Report

- **Technologies used:**

- Neo4j graph database
- Flask python framework
- Networks to visualize graph

- **App service:**

- Registration form to collect information from users like (firstname, email,…)
- Create graph database for register users.
- Graph database contains relationships between users and this address and company name.
- Every user has status (yellow, green, red), and you can search by username to know user status.
- Can change status for specific user.
- If you change status to red app automatically change all users in the same address or in same company and have green status to yellow status.
- If you register new user with green status and there is user with red status in same address or same company app automatically change register user status to yellow.
- Can visualize a graph for all user in same company.
- Can visualize a graph for all user in same address.

- **Design Overview:**

- This is three pages in app (registration page, search page, change state page).

- **Registration page**: implemented in registerAuthStaff() method and it have three parts:

1- collect data from form.

2- check user status and convert it if it green and there is user has red status in same address or company.

Report

3- add collected data to graph database (Neo4j).

- **search page**: implemented in searchForStatus() method and it have three parts:

  1- save all data related to user you need to search for in list and send this list to html to display the list as table.

  2- Get all users have the same address from database and create a graph to visualize user's data have same address using networks package in python and save this graph in image and send this image to html to display it.

  3- Get all users have the same company from database and create a graph to visualize user's data have same company using networks package in python and save this graph in image and send this image to html to display it.

- **Change status page:** implemented in changeStatu() method and it have two parts:

  1- If new status of specified user is red, it will firstly get all users have the same user address or company and green status and change their status to yellow finally change specified user status to red.

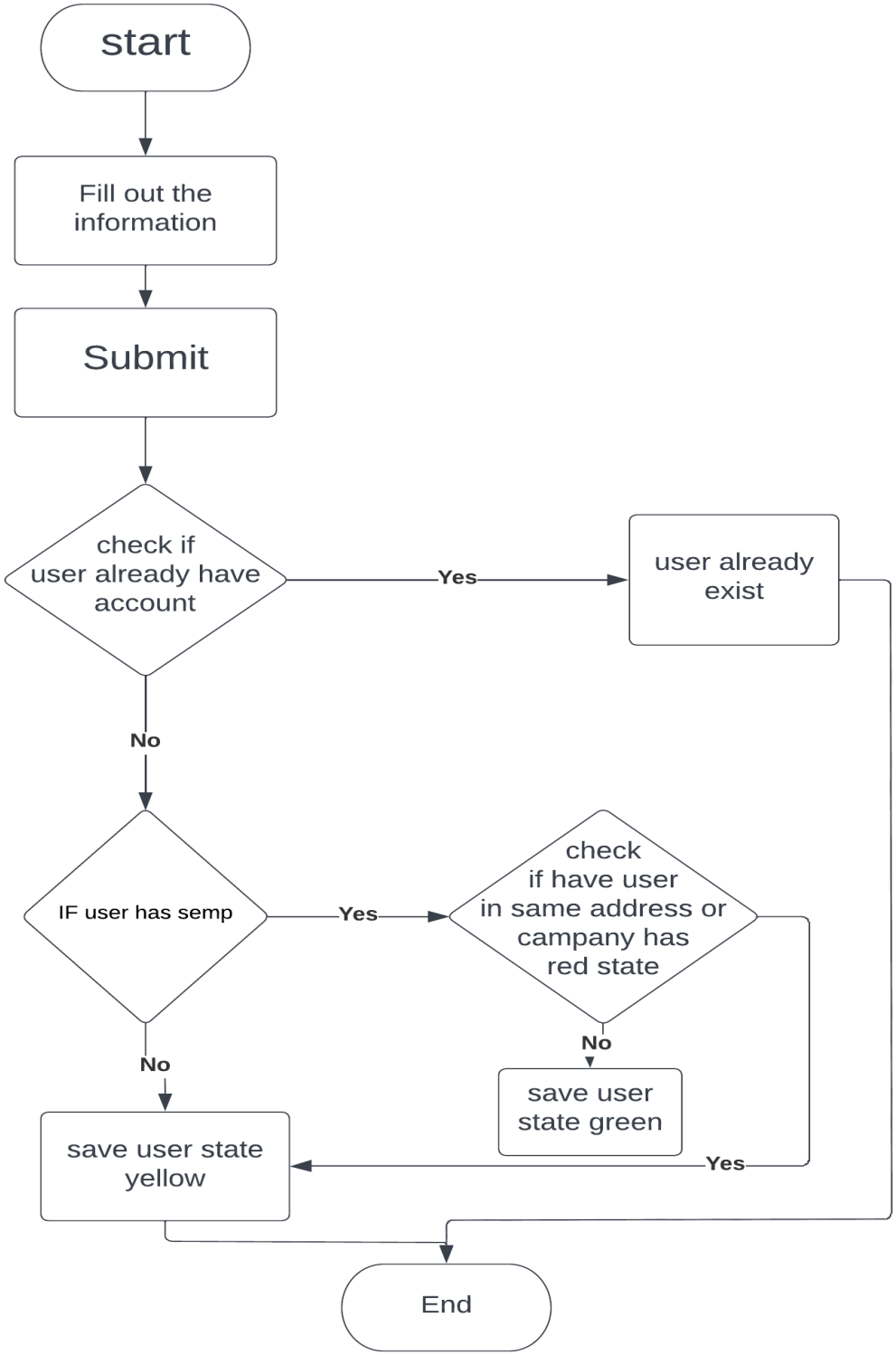
  2- If new status of specified user is green or yellow just change their status to green or yellow.

