

Doctor Appointment Making Chatbot Web Application

Introduction: This project is about making a chatbot web application where users can make an appointment to a doctor of a certain sector. The user will have to register on the website and login with their credentials. The user can talk with the chatbot and make the appointment easily. Plus they can see their appointment details, even they can edit their profile. There is also an Email sending feature where users will get an Email after they make an appointment.

Structure:

There are 3 main parts of this web application. These are:

1. Chatbot application
2. Backend with Python Django
3. Frontend with HTML, CSS, Bootstrap 5,4

Chatbot Application:

Chatbot application has 3 parts.

1. Dataset
2. Response getter
3. Implementing chatbot in web application

Dataset: Dataset is the conversation samples of normal users. It consists of questions and their relative answers. If a query is passed to the chatbot, chatbot will try to match the query with the dataset's question sets. If the query is match, then the chatbot will fetch the relative answers.

Response getter: When a query is passed to the chatbot, the response getter function will check the similarity score with the dataset. Here is the code snippet of the similarity function that has been used in this project.

```
def similar(a, b):  
    return SequenceMatcher(None, a, b).ratio()
```

It will return a floating number score of similarity level. If the similarity score is greater than greater than or equal to 0.9 , then it will reply the relative answer. If not it will show "IDKresponse". Thus we get the responses in this way.

Implementing chatbot in web application:

Here is the code snippet of implementation of the chatbot code in the web application.

```
def sector(request):  
    query = request.POST.get("query")  
    bot_reply = getResponse(query)  
    if bot_reply=="make_appointment":  
        context = {"state":2}  
        return render(request, "chatting1.html", context)  
    else:  
        response = redirect("/chatbot/chat/")  
        return response
```

We imported the response getter file and the method “getResponse”. Then we pass the result, process it, and show it to the user in a web application.

Backend with Python Django:

There are 3 main parts of the backend. These are:

1. Controller
2. Database
3. Routing

Controller: In this sector, the functions are declared which are used in the web application. This sector declares the instructions of how the website will be operated.

There are 2 roles for the web application.

1. Admin
2. Normal users

Database: For this project, MySQL database has been used. In the models.py file, the tables of the database have been created along with the table information. In the settings.py file, the information of the database system, and the name of the database has been stored. Finally after finishing these steps, two commands has been run in the command prompt.

1. python manage.py makemigrations
2. python manage.py migrate

Let's look at our database system.

Table	Action	Rows	Type	Collation	Size	Overhead
<input type="checkbox"/> account_user	★ Browse Structure Search Insert Empty Drop	3	InnoDB	utf8mb4_general_ci	16.0 KiB	-
<input type="checkbox"/> admin_doctor	★ Browse Structure Search Insert Empty Drop	9	InnoDB	utf8mb4_general_ci	16.0 KiB	-
<input type="checkbox"/> auth_group	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	32.0 KiB	-
<input type="checkbox"/> auth_group_permissions	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	48.0 KiB	-
<input type="checkbox"/> auth_permission	★ Browse Structure Search Insert Empty Drop	32	InnoDB	utf8mb4_general_ci	32.0 KiB	-
<input type="checkbox"/> auth_user	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	32.0 KiB	-
<input type="checkbox"/> auth_user_groups	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	48.0 KiB	-
<input type="checkbox"/> auth_user_user_permissions	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	48.0 KiB	-
<input type="checkbox"/> chatbot_appointment	★ Browse Structure Search Insert Empty Drop	1	InnoDB	utf8mb4_general_ci	16.0 KiB	-
<input type="checkbox"/> django_content_type	★ Browse Structure Search Insert Empty Drop	8	InnoDB	utf8mb4_general_ci	48.0 KiB	-
<input type="checkbox"/> django_migrations	★ Browse Structure Search Insert Empty Drop	18	InnoDB	utf8mb4_general_ci	16.0 KiB	-
<input type="checkbox"/> django_session	★ Browse Structure Search Insert Empty Drop	1	InnoDB	utf8mb4_general_ci	32.0 KiB	-
12 tables	Sum	72	InnoDB	utf8mb4_general_ci	384.0 KiB	0 B

⬅ ☐ Check all With selected: ▾

Routing:

In the Routing sector of the django backend, the urls are declared. Urls helps us to go to any web pages that we want. Routing sector will redirect our requests, and get us on the requested page. In the urls.py file, we declare the routes or url paths.

Let's look at some portion of the routing code snippet.

```
from django.urls import path
from . import views

urlpatterns = [
    path('', views.gotoHome , name = 'home-page'),
    path('available/' , views.available_doctors , name = 'available'),
    path('chat/' , views.gotoChat , name="goto-chat"),
    path('sector/' , views.sector , name="ch-sector"),
    path('doctors/' , views.cho_doctor , name="ch-doctor"),
    path('confirm/' , views.confirming , name="confirm-app"),
    path('app_list/' , views.gotoAppList , name="list-app"),
    path('gotoProfileUpdate/' , views.gotoUpdateProfile,
name="go-update-profile"),
```

```
path('update', views.updateProfile,name="update-profile")
]
```

Frontend with HTML, CSS, Bootstrap 5,4:

In this sector, we build the views to let users see what they are doing and get query from them. In the 'templates' folder, the html files are stored.
Let's look at our website.

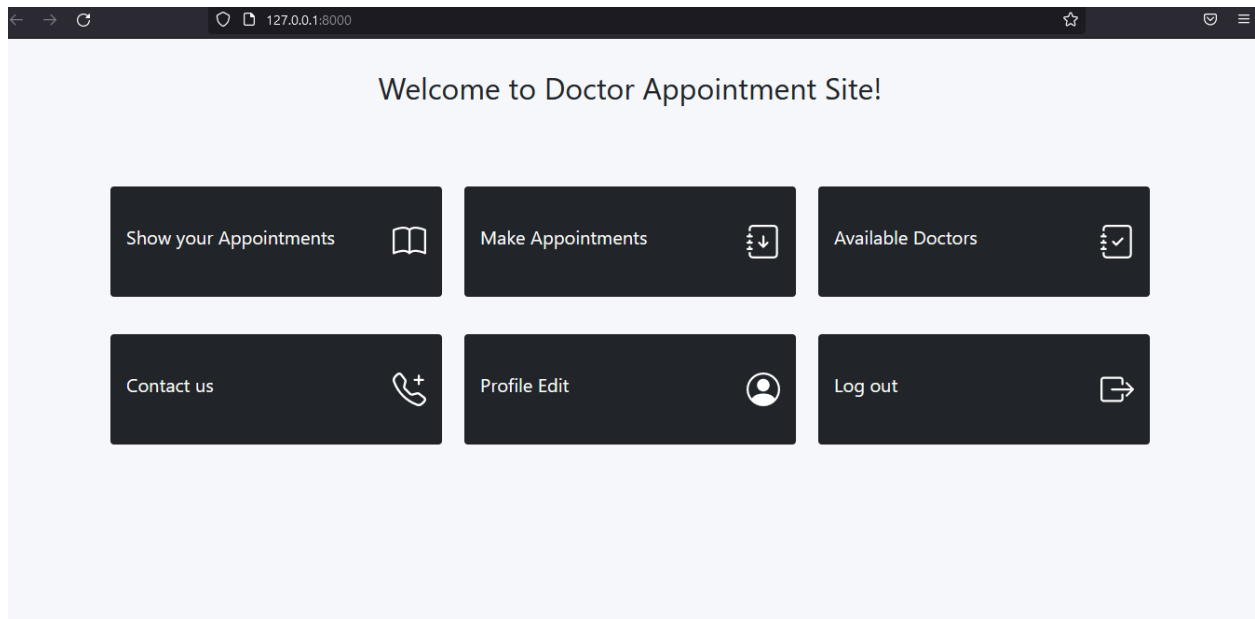


Fig: Normal user's home page

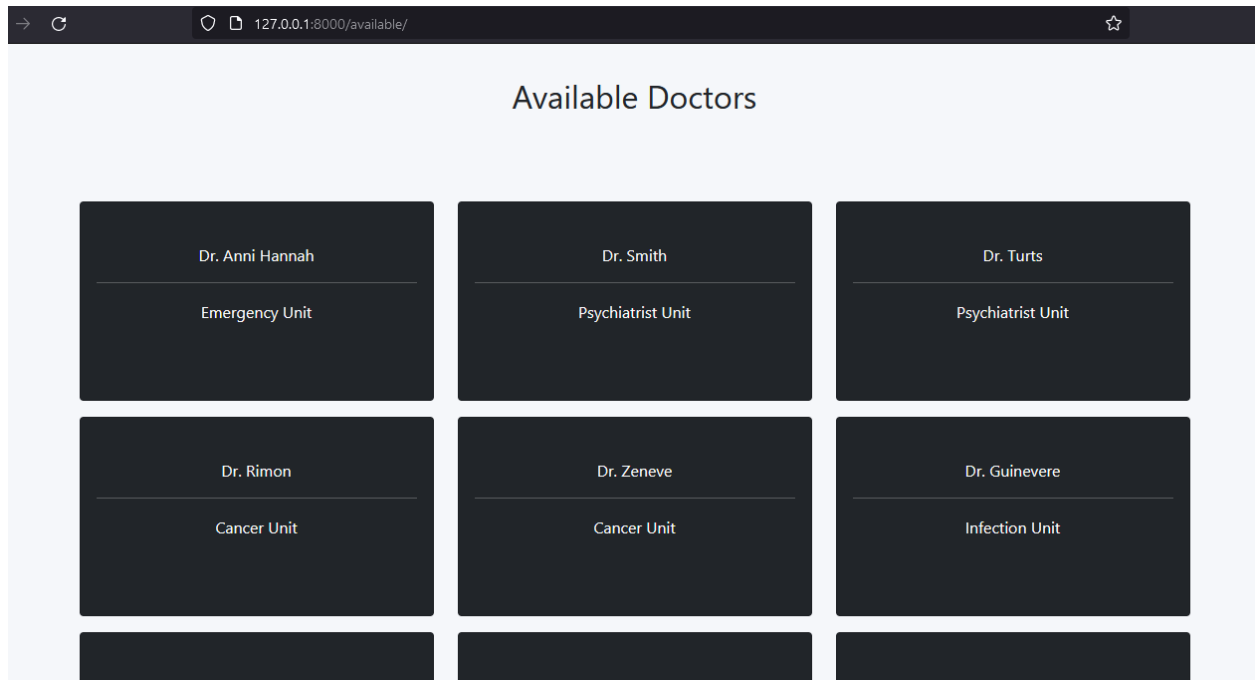



Fig: Doctor available list

Chat Bot





Hi thanks for coming here. Would you like to make an appointment?


Fig: Chatbot web page

127.0.0.1:8000/admin/

Admin Dashboard

User List

Doctor List

Add New Doctor


Log out

Fig: Admin Dashboard

So here are the main features of the web application. One extra feature is that users will get a confirmation email everytime they make an appointment. We initialized the email services in settings.py file and made implementations in the controller. Here is some code snippet of the email service.

```
EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'
EMAIL_HOST = 'smtp.gmail.com'
EMAIL_USE_TLS = True
EMAIL_PORT = 587
EMAIL_HOST_USER = 'email_address'
EMAIL_HOST_PASSWORD = "email password"
```

This is in settings.py file

```
user_mail = user.email
    user_name = user.fullname
    subject = 'Confirmation Email of Appointment Registration'
    message = f'Hi {user_name}, thank you for registering an Appointment
in our Website. Your Appointment details is here:\nDoctor name: {doctor},
Date: {str(datePick)}, Time: {timePick} \nPlease attend at the appointment
in time. Thank you!'
    email_from = settings.EMAIL_HOST_USER
    recipient_list = [user_mail, ]
    send_mail( subject, message, email_from, recipient_list )
```

This is in the controller.