

Chatbot Operation Manual

Tingfeng Lin z5147976

Notes:

- ◆ Please use Python 3.6 or above to test my program.
- ◆ In deployment instructions, I use red bold underline italic style to highlight the operations. You can copy and paste these commands directly following my instructions to run my program.
- ◆ In this project, I am using Mlab (remote MongoDB system) as my database. If you are going to check my service logic and data, I would be glad to help you access to my remote database.
 - Website: <https://mlab.com/>
 - Account: tracy1111
 - Password: ltf9495!
- ◆ Because of the network environment, message response needs to wait 2-3 seconds. When you press *Enter*, please wait 2-3 seconds for response.
- ◆ If you have any problem during deployment, please send me an email z5147976@ad.unsw.edu.au

1. Project design principle

In this project, I am going to use Docker to pack doctor module, timeslot module and chatbot module into three separate services. Each service is isolated and communicate with others.

```
emondesMBP:chatbot-app lemontracy$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS               NAMES
84aad90eef          time               "/bin/sh -c 'python3..."   About an hour ago   Up About an hour   0.0.0.0:3999->5000/tcp   hardcore_diffie
be2acfd4389c        chatbot             "/bin/sh -c 'python3..."   About an hour ago   Up About an hour   0.0.0.0:5002->5000/tcp   adoring_noether
2460c1bbf97         doctor              "/bin/sh -c 'python3..."   About an hour ago   Up About an hour   0.0.0.0:7777->5000/tcp   quizzical_goodall
```

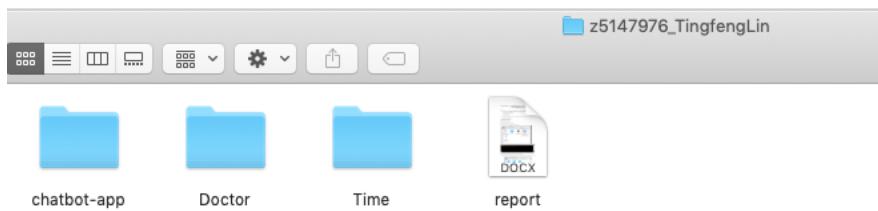
2. Deployment instructions

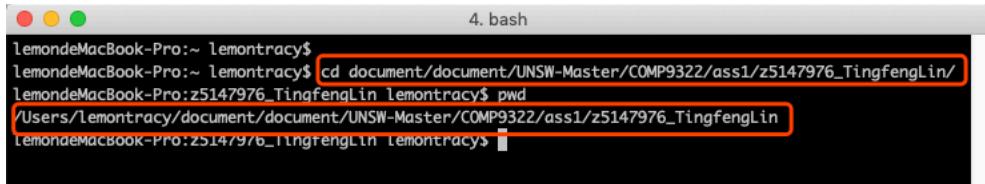
Step 1: Unzip the file in a folder

- Unzip the file in any folder.

Step 2: Using Terminal to access to the folder

- Using Terminal to access the folder you just unzipped

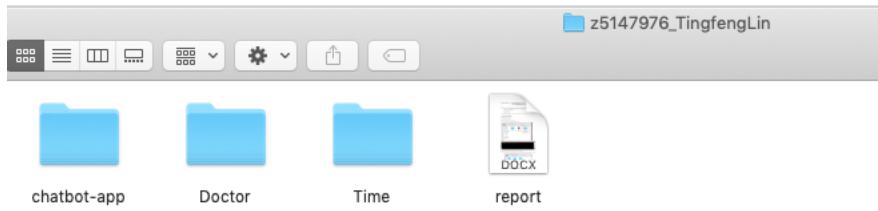




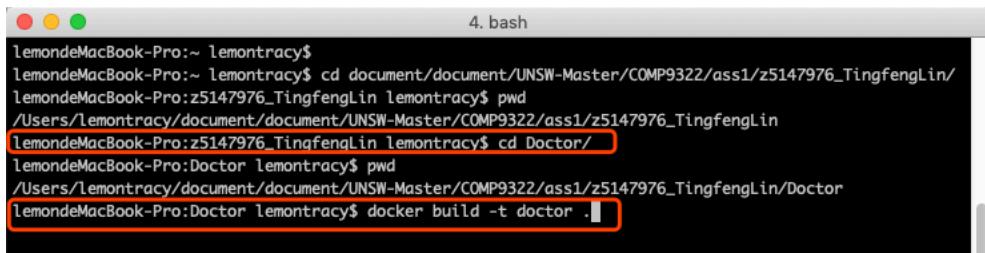
```
lemondeMacBook-Pro:~ lemontracy$ cd document/document/UNSW-Master/COMP9322/ass1/z5147976_TingfengLin/
lemondeMacBook-Pro:z5147976_TingfengLin lemontracy$ pwd
/Users/lemontracy/document/document/UNSW-Master/COMP9322/ass1/z5147976_TingfengLin
lemondeMacBook-Pro:z5147976_TingfengLin lemontracy$
```