NOTE: All methods implementation in in the end of report.

Report

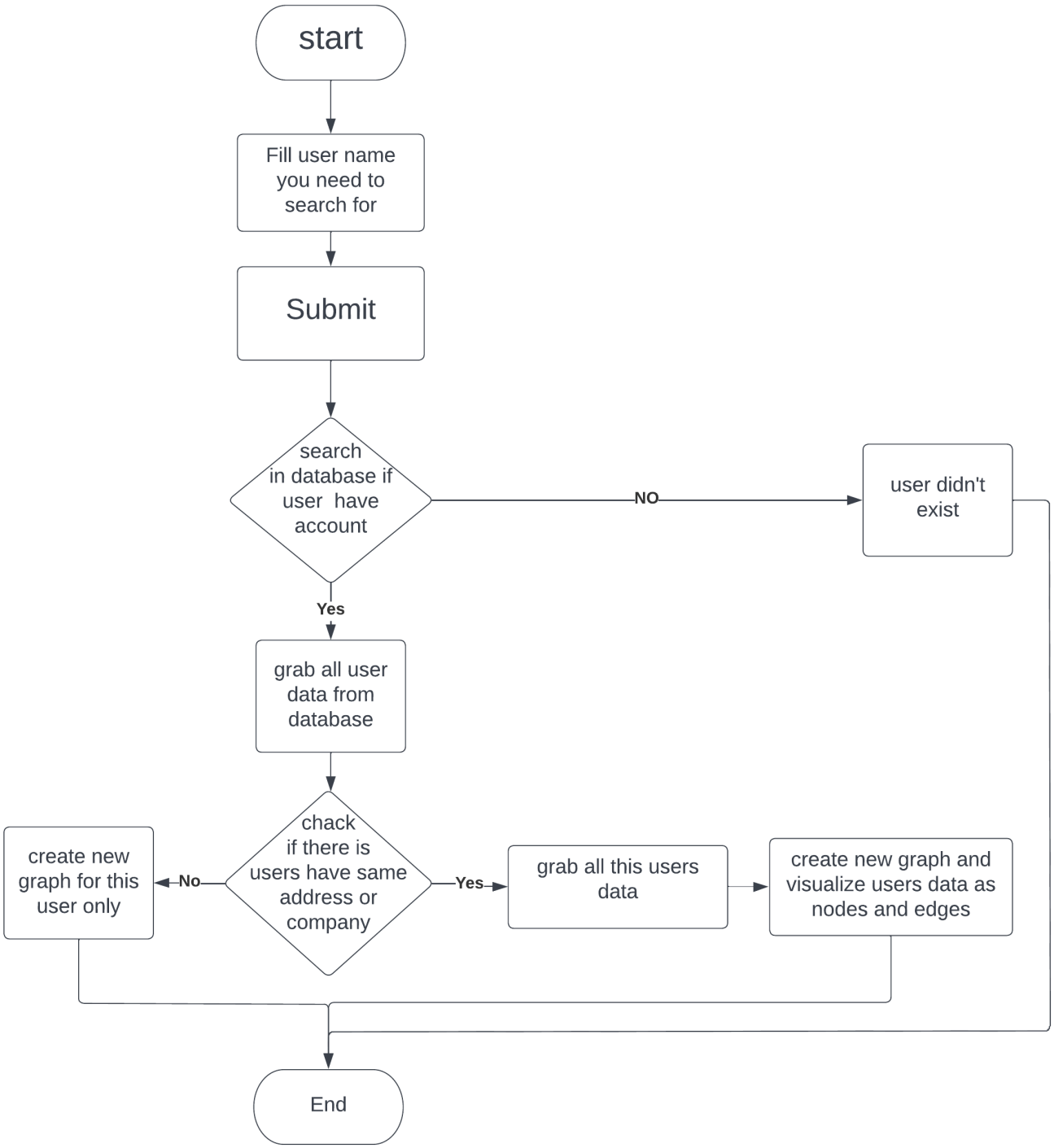
- **Flow charts:**

- Registration page:



