.figure(figsize=(10, 10))
        plt.bar(range(5), [degree_centrality[influencer] for influencer in influencers], color='skyblue')
        plt.xlabel('Influencers')
        plt.ylabel('Degree Centrality')
        plt.title('Top 5 Influencers')
        plt.xticks(range(5), influencers, rotation=45)
        plt.tight_layout()
        plt.show()

        # Plotting programming languages
        languages_count = {}
        for user_data in all_data:
            languages = user_data.get('languages', [])
            for lang in languages:
                if lang in languages_count:
                    languages_count[lang] += 1
                else:
                    languages_count[lang] = 1

        plt.figure(figsize=(10, 6))
        plt.bar(languages_count.keys(), languages_count.values(), color='skyblue')
        plt.xlabel('Programming Languages')
        plt.ylabel('Number of Users')
        plt.title('Users by Programming Languages')
        plt.xticks(rotation=45, ha='right')
        plt.tight_layout()
        plt.show()

    if __name__ == "__main__":
        main()
```

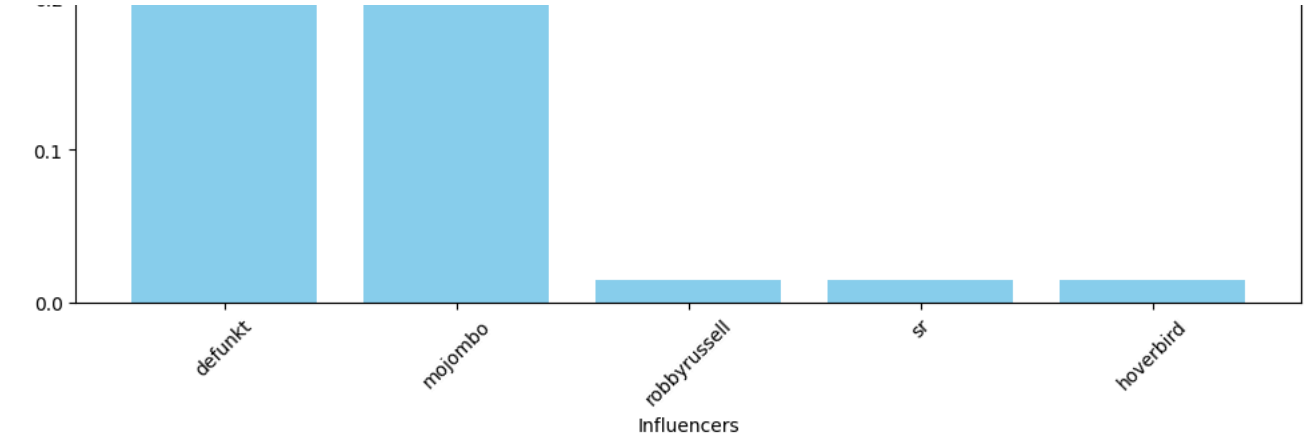


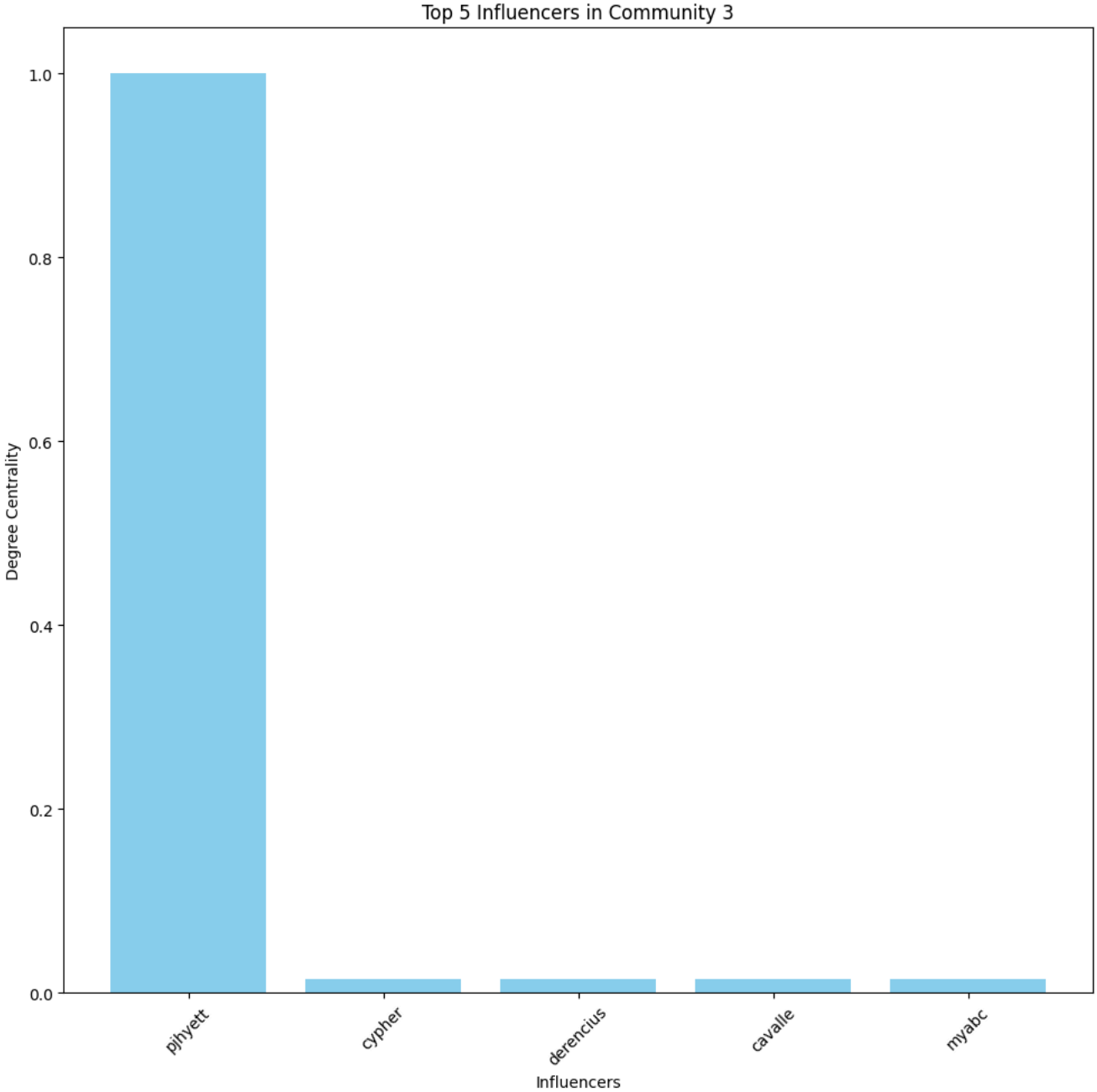

```
Communities:
Community 1: {'nathos', 'terjesb', 'amoeba', 'daviscabral', 'yuhonas', 'boticello', 'bkeepers', 'smt', 'angelo', 'pusewicz', 'bb', 'awhi
Community 2: {'trey', 'robbyrussell', 'mulder', 'daksis', 'jnewland', 'alexvollmer', 'sr', 'parabuzzle', 'samgranieri', 'anildigital', '
Community 3: {'cypher', 'derencius', 'cavalle', 'myabc', 'craigw', 'abraham', 'pointcom', 'fd', 'eventualbuddha', 'mikeymicrophone', 'sm
Community 4: {'jwarchol', 'mg', 'jarib', 'eugene', 'daeken', 'lgs', 'eastein', 'rapind', 'cheeaun', 'hojberg', 'fairchild', 'faisal', 'f

Top 5 Influencers in Community 1:
1. sindresorhus - Degree Centrality: 1.0
2. nathos - Degree Centrality: 0.013333333333333334
3. terjesb - Degree Centrality: 0.013333333333333334
4. amoeba - Degree Centrality: 0.013333333333333334
5. yuhonas - Degree Centrality: 0.013333333333333334

Top 5 Influencers in Community 2:
1. defunkt - Degree Centrality: 0.7071428571428571
2. mojombo - Degree Centrality: 0.6285714285714286
3. robbyrussell - Degree Centrality: 0.014285714285714285
4. sr - Degree Centrality: 0.014285714285714285
5. hoverbird - Degree Centrality: 0.014285714285714285

Top 5 Influencers in Community 3:
1. pjhyett - Degree Centrality: 1.0
2. cypher - Degree Centrality: 0.014492753623188406
3. derencius - Degree Centrality: 0.014492753623188406
4. cavalle - Degree Centrality: 0.014492753623188406
5. myabc - Degree Centrality: 0.014492753623188406

Top 5 Influencers in Community 4:
