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: (PICTURE)

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.

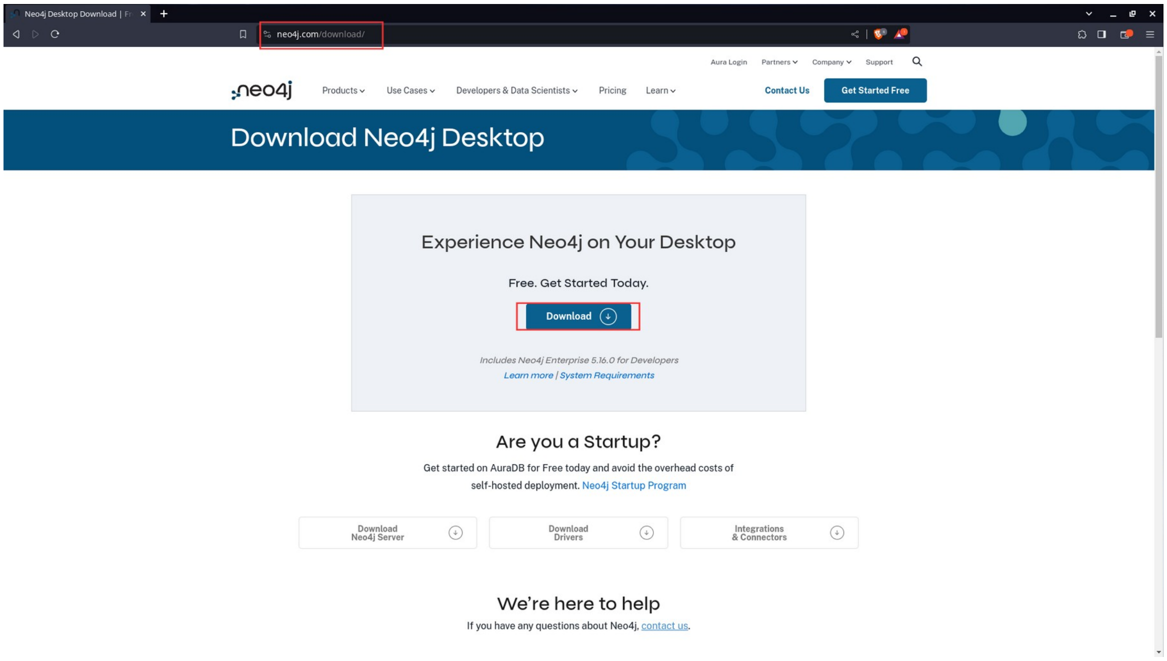
STEP 4: Create a new network model and retrieve the graph.

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

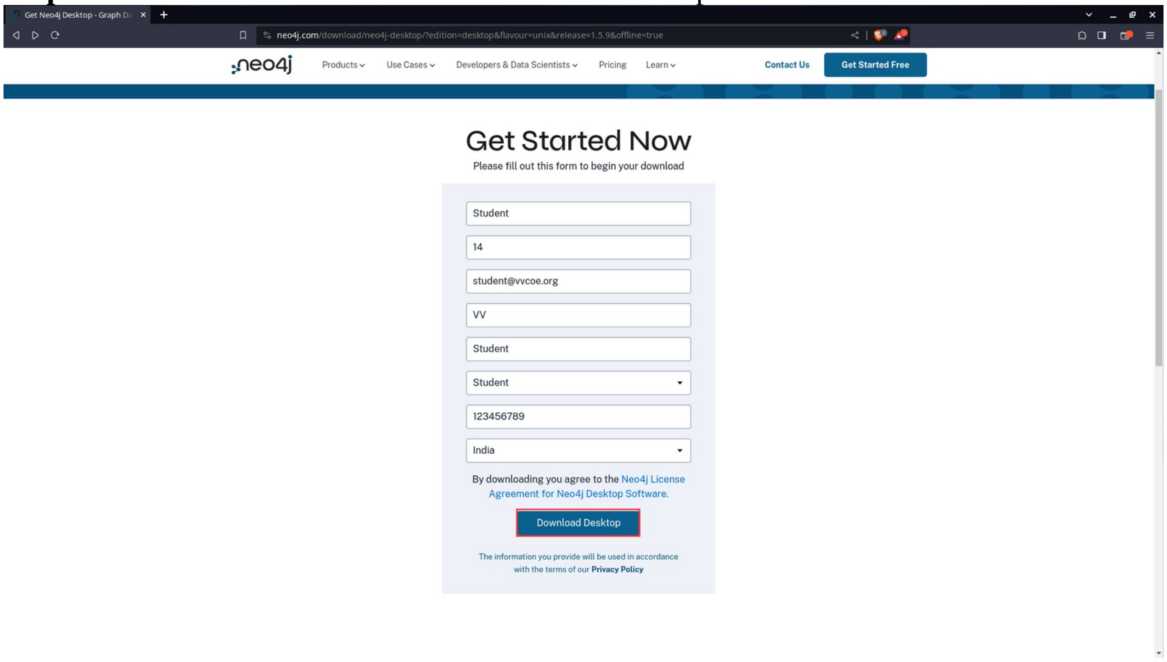
Are you a Startup?

