VV COLLEGE OF ENGINEERING
TISAIYANVILAI

LAB MANUAL

STUDENT NAME	: ....
REGISTER NUMBER	: ....
SUBJECT CODE	: CCS363
SUBJECT NAME	: SOCIAL NETWORK SECURITY LABORATORY
DEGREE /BRANCH	: BE / CSE
YEAR / SEM	: III / 06
ACADEMIC YEAR	: 2023 - 2024

V V COLLEGE OF ENGINEERING
(Approved By AICTE, New Delhi and Affiliated To Anna University Chennai)
V V Nagar, Arasoor, Tisaiyanvilai

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

College Vision and Mission Statement

Vision

"Emerge as a premier technical institution of global standards, producing enterprising, knowledgeable engineers and entrepreneurs."

Mission

- Impart quality and contemporary technical education for rural students.
- Have the state of the art infrastructure and equipment for quality learning.
- Enable knowledge with ethics, values and social responsibilities.
- Inculcate innovation and creativity among students for contribution to society.

Vision and Mission of the Department of Computer Science and Engineering

Vision

"Produce competent and intellectual computer science graduates by empowering them to compete globally towards professional excellence".

Mission

- Provide resources, environment and continuing learning processes for better exposure in latest and contemporary technologies in Computer Science and Engineering.
- Encourage creativity and innovation and the development of self-employment through knowledge and skills, for contribution to society
- Provide quality education in Computer Science and Engineering by creating a platform to enable coding, problem solving, design, development, testing and implementation of solutions for the benefit of society.

I. PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

Graduates of Computer Science Engineering can

- Apply their technical competence in computer science to solve real world problems, with
  technical and people leadership.
- Conduct cutting edge research and develop solutions on problems of social relevance.
- Work in a business environment, exhibiting team skills, work ethics, adaptability and lifelong learning.

II. PROGRAM OUTCOMES (POs)

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able

to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

III. PROGRAM SPECIFIC OUTCOMES (PSOs)

The Students will be able to

- Exhibit design and programming skills to build and automate business solutions using cutting edge technologies.
- Strong theoretical foundation leading to excellence and excitement towards research, to provide elegant solutions to complex problems.
- Ability to work effectively with various engineering fields as a team to design, build and develop system applications.

V V COLLEGE OF ENGINEERING
(Approved By AICTE, New Delhi and Affiliated To Anna University Chennai)
V V Nagar, Arasoor, Tisaiyanvilai

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

LABORATORY

LIST OF EXPERIMENTS – R2021

PRACTICAL SUBJECT NAME	SOCIAL NETWORK SECURITY
PRACTICAL SUBJECT CODE	CCS363
SEMESTER/ YEAR	06 / THIRD
TOTAL HOURS	30
DEGREE / DEPARTMENT	BE / COMPUTER SCIENCE AND ENGINEERING
STAFF IN-CHARGE	Mrs. S. ANGELIN MERLIN THAVA
LAB INSTRUCTOR	Mr. M. VINOTH
REGULATION	2021

CO1	Develop semantic web related simple applications
CO2	Address Privacy and Security issues in Social Networking
CO3	Explain the data extraction and mining of social networks
CO4	Discuss the prediction of human behavior in social communities
CO5	Describe the applications of social networks

V V COLLEGE OF ENGINEERING
(Approved By AICTE, New Delhi and Affiliated To Anna University Chennai)
V V Nagar, Arasoor, Tisaiyanvilai

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

S.No	Name of the Experiment	CO Mapping	PO Mapping
1	Design own social media applications.	CO1,CO2,CO3, CO4,CO5	PO3, PO5, PO6,PO9 PSO1-PSO3
2	Create a Network model using Neo4j.	CO1,CO2,CO3, CO4,CO5	PO1, PO4, PO5,PO10,PO11, PO12 PSO1-PSO3
3	Read and Write data from Graph database.	CO1,CO2,CO3, CO4,CO5	PO1, PO4, PO5,PO10,PO11, PO12 PSO1-PSO3
4	Find “Friend of Friends” using Neo4j.	CO1,CO2,CO3, CO4,CO5	PO1, PO4, PO5,PO10,PO11, PO12 PSO1-PSO3
5	Implement secure search in social media.	CO1,CO2,CO3, CO4,CO5	PO3, PO5, PO6,PO9 PSO1-PSO3
6	Create a simple security & privacy detector	CO1,CO2,CO3, CO4,CO5	PO3, PO5, PO6,PO9 PSO1-PSO3

V V COLLEGE OF ENGINEERING
(Approved By AICTE, New Delhi and Affiliated To Anna University Chennai)
V V Nagar, Arasoor, Tisaiyanvilai

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

RUBRICS FOR ASSESSING LABORATORY

SI. No.	Criteria	Total Marks	Excellent (25)	Good (20)	Average (10)	Poor (5)
			91% - 100%	71% - 90%	50% - 70%	<50%
1	Preparation	25	Gives clear idea about the aim and having good capability of executing experiments.	Capability of executing experiments but no proper clarification about the objective.	Gives clear idea about the target and has less capability of executing experiments.	Gives indistinct idea about the target and has less capability of executing experiments & who feel difficult to follow the objectives.
2	Viva	25	Have executed the experiments in an efficient way & make credible and unbiased judgments regarding the experiments.	Executed the experiments with less efficient & has partial judgments regarding the experiments.	Executed the experiments with less efficiency and has no judgments regarding experiments.	Incomplete experiments & lack of judgments regarding experiments.
3	Performance	25	Followed all the instructions given in the procedure and submitted the manual on time.	Followed all the instructions given in the procedure with some assisting.	Followed some of the instructions given in the procedure & late in submission of manual.	Unable to follow the instructions given in the procedure & late in submission of manual.

V V COLLEGE OF ENGINEERING
(Approved By AICTE, New Delhi and Affiliated To Anna University Chennai)
V V Nagar, Arasoor, Tisaiyanvilai

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Department of Computer Science and Engineering		
Preparation	25	
Viva	25	
Performance	25	
Total	75	
Lab Incharge	Date	

INDEX

EX.NO: 01	DESIGN OWN SOCIAL MEDIA APPLICATION
DATE:	

AIM:

To implement social media application.

ALGORITHM:

STEP 1: Create a new directory for your project. Inside this directory, create the following subdirectories and files.

STEP 2: Open a terminal and navigate to your project directory.

STEP 3: Flask : the web framework used for building the web application. render_template : A function from Flask that renders HTML templates. Graph, Namespace, Literal, URIRef : These are classes from the rdflib library, used for working with RDF (Resource Description Framework). RDF is a framework for representing information about resources on the web.

STEP 4: create an instance of the Flask class, representing the web application

STEP 5: social_graph: An instance of the RDF Graph used to store social data. FOAF: A Namespace object representing the Friend of a Friend (FOAF) vocabulary. FOAF is commonly used for describing people and relationships on the web.

STEP 6: URIRef: Represents a URI reference. Sample user data is added to the RDF graph, including user URIs and their names.

STEP 7: Adds a friendship relationship between user1 and user2 in the RDF graph.

STEP 8: Defines a route for the root URL (/). When a user accesses this URL, the index function is called. The index function retrieves a list of users from the RDF graph and renders the 'index.html' template, passing the users, social graph, and FOAF namespace to the template.

STEP 9: Defines a route for the '/profile/<user_id>' URL pattern. The <user_id> part is a dynamic parameter. The profile function takes the user_id as a parameter, retrieves the user's information from the RDF graph, and renders the 'profile.html' template, passing the user's name, friends, social graph, and FOAF namespace to the template.

STEP 10: Checks if the script is being run directly (not imported as a module). If so, it starts the Flask development server with debugging enabled.

PROGRAM:

index.html:

```
<!-- templates/index.html -->
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Social Media App</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
</head>

<body>
    <div style="text-align: center;">
        <h1>Users</h1>
    </div>
    <hr>
    <div class="image-container">
        {% for user in users %}
        <a href="{{ url_for('profile', user_id=user.split('/')[-1]) }}">
            <img src="https://tse1.mm.bing.net/th?
id=OIP.eoBtu339Epu84pJA0EY_QwAAAA&pid=Api&P=0&h=180"
                alt="User Image" class="avatar">
            <div class="overlay">{{ social_graph.value(user, FOAF.name) }}</div>
        </a>
        {% endfor %}
    </div>
</body>

</html>
```

profile.html:

```
<!-- templates/profile.html -->
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>User Profile</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
</head>

<body>
    <h1>User Profile</h1>
    <p>Name: {{ user_name }}</p>
    <h2>Friends</h2>
    <ul>
        {% for friend in friends %}
        <li>{{ social_graph.value(friend, FOAF.name) }}</li>
        {% endfor %}
    </ul>
</body>

</html>
```

styles.css:

```
/* static/styles.css */
body {
    font-family: 'Times New Roman', Times, serif;
    margin: 20px;
    background-color: aliceblue;
}

h1, h2 {
    color: #333;
}

ul {
    list-style-type: none;
    padding: 0;
}

li {
    margin-bottom: 10px;
}

/* Define a basic styling for the image container */
.image-container {
    display: flex;
    justify-content: space-evenly;
    max-width: 800px; /* Adjust the max-width based on your design */
    margin: auto; /* Center the container */
```

```
}

/* Style for each individual image container */
.image-container a {
    position: relative;
    text-decoration: none;
    display: inline-block; /* Ensure block-level layout for the anchor */
}

/* Style for each individual image */
.image-container img {
    width: 100%; /* Set the width to 100% to match the container size */
    height: auto; /* Auto-adjust height to maintain the aspect ratio */
    margin-right: 16px; /* Add some spacing between images */
    transition: transform 0.3s; /* Add a smooth transition effect */
    display: block; /* Ensure block-level layout for the image */
}

/* Style for the text overlay */
.image-container .overlay {
    position: absolute;
    top: 0;
    left: 0;
    width: 100%; /* Set the width to 100% to match the container size */
    height: 100%; /* Set the height to 100% to match the container size */
    display: flex;
    align-items: center;
    justify-content: center;
    opacity: 0;
    background: rgba(0, 0, 0, 0.5); /* Semi-transparent background */
    color: #fff; /* Text color */
    transition: opacity 0.3s; /* Add a smooth transition effect */
    pointer-events: none; /* Ensure the overlay doesn't block interactions with the
underlying image */
}

/* Hover effect on images */
.image-container a:hover .overlay {
    opacity: 1;
}

/* Style for the image links */
.image-container a {
    text-decoration: none; /* Remove underlines from links */
    color: inherit; /* Inherit text color from the parent */
}
```

app.py:

```
from flask import Flask, render_template, request

from rdflib import Graph, Namespace, Literal, URIRef

app = Flask(__name__)

# RDF graph to store social data
social_graph = Graph()

# Define Namespace
FOAF = Namespace("http://xmlns.com/foaf/0.1/")

# Sample user data
user_data = {
    "1": ("Luffy", ["2", "3", "4"]),
    "2": ("Zoro", ["1", "4", "3"]),
    "3": ("Nami", ["1", "4", "2"]),
    "4": ("Usopp", ["1", "3", "2"])
}

# Populate RDF graph with sample data
for user_id, (name, friends) in user_data.items():
    user_uri = URIRef(f"http://example.com/users/{user_id}")
    social_graph.add((user_uri, FOAF.name, Literal(name)))
    for friend_id in friends:
        friend_uri = URIRef(f"http://example.com/users/{friend_id}")
        social_graph.add((user_uri, FOAF.knows, friend_uri))

@app.route('/')
def index():
    # Display a list of users
    users = social_graph.subjects(predicate=FOAF.name)
    return render_template('index.html', users=users, social_graph=social_graph,
FOAF=FOAF)

@app.route('/profile/<user_id>')
def profile(user_id):
    try:
        user = URIRef(f"http://example.com/users/{user_id}")
        user_name = social_graph.value(user, FOAF.name)
        friends = social_graph.objects(subject=user, predicate=FOAF.knows)
        return render_template('profile.html', user_name=user_name, friends=friends,
                               social_graph=social_graph, FOAF=FOAF)
    except Exception as e:
        return render_template('error.html', error_message=str(e))

if __name__ == '__main__':
    app.run(debug=True)
```

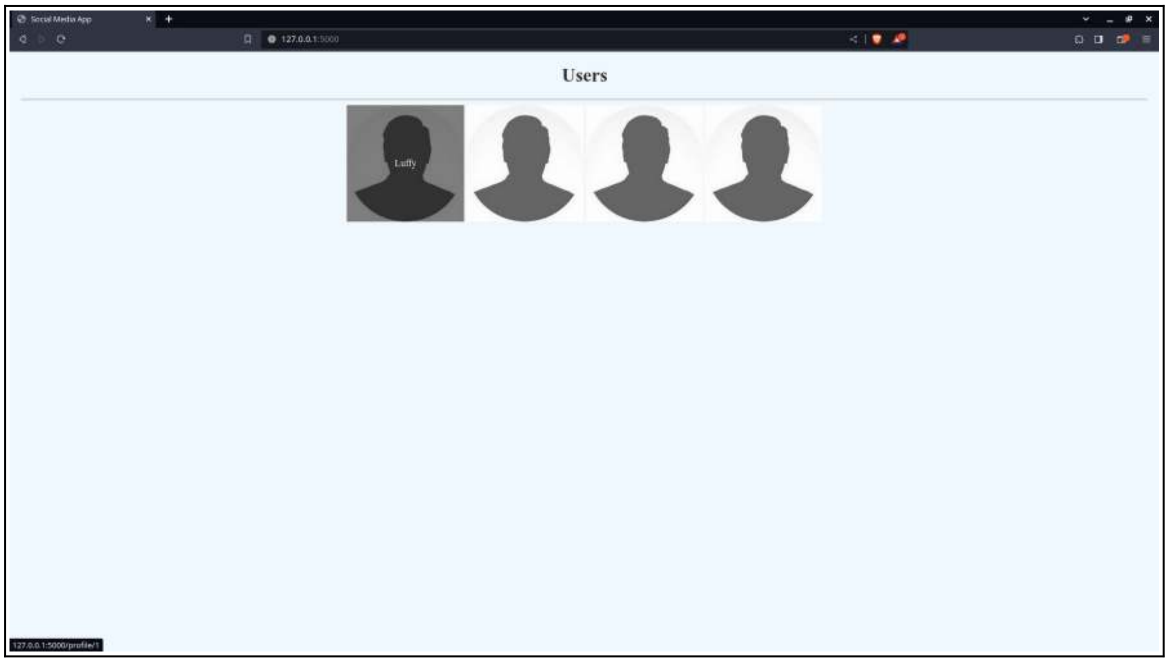
PROJECT STRUCTURE: (Just For Ref.)

```
.
├── app.py
├── static
│   └── styles.css
└── templates
    ├── index.html
    └── profile.html

2 directories, 4 files
```

Execution (Python (Programming Language)):

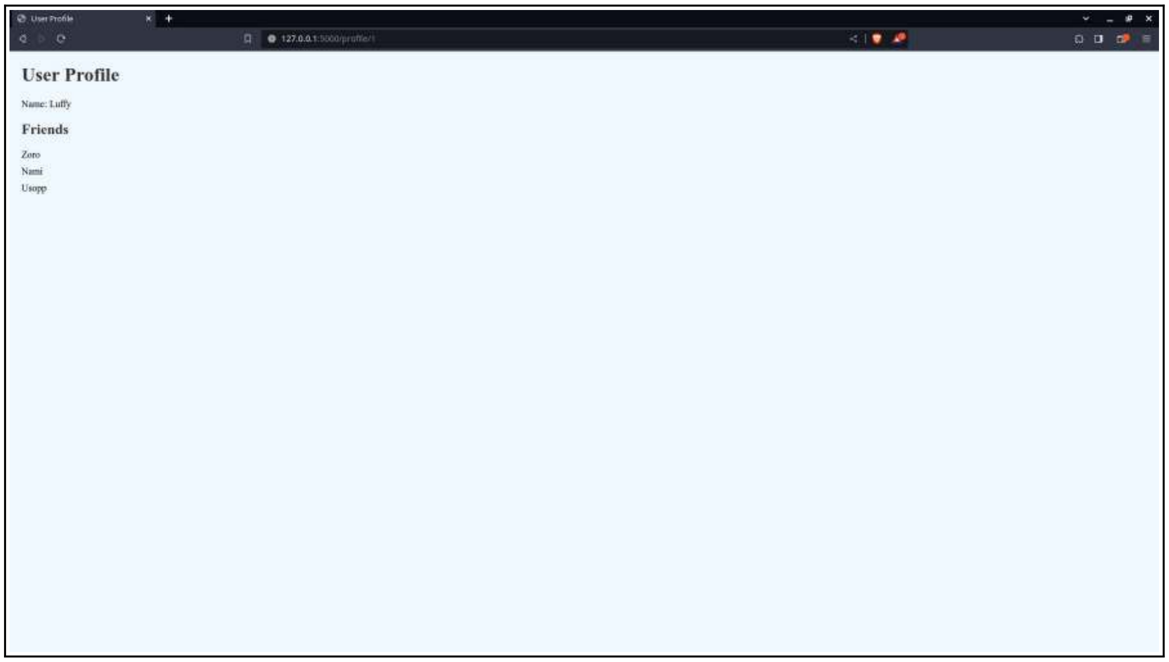
```
$ pip3 install Flask
$ pip3 install rdflib
$ python3 app.py
```

OUTPUT:



Users

Luffy

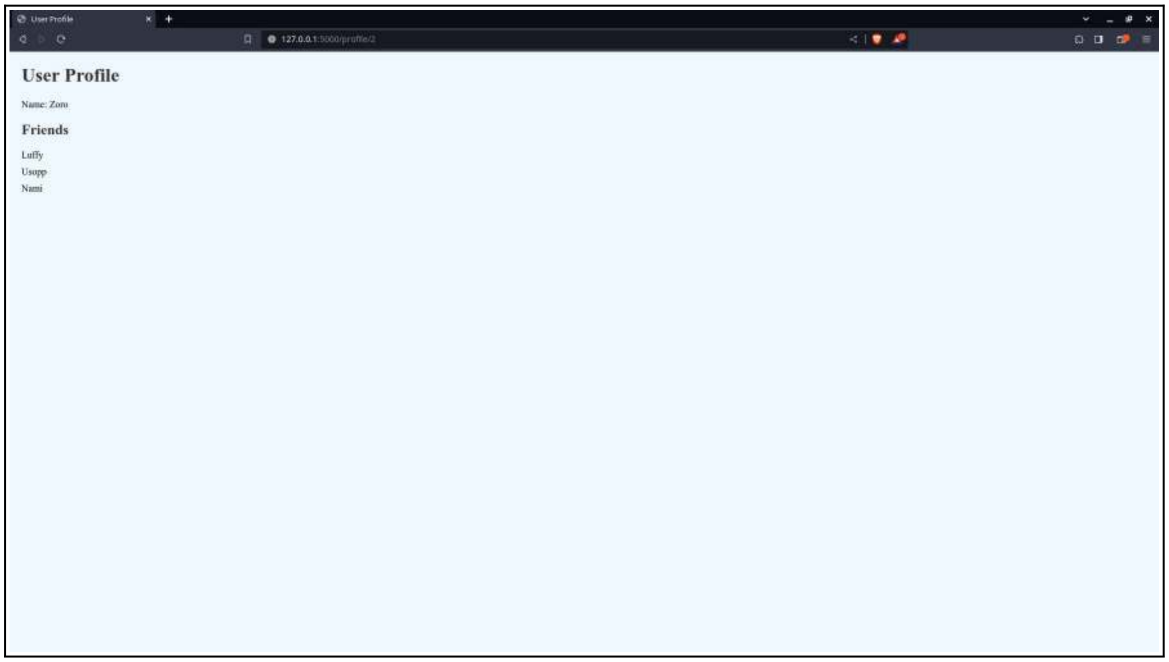


User Profile

Name: Luffy

Friends

Zoro
Nami
Usopp



User Profile

Name: Zoro

Friends

Luffy
Usopp
Nami

RESULT:

EX.NO: 02	**CREATE A NETWORK MODEL USING NEO4J**
DATE:	

AIM:

To create a network model using node4j.

ALGORITHM:

STEP 1: Start.

STEP 2: Download and install neo4j.

STEP 3: Open the Neo4j browser.

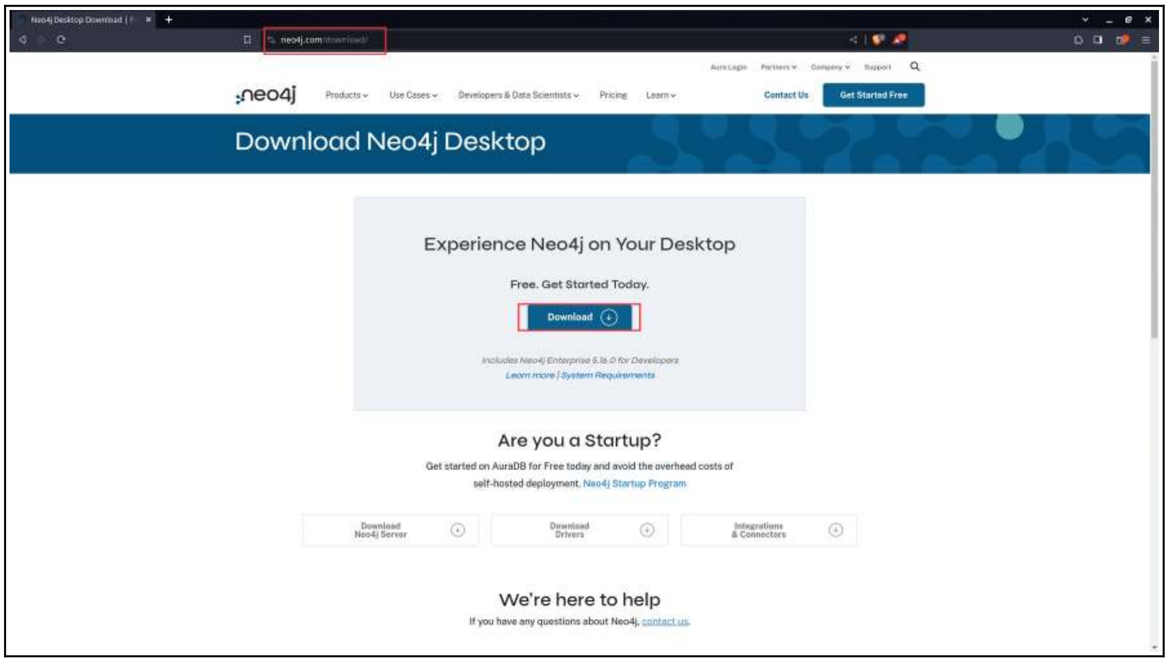
STEP 4: Create a new network model and retrieve the graph.

STEP 5: Stop.

INSTALLATION:

Step 1: Navigate to the Neo4j download page by visiting https://neo4j.com/download/.

Step 2: On the page, locate and click on the 'Download' button."



Download Neo4j Desktop

Experience Neo4j on Your Desktop

Free. Get Started Today.

Includes Neo4j Enterprise 5.16.0 for Developers
Learn more | System Requirements

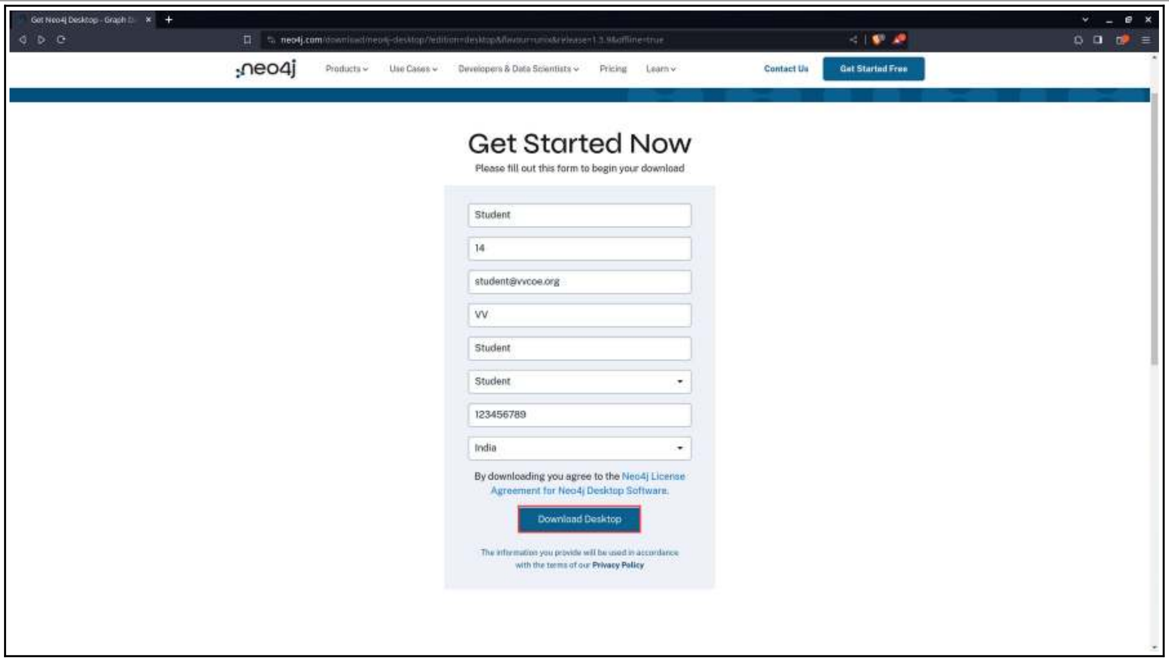
Are you a Startup?

Get started on AuraDB for Free today and avoid the overhead costs of self-hosted deployment. Neo4j Startup Program

We're here to help

If you have any questions about Neo4j, contact us.

Step 3: Fill out the form and click "Download Desktop".



Get Started Now

Please fill out this form to begin your download

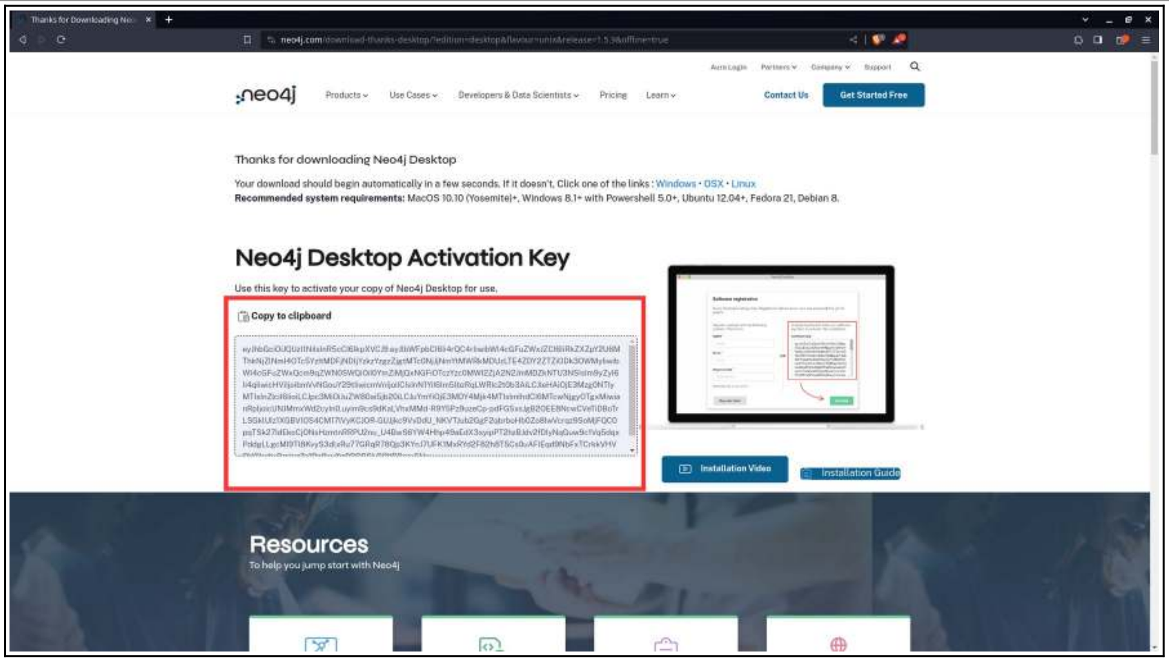
Student
14
student@vvcoe.org
VV
Student
Student
123456789
India

By downloading you agree to the Neo4j License Agreement for Neo4j Desktop Software.

The information you provide will be used in accordance with the terms of our Privacy Policy

(Note: The website automatically detects the desktop using the user-agent, and the suitable AppImage will begin downloading. Do not close the tab!)

Step 4: Copy the “Activation key” to the clipboard and wait for the download to finish.


Thanks for downloading Neo4j Desktop

Your download should begin automatically in a few seconds. If it doesn't, Click one of the links : Windows • OSX • Linux
Recommended system requirements: MacOS 10.10 (Yosemite)+, Windows 8.1+ with Powershell 5.0+, Ubuntu 12.04+, Fedora 21, Debian 8.

Neo4j Desktop Activation Key

Use this key to activate your copy of Neo4j Desktop for use.

Copy to clipboard

[illegible]

Step 5: To start Neo4j, verify the downloaded file in the `~/Downloads` directory.

$ ls -al | grep “neo4j”

Change the permissions to make it executable.

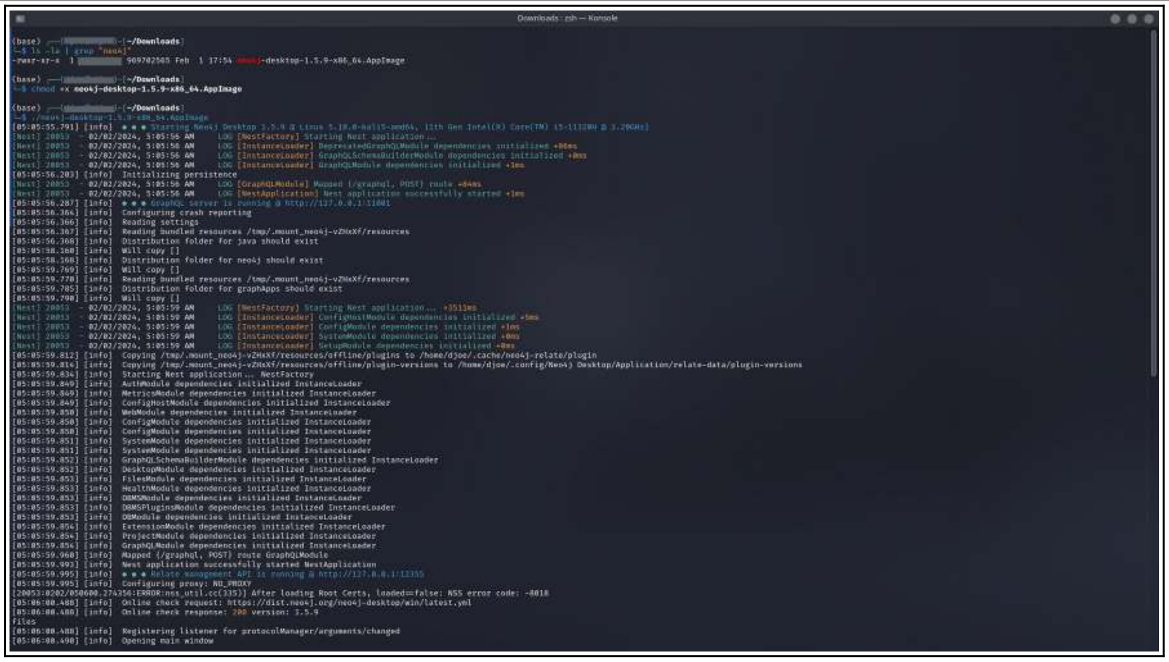
$ chmod +x neo4j-desktop-1.5.9-x86_64.AppImage

(Note: The “neo4j-desktop-1.5.9-x86_64.AppImage” may change according to the version you downloaded. Verify your AppImage name using “ls -al | grep “neo4j”)

Start the AppImage:

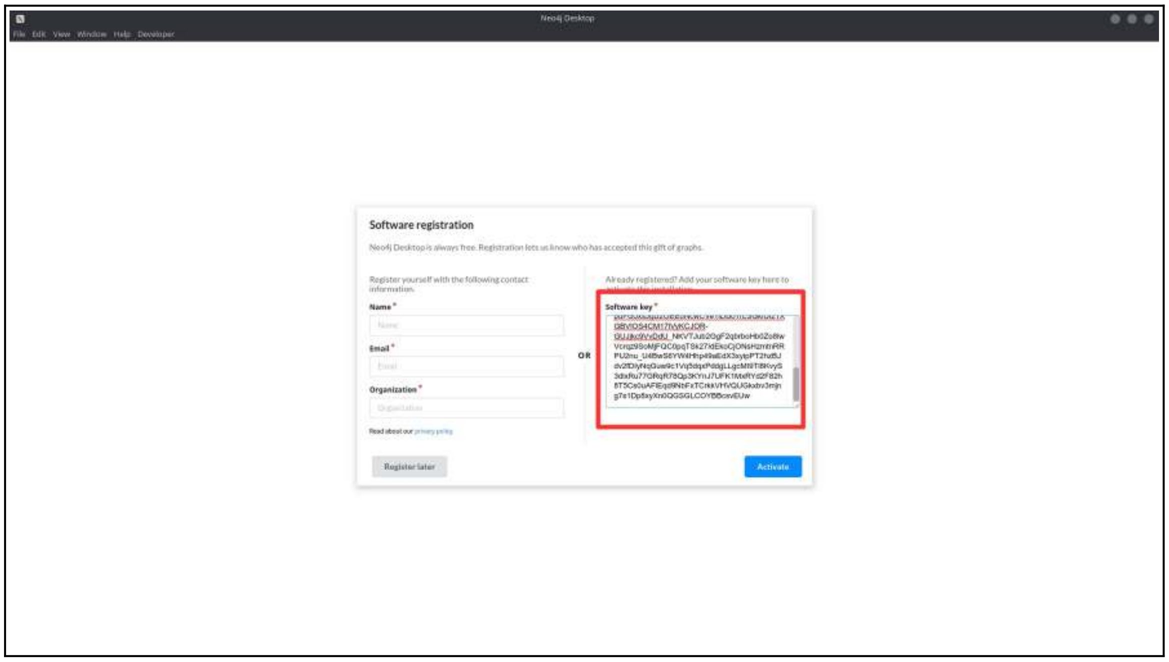
$ ./neo4j-desktop-1.5.9-x86_64.AppImage

(Note: The “neo4j-desktop-1.5.9-x86_64.AppImage” may change according to the version you downloaded. Verify your AppImage name using “ls -al | grep “neo4j”)



```
(base) ┌──([redacted])-[~/Downloads]
└─$ ls -la | grep "neo4j"
-rwxr-xr-x  1 [redacted] 909702565 Feb  1 17:54 neo4j-desktop-1.5.9-x86_64.AppImage

(base) ┌──([redacted])-[~/Downloads]
└─$ chmod +x neo4j-desktop-1.5.9-x86_64.AppImage

(base) ┌──([redacted])-[~/Downloads]
└─$ ./neo4j-desktop-1.5.9-x86_64.AppImage
[05:05:55.791] [info]  ● ● ● Starting Neo4j Desktop 1.5.9 @ Linux 5.18.0-kali5-amd64, 11th Gen Intel(R) Core(TM) i5-11320H @ 3.20GHz]
[Nest] 20053  - 02/02/2024, 5:05:56 AM     LOG [NestFactory] Starting Nest application...
[Nest] 20053  - 02/02/2024, 5:05:56 AM     LOG [InstanceLoader] DeprecatedGraphQLModule dependencies initialized +86ms
[Nest] 20053  - 02/02/2024, 5:05:56 AM     LOG [InstanceLoader] GraphQLSchemaBuilderModule dependencies initialized +0ms
[Nest] 20053  - 02/02/2024, 5:05:56 AM     LOG [InstanceLoader] GraphQLModule dependencies initialized +1ms
[05:05:56.203] [info]  Initializing persistence
[Nest] 20053  - 02/02/2024, 5:05:56 AM     LOG [GraphQLModule] Mapped {/graphql, POST} route +84ms
[Nest] 20053  - 02/02/2024, 5:05:56 AM     LOG [NestApplication] Nest application successfully started +1ms
[05:05:56.287] [info]  ● ● ● GraphQL server is running @ http://127.0.0.1:11001
[05:05:56.364] [info]  Configuring crash reporting
[05:05:56.366] [info]  Reading settings
[05:05:56.367] [info]  Reading bundled resources /tmp/.mount_neo4j-vZHxXf/resources
[05:05:56.368] [info]  Distribution folder for java should exist
[05:05:58.160] [info]  Will copy []
[05:05:58.168] [info]  Distribution folder for neo4j should exist
[05:05:59.769] [info]  Will copy []
[05:05:59.778] [info]  Reading bundled resources /tmp/.mount_neo4j-vZHxXf/resources
[05:05:59.785] [info]  Distribution folder for graphApps should exist
[05:05:59.790] [info]  Will copy []
[Nest] 20053  - 02/02/2024, 5:05:59 AM     LOG [NestFactory] Starting Nest application... +3511ms
[Nest] 20053  - 02/02/2024, 5:05:59 AM     LOG [InstanceLoader] ConfigHostModule dependencies initialized +5ms
[Nest] 20053  - 02/02/2024, 5:05:59 AM     LOG [InstanceLoader] ConfigModule dependencies initialized +1ms
[Nest] 20053  - 02/02/2024, 5:05:59 AM     LOG [InstanceLoader] SystemModule dependencies initialized +0ms
[Nest] 20053  - 02/02/2024, 5:05:59 AM     LOG [InstanceLoader] SetupModule dependencies initialized +0ms
[05:05:59.812] [info]  Copying /tmp/.mount_neo4j-vZHxXf/resources/offline/plugins to /home/djoe/.cache/neo4j-relate/plugin
[05:05:59.814] [info]  Copying /tmp/.mount_neo4j-vZHxXf/resources/offline/plugin-versions to /home/djoe/.config/Neo4j Desktop/Application/relate-data/plugin-versions
[05:05:59.834] [info]  Starting Nest application... NestFactory
[05:05:59.849] [info]  AuthModule dependencies initialized InstanceLoader
[05:05:59.849] [info]  MetricsModule dependencies initialized InstanceLoader
[05:05:59.849] [info]  ConfigHostModule dependencies initialized InstanceLoader
[05:05:59.850] [info]  WebModule dependencies initialized InstanceLoader
[05:05:59.850] [info]  ConfigModule dependencies initialized InstanceLoader
[05:05:59.850] [info]  ConfigModule dependencies initialized InstanceLoader
[05:05:59.851] [info]  SystemModule dependencies initialized InstanceLoader
[05:05:59.851] [info]  SystemModule dependencies initialized InstanceLoader
[05:05:59.852] [info]  GraphQLSchemaBuilderModule dependencies initialized InstanceLoader
[05:05:59.852] [info]  DesktopModule dependencies initialized InstanceLoader
[05:05:59.853] [info]  FilesModule dependencies initialized InstanceLoader
[05:05:59.853] [info]  HealthModule dependencies initialized InstanceLoader
[05:05:59.853] [info]  DBMSModule dependencies initialized InstanceLoader
[05:05:59.853] [info]  DBMSPluginsModule dependencies initialized InstanceLoader
[05:05:59.853] [info]  DBModule dependencies initialized InstanceLoader
[05:05:59.854] [info]  ExtensionModule dependencies initialized InstanceLoader
[05:05:59.854] [info]  ProjectModule dependencies initialized InstanceLoader
[05:05:59.856] [info]  GraphQLModule dependencies initialized InstanceLoader
[05:05:59.960] [info]  Mapped {/graphql, POST} route GraphQLModule
[05:05:59.993] [info]  Nest application successfully started NestApplication
[05:05:59.995] [info]  ● ● ● Relate management API is running @ http://127.0.0.1:12355
[05:05:59.995] [info]  Configuring proxy: NO_PROXY
[20053:0202/050600.274356:ERROR:nss_util.cc(335)] After loading Root Certs, loaded==false: NSS error code: -8018
[05:06:00.488] [info]  Online check request: https://dist.neo4j.org/neo4j-desktop/win/latest.yml
[05:06:00.488] [info]  Online check response: 200 version: 1.5.9
Files
[05:06:00.488] [info]  Registering listener for protocolManager/arguments/changed
[05:06:00.498] [info]  Opening main window
```

Step 6: After opening Neo4j, navigate to the 'Software Key' section and paste the previously copied 'Activation Key'. Then, click the 'Activate' button to complete the activation process.



Software registration

Neo4j Desktop is always free. Registration lets us know who has accepted this gift of graphs.

Register yourself with the following contact information.

Name *

Email *

Organization *

Read about our privacy policy

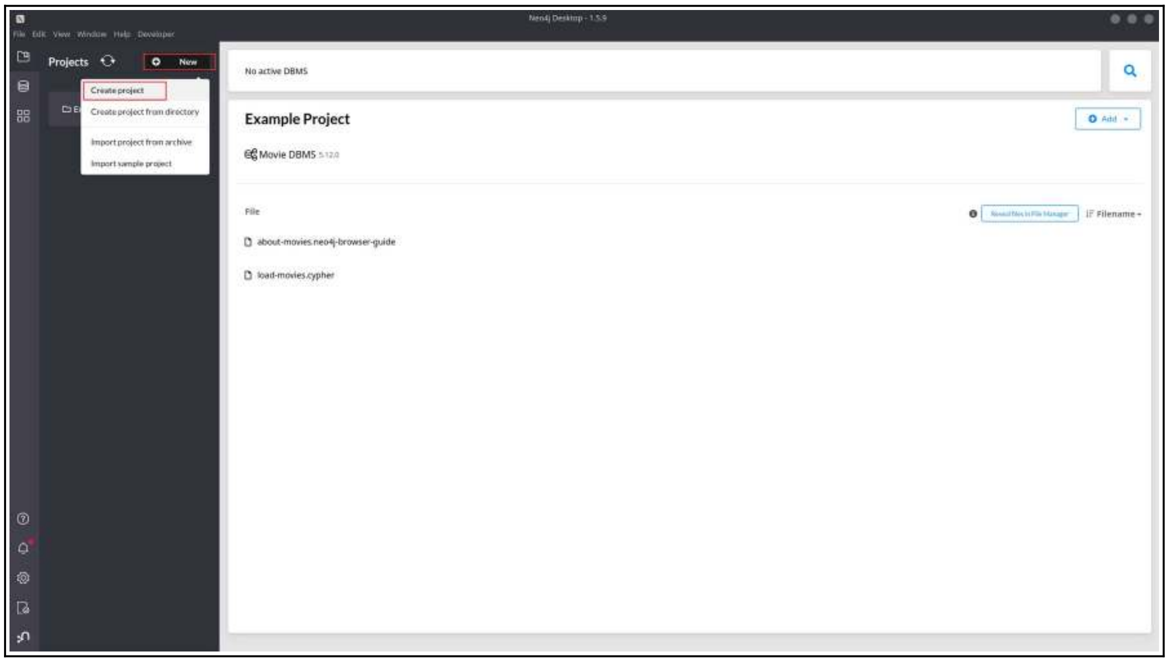
OR

Already registered? Add your software key here to activate this installation.

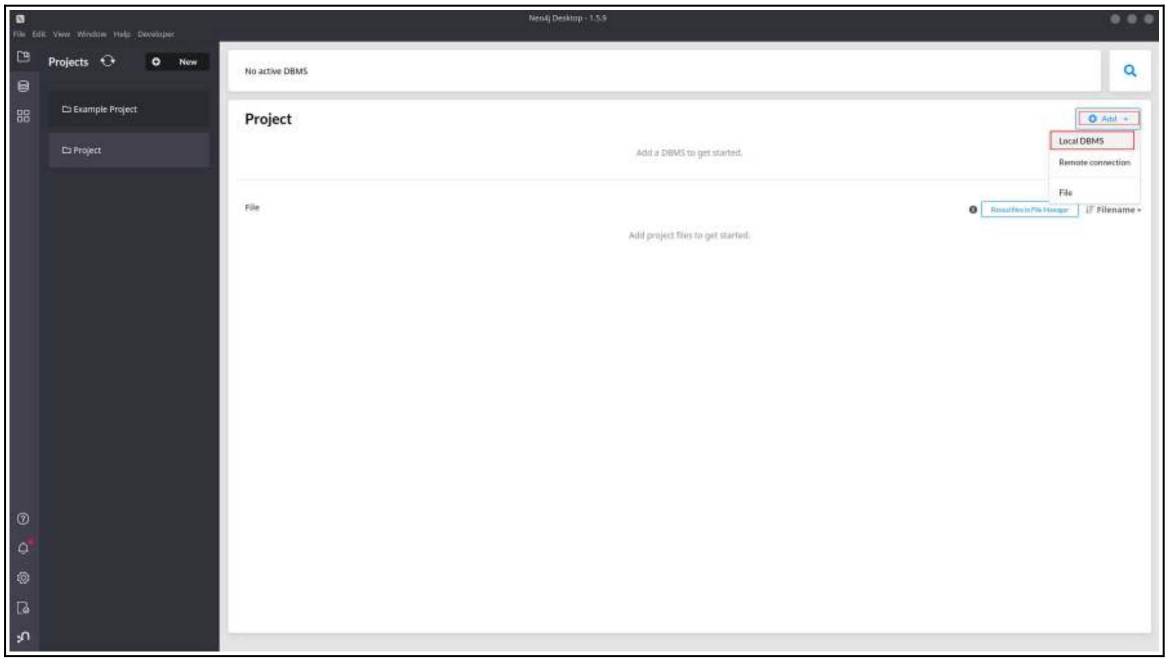
Software key *

QBVlO54CM17lVyKCJOR-
QUJjkc9VvDdU_NKVTJub2GgF2qbrboHb0Zo8w
Vcrqz9SoMjFQC0pqTSk27idEkoCjONsHzmtnRR
PU2nu_U4BwS6YW4Hhp49aEdX3sytpPT2hzBJ
dv2fDlyNqGuw9c1Vq5dqxPddgLLgcMl9TI8KvyS
3dxRu77GRqR78Qp3KYnJ7UFK1MxRYd2F82h
8T5Cs0uAFlEqd9NbFxTCnkVHVQUGkxbv3mjn
g7s1Dp8xyXn0QGSGLCOYBBcevEUw

Step 7: Within Neo4j, click on the 'New' button to create a new project.



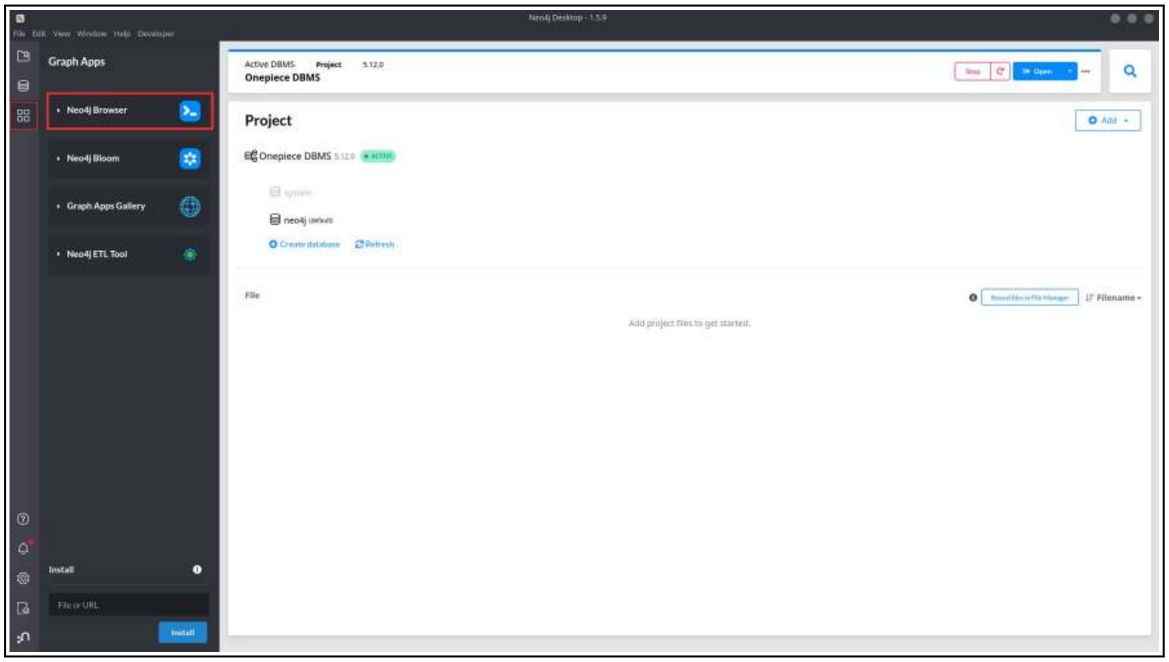
Step 8: Once you have created a new project, click on 'Add' and then select 'Local DBMS' to add a new database to our project.



When prompted, enter any desired DBMS name and password.

Step 9: Click the 'Start' button in the right corner of your newly created database.

Step 10: Once started, open the ‘Neo4j Browser’.



Once the Neo4j Browser has started successfully, this is where you can execute your 'Cypher query'.

PROGRAM:

Creating character nodes:

```
CREATE (:Character {name: 'Monkey D. Luffy', role: 'Main Protagonist'})
CREATE (:Character {name: 'Roronoa Zoro', role: 'Swordsman'})
CREATE (:Character {name: 'Nami', role: 'Navigator'})
CREATE (:Character {name: 'Usopp', role: 'Sniper'})
CREATE (:Character {name: 'Sanji', role: 'Cook'})
```

Creating crew relationship:

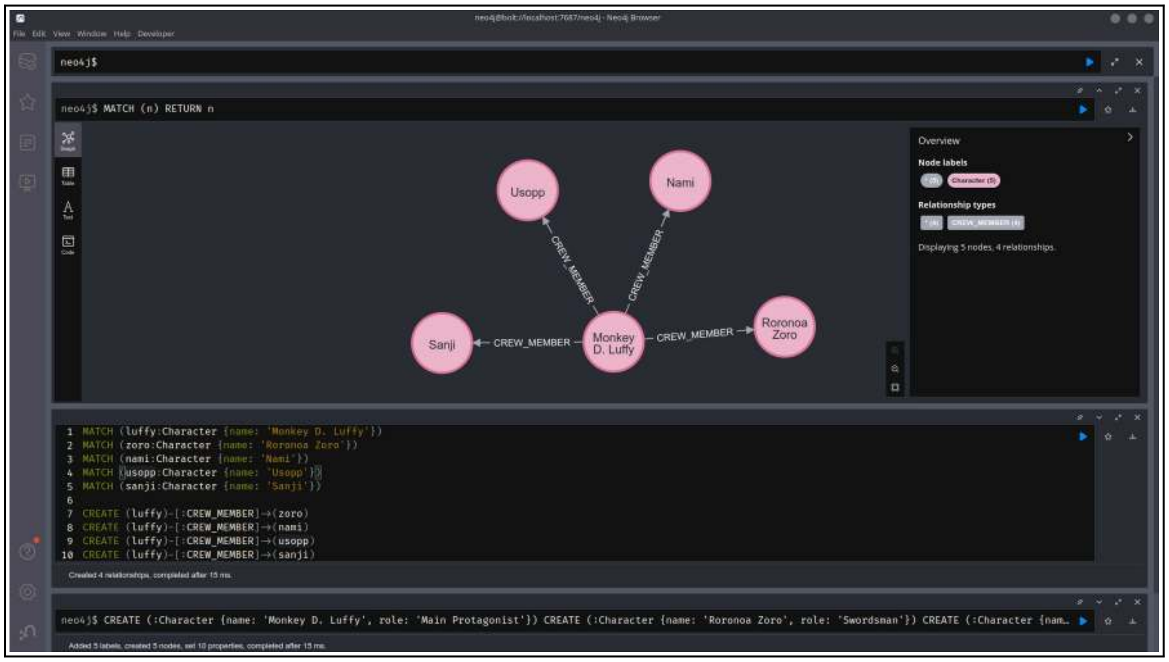
```
MATCH (luffy:Character {name: 'Monkey D. Luffy'})
MATCH (zoro:Character {name: 'Roronoa Zoro'})
MATCH (nami:Character {name: 'Nami'})
MATCH (usopp:Character {name: 'Usopp'})
MATCH (sanji:Character {name: 'Sanji'})

CREATE (luffy)-[:CREW_MEMBER]->(zoro)
CREATE (luffy)-[:CREW_MEMBER]->(nami)
CREATE (luffy)-[:CREW_MEMBER]->(usopp)
CREATE (luffy)-[:CREW_MEMBER]->(sanji)
```

Returning graph:

```
MATCH (n) RETURN n
```

(Interact with graph).

OUTPUT:



```
neo4j$
```

```
neo4j$ MATCH (n) RETURN n
```

```
MATCH (luffy:Character {name: 'Monkey D. Luffy'})
MATCH (zoro:Character {name: 'Roronoa Zoro'})
MATCH (nami:Character {name: 'Nami'})
MATCH (usopp:Character {name: 'Usopp'})
MATCH (sanji:Character {name: 'Sanji'})

CREATE (luffy)-[:CREW_MEMBER]→(zoro)
CREATE (luffy)-[:CREW_MEMBER]→(nami)
CREATE (luffy)-[:CREW_MEMBER]→(usopp)
CREATE (luffy)-[:CREW_MEMBER]→(sanji)
```

```
Created 4 relationships, completed after 15 ms.
```

```
neo4j$ CREATE (:Character {name: 'Monkey D. Luffy', role: 'Main Protagonist'}) CREATE (:Character {name: 'Roronoa Zoro', role: 'Swordsman'}) CREATE (:Character {nam…
```

```
Added 5 labels, created 5 nodes, set 10 properties, completed after 15 ms.
```

RESULT:

EX.NO: 03	READ AND WRITE DATA FROM GRAPH DATABASE
DATE:	

AIM:

To read and write data from graph database.

ALGORITHM:

STEP 1: Start.

STEP 2: Initiate the process by preparing for data management within the Neo4j graph database.

STEP 3: Utilize the `CREATE` command to seamlessly integrate new data into the graph database. This step involves the structured insertion of information, conforming to the predefined data model.

STEP 4: Employ the powerful `MATCH` clause to pinpoint specific data nodes or relationships within the graph. Further enhance the query by using the `RETURN` statement to elegantly present the desired information.

STEP 5: Stop.

PROGRAM:

Write data to graph database:

```
// nodes for characters
CREATE (:Character {name: 'Monkey D. Luffy', position: 'Captain'})
CREATE (:Character {name: 'Roronoa Zoro', position: 'Swordsman'})
CREATE (:Character {name: 'Nami', position: 'Navigator'})

// nodes for islands
CREATE (:Island {name: 'Dressrosa', type: 'Kingdom'})
CREATE (:Island {name: 'Alabasta', type: 'Kingdom'})
WITH 1 as dummy

// relationships between characters and islands
MATCH (luffy:Character {name: 'Monkey D. Luffy'})
MATCH (zoro:Character {name: 'Roronoa Zoro'})
MATCH (nami:Character {name: 'Nami'})
MATCH (dressrosa:Island {name: 'Dressrosa'})
MATCH (alabasta:Island {name: 'Alabasta'})

CREATE (luffy)-[:VISITS]->(dressrosa)
CREATE (zoro)-[:VISITS]->(alabasta)
CREATE (nami)-[:VISITS]->(dressrosa)
```

Reading data from graph database:

```
// Retrieve all characters
MATCH (c:Character)
RETURN c
```

Retrieve characters visiting a specific island:

```
MATCH (character)-[:VISITS]->(island:Island {name: 'Dressrosa'})
RETURN character
```

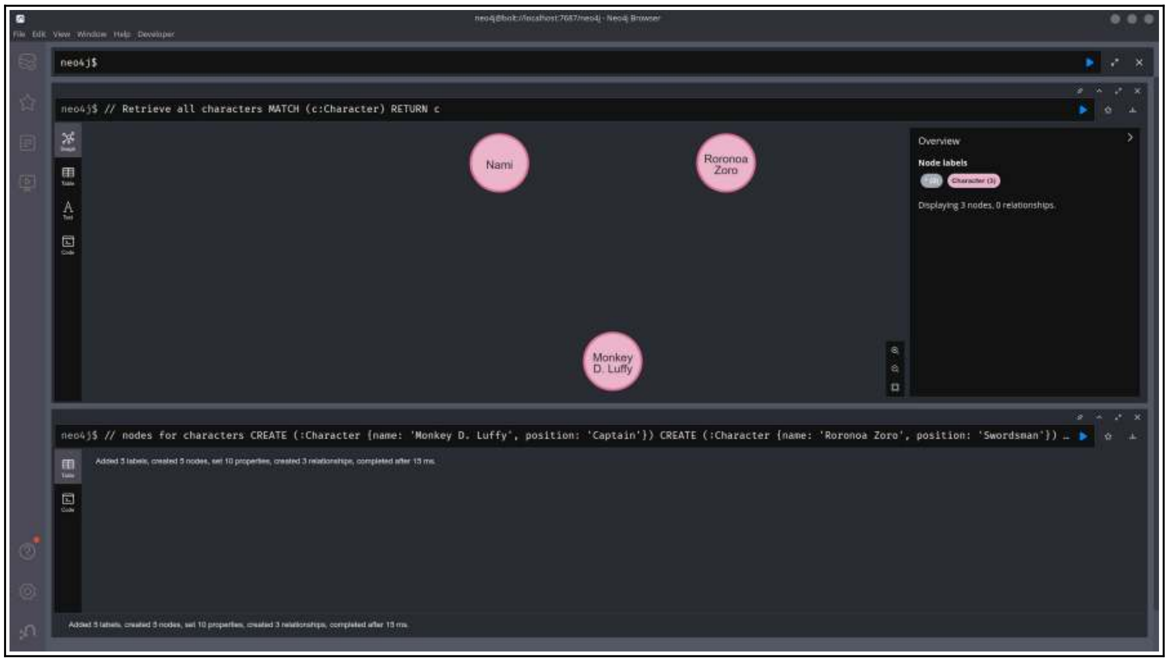
Updating data:

```
// Update character’s position
MATCH (luffy:Character {name: 'Monkey D. Luffy'})
SET luffy.position = 'Pirate King'
RETURN luffy
```

Deleting data:

```
MATCH (character:Character {name: 'Nami'})-[r]-()
DELETE character, r
```

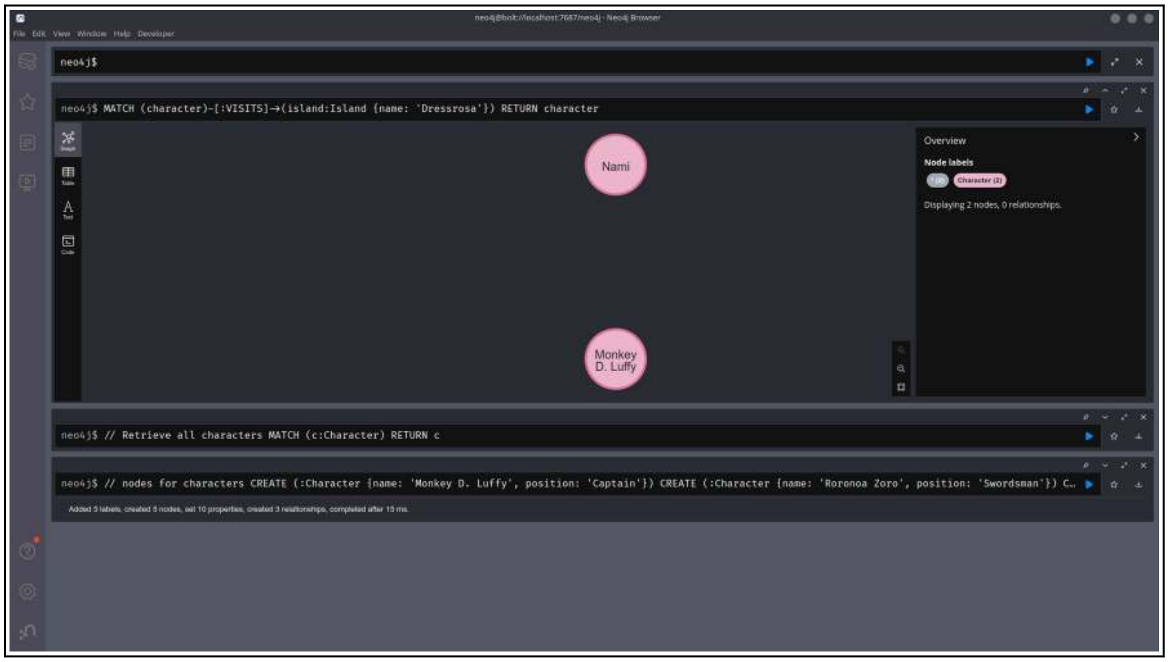
OUTPUT:



```
neo4j$ // Retrieve all characters MATCH (c:Character) RETURN c
```

```
neo4j$ // nodes for characters CREATE (:Character {name: 'Monkey D. Luffy', position: 'Captain'}) CREATE (:Character {name: 'Roronoa Zoro', position: 'Swordsman'}) …
```

```
Added 5 labels, created 5 nodes, set 10 properties, created 3 relationships, completed after 15 ms.
```

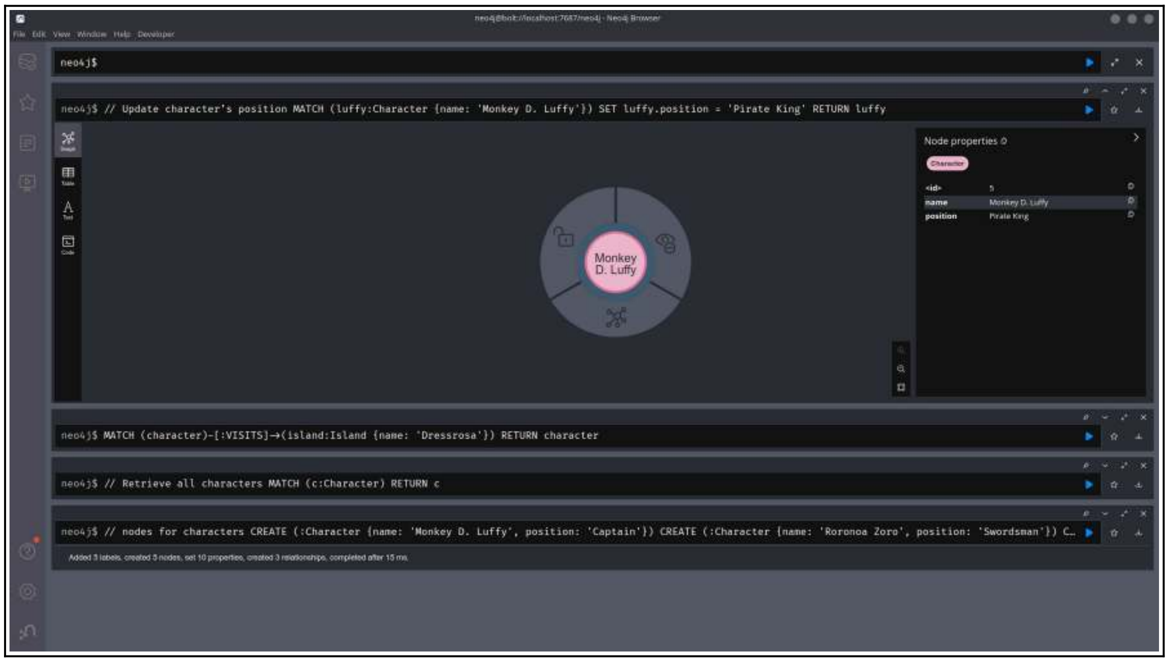


```
neo4j$ MATCH (character)-[:VISITS]→(island:Island {name: 'Dressrosa'}) RETURN character
```

```
neo4j$ // Retrieve all characters MATCH (c:Character) RETURN c
```

```
neo4j$ // nodes for characters CREATE (:Character {name: 'Monkey D. Luffy', position: 'Captain'}) CREATE (:Character {name: 'Roronoa Zoro', position: 'Swordsman'}) C…
```

```
Added 5 labels, created 5 nodes, set 10 properties, created 3 relationships, completed after 15 ms.
```



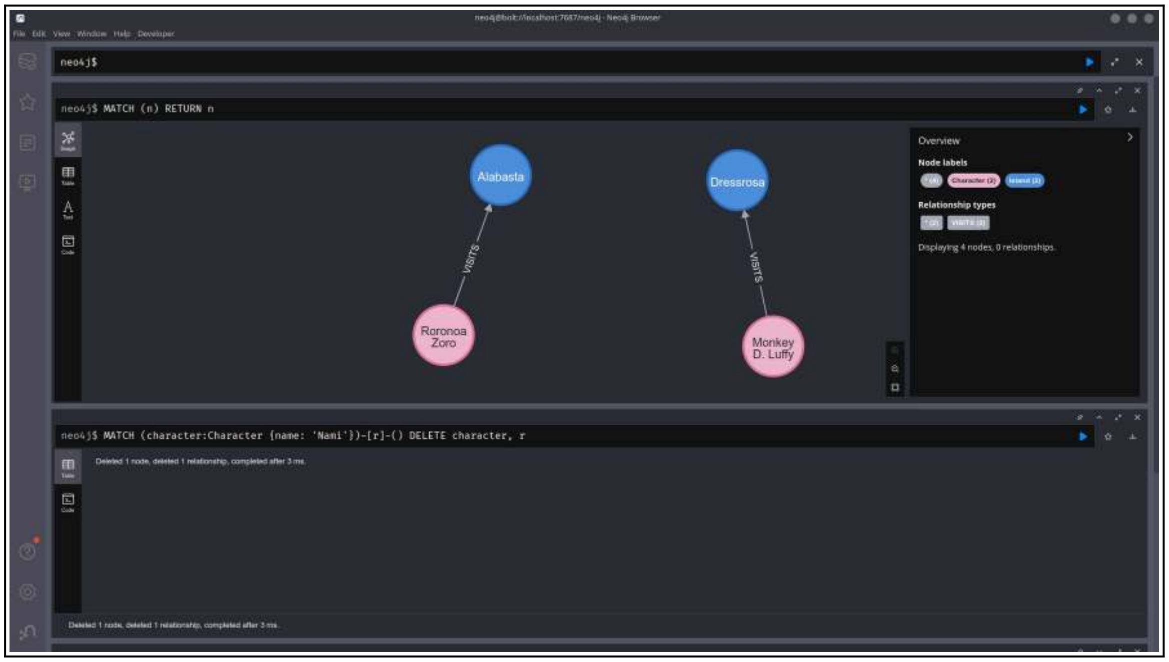
```
neo4j$

neo4j$ // Update character's position MATCH (luffy:Character {name: 'Monkey D. Luffy'}) SET luffy.position = 'Pirate King' RETURN luffy
```

Node properties

Character

<id>	5
name	Monkey D. Luffy
position	Pirate King

```
neo4j$ MATCH (character)-[:VISITS]→(island:Island {name: 'Dressrosa'}) RETURN character

neo4j$ // Retrieve all characters MATCH (c:Character) RETURN c

neo4j$ // nodes for characters CREATE (:Character {name: 'Monkey D. Luffy', position: 'Captain'}) CREATE (:Character {name: 'Roronoa Zoro', position: 'Swordsman'}) C…
Added 3 labels, created 5 nodes, set 10 properties, created 3 relationships, completed after 15 ms.
```



```
neo4j$

neo4j$ MATCH (n) RETURN n
```

Overview

Node labels

Relationship types

Displaying 4 nodes, 0 relationships.

```
neo4j$ MATCH (character:Character {name: 'Nami'})-[r]-() DELETE character, r
Deleted 1 node, deleted 1 relationship, completed after 3 ms.

Deleted 1 node, deleted 1 relationship, completed after 3 ms.
```

RESULT:

EX.NO: 04	**FIND “FRIEND OF FRIENDS” USING NEO4J**
DATE:	

AIM:

To find “friend of friends” using neo4j.

ALGORITHM:

STEP 1: Start.

STEP 2: Initiate the process by preparing for data management within the Neo4j graph database.

STEP 3: Write and execute Cypher queries to create nodes for characters.

STEP 4: Write and execute Cypher queries to establish friendship relationships between characters.

STEP 5: Write and execute Cypher queries to find "Friend of Friends" for a specific character.

STEP 6: Write and execute Cypher queries to visualize the graph in Neo4j Browser.

STEP 7: Explore the graph.

STEP 8: Stop.

PROGRAM:

Create nodes for characters:

// Create nodes for characters

```
CREATE (luffy:Character {name: 'Monkey D. Luffy'})
CREATE (zoro:Character {name: 'Roronoa Zoro'})
CREATE (nami:Character {name: 'Nami'})
CREATE (usopp:Character {name: 'Usopp'})
CREATE (sanji:Character {name: 'Sanji'})
```

// Create relationship between characters

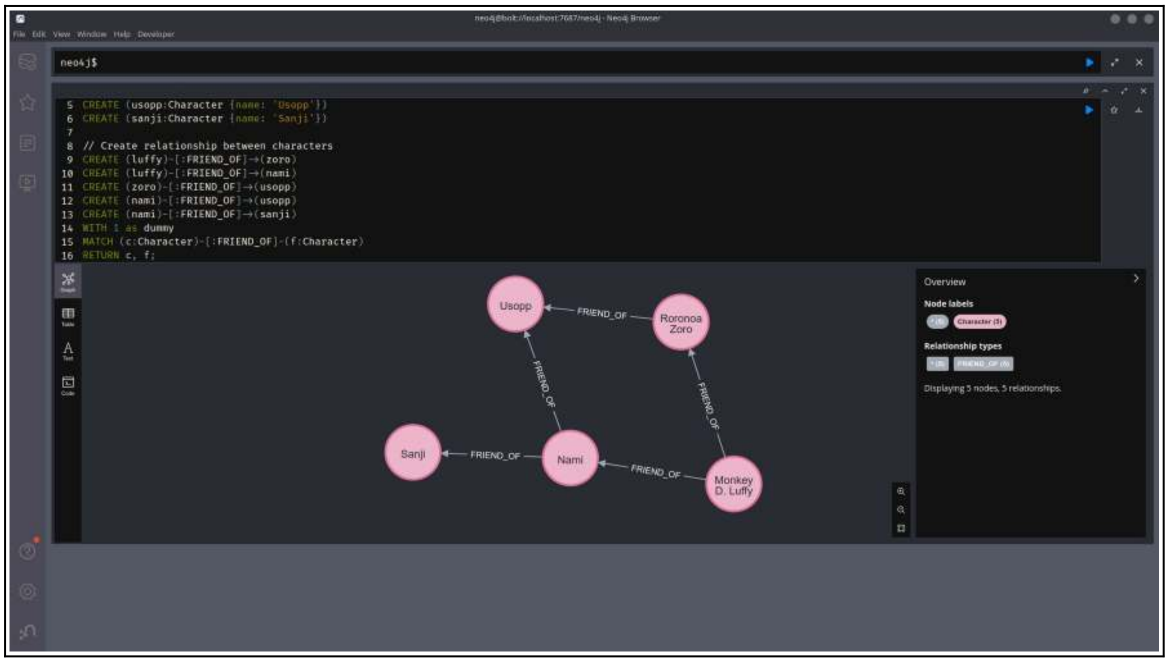
```
CREATE (luffy)-[:FRIEND_OF]→(zoro)
CREATE (luffy)-[:FRIEND_OF]→(nami)
CREATE (zoro)-[:FRIEND_OF]→(usopp)
CREATE (nami)-[:FRIEND_OF]→(usopp)
CREATE (nami)-[:FRIEND_OF]→(sanji)
WITH 1 as dummy
MATCH (c:Character)-[:FRIEND_OF]-(f:Character)
RETURN c, f;
```

Finding friends of friends

// Find friends of friends for luffy

```
MATCH (start:Character {name: 'Monkey D. Luffy'})-[:FRIEND_OF]→(friend)-
[:FRIEND_OF]→(friendOfFriend)
WHERE friendOfFriend <> start
RETURN DISTINCT friendOfFriend.name
```

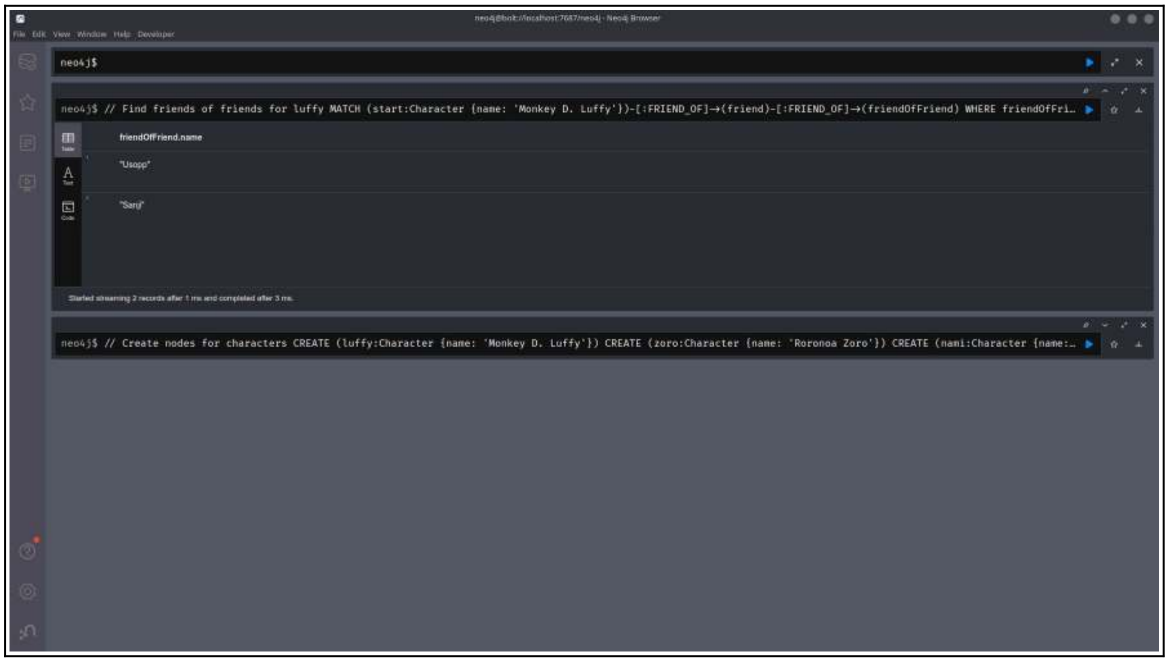
OUTPUT:



```
neo4j$
```

```
CREATE (usopp:Character {name: 'Usopp'})
CREATE (sanji:Character {name: 'Sanji'})

// Create relationship between characters
CREATE (luffy)-[:FRIEND_OF]→(zoro)
CREATE (luffy)-[:FRIEND_OF]→(nami)
CREATE (zoro)-[:FRIEND_OF]→(usopp)
CREATE (nami)-[:FRIEND_OF]→(usopp)
CREATE (nami)-[:FRIEND_OF]→(sanji)
WITH 1 as dummy
MATCH (c:Character)-[:FRIEND_OF]-(f:Character)
RETURN c, f;
```

Displaying 5 nodes, 5 relationships.



```
neo4j$

neo4j$ // Find friends of friends for luffy MATCH (start:Character {name: 'Monkey D. Luffy'})-[:FRIEND_OF]→(friend)-[:FRIEND_OF]→(friendOfFriend) WHERE friendOfFri…
```

friendOfFriend.name
"Usopp"
"Sanji"

Started streaming 2 records after 1 ms and completed after 3 ms.

```
neo4j$ // Create nodes for characters CREATE (luffy:Character {name: 'Monkey D. Luffy'}) CREATE (zoro:Character {name: 'Roronoa Zoro'}) CREATE (nami:Character {name:…
```

RESULT:

EX.NO: 05	**IMPLEMENT SECURE SEARCH IN SOCIAL MEDIA**
DATE:	

AIM:

To implement secure search in social media.

ALGORITHM:

STEP 1: Start.

STEP 2: Create a new project directory with subdirectories and files.

STEP 3: Import necessary modules in app.py.

STEP 4: Flask : the web framework used for building the web application. render_template : A function from Flask that renders HTML templates. Re: Stands for Regular Expression used to match string patterns.

STEP 5: Define sample user data.

STEP 6: Define a search query sanitizer to securely get input from user.

STEP 7: Return the search results.

STEP 8: Stop.

PROGRAM:

app.py:

```
from flask import Flask, render_template, request, redirect, url_for
import re

app = Flask(__name__)

user_data = {
    "luffy": {"name": "Monkey D. Luffy", "role": "Captain", "goal": "King of pirates"},
    "zoro": {"name": "Roronoa Zoro", "role": "Swordsman", "goal": "World's greatest
swordsman"},
    "nami": {"name": "Nami", "role": "Navigator", "goal": "Map the entire world"},
    "usopp": {"name": "Usopp", "role": "Sniper", "goal": "Brave warrior of the sea"},
    "sanji": {"name": "Sanji", "role": "Chef", "goal": "Find the All Blue"},
    "chopper": {"name": "Tony Tony Chopper", "role": "Doctor", "goal": "Cure any
disease"},
    "robin": {"name": "Nico Robin", "role": "Archaeologist", "goal": "Learn the true
history"},
    "franky": {"name": "Franky", "role": "Shipwright", "goal": "Build the best ship"},
    "brook": {"name": "Brook", "role": "Musician", "goal": "Reunite with Laboon"},
    "jinbe": {"name": "Jinbe", "role": "Helmsman", "goal": "Achieve true justice"},
}

def query_sanitizer(query):
    sanitized_query = re.sub(r'[^\w\s]', '', query.strip()) or "query"
    return sanitized_query

@app.route("/")
def index():
    return render_template("index.html", users=user_data.items())

@app.route("/search", methods=["POST"])
def search():
    query = request.form.get("search_query")
    return redirect(url_for("search_results", query=query_sanitizer(query)))

@app.route("/search/<query>")
def search_results(query):
    results = [(key, user) for key, user in user_data.items() if query.lower() in
user['name'].lower()]
    return render_template("search_results.html", query=query, results=results)
```

```
if __name__ == '__main__':
    app.run(debug=True)
```

templates/index.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Secure Search</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
</head>
<body>
    <header>
        <form action="/search" method="post">
            <label for="search_query">Search:</label>
            <input type="text" name="search_query" id="search_query" placeholder="Type
here.." required>
            <button type="submit">Search</button>
        </form>
    </header>
    <div class="users-container">
        <h3>Your Friends:</h3>
        <br />
        <div class="users">
            <ul>
                {% for member, details in users %}
                    <li>{{ details.name }}</li>
                    <br />
                {% endfor %}
            </ul>
        </div>
    </div>
</body>
</html>
```

templates/search_results.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Search Results</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
</head>
<body>
    <h1>Search Results for "{{ query }}"</h1>
    <div class="results-container">
        <section>
            <ul>
                {% for member, details in results %}
                    <li>
                        <h3>{{ details.name }}</h3>
                        <ul>
                            <li><span>Role: </span>{{ details.role }}</li>
                            <li><span>Dream: </span>{{ details.goal }}</li>
                        </ul>
                    </li>
                {% endfor %}
            </ul>
        </section>
    </div>
</body>
</html>
```

static/styles.css:

```
* {
    padding: 0px;
    margin: 0px;
}

.form-body {
    display: flex;
    align-items: center;
    place-content: center;
    place-items: center;
    margin-top: 5%;
    flex-direction: column;
    padding: 30px;
}

.form-container {
    display: flex;
}
```

```
.form-container form {
    display: flex;
    flex-direction: column;
}

.form-container form input[type="text"], input[type="email"], input[type="password"] {
    width: 80%;
    border: none;
    border-bottom: 1px solid grey;
    height: 30px;
    outline: none;
}

.checkboxes {
    display: flex;
    flex-direction: row;
    margin-top: 10px;
    padding: 3px;
}

select {
    margin-bottom: 10px;
}

.form-body h2 {
    margin-bottom: 20px;
}

button {
    width: 90px;
}

.result-body {
    padding: 30px;
}
```

PROJECT STRUCTURE: (Just For Ref.)

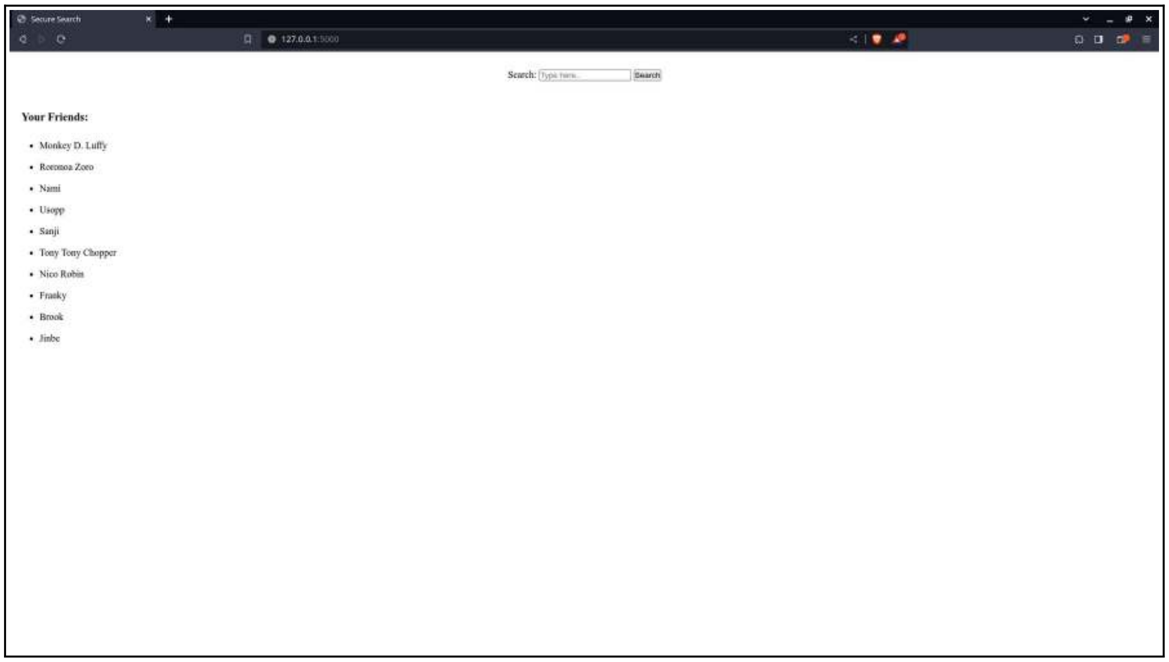
```
.
├── app.py
├── static
│   └── styles.css
└── templates
    ├── index.html
    └── search_results.html
```

2 directories, 4 files

Execution (Python (Programming Language)):

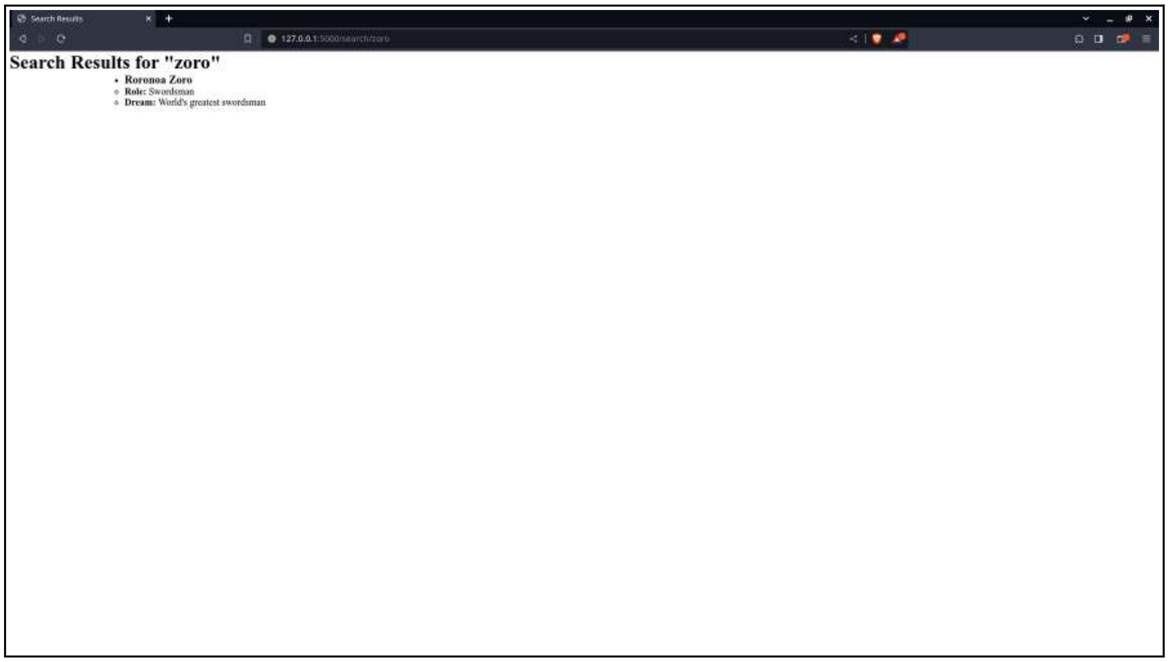
```
$ pip3 install Flask
$ python3 app.py
```

OUTPUT:



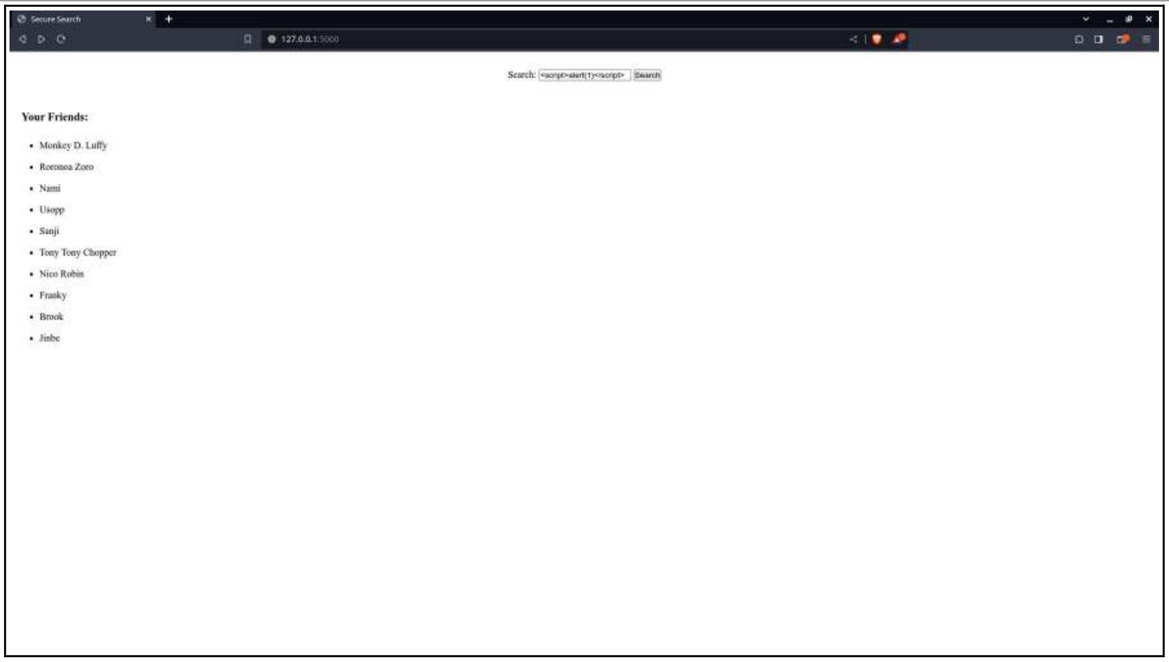
Your Friends:

- Monkey D. Luffy
- Roronoa Zoro
- Nami
- Usopp
- Sanji
- Tony Tony Chopper
- Nico Robin
- Franky
- Brook
- Jinbe



Search Results for "zoro"

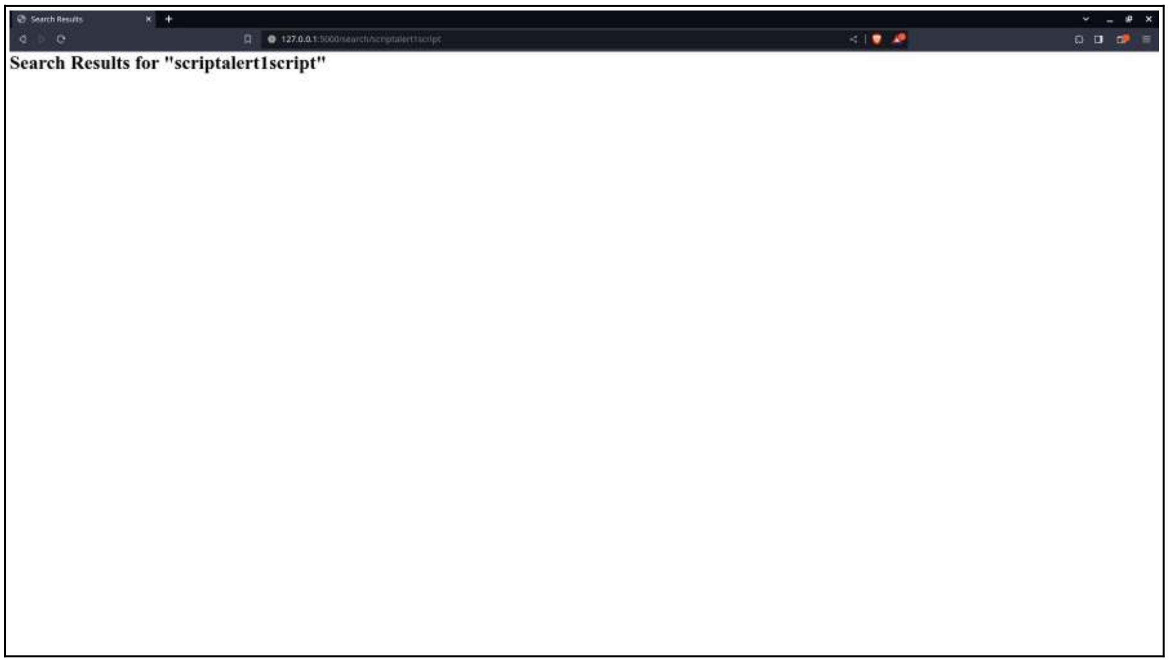
- **Roronoa Zoro**
  - **Role:** Swordsman
  - **Dream:** World's greatest swordsman



Search: <script>alert(1)</script> Search

Your Friends:

- Monkey D. Luffy
- Roronoa Zoro
- Nami
- Usopp
- Sanji
- Tony Tony Chopper
- Nico Robin
- Franky
- Brook
- Jinbe



Search Results for "scriptalert1script"

RESULT:

EX.NO: 06	CREATE A SIMPLE SECURITY & PRIVACY DETECTOR
DATE:	

AIM:

To create a simple security and privacy detector.

ALGORITHM:

STEP 1: Start.

STEP 2: Create a new project directory with subdirectories and files.

STEP 3: Import necessary modules in app.py. Flask, re

STEP 4: Define index and result functions.

STEP 5: Get form data and calculate ratio in ‘result’ function.

STEP 6: Return the results.

STEP 7: Stop.

PROGRAM:

app.py:

```
from flask import Flask, render_template, request
import re

app = Flask(__name__)

@app.route("/")
def index():
    return render_template("index.html")

@app.route("/result", methods=["POST"])
def result():
    username = request.form.get("username")
    password = request.form.get("password")
    _2fa = request.form.get("2fa")
    private = request.form.get("private")
    fields_to_check = ["priv_activity", "priv_pfp", "priv_bio", "priv_call"]
    privacy_values = {field: request.form.get(field) for field in fields_to_check}

    security_level = sum([
        len(password) >= 8,
        bool(re.compile(r'[^a-zA-Z0-9\s]').search(password)),
        bool(re.compile(r'\d').search(password)),
        bool(_2fa),
    ])

    privacy_level = sum([
        bool(private),
        sum(2 if value == "nobody" else 1 for value in privacy_values.values() if value ==
"nobody"),
        sum(1 for value in privacy_values.values() if value == "friends"),
    ])

    sec_ratio = "{:.2f}/10".format((security_level / 4) * 10)
    priv_ratio = "{:.2f}/10".format((privacy_level / 9) * 10)

    data = {
        "username": username,
        "sec_ratio": sec_ratio,
        "priv_ratio": priv_ratio
    }
    return render_template("detected_result.html", data=data)

if __name__ == "__main__":
    app.run(debug=True)
```

templates/index.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Security and Privacy Detector</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
</head>
<body class="form-body">
    <h2>Security and Privacy Detector</h2>

    <div class="form-container">
        <form action="/result" method="post">
            <input type="text" placeholder="Username" name="username" required>
            <input type="text" placeholder="First Name" name="first_name">
            <input type="text" placeholder="Last Name" name="last_name">
            <input type="email" placeholder="Email" name="email">

            <!-- Security -->
            <input type="password" placeholder="Password" name="password" required>

            <div class="checkboxes">
                <input type="checkbox" value="2fa" name="2fa" id="2fa">
                <label for="2fa">Two-Step Verification</label>
            </div>

            <!-- Privacy -->
            <div class="checkboxes">
                <input type="checkbox" value="private" id="private" name="private">
                <label for="private">Private Account</label>
            </div>

            <label for="priv_email">Who can see my email</label>
            <select name="priv_email" id="priv_email">
                <option value="everybody">Everybody</option>
                <option value="friends">My Friends</option>
            <option value="nobody">Nobody</option>
          </select>

          <label for="priv_activity">Last seen and online</label>
          <select name="priv_activity" id="priv_activity">
            <option value="everybody">Everybody</option>
            <option value="friends">My Friends</option>
            <option value="nobody">Nobody</option>
          </select>

          <label for="priv_pfp">Who can see my profile photo</label>
          <select name="priv_pfp" id="priv_pfp">
            <option value="everybody">Everybody</option>
            <option value="friends">My Friends</option>
            <option value="nobody">Nobody</option>
          </select>

          <label for="priv_bio">Who can see my bio</label>
          <select name="priv_bio" id="priv_bio">
            <option value="everybody">Everybody</option>
            <option value="friends">My Friends</option>
            <option value="nobody">Nobody</option>
          </select>

          <label for="priv_call">Who can call me</label>
          <select name="priv_call" id="priv_call">
            <option value="everybody">Everybody</option>
            <option value="friends">My Friends</option>
            <option value="nobody">Nobody</option>
          </select>

          <button type="submit">Detect</button>
        </form>
    </div>
</body>
</html>
```

templates/detected_result.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Security and Privacy Results</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
</head>
<body class="result-body">
    <h3>Hello, {{data.username}}...</h3>

    <br>

    <ul>
        <li><span style="font-weight: 700;">Your security level:</span>
{{data.sec_ratio}}</li>
        <li><span style="font-weight: 700;">Your privacy level :</span>
{{data.priv_ratio}}</li>
    </ul>
</body>
</html>
```

static/styles.css:

```
* {
    padding: 0px;
    margin: 0px;
}

.form-body {
    display: flex;
    align-items: center;
    place-content: center;
    place-items: center;
    margin-top: 5%;
    flex-direction: column;
    padding: 30px;
}

.form-container {
    display: flex;
}

.form-container form {
    display: flex;
    flex-direction: column;
}
```

```
.form-container form input[type="text"], input[type="email"], input[type="password"] {
    width: 80%;
    border: none;
    border-bottom: 1px solid grey;
    height: 30px;
    outline: none;
}

.checkboxes {
    display: flex;
    flex-direction: row;
    margin-top: 10px;
    padding: 3px;
}

select {
    margin-bottom: 10px;
}

.form-body h2 {
    margin-bottom: 20px;
}

button {
    width: 90px;
}

.result-body {
    padding: 30px;
}
```

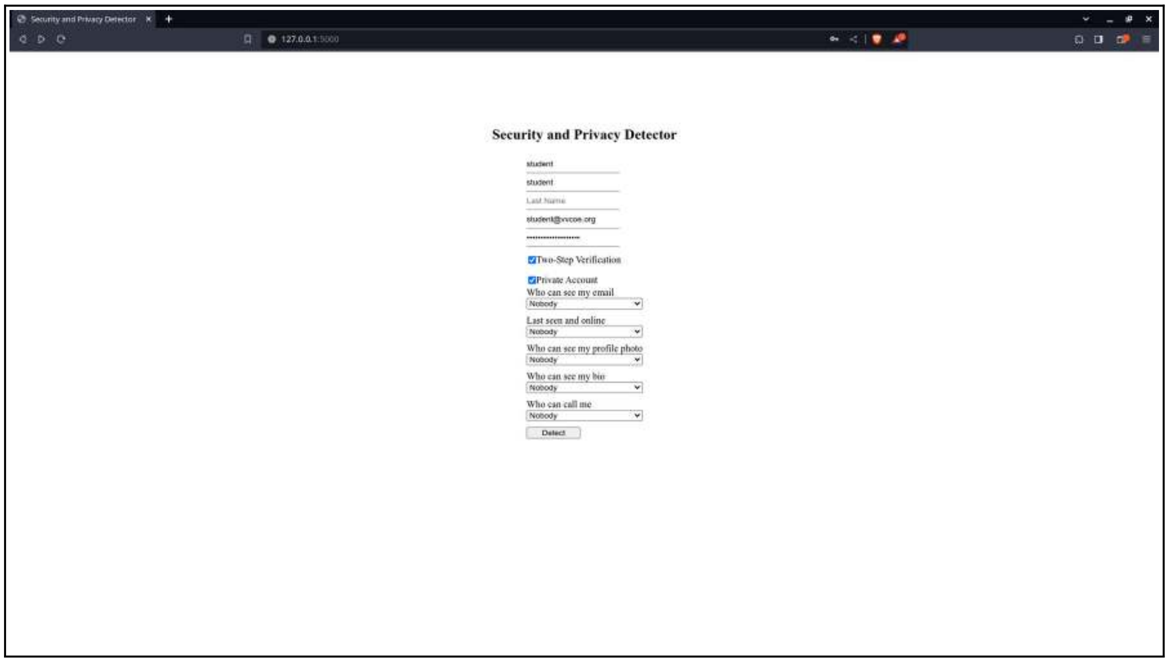
PROJECT STRUCTURE: (Just For Ref.)

```
.
├── app.py
├── static
│   └── styles.css
└── templates
    ├── detected_result.html
    └── index.html

2 directories, 4 files
```

Execution (Python (Programming Language)):

```
$ pip3 install Flask
$ python3 app.py
```

OUTPUT:



Security and Privacy Detector

student

student

Last Name

student@vvcoe.org

••••••••••••••••••

✓ Two-Step Verification

✓ Private Account

Who can see my email: Nobody

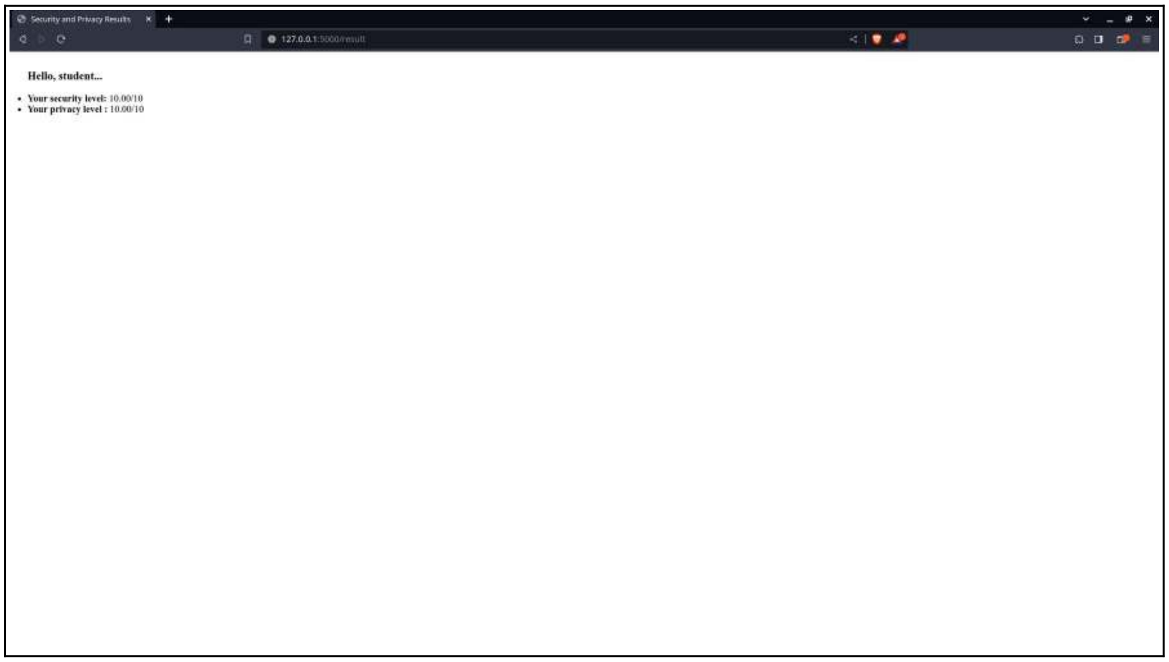
Last seen and online: Nobody

Who can see my profile photo: Nobody

Who can see my bio: Nobody

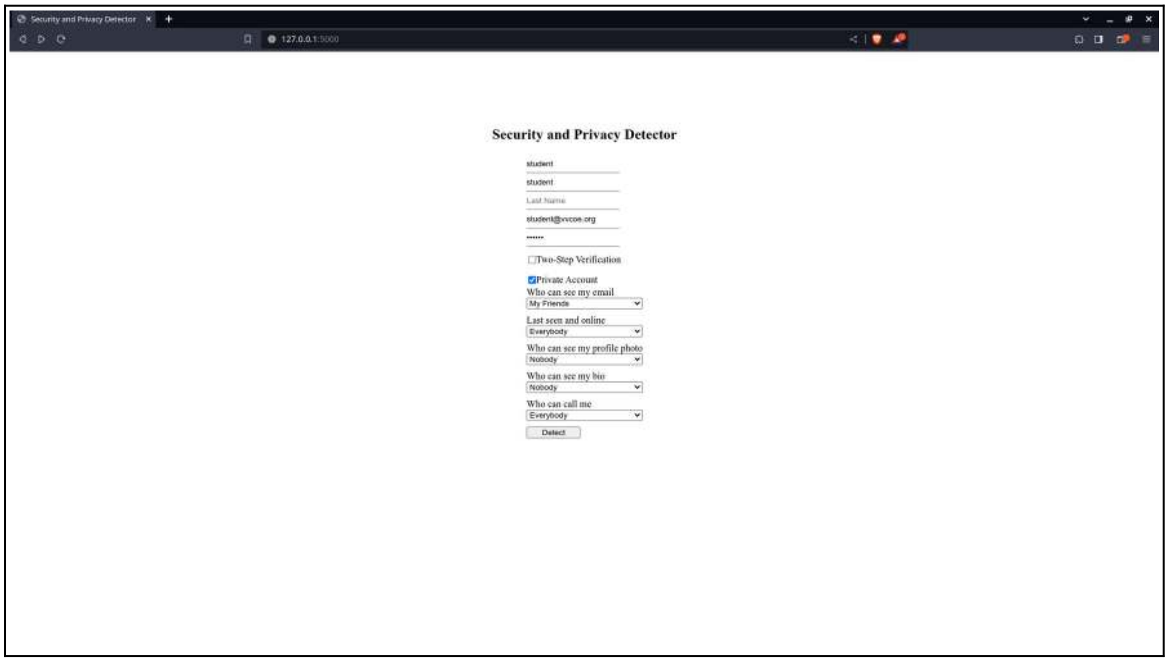
Who can call me: Nobody

Detect



Hello, student...

- **Your security level:** 10.00/10
- **Your privacy level :** 10.00/10



Security and Privacy Detector

student

student

Last Name

student@vvcoe.org

••••••

☐ Two-Step Verification

☑ Private Account

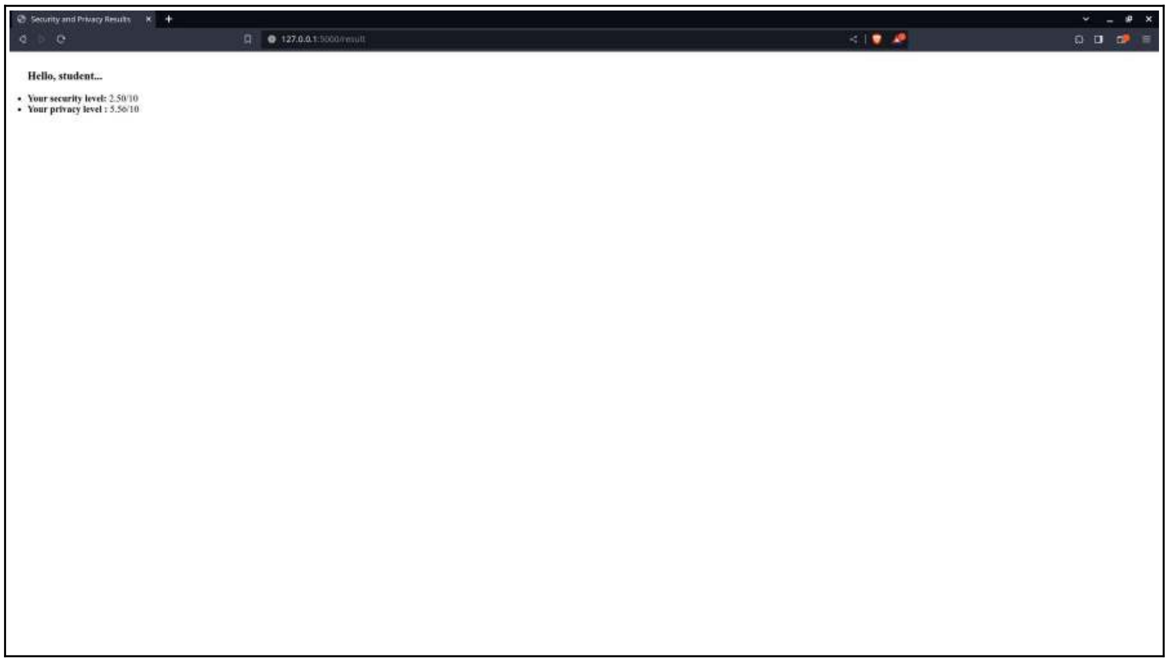
Who can see my email: My Friends

Last seen and online: Everybody

Who can see my profile photo: Nobody

Who can see my bio: Nobody

Who can call me: Everybody



Hello, student...

- **Your security level:** 2.50/10
- **Your privacy level :** 5.56/10

RESULT:

Viva Questions

1. Define semantic Web.

The Semantic Web is an extension of the Web through standards by the World Wide Web Consortium (W3C). The standards promote common data formats and exchange protocols on the Web, most fundamentally the Resource Description Framework (RDF).

2. What is the key concept of network analysis?

There are several key terms associated with social network analysis. They are density,centrality, indegree,outdegree,Sociogram.

3. Give the limitations of current Web.

The current Web has its limitations when it comes to:

1. Finding relevant information
2. Extracting relevant information
3. Combining and reusing information

4.What is social network analysis?

Social network analysis [SNA] is the mapping and measuring of relationships and flows between people, groups, organizations, computers, URLs, and other connected information/knowledge entities. The nodes in the network are the people and groups while the links show relationships or flows between the nodes.

5.What is a Web Community?

A web community is a web site (or group of web sites) where specific content or links are only available to its members. A web community may take the form of a social network service, an Internet forum, a group of blogs, or another kind of social software web application.

6.what is Neo4j?

Neo4j is an open source NOSQL graph database, implemented in Java. It saves data structured in graphs rather than in tables.

7. For what Neo4j is widely used for?

Neo4j is widely used for

- Highly connected data – Social Network
- Recommendation- (e-commerce)
- Path Finding
- Data First Schema (bottom-up)
- Schema Evolution
- A* (Least Cost Path)

8. Mention what is the difference between Neo4j graph database and MySQL?

Neo4j	**MySQL**
• It consists of vertices and edges. Each vertex or node represent a key value or attribute	• In relational databases, attributes are appended in plain table format
• It is possible to store dynamic content like images, videos, audio,	• In relational databases, such as MySQL, it’s difficult to store videos, audios, images,
• It has the capability for deep search into the database without affecting the performance along with efficient timing	• It takes longer time for database search and also inconvenient compared to neo4j

• We can relate any two objects in neo4j by the mean of making relationship between any two nodes	• It lacks relationship and difficult to use them for connected graphs and data

9.Mention some of the important characteristics of neo4j?

Some important characteristics of neo4j includes

- Materializing of relationship at creation time, resulting in no penalties for runtime queries
- Continuous time traversals for relationship in the graph both in breadth and depth due to double linking on the storage level between nodes and relationships
- Relationship in Neo4j is fast and make it possible to materialize and use new relationships later on to "shortcut" and speed up the domain data when new requirement arise

10. Explain the role of building blocks like Nodes, Relationships, Properties and Labels in Neo4j?

The role of building blocks

- Nodes: They are entities
- Relationship: It connects entities and structure domain
- Properties: It consists of meta-data and attributes
- Labels: It group nodes by role

11. Mention which query language does Neo4j use and what is consist of?

Neo4j uses Cypher query language, which is unique to Neo4j. Traversing the graph requires to know where you want to begin (Start), the rules that allow traversal (Match) and what data you are expecting back (Return). The basic query consists of

- **START n**
- **MATCH n-[r]- m**
- **RETURN r;**

12.Mention how files are stored in Neo4j?

Neo4j stores graph data in a number of different store files, and each store file consists of the data for a specific part of the graph for example relationships, nodes, properties etc. for example Neostore.nodestore.db, neostore.propertystore.db and so on.

13. Mention what are the different types of object caches in Neo4j?There are two different types of object caches in Neo4j

- **Reference Caches:** With this cache, Neo4j will use as much as allocated JVM heap memory as it can hold nodes and relationships
- **High-performance Caches:** It get assigned a certain maximum amount of space on the JVM heap and will delete objects whenever it grows bigger than that.

14.Define Neo4j?

It is one of the most popular open-source free NOSQLGraph DBMS (database management system) developed by Neo4j, Inc. It written in Java and Scala..The development of Neo4j was started in 2003, it has been publicly available since 2007. The source code and issue tracking of Neo4j is available on GitHub, with support readily available on Stack Overflow and the Neo4j Google group

15.What is a graph database?

A Neo4j graph database stores nodes and relationships instead of tables or documents. Data is stored just like you might sketch ideas on a whiteboard. Your data is stored without restricting it to a pre-defined model, allowing a very flexible way of thinking about and using it.

16.Where and how is Neo4j used?

Neo4j is used today by thousands of startups, educational institutions, and large enterprises in all sectors including financial services, government, energy, technology, retail, and manufacturing. From innovative new technology to driving businesses, users are generating insights with graph, generating new revenue, and improving their overall efficiency.

17.RDBMS Vs Graph Database

Following is the table which compares Relational databases and Graph databases.

Sr.No	RDBMS	Graph Database
1	Tables	Graphs
2	Rows	Nodes
3	Columns and Data	Properties and its values
4	Constraints	Relationships
5	Joins	Traversal

18.List some Popular Graph Databases

Neo4j is a popular Graph Database. Other Graph Databases are Oracle NoSQL Database, OrientDB, HypherGraphDB, GraphBase, InfiniteGraph, and AllegroGraph.

19.What is a Graph Database?

A graph is a pictorial representation of a set of objects where some pairs of objects are connected by links. It is composed of two elements - nodes (vertices) and relationships (edges).

Graph database is a database used to model the data in the form of graph. In here, the nodes of a graph depict the entities while the relationships depict the association of these nodes.

20.Why Graph Databases?

Nowadays, most of the data exists in the form of the relationship between different objects and more often, the relationship between the data is more valuable than the data itself.

Relational databases store highly structured data which have several records storing the same type of data so they can be used to store structured data and, they do not store the relationships between the data.

Unlike other databases, graph databases store relationships and connections as first-class entities.