1. torvalds - Degree Centrality: 1.0
2. jwarchol - Degree Centrality: 0.011904761904761904
3. mg - Degree Centrality: 0.011904761904761904
4. jarib - Degree Centrality: 0.011904761904761904
5. eugene - Degree Centrality: 0.011904761904761904
```

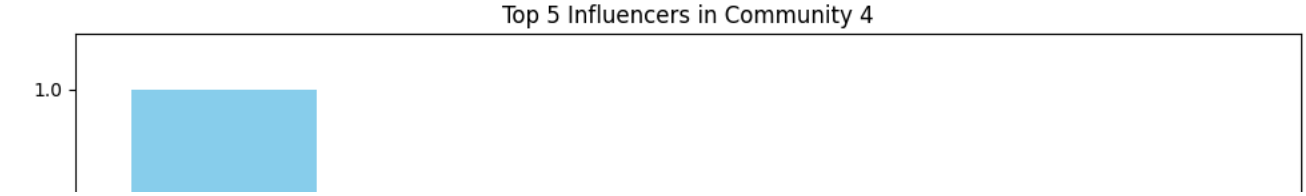


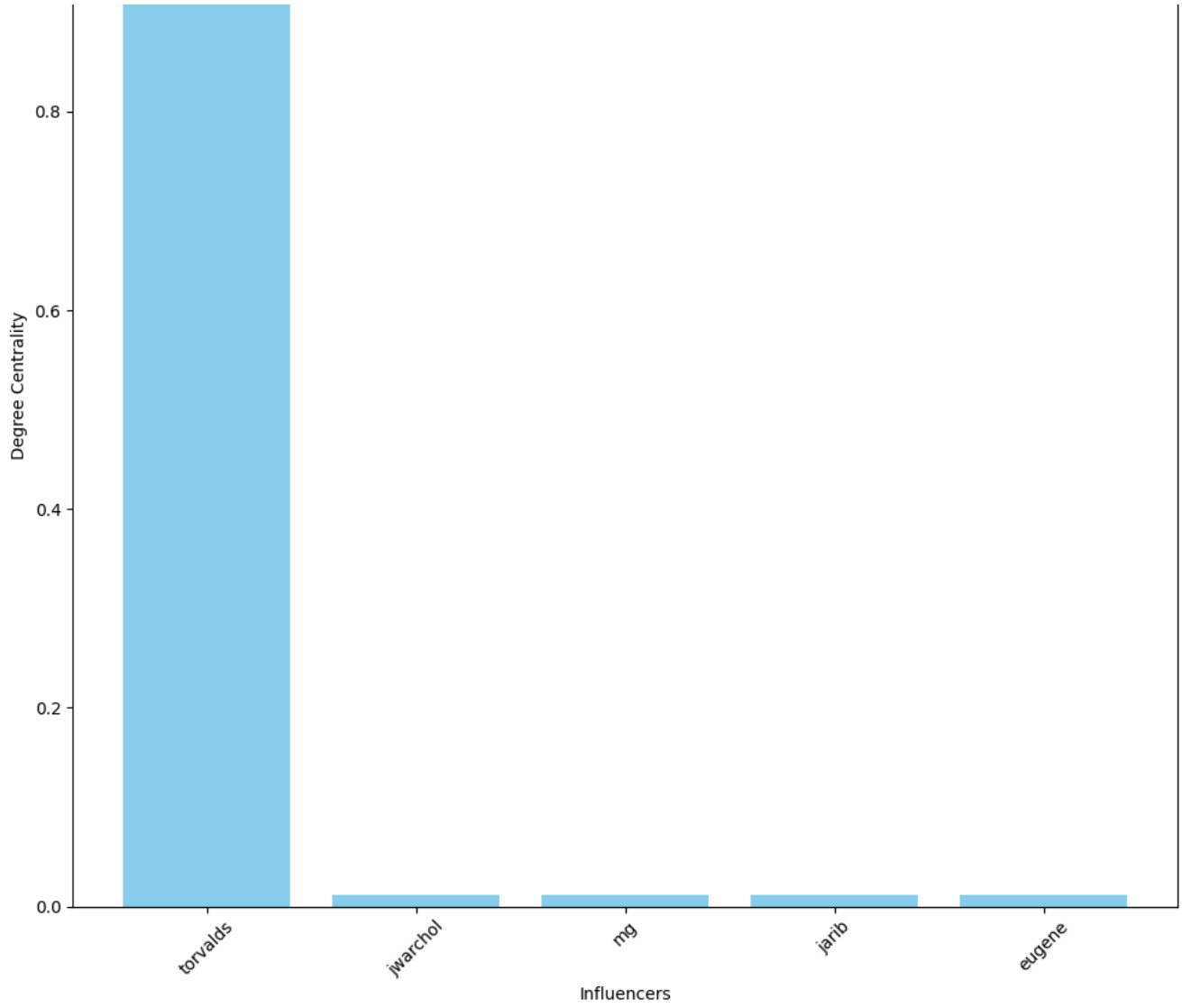

0.8
0.6
0.4
0.2
0.0
Degree Centrality
sindresorhus
nathos
terjesb
amoeba
yuhonas
Influencers


Top 5 Influencers in Community 2
0.7
0.6
0.5
0.4
0.3
0.2
Degree Centrality

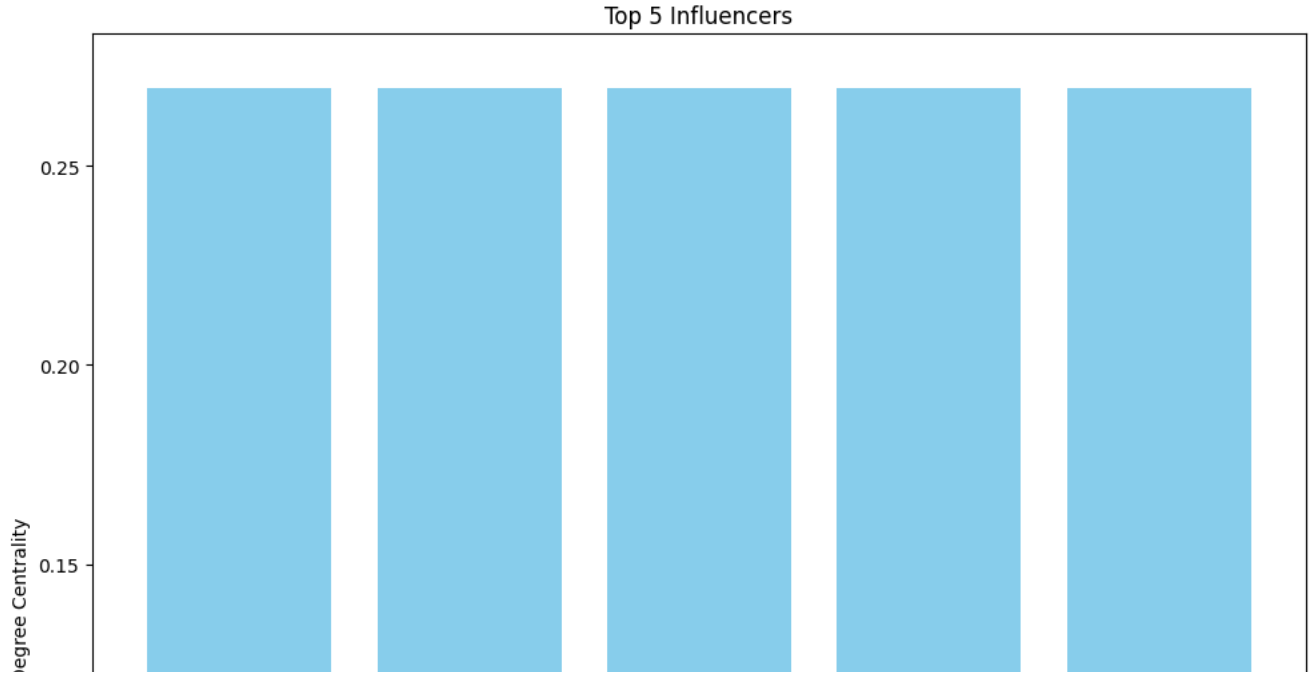

0.1
0.0
defunkt
mojombo
robbyrussell
sr
hoverbird
Influencers

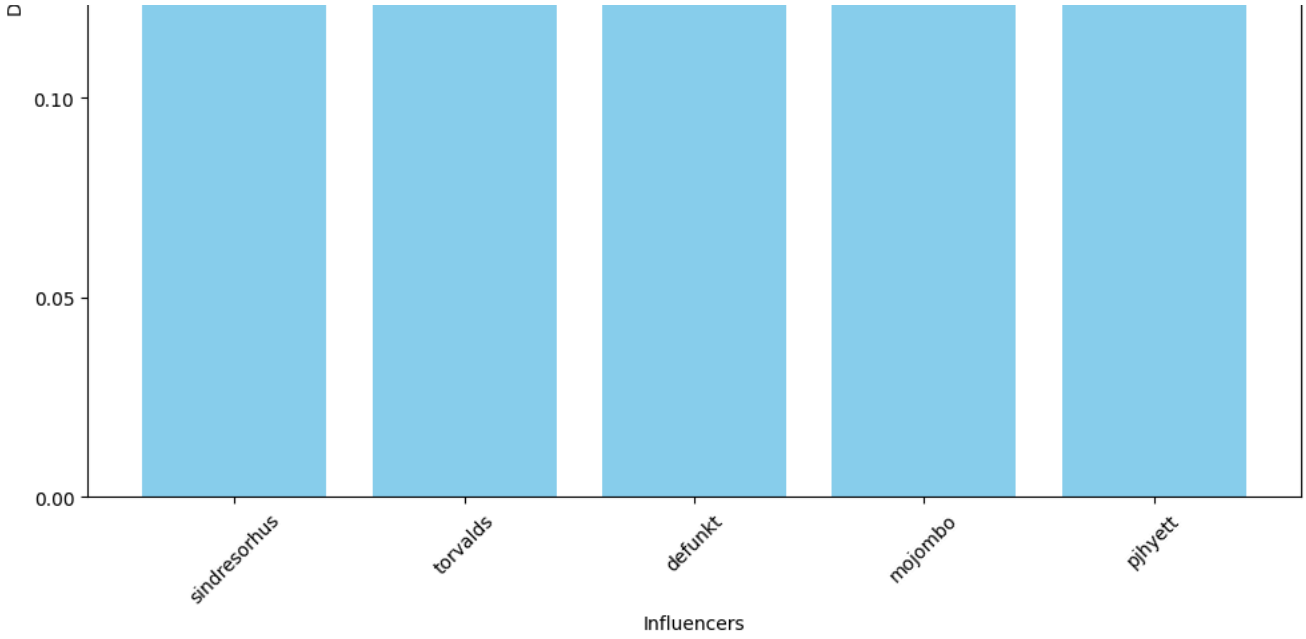

Top 5 Influencers in Community 3