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 Neo4j Desktop - Graph Da
neo4j.com/download/neo4j-desktop/?edition=desktop&flavour=unix&release=1.5.9&offline=true
neo4j
Products
Use Cases
Developers & Data Scientists
Pricing
Learn
Contact Us
Get Started Free

Get Started Now

Please fill out this form to begin your download

Student

14

student@vvcoe.org

VV

Student

Student

123456789

India

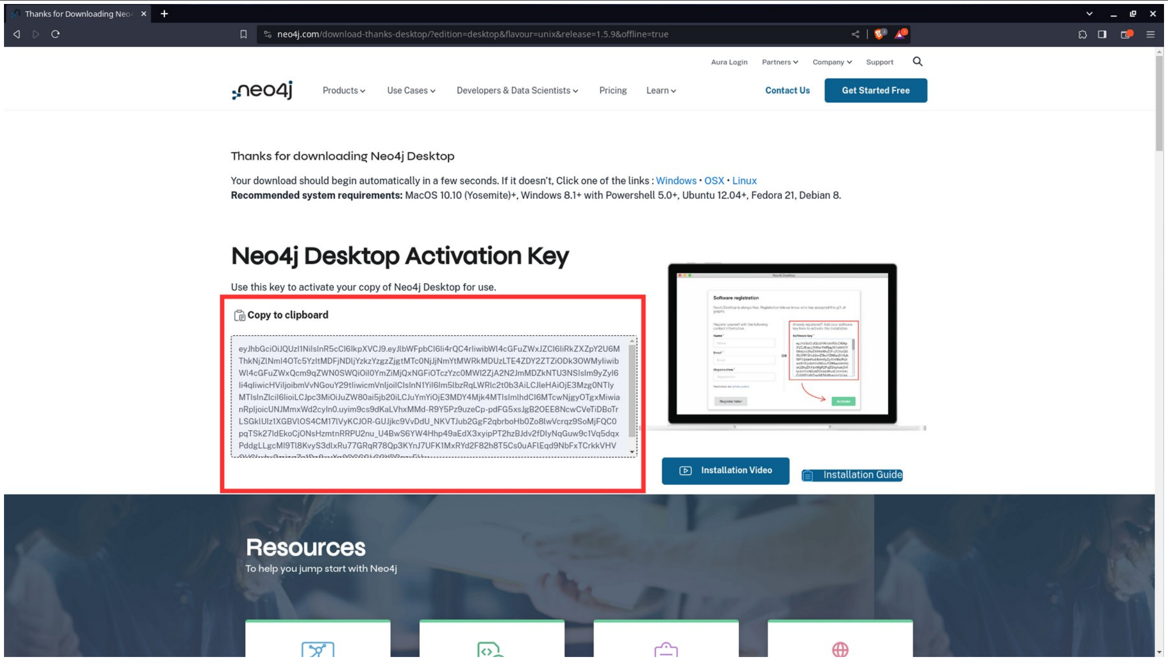
By downloading you agree to the Neo4j License Agreement for Neo4j Desktop Software.