Step 3: Accessing the specific folder and execute the relevant commands

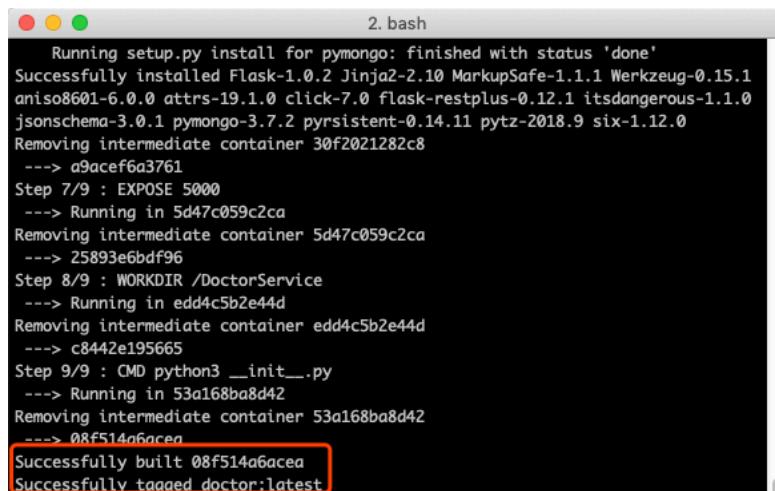
- In this folder, there are three sub-folders. One is Doctor which is the Doctor Service. Another is Time which is obviously the Time Service. The other is chatbot-app which is the interactive part (front end).



- For Doctor Service, cd Doctor folder and type command docker build -t doctor. to create an image (Notes: There is a comma at the end of the command). After building images, it will show ‘Successfully built XXXX’. This means image is created perfectly.



```
lemondeMacBook-Pro:~ lemontracy$ cd document/document/UNSW-Master/COMP9322/ass1/z5147976_TingfengLin/
lemondeMacBook-Pro:z5147976_TingfengLin lemontracy$ pwd
/Users/lemontracy/document/document/UNSW-Master/COMP9322/ass1/z5147976_TingfengLin
lemondeMacBook-Pro:z5147976_TingfengLin lemontracy$ cd Doctor/
lemondeMacBook-Pro:Doctor lemontracy$ pwd
/Users/lemontracy/document/document/UNSW-Master/COMP9322/ass1/z5147976_TingfengLin/Doctor
lemondeMacBook-Pro:Doctor lemontracy$ docker build -t doctor .
```



```
Running setup.py install for pymongo: finished with status 'done'
Successfully installed Flask-1.0.2 Jinja2-2.10 MarkupSafe-1.1.1 Werkzeug-0.15.1
aniso8601-6.0.0 attrs-19.1.0 click-7.0 flask-restplus-0.12.1 itsdangerous-1.1.0
jsonschema-3.0.1 pymongo-3.7.2 pyrsistent-0.14.11 pytz-2018.9 six-1.12.0
Removing intermediate container 30f2021282c8
--> a9acef6a3761
Step 7/9 : EXPOSE 5000
--> Running in 5d47c059c2ca
Removing intermediate container 5d47c059c2ca
--> 25893e6bdf96
Step 8/9 : WORKDIR /DoctorService
--> Running in edd4c5b2e44d
Removing intermediate container edd4c5b2e44d
--> c8442e195665
Step 9/9 : CMD python3 __init__.py
--> Running in 53a168ba8d42
Removing intermediate container 53a168ba8d42
--> 08f514a6acea
Successfully built 08f514a6acea
Successfully tagged doctor:latest
```

- For Time Service, cd .. then cd Time and type command docker build -t time. to create an image (Notes: There is a comma at the end of the command). After building images, it will show ‘Successfully built XXXX’. This means image is created perfectly.

4. bash

```
lemondeMacBook-Pro:~ lemontracy$ 
lemondeMacBook-Pro:~ lemontracy$ cd document/document/UNSW-Master/COMP9322/ass1/z5147976_TingfengLin/
lemondeMacBook-Pro:z5147976_TingfengLin lemontracy$ pwd
/Users/lemontracy/document/document/UNSW-Master/COMP9322/ass1/z5147976_TingfengLin
lemondeMacBook-Pro:z5147976_TingfengLin lemontracy$ cd Doctor/
lemondeMacBook-Pro:Doctor lemontracy$ pwd
/Users/lemontracy/document/document/UNSW-Master/COMP9322/ass1/z5147976_TingfengLin/Doctor
lemondeMacBook-Pro:Doctor lemontracy$ docker build -t doctor .
lemondeMacBook-Pro:