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, URIRefapp =

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)
```

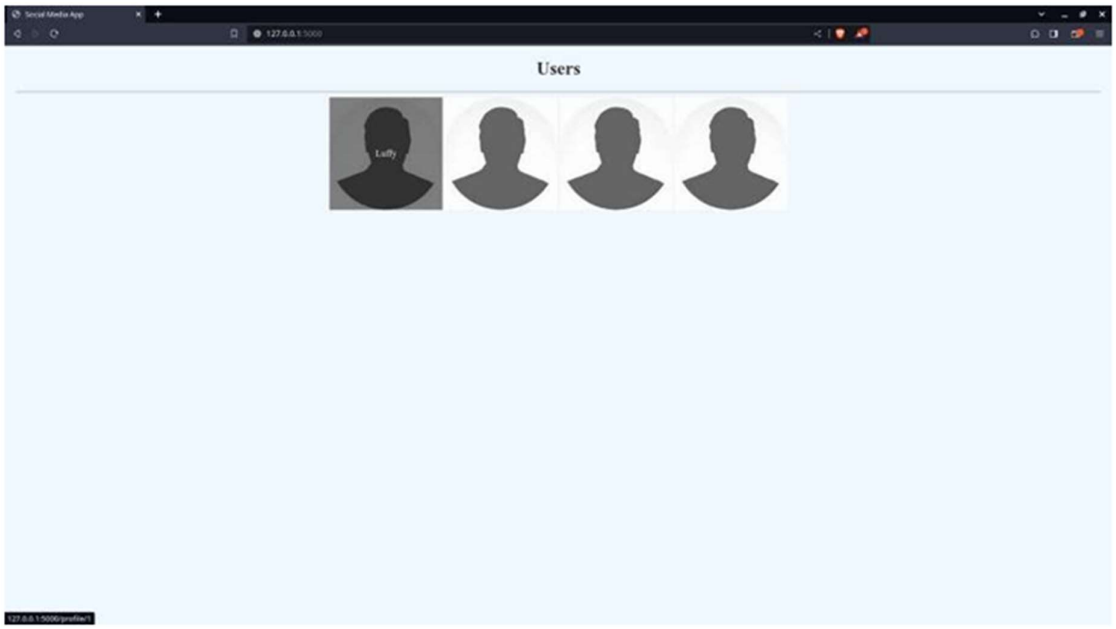
PROJECT STRUCTURE: (Just For Ref.)

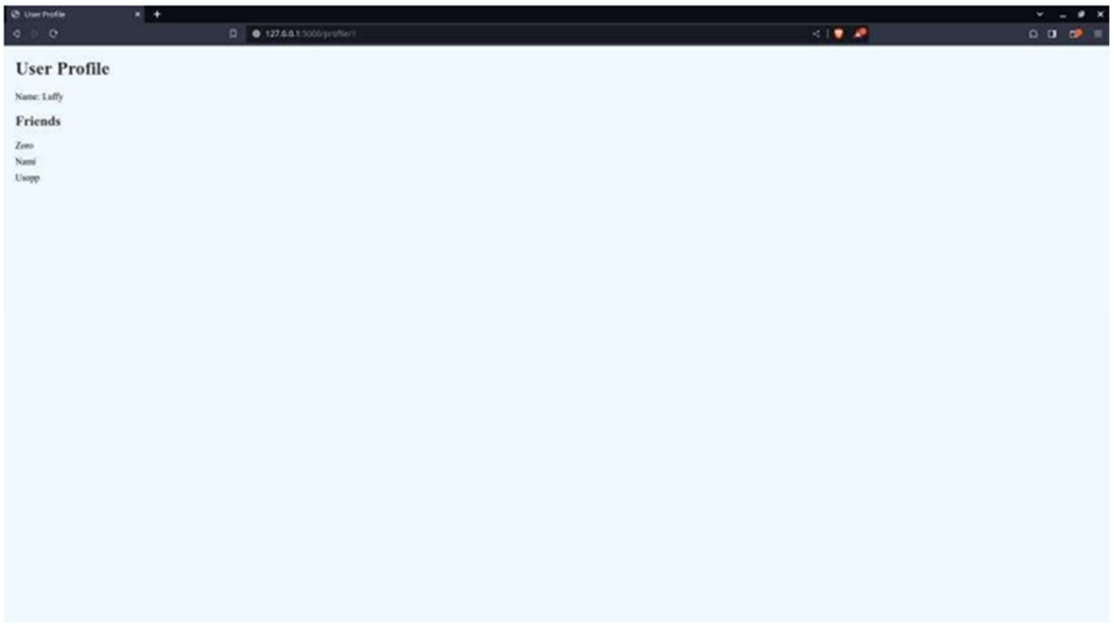
```
.
├── app.py
├── static
│   └── styles.css
└── templates
    ├── index.html
    └── profile.html