Download Desktop

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.

```
eyJhbGciOiJQUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6Ii4rQC4rIiwibWl4cGFuZWxJZCI6IiRkZXZpY2U6M
ThkNjZlNmI4OTc5YzItMDFjNDljYzkzYzgzZjgtMTc0NjJjNmYtMWRkMDUzLTE4ZDY2ZTZiODk3OWMyIiwib
Wl4cGFuZWxQcm9qZWN0SWQiOiI0YmZiMjQxNGFiOTczYzc0MWI2ZjA2N2JmMDZkNTU3NSIsIm9yZyI6
Ii4qIiwicHViIjoibmVvNGouY29tIiwicmVnIjoiIClsInN1YiI6Im5lbzRqLWRlc2t0b3AiLCJleHAiOjE3Mzg0NTIy
MTIsInZlciI6IioiLCJpc3MiOiJuZW80ai5jb20iLCJuYmYiOjE3MDY4Mjk4MTIsImlhdCI6MTcwNjgyOTgxMiwia
nRpIjoicUNJMmxWd2cyIn0.uyim9cs9dKaLVhxMMd-R9Y5Pz9uzeCp-pdFG5xsJgB2OEE8NcwCVeTiDBoTr
LSGklUIz1XGBVlOS4CM17lVyKCJOR-GUJjkc9VvDdU_NKVTJub2GgF2qbrboHb0Zo8lwVcrqz9SoMjFQC0
pqTSk27IdEkoCjONsHzmtnRRPU2nu_U4BwS6YW4Hhp49aEdX3xyipPT2hzBJdv2fDlyNqGuw9c1Vq5dqx
PddgLLgcMI9Tl8KvyS3dlxRu77GRqR78Qp3KYnJ7UFK1MxRYd2F82h8T5Cs0uAFlEqd9NbFxTCrkkVHV
```

Resources

To help you jump start with Neo4j

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

```
$ ls -al | grep “neo4j”
```

Change the permissions to make it executable.

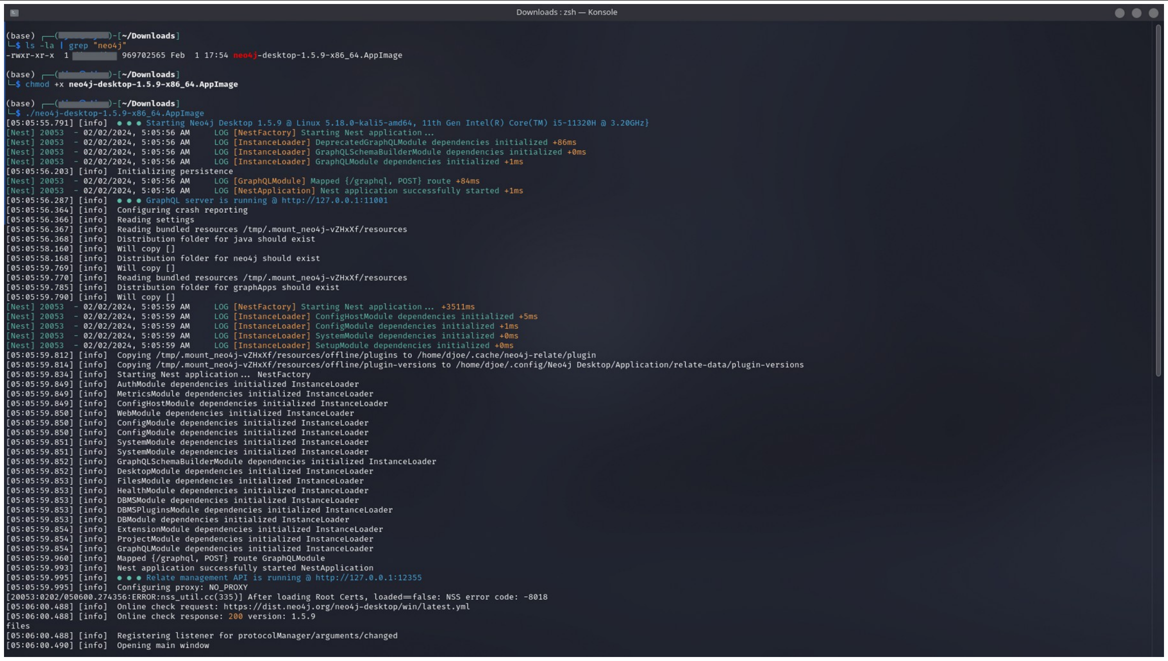
```
$ chmod +x neo4j-desktop-1.5.9-x86_64.AppImage
```

