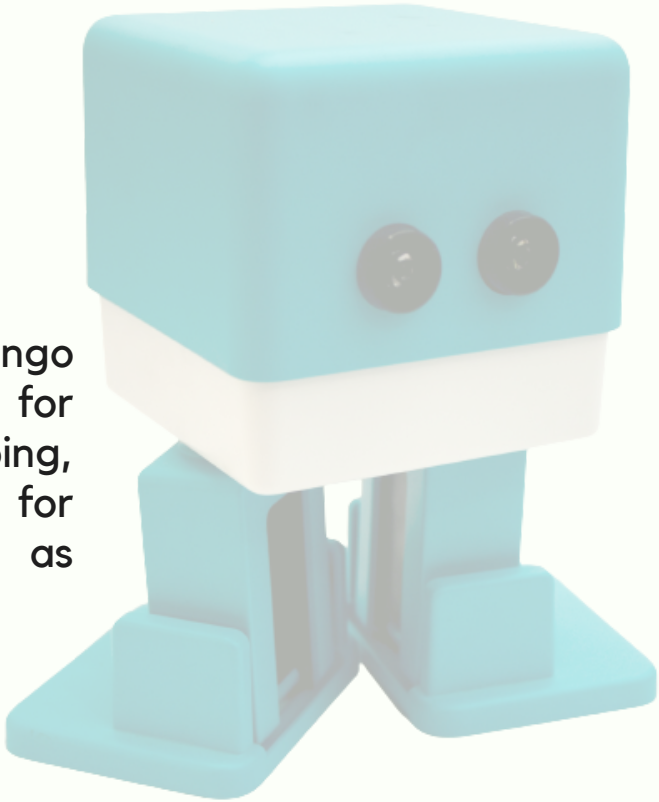
AI2101 - PROG FOR AI - A2

STELLA

AI - ChatBot

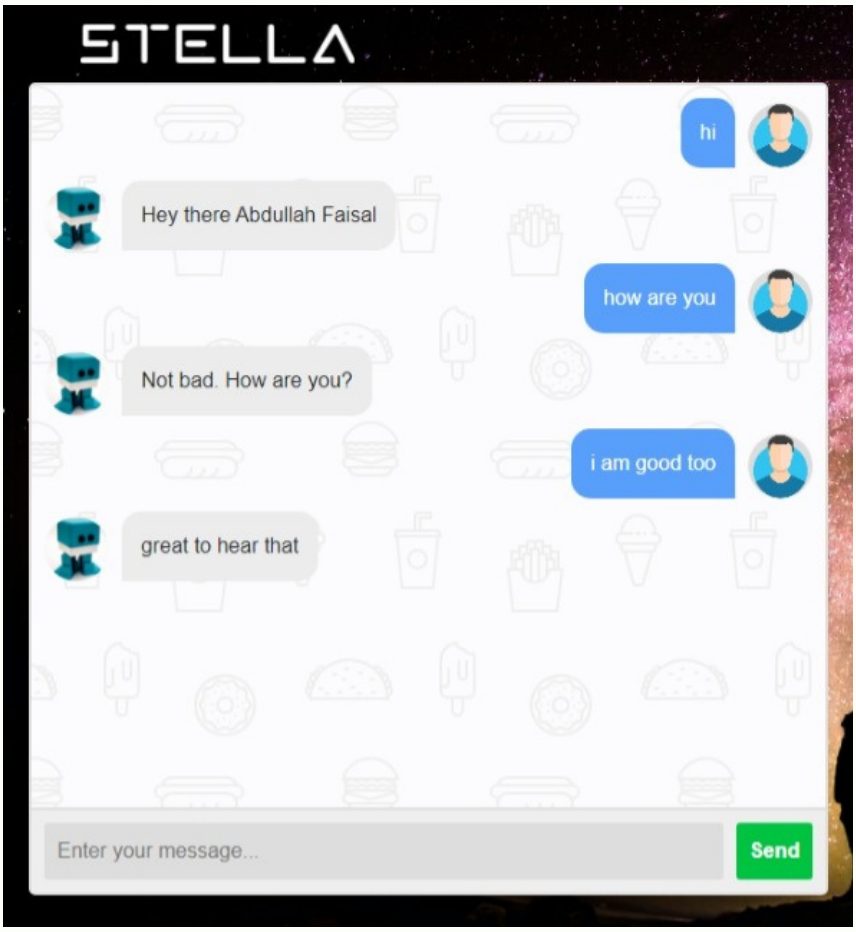


Project Documentation

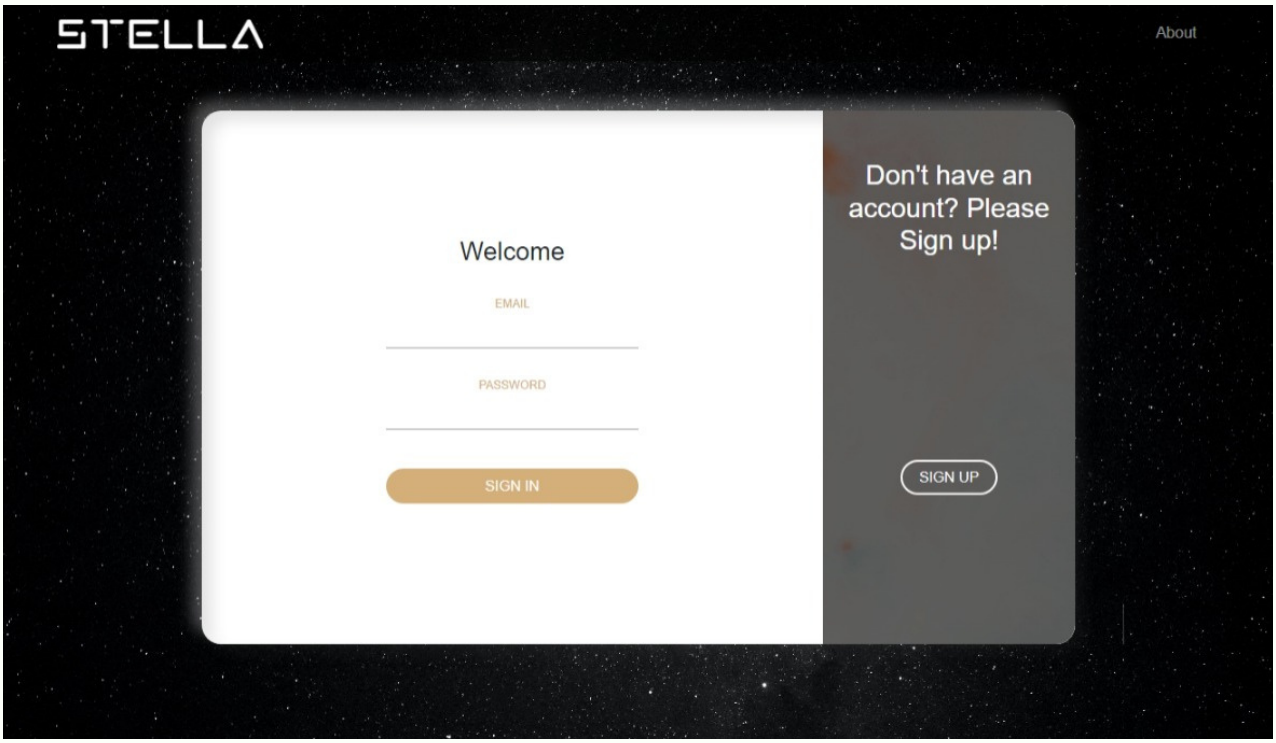
AI chat bot that uses Python-Django for server side, HTML, CSS and JS for frontend, AIML, Prolog, web scrapping, WordNet and machine learning for backend functionality and neo4j as database

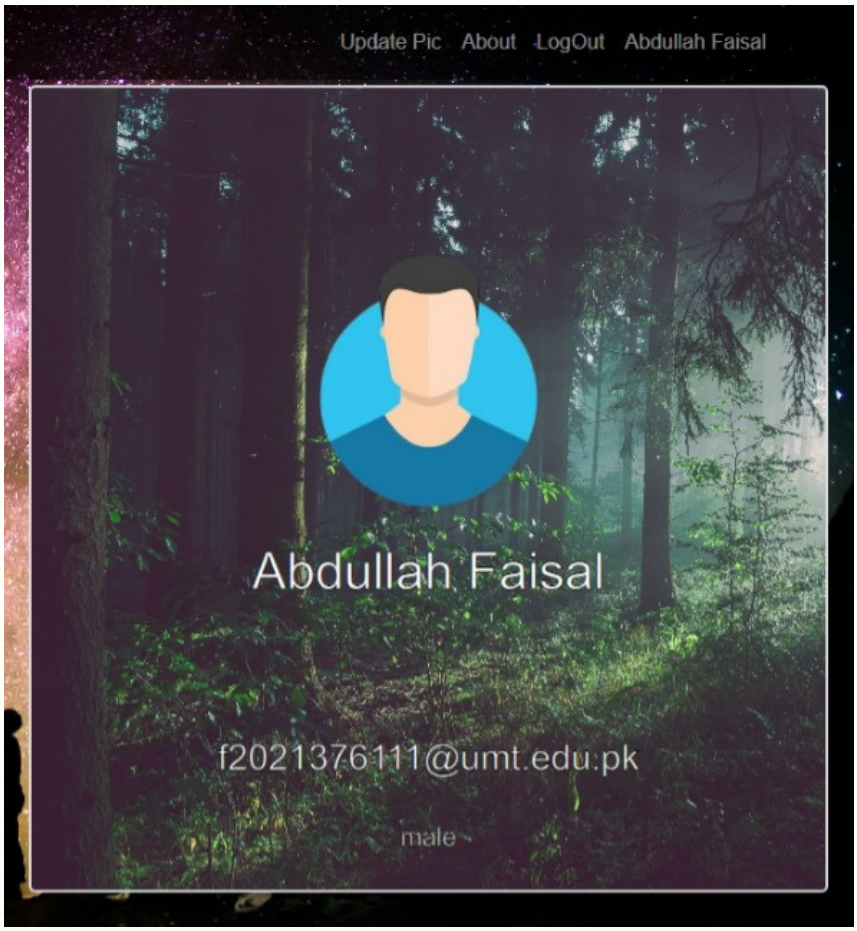
Prepared by: Abdullah Faisal,
F2021376111

Resource Person: Prof. Mahmood Hussain
BS AI

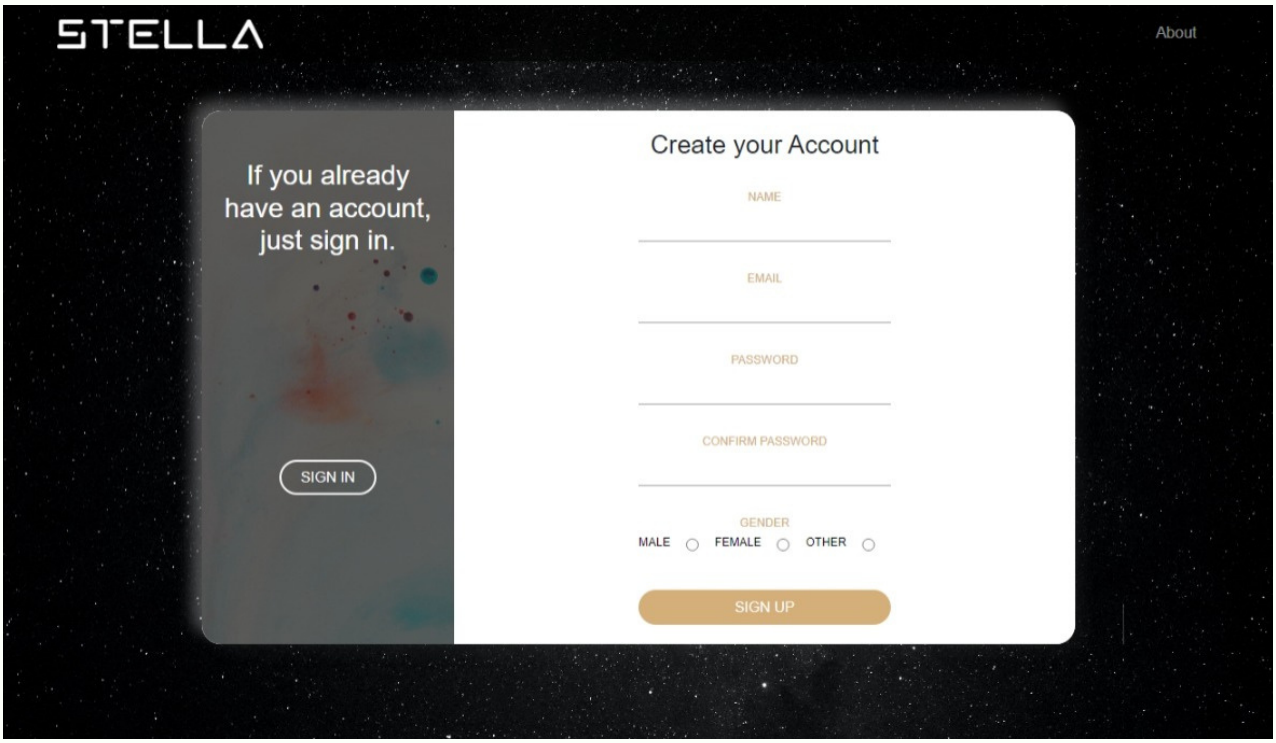

STELLA
hi
Hey there Abdullah Faisal
how are you
Not bad. How are you?
i am good too
great to hear that
Enter your message...
Send

SIGN-IN


STELLA
About
Welcome
EMAIL
PASSWORD
SIGN IN
Don't have an account? Please Sign up!
SIGN UP


Update Pic
About
LogOut
Abdullah Faisal
Abdullah Faisal
f2021376111@umt.edu.pk
male

SIGN-UP


STELLA
About
If you already have an account, just sign in.
SIGN IN
Create your Account
NAME
EMAIL
PASSWORD
CONFIRM PASSWORD
GENDER
MALE
FEMALE
OTHER
SIGN UP

File Structure

```
> chatbot
 |_>chatbot (django's default)
 |_>stella (my app)
   |_>chatbot (functionality of bot)
     |_>aiml_files
     |_>ML (5 models)
     |_ bot.py (main bot file)
     |_ knowledge.pkl (prolog knowledge)
     |_ my_neo4j.py
     |_ prolog.py
     |_ scrapping.py (wikipedia web scrap)
     |_ spell_training_data.txt
   |_>migrations
   |_>static
     |_>css  (styles of web pages)
     |_>js  (scripts of web pages)
     |_>pics  (pics used in project)
     |_>profile_pics  (user pfps)
   |_>templates  (all html files)
     # some more files
     decorators.py
     models.py
     urls.py
     views.py
 manage.py
>venv
```

Code is available at https://github.com/pmchohan/stella

Sign-IN

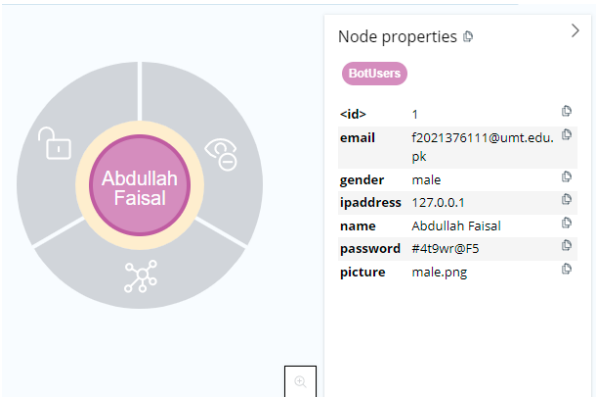
If a user is signed up already. S/He can sign in. There are some checks on sign in. If an email is not registered it will show an error and same is the case if password is incorrect. When a user is successfully logged in or signed up its id, name and email from neo4j is saved in django-session.

```
38  def signin(requests):
39      email = requests.POST.get('email')
40      key = requests.POST.get('key')
41      errors = problems_in_signin(email, key)
42      if len(errors['prob']) != 0:
43          return JsonResponse(errors)
44      user = BotUsers.nodes.get(email=email)
45      requests.session['user_id'] = user.id
46      requests.session['user_email'] = user.email
47      requests.session['user_name'] = user.name
48      return redirect('bot')
```

Sign-UP

When user signs up a node is created in neo4j database using django-neomodel. There are also some checks like if an email is already registered it will show an error, same is the case if password field does not match with confirm password field.

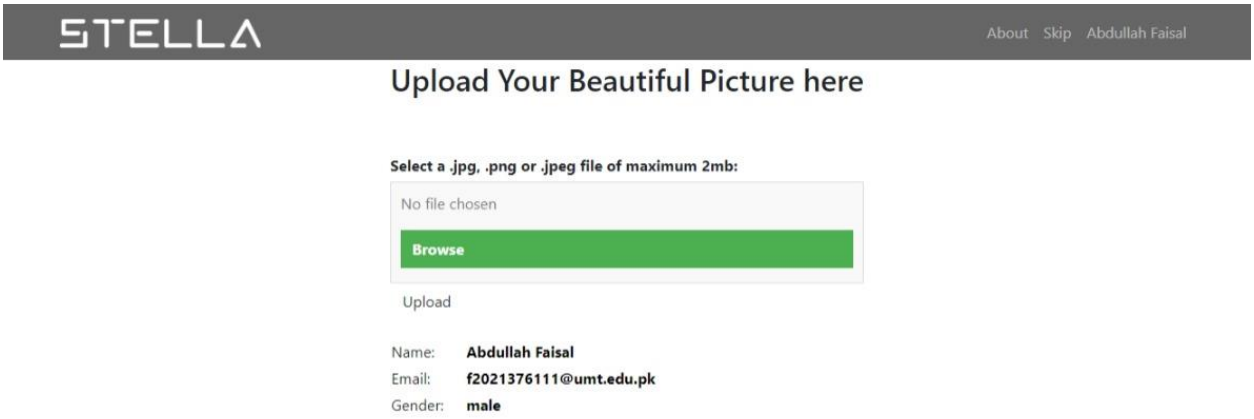
```python
def signup(requests):
    ip = requests.META['REMOTE_ADDR']
    email = requests.POST.get('email')
    key1 = requests.POST.get('key1')
    name = str(requests.POST.get('name'))
    key2 = requests.POST.get('key2')
    gender = requests.POST.get('gender', 'other')
    errors = problems_in_signup(email, key1, key2)
    if len(errors['prob']) != 0:
        return JsonResponse(errors)
    pfp = gender+'.png'
    user = BotUsers(name=name.title(), gender=gender,
    user.save()
    print('user created')
    requests.session['user_id'] = user.id
    requests.session['user_email'] = user.email
    requests.session['user_name'] = user.name
    print('sessions set')
    return redirect('upload')
```



Profile Picture

When user signs up, s/he is prompted to upload their profile picture. If they chose to skip uploading their profile picture, default picture based on their gender will be set. Default profile picture could be male.png, female.png or others.png, but if a user uploads their profile picture it is saved with the name derived from email (abcxyz for abc@xyz.com) and the picture is saved in chatbot > static > profile_pics.

```python
@login_required
def uploaded(requests):
    if requests.method == 'POST':
        print('pfp post')
        image = requests.FILES['file']
        user_email = str(requests.POST.get('email'))
        pfp = create_file_name(user_email)
        file_path = './stella/static/profile_pics/'+pfp
        with open(file_path, 'wb') as f:
            for chunk in image.chunks():
                f.write(chunk)
        user = BotUsers.nodes.get(email=user_email)
        user.picture = pfp
        user.save()
        return redirect('bot')
```



Decorators

Django decorators are used to restrict user from manually going on bot without login through url. For this purpose checks are added on sessions. If session is set user will be redirected to chat screen without login password. But if user is not logged in S/He will be redirected to sign in screen

```
4 usages  pmchohan *
def login_required(view_func):
    pmchohan *
    def wrapped_func(requests, *args, **kwargs):
        if not requests.session.get('user_email'):
            return redirect('home')

        return view_func(requests, *args, **kwargs)
    return wrapped_func


2 usages  pmchohan *
def logged_in(view_func):
    pmchohan *
    def wrapped_func(requests, *args, **kwargs):
        if requests.session.get('user_email'):
            return redirect('bot')

        return view_func(requests, *args, **kwargs)
    return wrapped_func
```

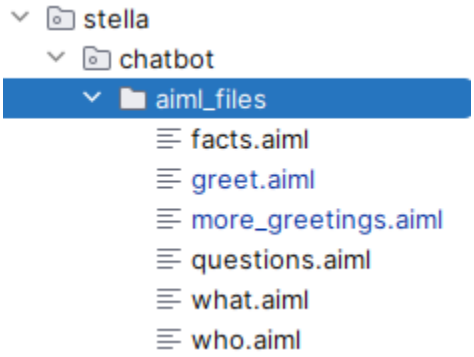
Chat Screen

After successfully logging in or signing up, the user is finally at a screen where S/He can chat with STELLA. This page is divided in two parts, on left there is a chatbox div and on right lies the user information.

Functionalities

AIML

For basic chat aiml is used and for this purpose python module ***pyaiml21*** is used. There are only few aiml files. I might update this project in free time.



GET/SET Predicate

Instead of using nlp I have used AIML get set predicates which helps me to decide that when to give response from AIML, Wikipedia or Prolog knowledge base.

```
52          result = check_predicates(myBot, u_id)
53          if result:
54              print('entered prolog')
55              response = result
56          asked_about = myBot.get_predicate('what', u_id)
57          if asked_about != 'unknown':
58              response = what_is(asked_about)
59              myBot.respond('remove what predicate', u_id)
60          asked_about_person = myBot.get_predicate('who', u_id)
61          if asked_about_person != 'unknown':
62              response = what_is(asked_about_person, 'person')
63              myBot.respond('remove who predicate', u_id)
64          responses += response + '. '
65          print('response:', responses)
```

```
2 usages  pmchohan
def check_predicates(mybot, user_id):
    global bot, uid
    bot = mybot
    uid = user_id
    male = bot.getPredicate('male', uid)
    female = bot.getPredicate('female', uid)
    parent = bot.getPredicate('parent', uid)
    relation = bot.getPredicate('relation', uid)
    child = bot.getPredicate('child', uid)
    who_is = bot.getPredicate('who_is', uid)
    who_is_of = bot.getPredicate('who_is_of', uid)

    result = None

    if male != 'unknown':
        set_fact('male', male)
    elif female != 'unknown':
        set_fact('female', female)
    elif relation != 'unknown':
        set_fact(relation, parent, child)
    elif who_is != 'unknown':
        result = query_kb(who_is, who_is_of)
        if result:
            result = who_is.lower() + ' of ' + who_is_of + ' is/are ' + result

    return result
```

Spell Checker

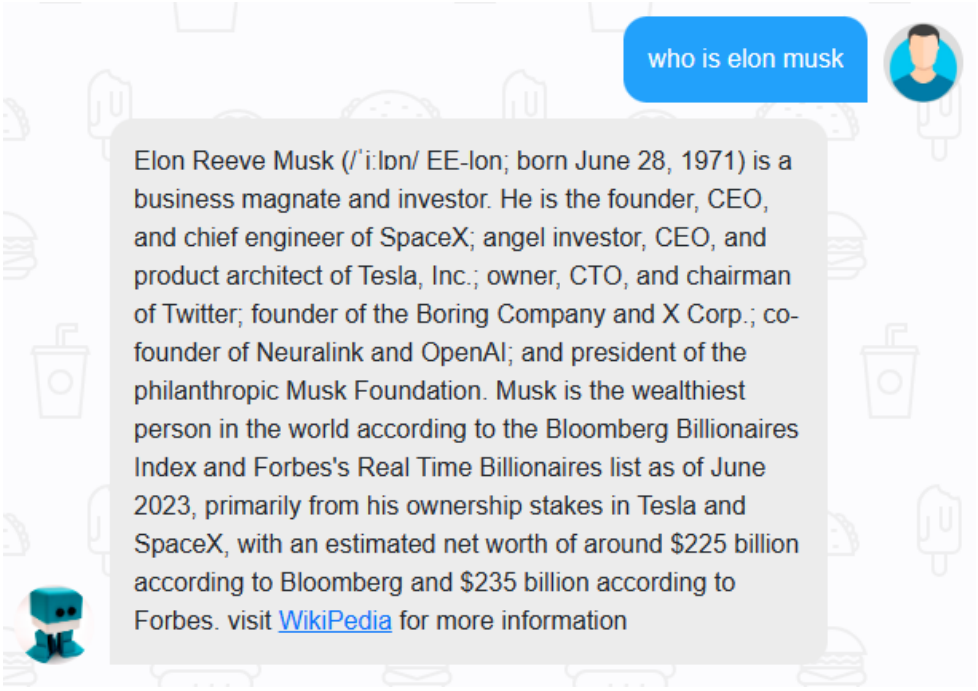
Instead of using normal spell checker like textblob I have used ***spello,*** I have trained it on the type of texts that it can expect to receive based on my aiml files.

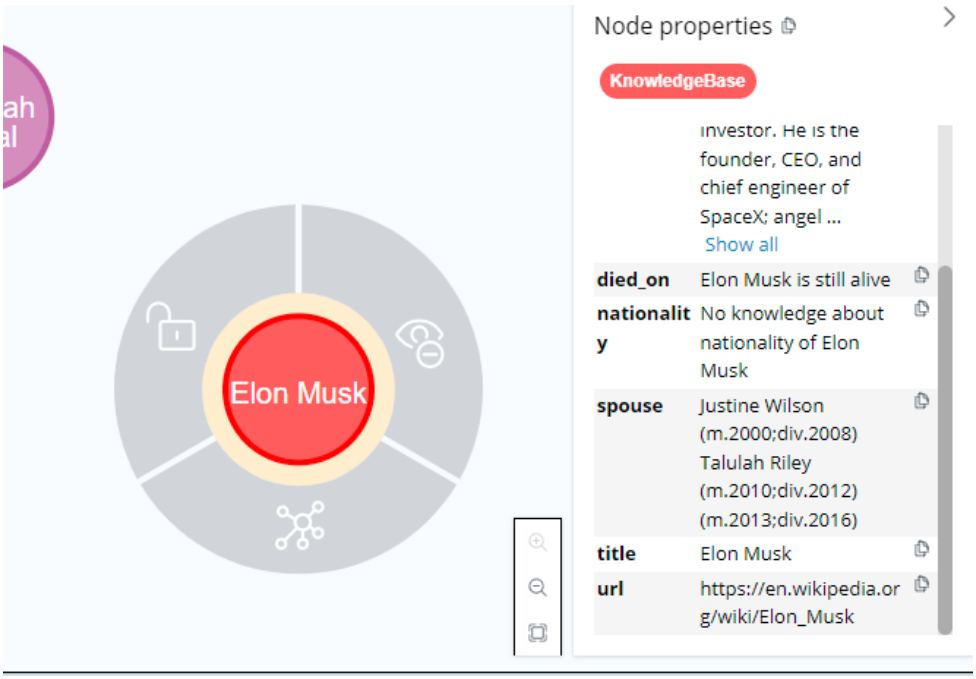
```
1 usage  pmchohan
def spell_checker():
    # spell checker/corrector initialization
    global corrector
    corrector = SpellCorrectionModel(language="en")
    with open(r'C:\Users\abdul\PycharmProjects\chatbo
        data = file.readlines()

    data = [i.strip() for i in data]
    corrector.train(data)
```

Web Scrapping

Since this bot can not answer everything so I am scrapping Wikipedia. It scrapes Wikipedia using ***beautifulsoup*** for basic info and If an entity for which we are searching is a person I also get birth_date, death_date (if dead), nationality and spouse(s) information. For Organization along with basic information it also gets headquarter location, founding date, founder and website of that organization. It also saves it in neo4j. And when user asks again for it it will not scrape Wikipedia instead it will get it from neo4j.

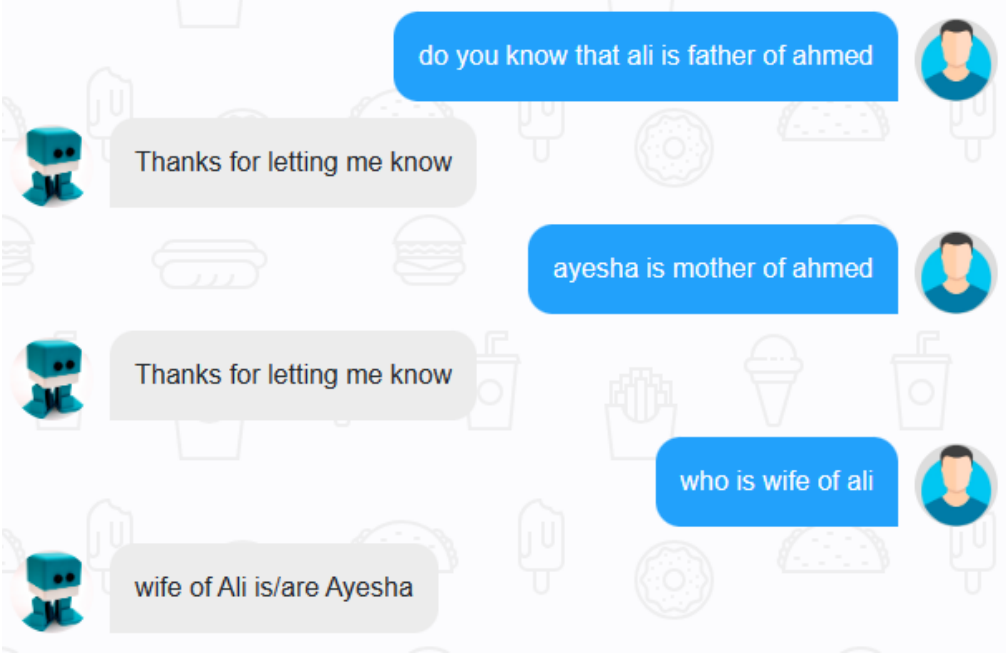




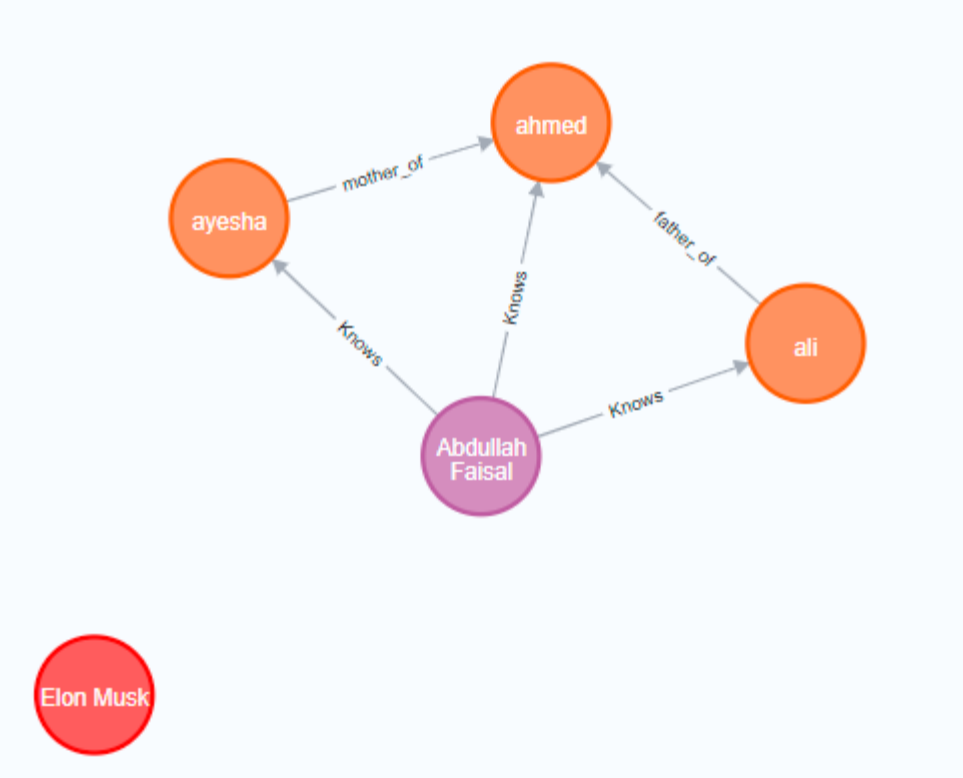
Prolog

To deal with relationships and inference on base of some facts prolog is implemented using the ***pytholog*** library. A list with rules for relationships is already present and if user gives us more facts and asks us to infer some relationship we do it and update the knowledge list and save it in a pickle file. The relationships and facts also goes to neo4j.

Inference:



Neo4j:



Machine Learning

While creating a prolog relationship if person fact does not already exists his/her gender will be predicted using machine learning and a fact with gender will be created first. Since my names dataset was not large enough I wrote 5 machine learning models (logistic Regression, Multinomial Naïve Bayes, SVM SVC, Recurrent neural network and deep neural network) using **tensorflow** and **scikit-learn**. And at the end average is calculated of result of all predictions.

```
3 usages   pmchohan
def predict_gender(name):
    lr_p = lr(name)
    nb_p = nb(name)
    svm_p = svc(name)
    rnn_p = rnn(name)
    dnn_p = dnn(name)
    result = (lr_p + nb_p + svm_p + rnn_p + dnn_p)/5
    print(result)
    return 'male' if result <= 0.5 else 'female'
```

The models, their vectorizer and tokenizer are saved in pickle files, so instead of training models again and again it will learn from file to save time.