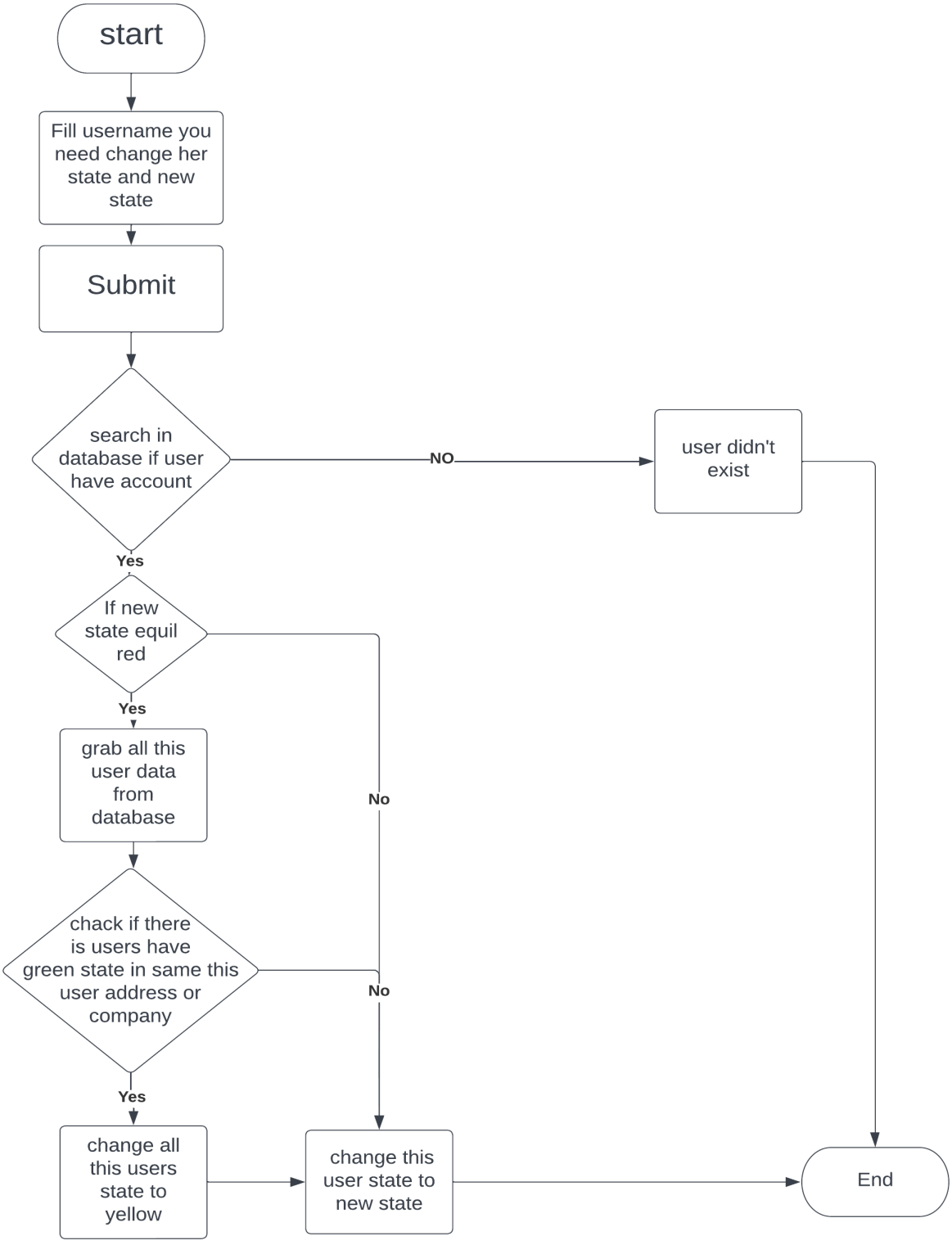
Report

- search page:



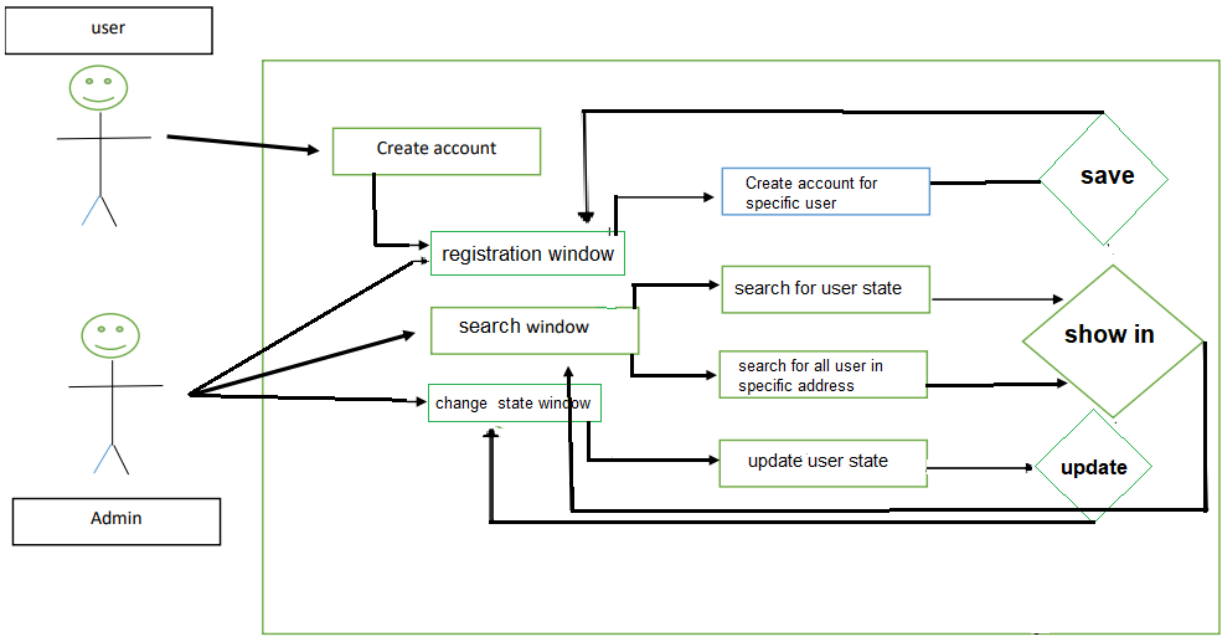
Report

- Change status page:

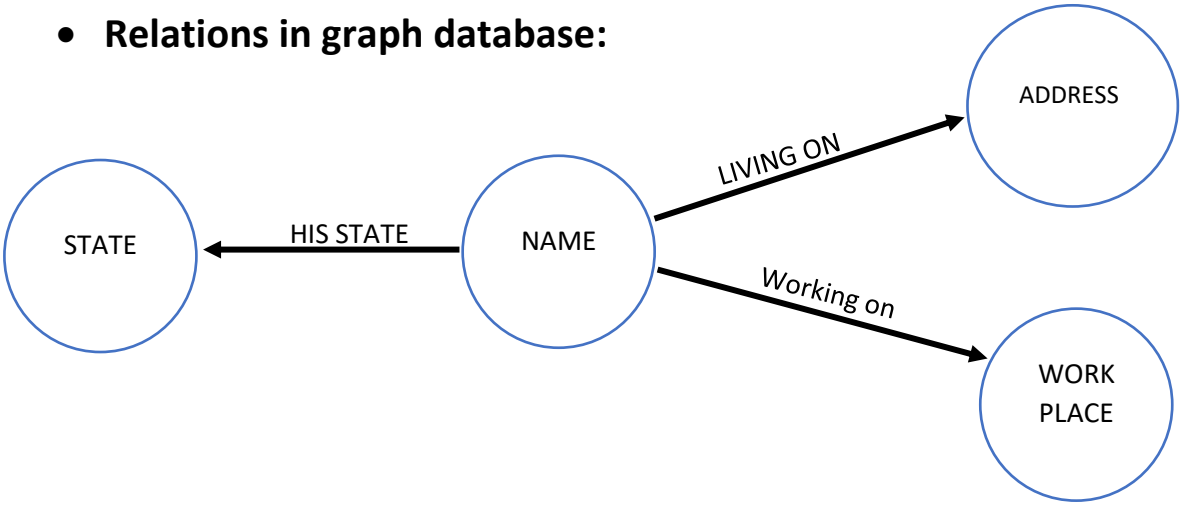


Report

- **Use case:**

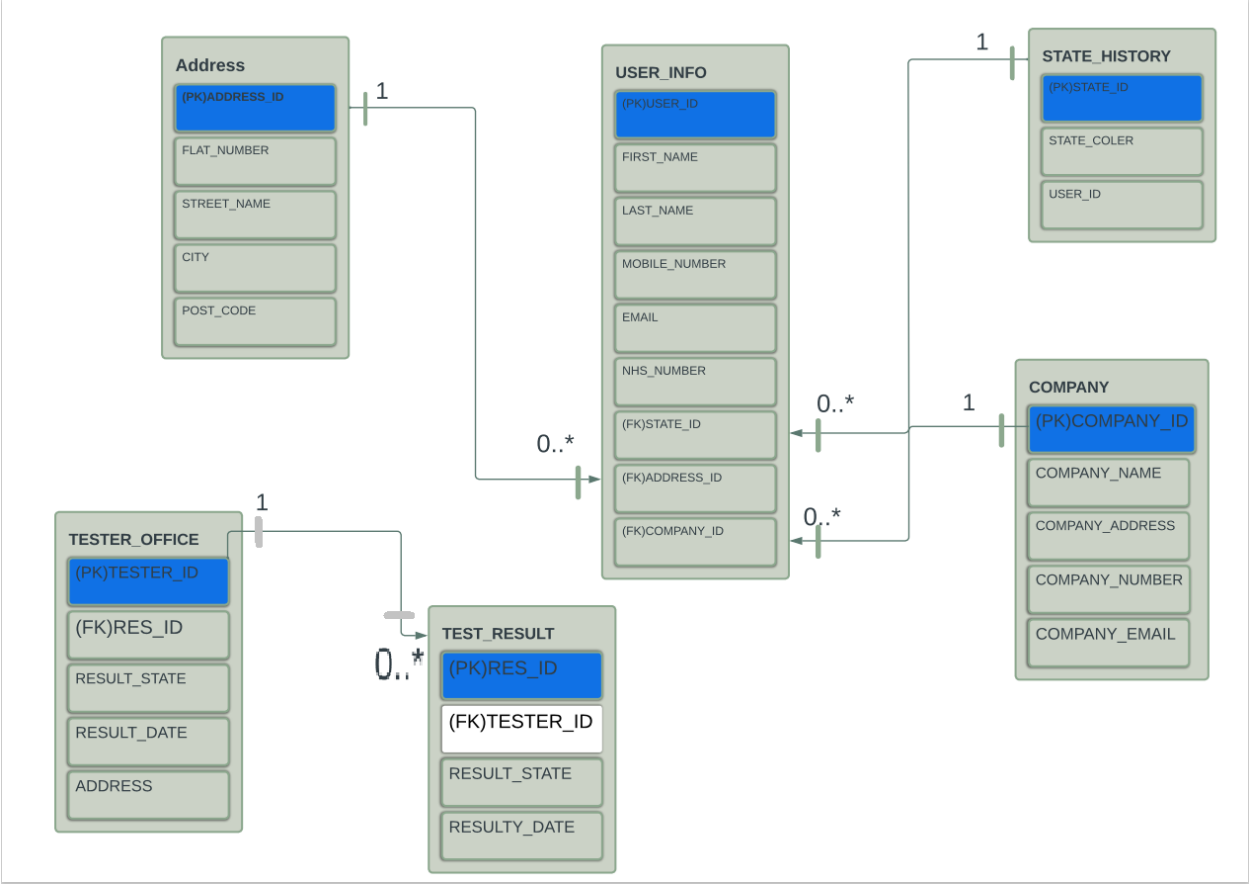


- **Relations in graph database:**

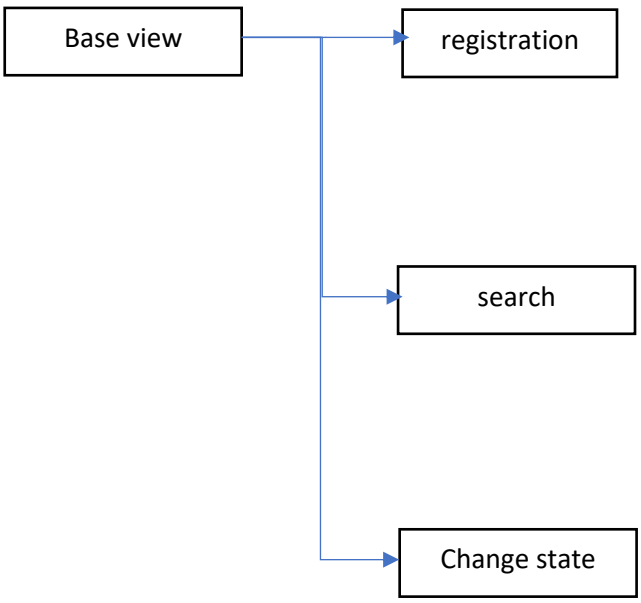


Report

- **ERD:**

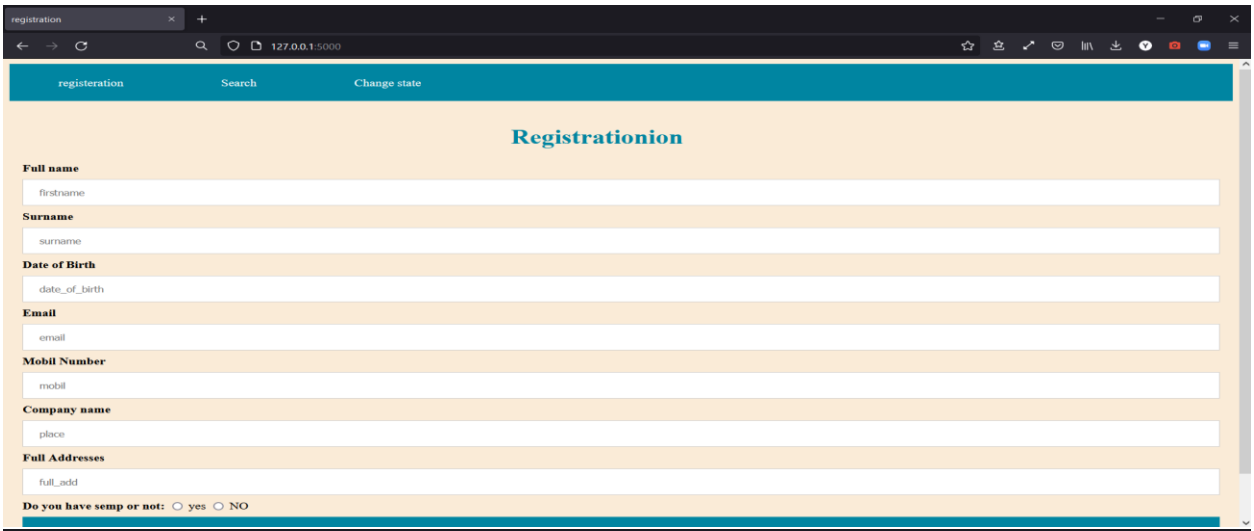


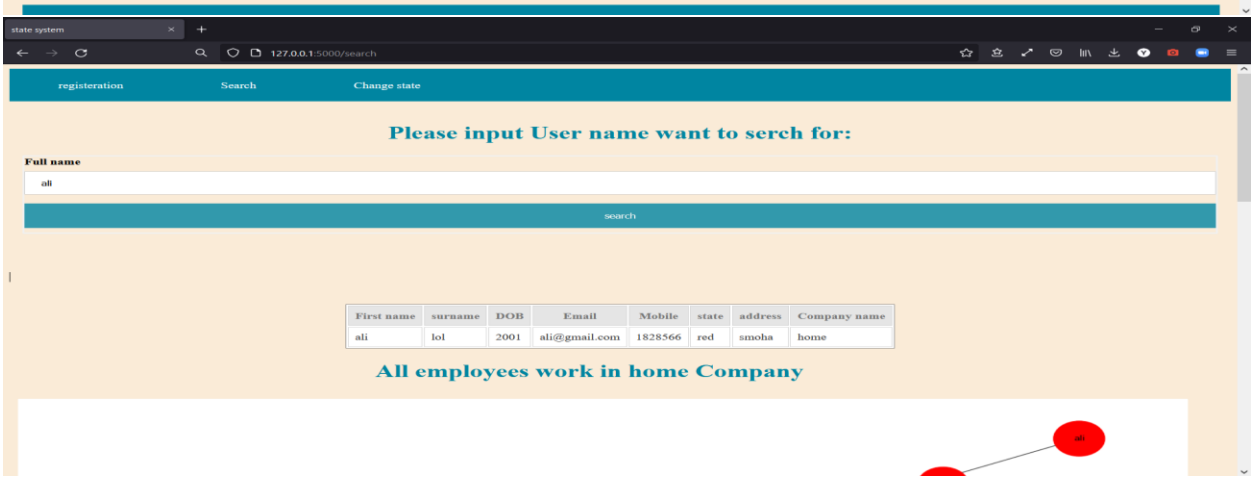
- **Class View Diagram Tree:**



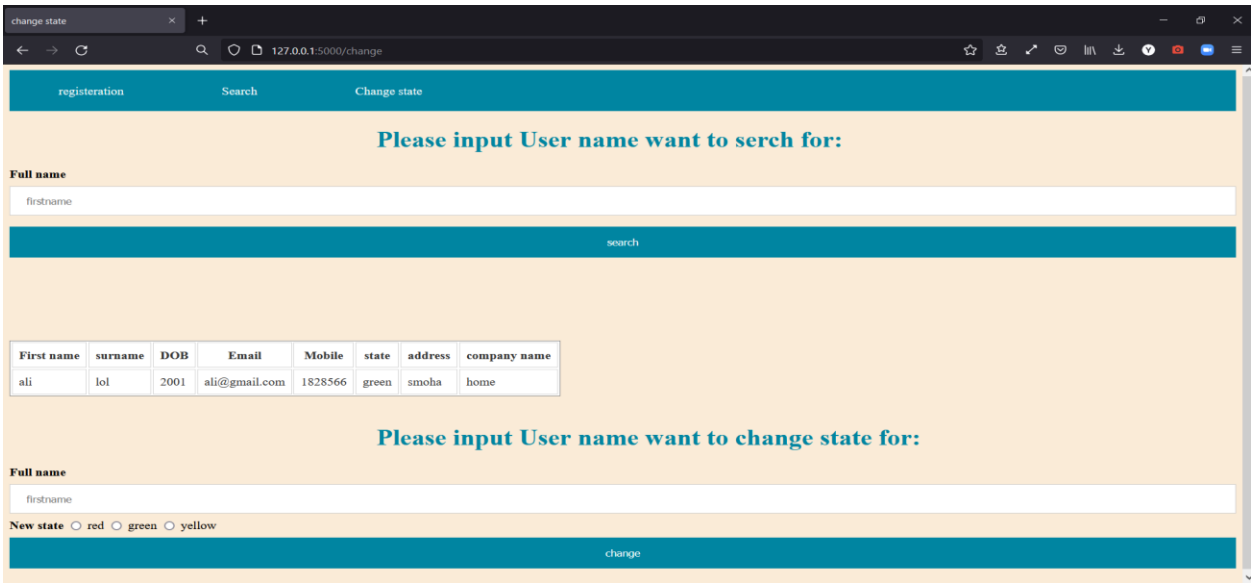
Report

- **sample Runs:**





Change ali status to green



Report

- **registerAuthStaff() method:**

```
#get submission from the registeration form
@app.route('/registerAuthStaff', methods=['GET', 'POST'])
def registerAuthStaff():
    codeid=+1
    # grabs information from the forms
    firstname = request.form['firstname']
    surname = request.form['surname']
    dob = request.form['date_of_birth']
    email = request.form['email']
    mobil = request.form['mobil']
    state = request.form['state']
    place = request.form['place']
    full_add = request.form['full_add']

    if (state =="yes"):
        state="yellow"
    else:
        state="green"

    # session used to send queries
    session=db.session()
    # executes query
    query = "match (e:names{{firstname:\"{}\"}}) return e"
    nodes=session.run(query.format(firstname))

    error = None
    #check if input username exists in database or not
    if  (nodes.single()):
        # If the previous query returns data, then user exists so raise an error
        error = "This user already exists"
        return render_template('register.html', error=error)
    else:
        #check if the input state is green
        # and there is a person with red state in same address or same company so will convert state to yellow
        if(state=="green"):
            statelist=[]
            #get all users state in input address
            ds="match (e:names)-[w:youraddress]->(d:addresses{{full_add: \"{}\" }}) return e.state"
            add_state=session.run(ds.format(full_add))
            #add states in statelist
            for i in add_state:
                statelist.append(i[0])
            #get all users state  in input company
```

Report

```
        ps="match (e:names)-[w:workplace] ->(d:work{{ place:\"{}\" }}) return e.state"
        place_state=session.run(ps.format(place))
        #add states in statelist
        for i in place_state:
            statelist.append(i[0])
        #search for red state in statelist and if it contains red state the input state will be converted from green to yellow
        for i in statelist:
            if(i=="red"):
                state="yellow"
                statelist.clear()
                break
    #create new node with input data
    ins = "create(e:names{{codeid: \"{}\" ,firstname: \"{}\" ,surname: \"{}\" ,dob: \"{}\" ,email: \"{}\" ,mobil: \"{}\" ,state:
\"{}\" }})"
    session.run(ins.format(codeid,firstname, surname, dob, email, mobil,state))
    #search for input place
    q1 = "match (e:work{{place:\"{}\"}}) return e"
    node1=session.run(q1.format(place))
    #if database dosen't contain input place :will create place node else it will connect user with this place
    if not(node1.single()):
        #create place node
        ins2="create(e:work{{place:\"{}\"}})"
        session.run(ins2.format(place))
    #connect user with input place
    ins22="match(e:names{{firstname:\"{}\"}}) ,(d:work{{place:\"{}\"}}) create(e)- [w:workplace]->(d)"
    session.run(ins22.format(firstname,place))
    #search for input address
    q2 = "match (e:addresses{{full_add:\"{}\"}}) return e"
    node2=session.run(q2.format(full_add))
    #if database dosen't contain input address :will create address node else it will connect user with this address
    if not(node2.single()):
        #create address node
        ins3="create(e:addresses{{full_add:\"{}\"}})"
        session.run(ins3.format(full_add))
    #connect user with input address
    ins33="match(e:names{{firstname:\"{}\"}}) ,(d:addresses{{full_add:\"{}\"}}) create(e)- [w:youraddress]->(d)"
    session.run(ins33.format(firstname,full_add))
    #end session
    session.close()
    #after submission render same page
    return render_template('register.html', results=0)
```

Report

- **searchForStatus() method:**

```
#search by user name
@app.route('/search', methods=['POST'])
def searchForStatus():
    #delete all images in img die
    dir=os.getcwd() + "\static\img"
    for file_name in os.listdir(dir):
        os.remove(dir + '\\' +file_name )
    #get user name you need to search for from form
    name = request.form['firstname']
    session=db.session()
    #search in database for user data  by firstname
    query = "match (e:names{{firstname:\"{}\"}}) return e.firstname,e.surname,e.dob,e.email,e.mobil,e.state"
    data=session.run(query.format(name))
    #add all user information l list
    l=list()
    for i in data:
        l.append(i[0])
        l.append(i[1])
        l.append(i[2])
        l.append(i[3])
        l.append(i[4])
        l.append(i[5])
    #search in database what is the address of specified user name
    query = "match (e:names{{firstname:\"{}\"}})-[w:youraddress]->(d:addresses) return d.full_add"
    full_add=session.run(query.format(name))
    #add address  into l list
    for i in full_add:
        l.append(i[0])
    #search in database what is the company name of specified user name
    query = "match (e:names{{firstname:\"{}\"}})-[w:workplace]->(d:work) return d.place"
    place=session.run(query.format(name))
    #add company name  into l list
    for i in place:
        l.append(i[0])
    session=db.session()
    #get states of all user in this company
    q10="match (e:names)-[w:workplace] ->(d:work{{place: \"{}\" }}) return e.firstname,e.state"
    w=session.run(q10.format(l[7]))
    #create graph to vizalize all user in this company
    g = nx.Graph()
    #create company node in createdgraph
    g.add_node(l[7],color='nothing',pos=(1,2))
```

Report

```
p=1
#create nodes for all user and set color node as state color .
for i in w:
    g.add_node(i[0] ,color=i[1],pos=(p,p))
    g.add_edge(i[0], l[7])
    p=p+1
    #draw edge btween user and company node
    nx.draw_networkx_edge_labels(g, pos=nx.spring_layout(g),edge_labels={(i[0], l[7]): ''},font_color='black')
color=nx.get_node_attributes(g,'color')
color_map = []
#set node color as users state color
for node in color.values():
    if(node=='green'):
        color_map.append('green')
    elif(node=='red'):
        color_map.append('red')
    elif(node=='yellow'):
        color_map.append('yellow')
    else:
       color_map.append('Cyan')
pos=nx.get_node_attributes(g,'pos')
#drow graph
nx.draw(g, pos,with_labels = True,node_size=4000,node_color=color_map)
plt.tight_layout()
#save the graph as image in img directory
plt.savefig(os.getcwd()+"\static\img\Graph.png", format="PNG")
plt.clf()
# get states of all user in this address
q5="match (e:names)-[w:youraddress]->(d:addresses{{full_add: \"{}\" }}) return e.firstname,e.state"
w=session.run(q5.format(l[6]))
#create graph to vizalize all user in this address
g = nx.Graph()
#create address node in created graph
g.add_node(l[6],color='nothing',pos=(1,2))
p=1
#create nodes for all user and set color node as state color .
for i in w:
    g.add_node(i[0] ,color=i[1],pos=(p,p))
    g.add_edge(i[0], l[6])
    p=p+1
    #draw edge btween user and address node
    nx.draw_networkx_edge_labels(g, pos=nx.spring_layout(g),edge_labels={(i[0], l[6]): ''},font_color='black')
color=nx.get_node_attributes(g,'color')
color_map = []
#set node color as users state color
```

Report

```
    for node in color.values():
        if(node=='green'):
            color_map.append('green')
        elif(node=='red'):
            color_map.append('red')
        elif(node=='yellow'):
            color_map.append('yellow')
        else:
            color_map.append('Cyan')
    pos=nx.get_node_attributes(g,'pos')
    #drow graph
    nx.draw(g, pos,with_labels = True,node_size=4000,node_color=color_map)
    plt.tight_layout()
    #save the graph as image in img directory
    plt.savefig(os.getcwd()+"\static\img\Graph1.png", format="PNG")
    plt.clf()
    #send results to html page and update it to display images
    return render_template('index.html', results=l)
```

- **changeStatu() method:**

```
@app.route('/change/state', methods=['POST'])
def changeStatu():
    #get user name and new state from form
    firstname = request.form['firstname']
    newstate = request.form['newstate']
    session=db.session()
    #if new state is red you need to change all users has green state and in same address or company of spacified
user
    if (newstate=="red"):
        #get user addresse
        query = "match (e:names{{firstname:\"{}\"}})-[w:youraddress]->(d:addresses) return d.full_add"
        full_add=session.run(query.format(firstname))
        #create l list and add address to this list
        l=list()
        for i in full_add:
            l.append(i[0])
        full_add=l[0]
        #get user company
        query = "match (e:names{{firstname:\"{}\"}})-[w:workplace]->(d:work) return d.place"
        place=session.run(query.format(firstname))
        #add company to the l list
        for i in place:
            l.append(i[0])
```

Report

```
        place=l[1]
        #get all users have the same addresse and state green and set there state to yellow
        query="match (e:names{{state:'green'}})-[w:youraddress]->(d:addresses{{full_add:\"{}\"}}) set
e.state='yellow'"
        session.run(query.format(full_add))
        #get all user have the same company and state green and set there state to yellow
        query="match (e:names{{state:'green'}})-[w:workplace] ->(d:work{{place:\"{}\"}})  set e.state='yellow'"
        session.run(query.format(place))
        #finally change state of user to red
        query = "match (e:names{{firstname:\"{}\"}}) set e.state=\"{}\" "
        session.run(query.format(firstname,newstate))
    else:
        #change state of user to green or yellow as he choice
        query = "match (e:names{{firstname:\"{}\"}}) set e.state=\"{}\" return e"
        session.run(query.format(firstname,newstate))
    return render_template('change.html', results=0)
```

****