Degree Centrality
1.0
0.8
0.6
0.4
0.2
0.0
pjhyett
cypher
derencius
cavalle
myabc
Influencers

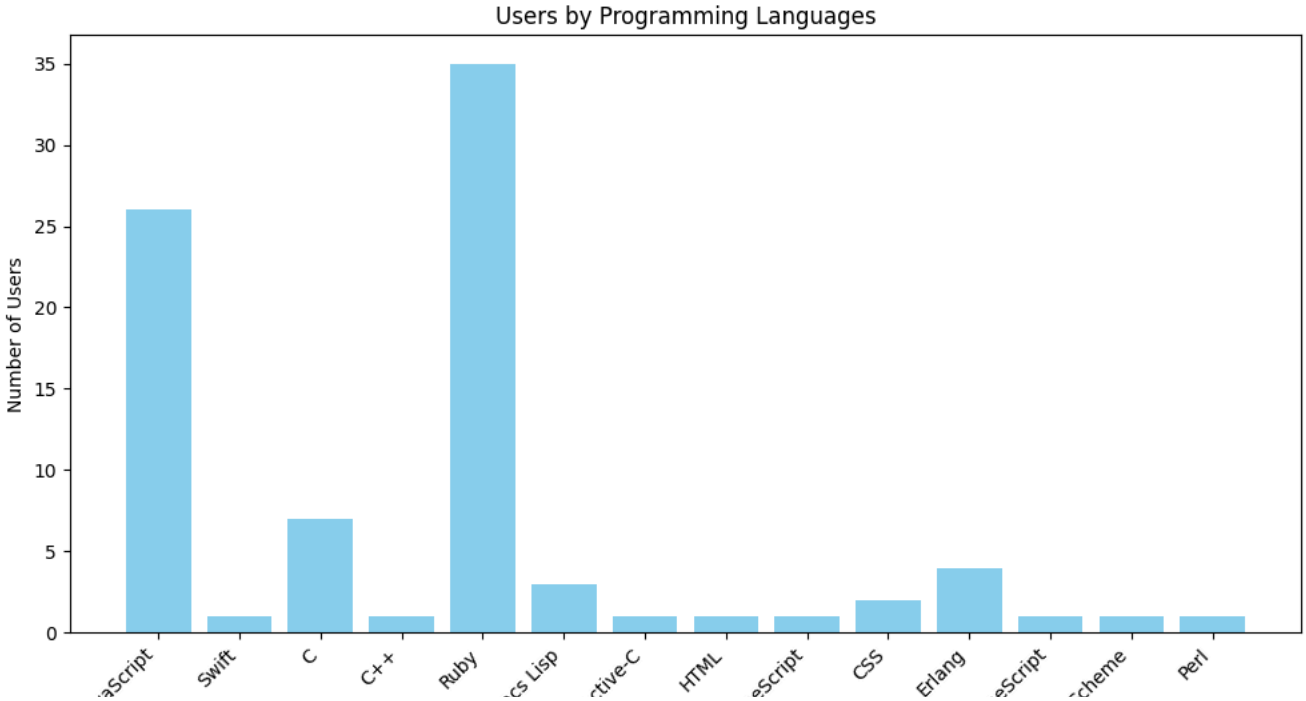

Top 5 Influencers in Community 4
1.0




```
Top 5 Influencers:
1. sindresorhus - Degree Centrality: 0.2695417789757413
2. torvalds - Degree Centrality: 0.2695417789757413
3. defunkt - Degree Centrality: 0.2695417789757413
4. mojombo - Degree Centrality: 0.2695417789757413
5. pjhyett - Degree Centrality: 0.2695417789757413
```




0.10
0.05
0.00
sindresorhus
torvalds
defunkt
mojombo
pjhyett
Influencers


Users by Programming Languages
Number of Users
35
30
25
20
15
10
5
0
Swift
C
C++
Ruby
HTML
CSS
Erlang
Perl