(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
```

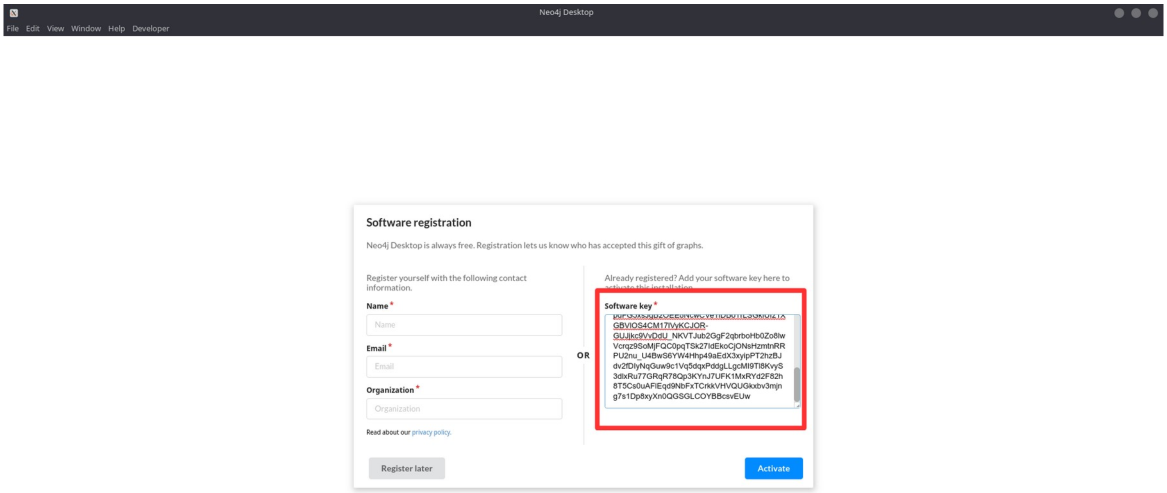
(Note: The “neo4j-desktop-1.5.9-x86_64.AppImage” may change according to the version you downloaded. Verify your AppImage name using “ls -al | grep “neo4j”)



Downloads : zsh — Konsole

```
(base) ┌──([redacted])-[~/Downloads]
└─$ ls -la | grep "neo4j"
-rwxr-xr-x  1 [redacted] 969702565 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 []
[05:05:59.770] [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
[05:05:59.854] [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
[05:06:00.490] [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.

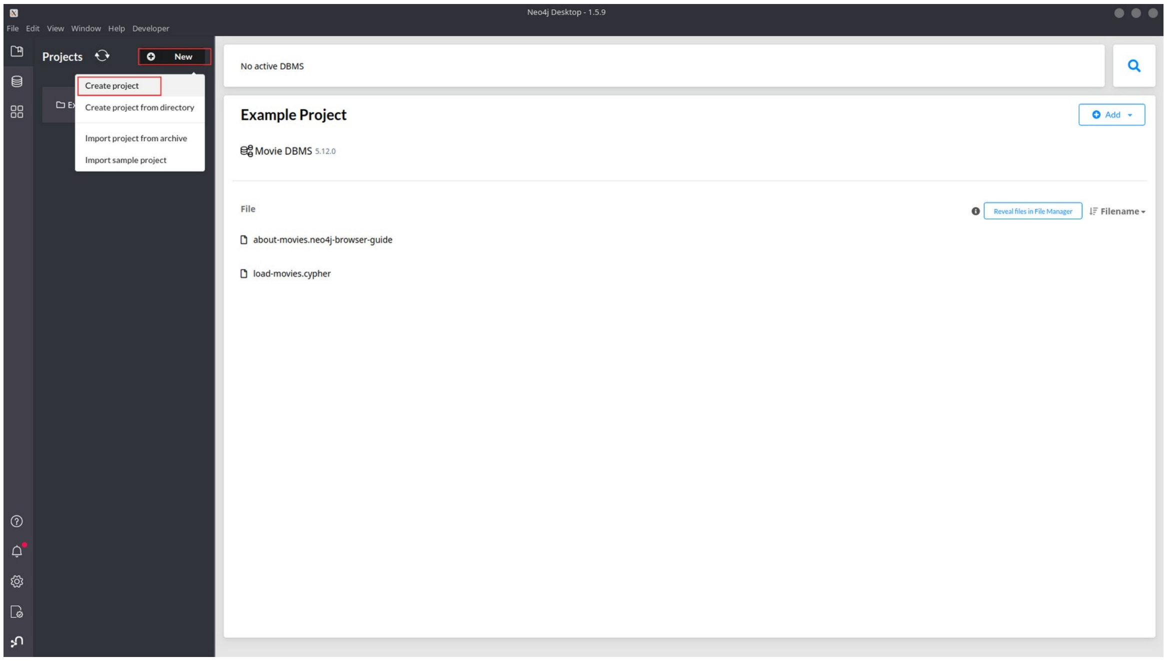
OR

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

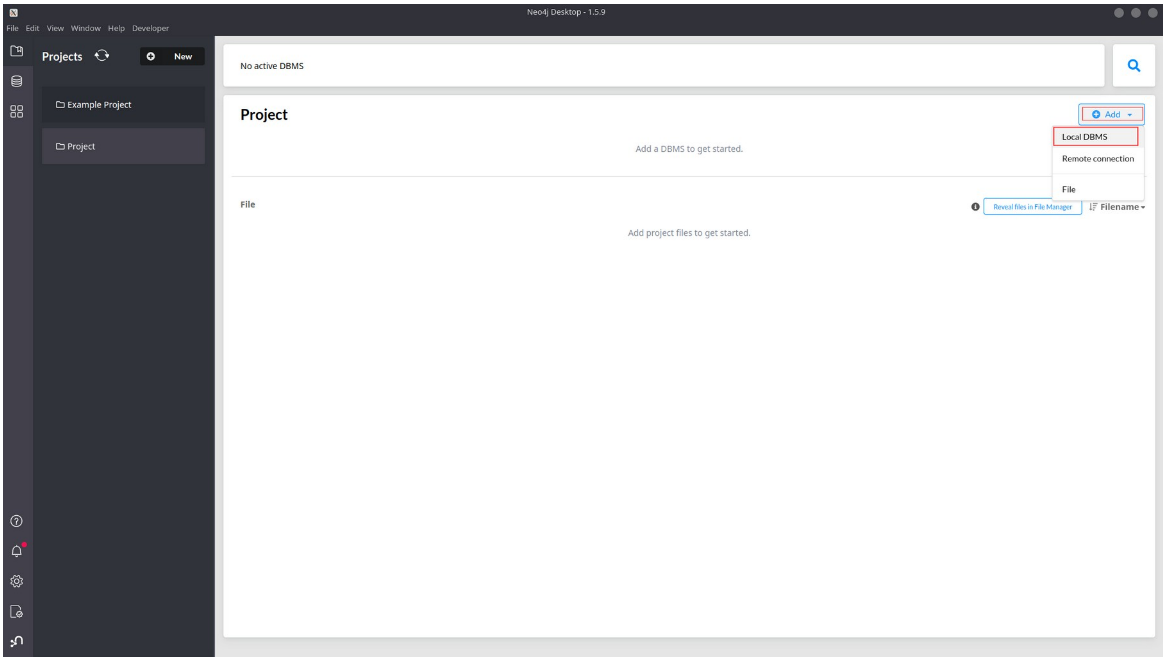
Software key *

```
GBVIOS4CM17IVyKCJOR-
GUJjkc9VvDdU_NKVTJub2GgF2qbrboHb0Zo8lw
Vcrqz9SoMjFQC0pqTSk27IdEkoCjONsHzmtnRR
PU2nu_U4BwS6YW4Hhp49aEdX3xyipPT2hzBJ
dv2fDlyNqGuw9c1Vq5dqxPddgLLgcMI9TI8KvyS
3dlxRu77GRqR78Qp3KYnJ7UFK1MxRYd2F82h
8T5Cs0uAFIEqd9NbFxTCrkkVHVQUGkxbv3mjn
g7s1Dp8xyXn0QGSGLCOYBBcsvEUw
```

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


Neo4j Desktop - 1.5.9
File Edit View Window Help Developer
Projects
New
Create project
Create project from directory
Import project from archive
Import sample project
No active DBMS
Example Project
Add
Movie DBMS 5.12.0
File
Reveal files in File Manager
Filename
about-movies.neo4j-browser-guide
load-movies.cypher

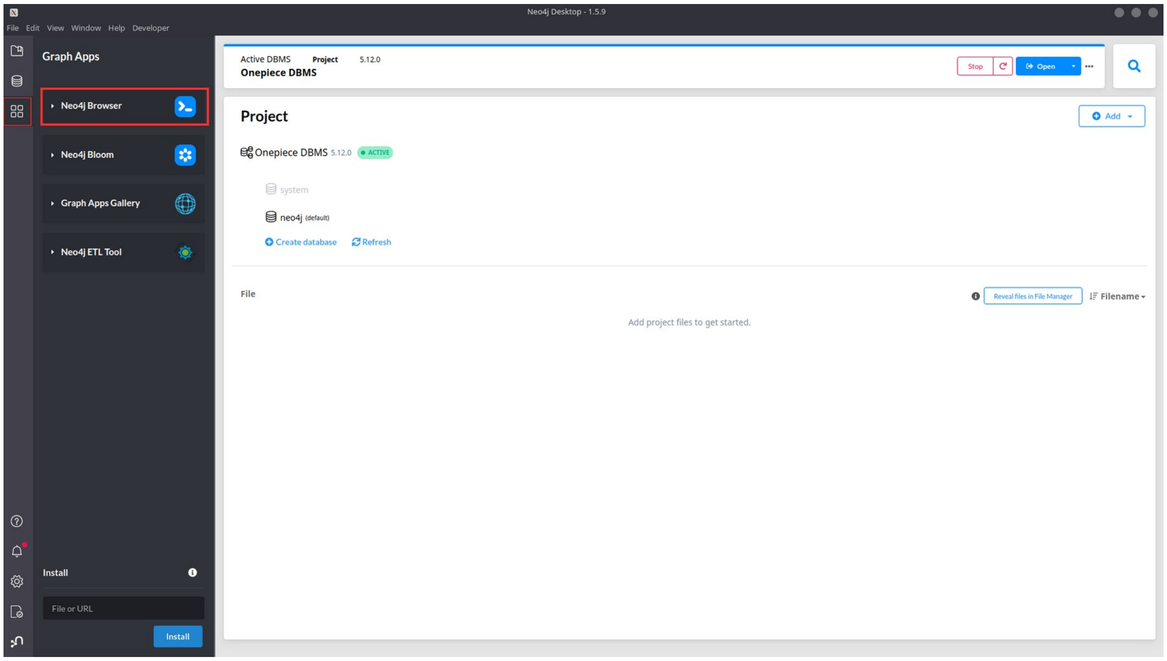
Step 8: Once you have created a new project, click on 'Add' and then select 'Local DBMS' to add a new database to our project.