2 directories, 4 files
```

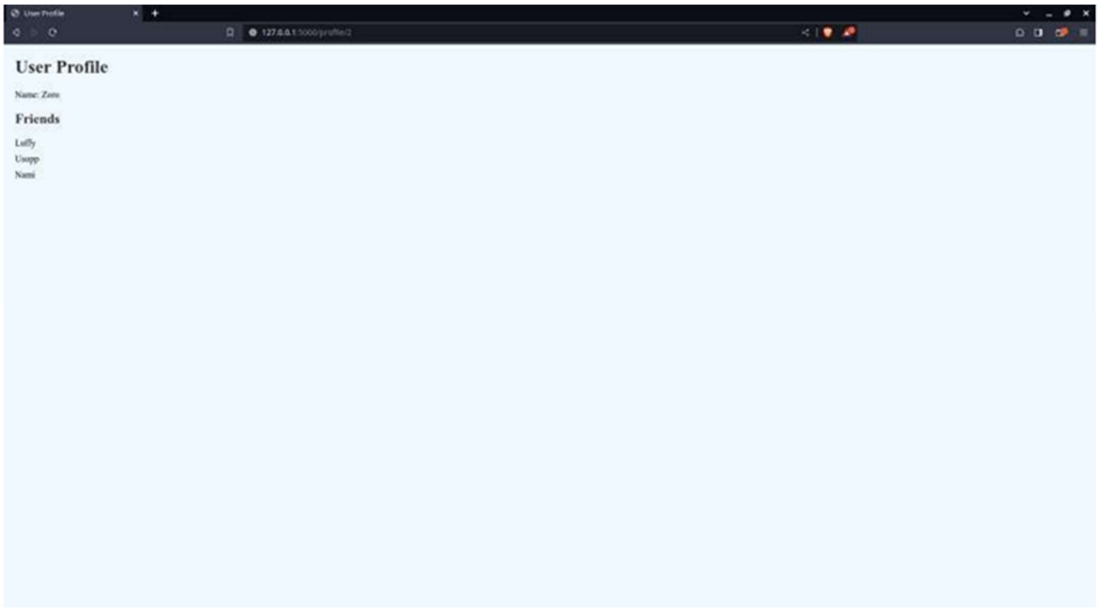
Execution (Python (Programming Language)):

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

OUTPUT:


Social Media App
127.0.0.1:5000
Users
Luffy
127.0.0.1:5000/profile/1


User Profile
127.0.0.1:5000/profile/1
User Profile
Name: Luffy
Friends
Zoro
Nami
Usopp

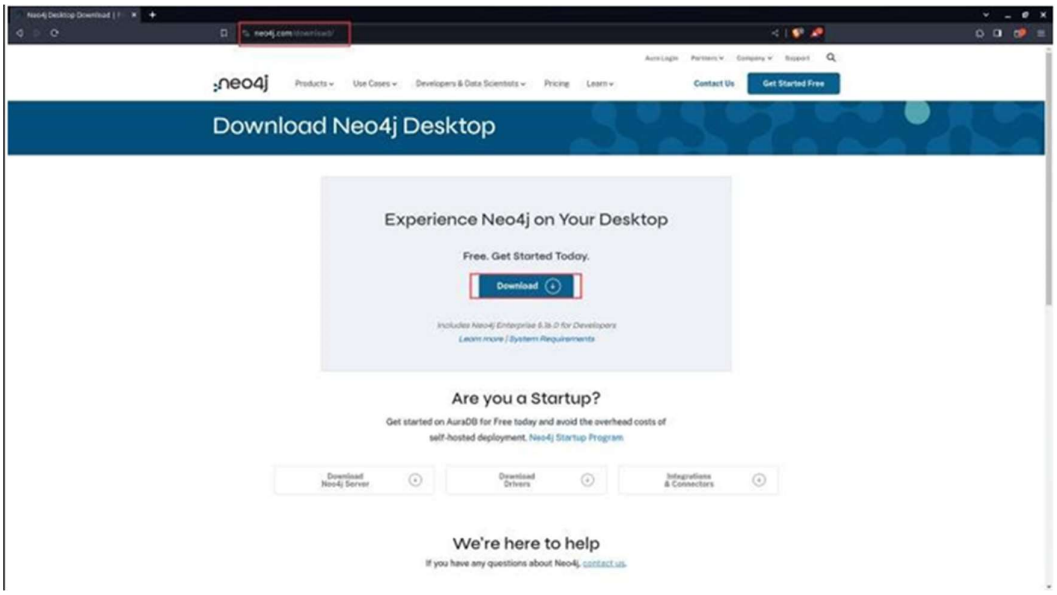

User Profile
127.0.0.1:5000/profile/2
User Profile
Name: Zoro
Friends
Luffy
Usopp
Nami

RESULT:

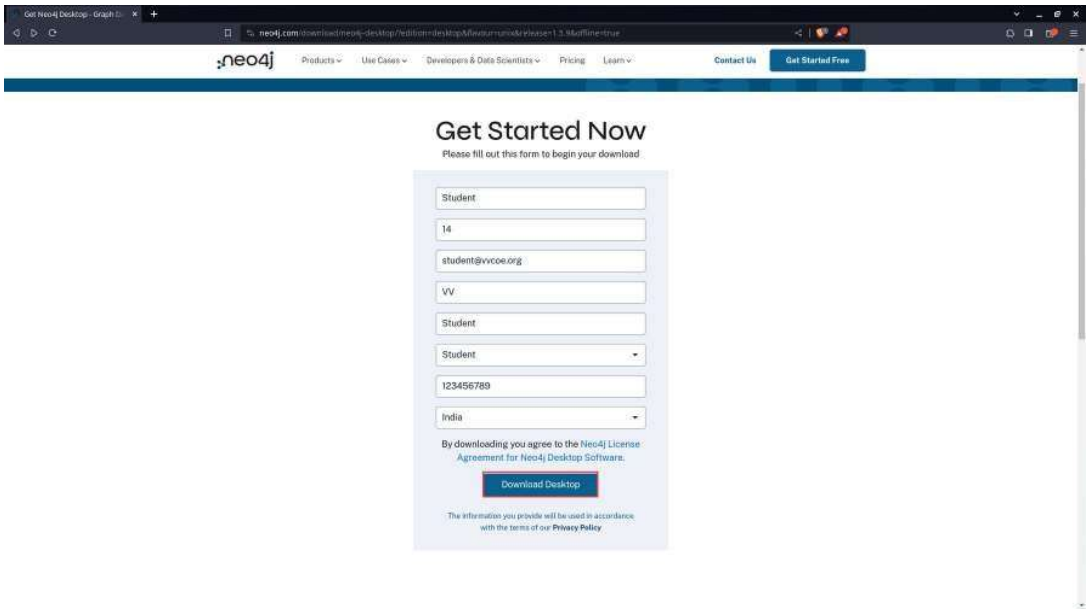
INSTALLATION:

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

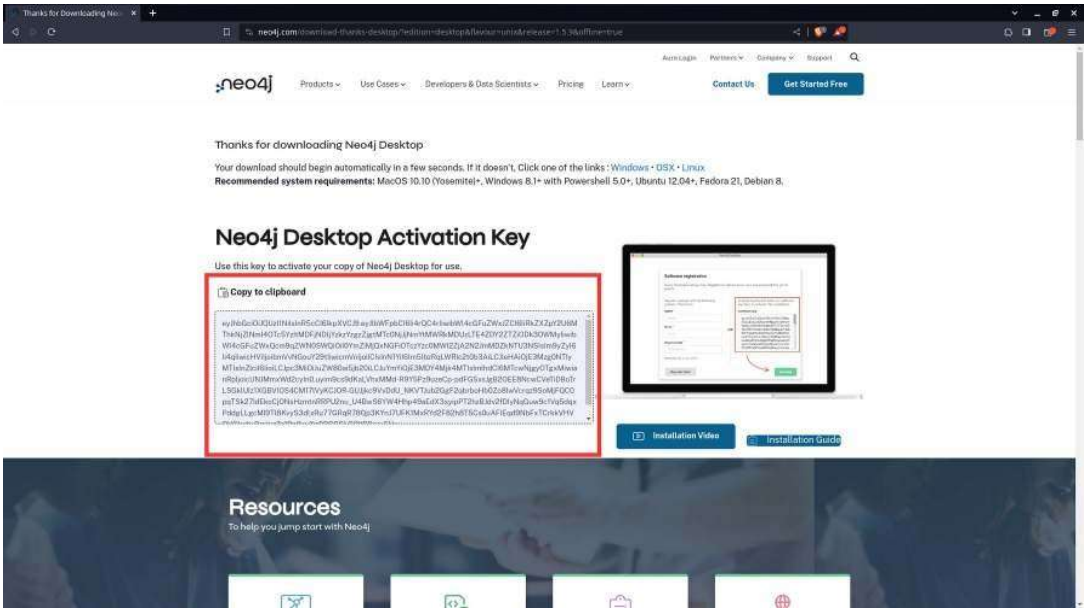
Step 2: On the page, locate and click on the 'Download' button."



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



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



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

$ ls -al | grep “neo4j”

Change the permissions to make it executable.

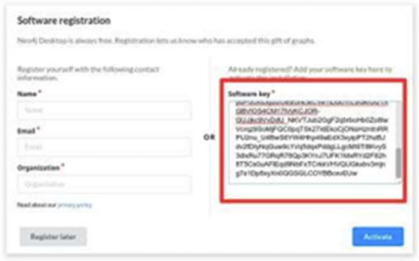
$ chmod +x neo4j-desktop-1.5.9-x86_64.AppImage

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

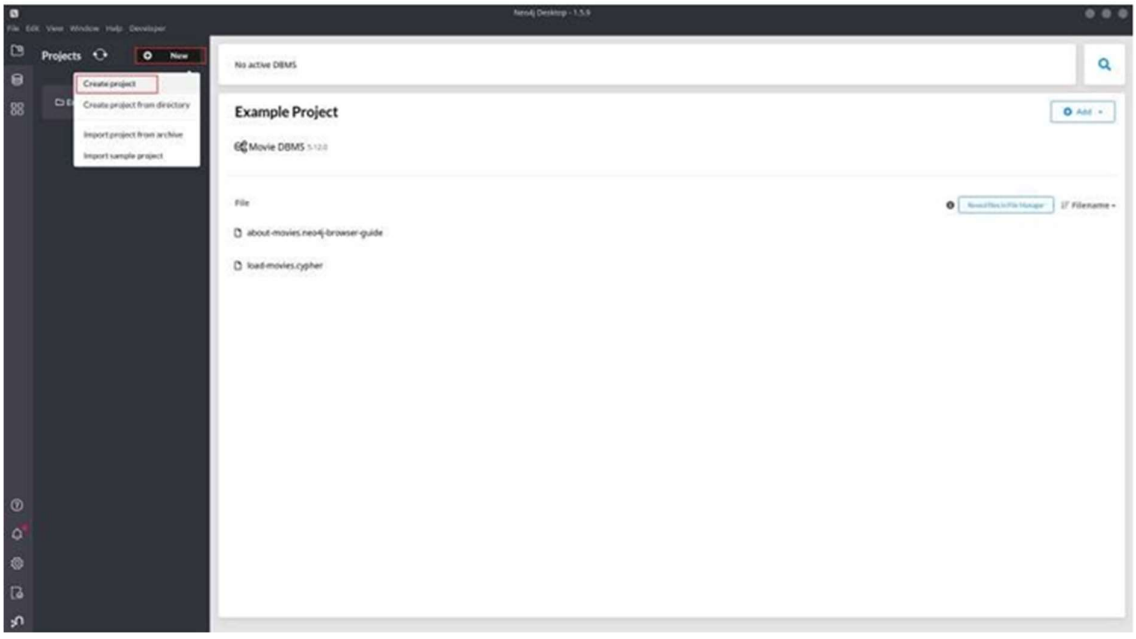
Start the AppImage:

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

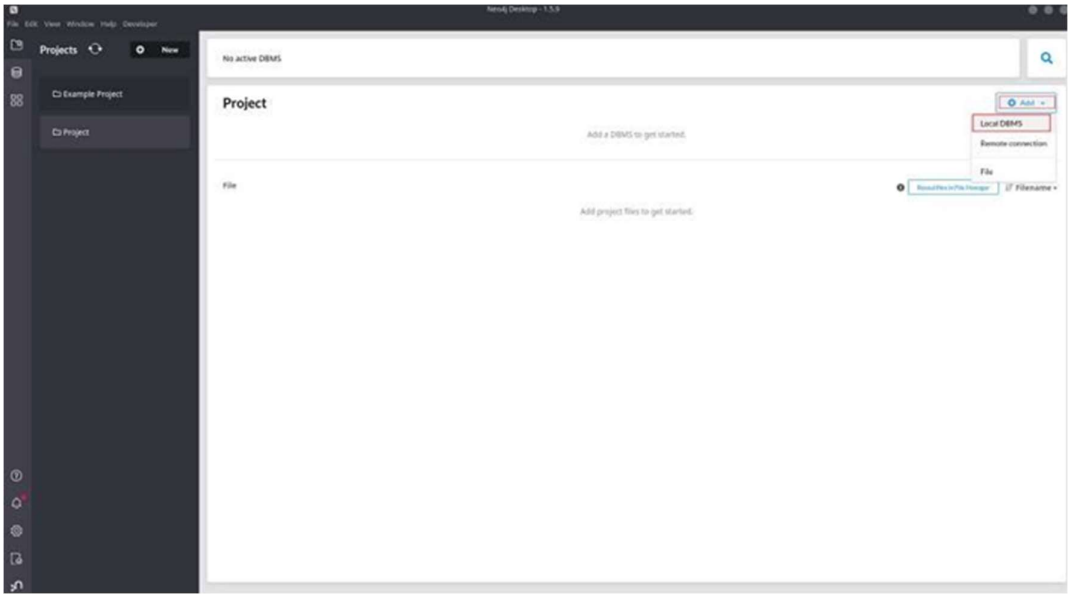
Step 6: After opening Neo4j, navigate to the 'Software Key' section and paste the previously copied 'Activation Key'. Then, click the 'Activate' button to complete theactivation process.



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



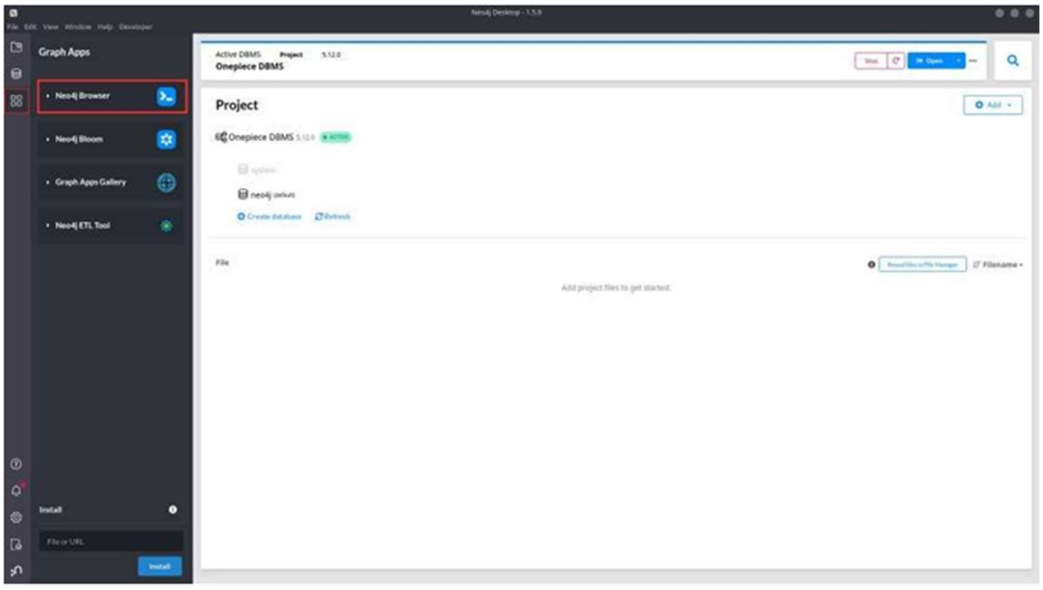
Step 8: Once you have created a new project, click on 'Add' and then select 'Local DBMS'to add a new database to our project.



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'})
```

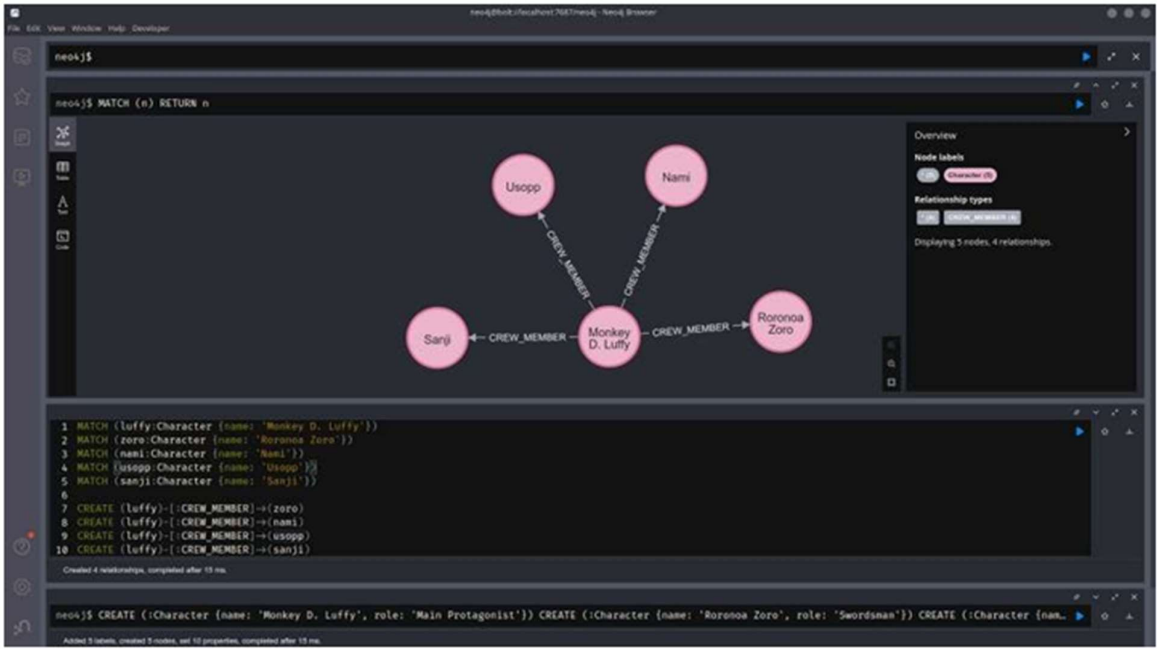
Creating crew relationship:

```
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
```

OUTPUT:



RESULT:

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
```

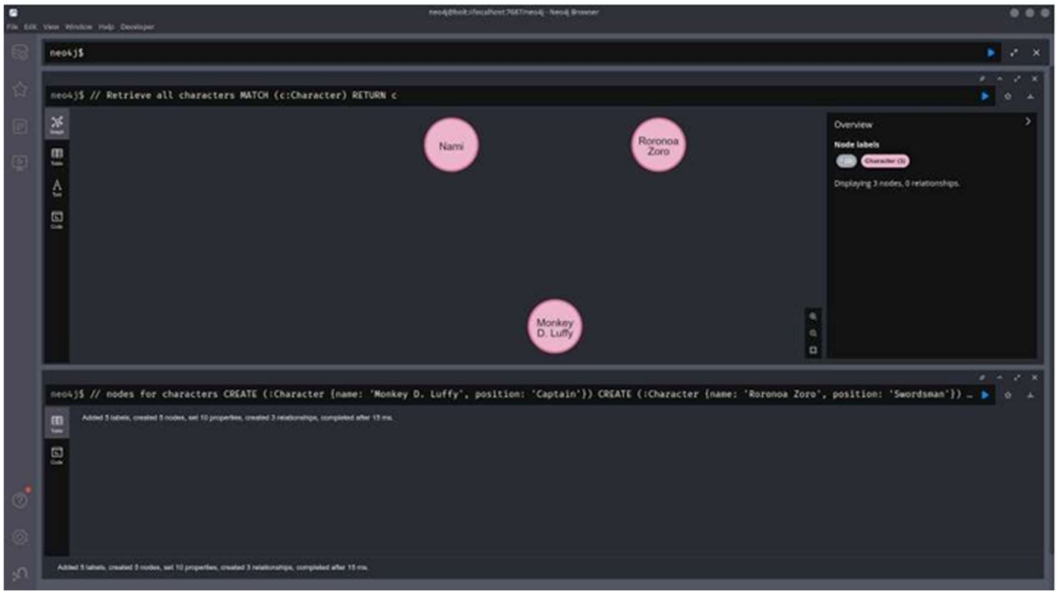
Updating data:

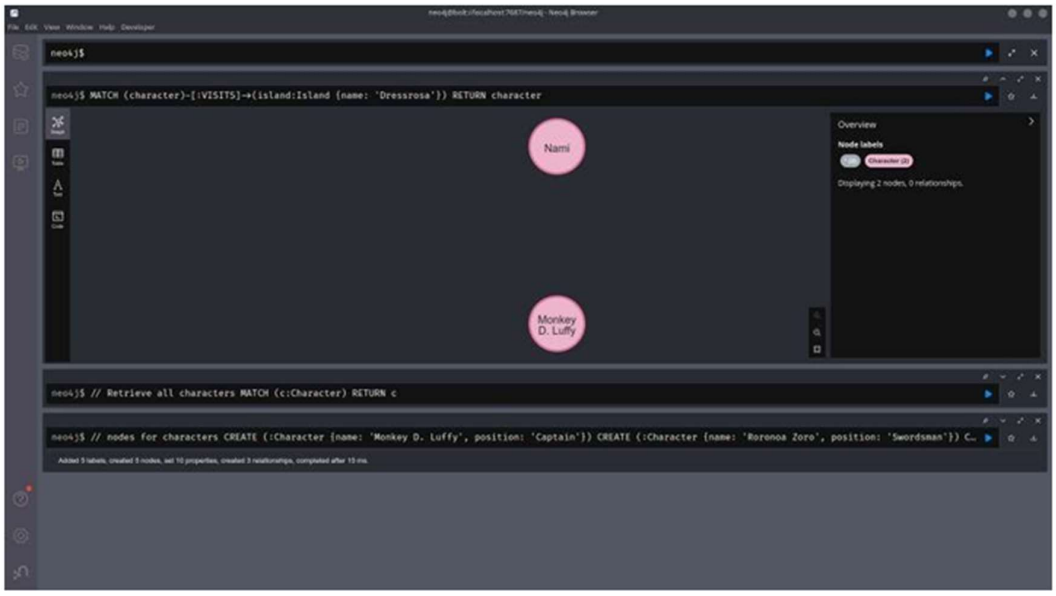
```
// Update character's position
MATCH (luffy:Character {name: 'Monkey D. Luffy'})
SET luffy.position = 'Pirate King'
RETURN luffy
```

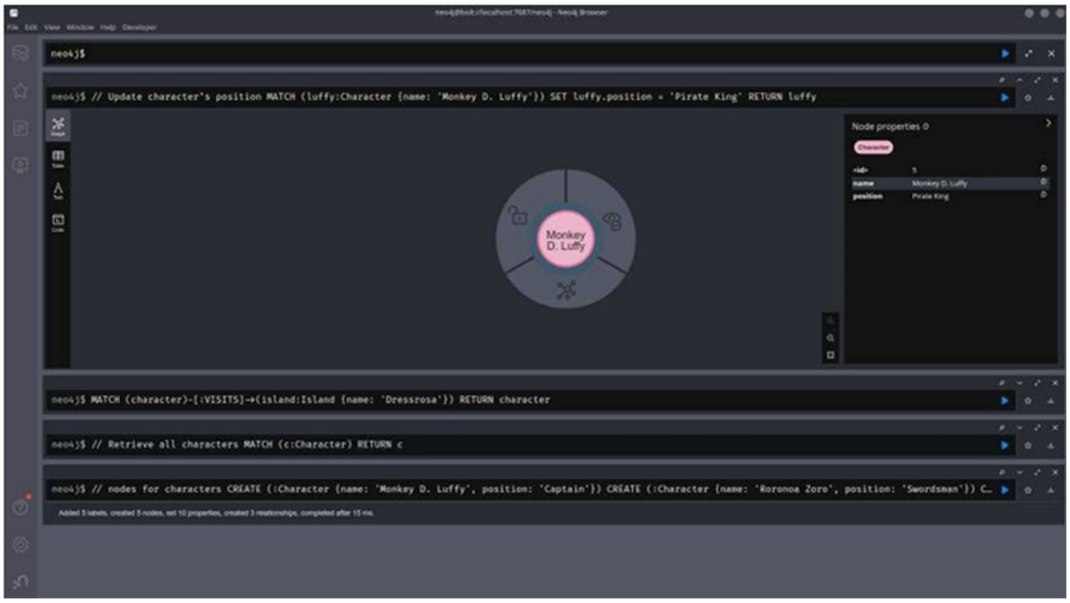
Deleting data:

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

OUTPUT:


neo4j$
neo4j$ // Retrieve all characters MATCH (c:Character) RETURN c
Nami
Roronoa Zoro
Monkey D. Luffy
Overview
Node labels
Character (3)
Displaying 3 nodes, 0 relationships.
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$
neo4j$ MATCH (character)-[:VISITS]→(island:Island {name: 'Dressrosa'}) RETURN character
Nami
Monkey D. Luffy
Overview
Node labels
Character (2)
Displaying 2 nodes, 0 relationships.
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
Monkey D. Luffy
Node properties 0
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 5 labels, created 5 nodes, set 10 properties, created 3 relationships, completed after 15 ms.


neo4j$
neo4j$ MATCH (n) RETURN n
Alabasta
Dressrosa
VISITS
VISITS
Roronoa Zoro
Monkey D. Luffy
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.

RESULT:

PROGRAM:

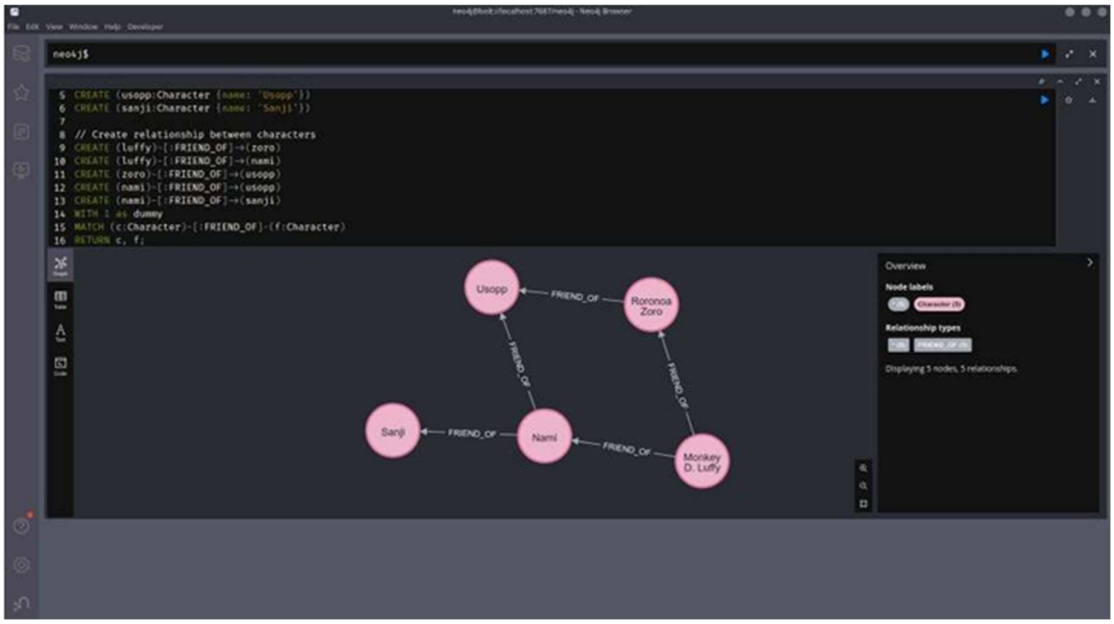
Create nodes for characters:

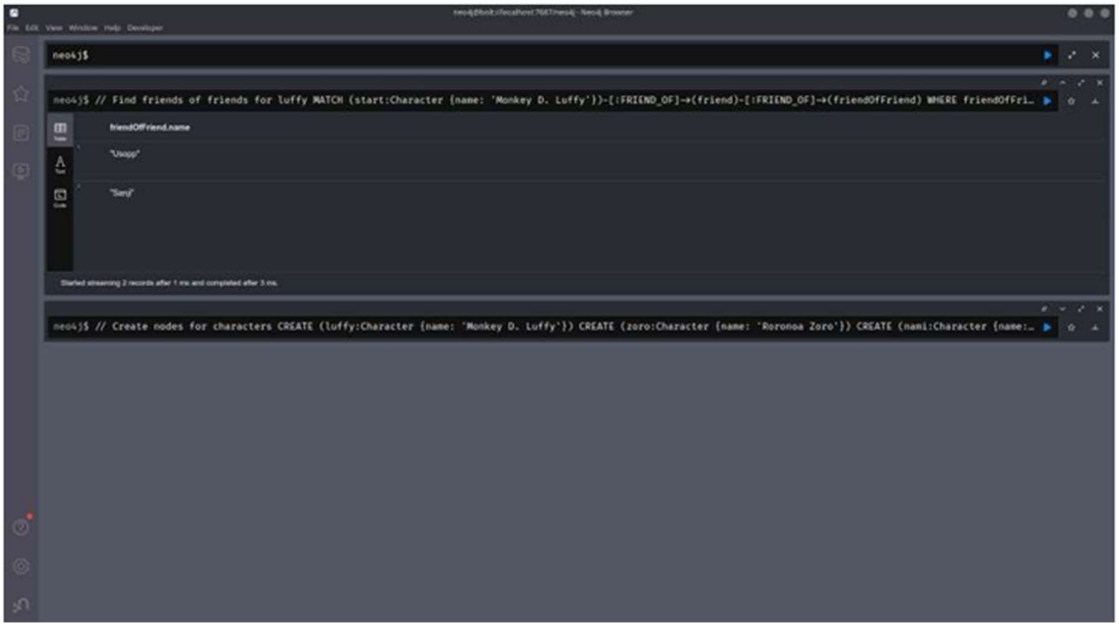
```
// Create nodes for characters
CREATE (luffy:Character {name: 'Monkey D. Luffy'})
CREATE (zoro:Character {name: 'Roronoa Zoro'}) CREATE
(nami:Character {name: 'Nami'})
CREATE (usopp:Character {name: 'Usopp'})CREATE
(sanji:Character {name: 'Sanji'})

// 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:

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 truehistory"},
   "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="Typehere.."
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
```

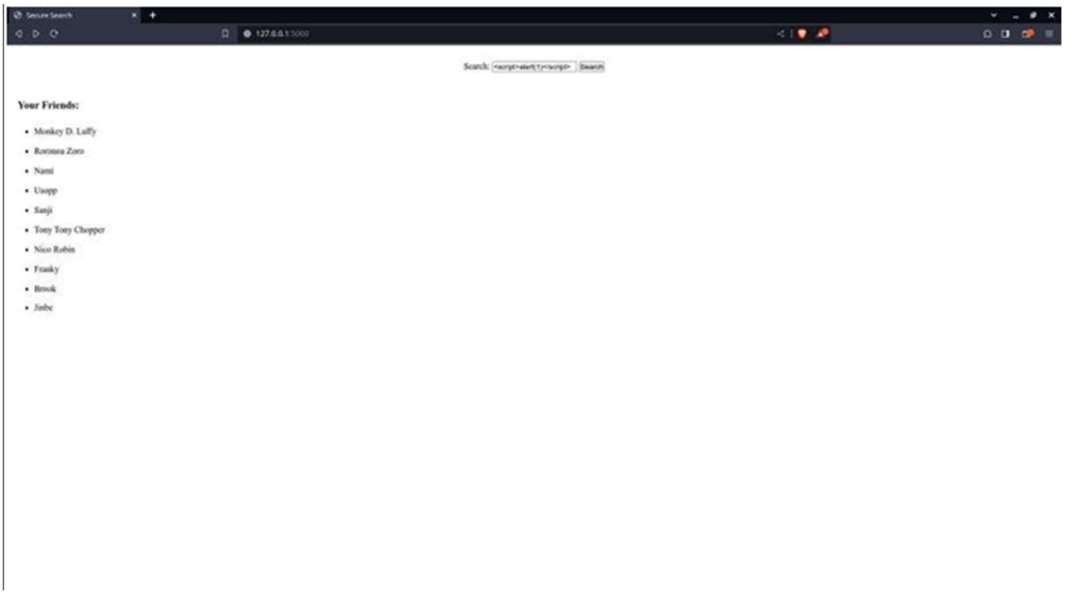
Execution (Python (Programming Language)):

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

OUTPUT:


Secure Search
127.0.0.1:5000
Search: Type here. Search
Your Friends:
Monkey D. Luffy
Roronoa Zoro
Nami
Usopp
Sanji
Tony Tony Chopper
Nico Robin
Franky
Brook
Jinbe


Search Results
127.0.0.1:5000/search/zoro
Search Results for "zoro"
Roronoa Zoro
Role: Swordsman
Dream: World's greatest swordsman


Secure Search
127.0.0.1:5000
Search: Search
Your Friends:
Monkey D. Luffy
Roronoa Zoro
Nami
Usopp
Sanji
Tony Tony Chopper
Nico Robin
Franky
Brook
Jinbe

Search Results for "scriptalert1script"

RESULT:

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;
}
```

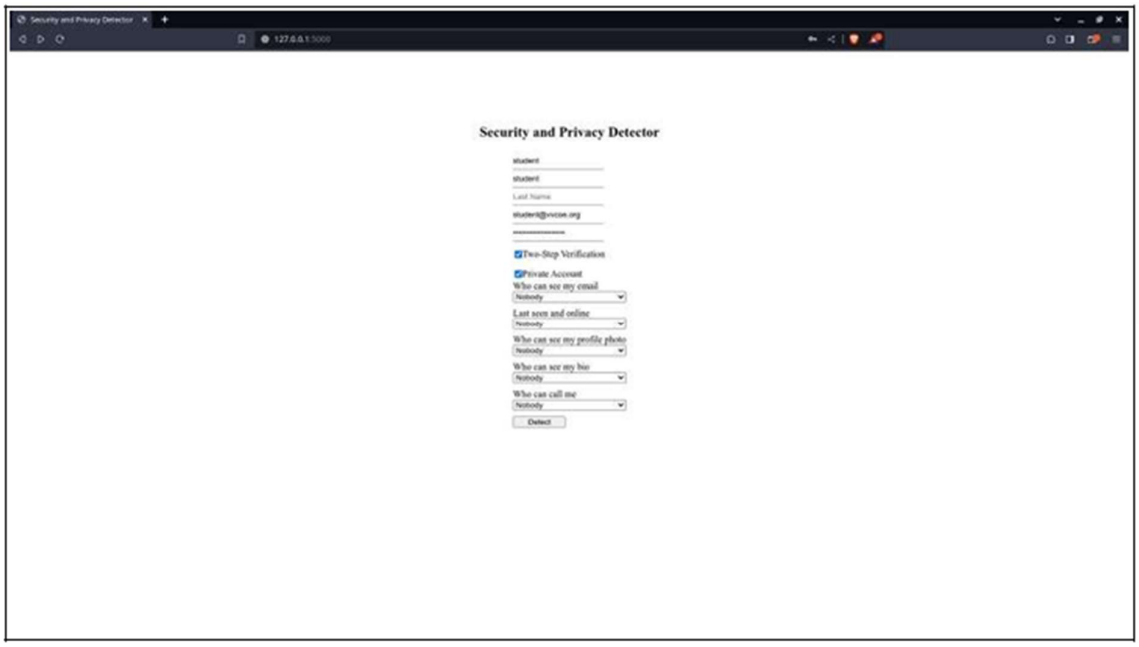
PROJECT STRUCTURE: (Just For Ref.)

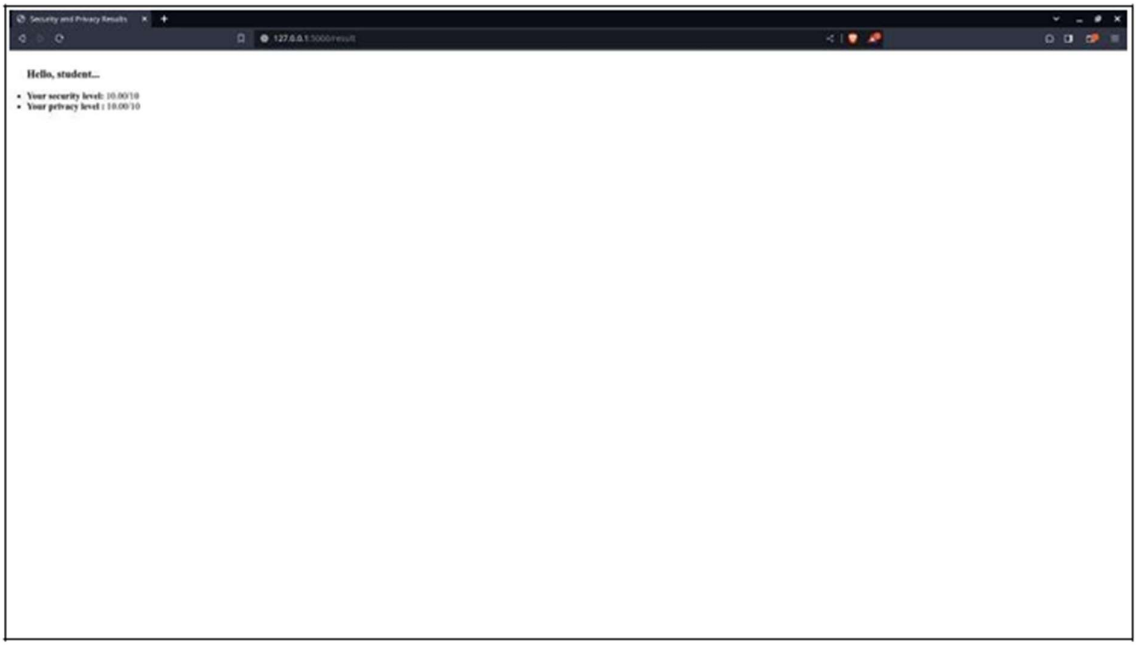
```
.
├── app.py
├── static
│   └── styles.css
└── templates
    ├── detected_result.html
    └── index.html

2 directories, 4 files
```

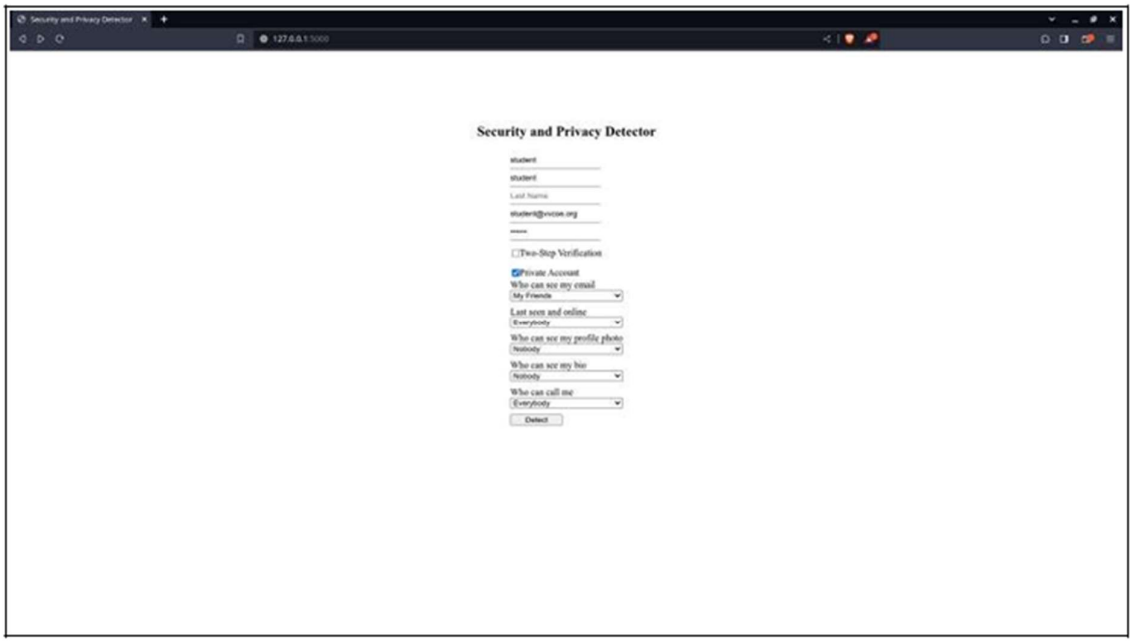
Execution (Python (Programming Language)):

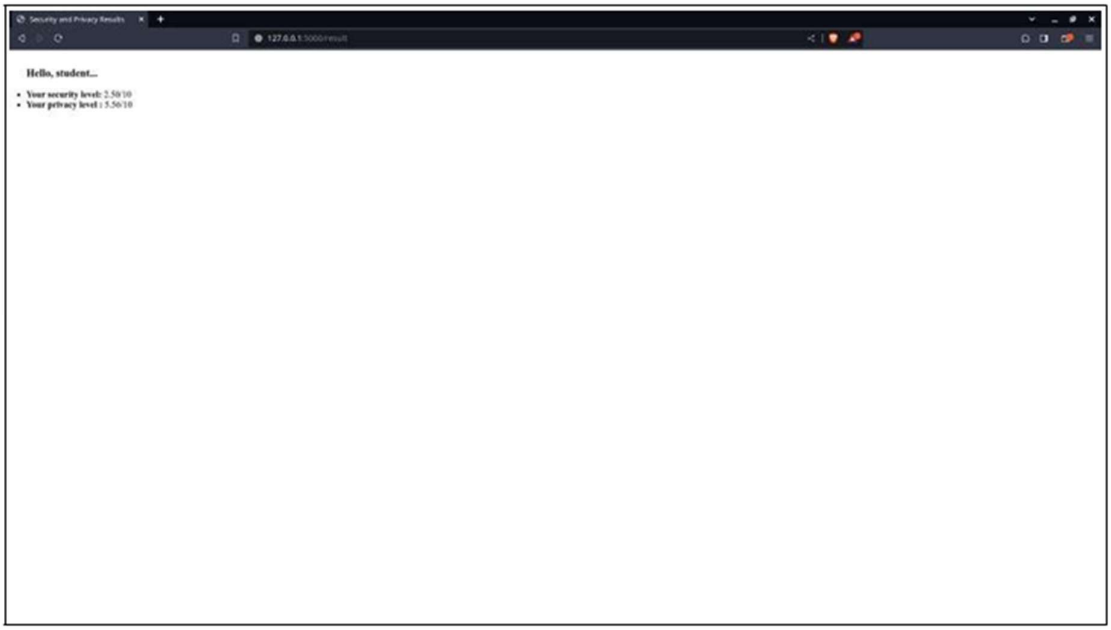
```
$ pip3 install Flask
$ python3 app.py
```

OUTPUT:


Security and Privacy Detector
Two-Step Verification
Private Account
Who can see my email
Last seen and online
Who can see my profile photo
Who can see my bio
Who can call me
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
Detect


Hello, student...
Your security level: 2.50/10
Your privacy level : 5.56/10

RESULT: