CHATBOT PROJECT DOCUMENTATION

DESCRIPTION

The project is developed with a ACBR algorithm .It is developed for customer service interaction

The idea is to give a automatic response to the customer who reach company website for any requirement, so that they can interact with the chatbot

The customer at one end and the admin at the other end can effectively chat .

AGENTA

LIST OF API - total API's	2
DB STRUCTURE	3
FUNCTIONALITY	9
API EXAMPLES	16

**** RUD - retrieve update , delete**
**** LC - List and create**

LIST OF API's

User

1. User Login
2. User Logout
3. User List
4. User Create
5. User List <id>
6. User RUD <id>

User info

User info <id>

In-app chat
(automatic chat reply)

7. keyword
8. domain
9. services
10. Technologies
11. Messages
12. category
13. Full Length keyword
14. General keyword
15. Static keyword
16. Grammatical keyword

Admin login status

17. Login status

Customer service
Interaction

18. Human
19. Human <id>

DB TABLE

- **TABLE NAME : USER**

FIELDS	**TYPE**	**OTHERS**
Email **	emailfield	Null = true
password **	varchar	
Phone	varchar	Len = 15
Name **	varchar	Default = “Test”
user_type	varchar	Default = “anonymous”
scope	varchar	Len = 500
domain	varchar	Len = 50
is_active	varchar	Default = True
is_staff	varchar	Default = false
is_online	varchar	Default = false

** - required fields

- **TABLE NAME : USER-INFO**

FIELDS	**TYPE**	**OTHERS**
city	varchar	
region	varchar	
country	varchar	
zip_code	varchar	
ip_address	varchar	
Country code	varchar	
lat	decimal	
long	decimal	
user	Onetoone (**user**)	

OTHER TABLES

1. DOMAIN

TABLE NAME : DOMAIN

FIELD	TYPE
name	varchar

2. SERVICE

TABLE NAME : SERVICE

FIELD	TYPE
service	varchar
did	foreignkey(domain)

3. TECHNOLOGIES

TABLE NAME : technologies

FIELD	TYPE

technology	varchar
sid	foreignkey(service)

4. CATEGORIES

TABLE NAME : categories

FIELD	TYPE
category	varchar
tid	foreignkey(technology)

5. KEYWORD

TABLE NAME : keyword

FIELD	TYPE
Keyword	varchar
key_answer	text
tid	foreignkey(technology)

6. FULL LENGTH KEYWORD

TABLE NAME : full length keyword

FIELD	TYPE
keyword	varchar

full_answer	text

7. GENERAL KEYWORD

TABLE NAME : general keyword

FIELD	TYPE
Keyword	varchar
gen_answer	text

8. STATIC KEYWORD

TABLE NAME : static keyword

FIELD	TYPE
keyword	varchar

9. GRAMMATICAL KEYWORD

TABLE NAME : grammatical keyword

FIELD	TYPE
keyword	varchar

10. MESSAGES

TABLE NAME : messages

FIELD	TYPE
msg	textfield
time	Timefield
aid	foreignkey(keyword)
faid	foreignkey(fulllengthkeyword)
gaid	foreignkey(generalkeyword)
did	foreignkey(Domains)
sid	foreignkey(services)
tid	foreignkey(technologies)
cid	foreignkey(categories)
uid	foreignkey(User)
relevant	Boolean
user	Foreignkey(User)
other	ForeignKey(User)
is_read	Boolean

FUNCTIONALITY FOR AUTH API

1. User login

Fields : **email** and **password**
Response :


```
{
    "token": "f49131dccc7a99fe1d8bd80bcfdfab85b0baf3c1",
    "status": true
}
```


2. User create

Fields : **email , password , name** { default = “test” }
Response :


```
 {
        "id": 1,
        "email": "12435245246538930@test.com",
        "phone": null,
        "name": "Test",
        "user_type": "anonymous",
        "last_login": "2021-10-21T17:29:30.789206Z",
        "scope": "",
        "domain": "",
        "user_name": null,
        "is_staff": false,
        "is_online": false
    }
]
```


3. User Update

Fields : __all__

Response after update :


```
{
        "id": 1,
        "email": "anand@gmail.com",
        "phone": null,
        "name": "anand",
        "user_type": "customer",
        "last_login": "2021-10-21T17:29:30.789206Z",
        "scope": "",
        "domain": "",
        "user_name": null,
        "is_staff": false,
        "is_online": false
    }
]
```


Mail action

* whenever user details are updates it should send email to the updated user's **email** as message "welcome to desss inc" and share the updated user details to dev@desss.com and 7135578001@vtext.com

** from the angular side the user created by means of {timestamp@test.com) , so when user update the email to **{testuser@gmail.com} ,** it should perform the mail action .

4. **User Info**

Fields : __all__

*whenever user login , the user's geolocation and ip address are captured and saved to the User info table .

FUNCTIONALITY FOR ADMIN LOGIN

1. **Admin status**

   *When an admin user login in this api it should return a response as and need to change **false** when logged out .

```
{
    "status": true
}
```

FUNCTIONALITY FOR IN-CHAT

1. **Messages**

In Message API , users send a message in the chat screen to the chatbot , and meanwhile get an auto reply based on what he/she typed .

Fields : check message table no.10

Response :

```
** some black and bold fields are extra fields merged from another table
```

```
[
    {
        "id": 1,
        "time": "05:23:11.543003",
        "uid": 2,
        "user_email": "ganesh@desss.com",
        "user_phone": "1234567890",
        "user_name": "ganesh",
        "last_login": "2021-10-04T04:24:35.852356Z",
        "scope": "",
        "u_domain": "",
        "msg": "hey",
        "aid": null,
        "key_answer": null,
        "gaid": null,
        "gen_answer": null,
        "faid": null,
        "full_answer": null,
        "did": null,
        "domain": null,
        "sid": null,
        "service": null,
        "tid": null,
        "tech": null,
        "relevant": false,
        "cid": null,
        "user": 2,
        "other": 1,
        "is_read": true
    },
```


2. Automatic chat response structure

Domain -> service -> Technologies

FUNCTIONALITY FOR CUSTOMER SERVICE INTERACTION

1. human

In the customer service interaction API , the user starts to message to the admin who is on the other end of the chat screen .

Fields : check message table no.10

Response :

**** some black and bold fields are extra fields merged from another table**


```
[
    {
        "id": 1,
        "time": "05:23:11.543003",
        "uid": 2,
        "user_email": "ganesh@desss.com",
        "user_phone": "1234567890",
        "user_name": "ganesh",
        "last_login": "2021-10-04T04:24:35.852356Z",
        "scope": "",
        "u_domain": "",
        "msg": "hey",
        "aid": null,
        "key_answer": null,
        "gaid": null,
        "gen_answer": null,
        "faid": null,
        "full_answer": null,
        "did": null,
        "domain": null,
        "sid": null,
        "service": null,
        "tid": null,
        "tech": null,
        "relevant": false,
        "cid": null,
        "user": 2,
```

```
        "other": 1,
        "is_read": true
    },
```


2. List messages between sender and receiver

In human interaction API , both the customer at one end and the admin at the other end communicate with each other . It is a sort of real time chat.

Url : xyz.com/human/?user=1&other=2

Response :


```
{
    "data": [
        {
            "id": 3,
            "time": "13:42:43.498359",
            "uid": 1,
            "user_email": "anand@desss.com",
            "user_phone": null,
            "user_name": "Test",
            "last_login": "2021-10-21T17:40:12.671754Z",
            "scope": "",
            "u_domain": "",
            "msg": "hi customer 1",
            "aid": null,
            "key_answer": null,
            "gaid": null,
            "gen_answer": null,
            "faid": null,
            "full_answer": null,
            "did": null,
            "domain": null,
            "sid": null,
            "service": null,
            "tid": null,
            "tech": null,
            "relevant": false,
            "cid": null,
            "user": 1,
            "other": 2,
            "is_read": false
        },
```

```
        {
            "id": 1,
            "time": "12:52:27.447215",
            "uid": 2,
            "user_email": "customer@gmail.com",
            "user_phone": "789456123",
            "user_name": "customer",
            "last_login": null,
            "scope": "",
            "u_domain": "",
            "msg": "hey this is customer",
            "aid": null,
            "key_answer": null,
            "gaid": null,
            "gen_answer": null,
            "faid": null,
            "full_answer": null,
            "did": null,
            "domain": null,
            "sid": null,
            "service": null,
            "tid": null,
            "tech": null,
            "relevant": false,
            "cid": null,
            "user": 2,
            "other": 1,
            "is_read": true
        }
    ]
}
```


3. Message count

Count the number of unread messages for the admin user side , from whom the customer has sent messages to the admin .
Implement this count response in **api/human/ -** message list api

Fields : count : **{** “user_id” **:** “no.of unread_messages” **}**

Response :


```
{
    "count": {
        "3": 2,
        "2": 1
    },
```

API EXAMPLES

1. api/ user/login/ **[USER-LOGIN]**
2. api/ user/logout **[USER-LOGOUT]**
3. api/ user/ **[USER-LIST]**
4. api/ user/<int:pk>/ **[USER-RUD]**
5. api/ userinfo/ **[USER-INFO]**
6. api/ userinfo/<int:pk>/ **[USER-INFO RUD]**
7. api/ keyword/ **[KEYWORD- LIST & CREATE]**
8. api/ keyword/<int:pk>/ **[KEYWORD RUD]**
9. api/ domain/ **[DOMAIN- LIST & CREATE]**
10. api/ domain/<int:pk>/ **[DOMAIN RUD]**
11. api/ service/ **[SERVICE- LIST AND CREATE]**
12. api/ service/<int:pk>/ **[SERVICE RUD]**
13. api/ tech/ **[TECHNOLOGY LIST & CREATE]**
14. api/ tech/<int:pk>/ **[TECHNOLOGY RUD]**
15. api/ category/ **[CATEGORY LIST & CREATE]**
16. api/ category/<int:pk>/ **[CATEGORY RUD]**
17. api/ message/ **[MESSAGES LIST & CREATE]**
18. api/ message/<int:pk>/ **[MESSAGES RUD]**
19. api/ fulllenkeyword/ **[KEYWORD LIST AND CREATE]**
20. api/ fulllenkeyword/<int:pk>/
21. api/ genkeyword/
22. api/ genkeyword/<int:pk>/
23. api/ statickeyword/

24. api/ statickeyword/<int:pk>/
25. api/ gramaticalkeyword/
26. api/ gramaticalkeyword/<int:pk>/
27. api/ adminstatus/ **[ADMIN - LOGIN STATUS]**
28. api/ human/ **[CUSTOMER SERVICE INTERACTION]**
29. api/ human/<int:pk>/

E.G , api/human/?user=1&other=2

[LIST CHAT BETWEEN USER AND ADMIN]