Neo4j Desktop - 1.5.9
File Edit View Window Help Developer
Projects
New
Example Project
Project
No active DBMS
Project
Add
Local DBMS
Remote connection
File
Add a DBMS to get started.
File
Reveal files in File Manager
Filename
Add project files to get started.

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’.



Add project files to get started.

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:

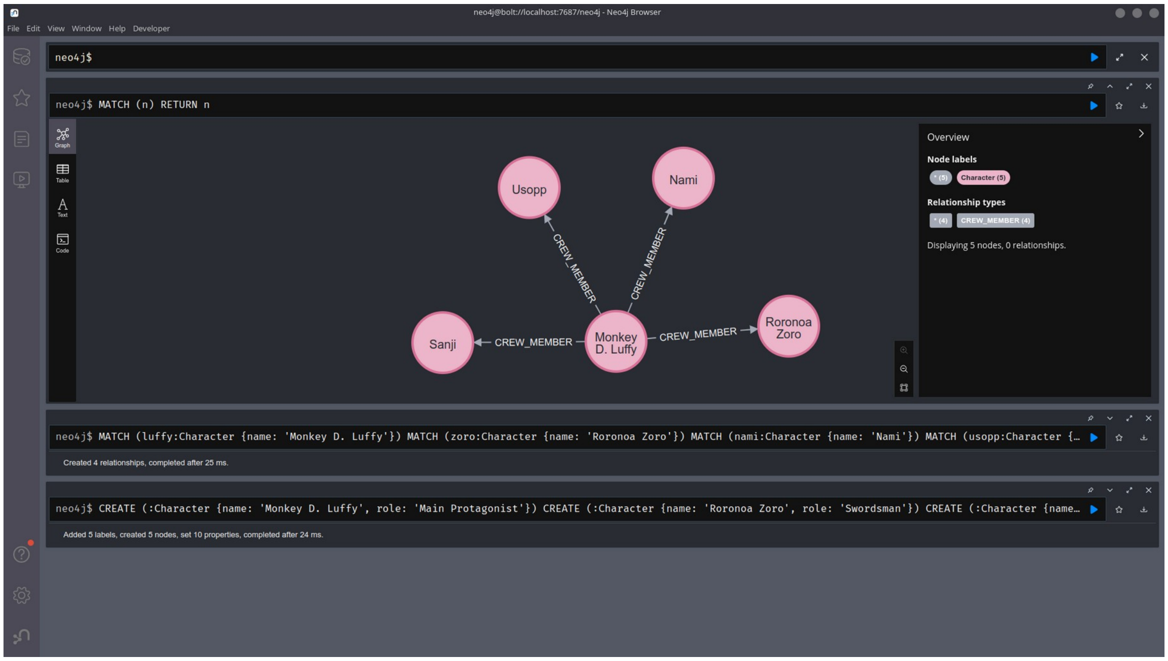
```
MATCH (luffy:Character {name: 'Monkey D. Luffy'})
MATCH (zoro:Character {name: 'Roronoa Zoro'})
MATCH (nami:Character {name: 'Nami'})
MATCH (usopp:Character {name: 'Usopp'})
MATCH (sanji:Character {name: 'Sanji'})

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
```

Overview

Node labels

* (5) Character (5)

Relationship types

* (4) CREW_MEMBER (4)

Displaying 5 nodes, 0 relationships.

```
neo4j$ MATCH (luffy:Character {name: 'Monkey D. Luffy'}) MATCH (zoro:Character {name: 'Roronoa Zoro'}) MATCH (nami:Character {name: 'Nami'}) MATCH (usopp:Character {…
```

Created 4 relationships, completed after 25 ms.

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

Added 5 labels, created 5 nodes, set 10 properties, completed after 24 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:

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:

RESULT: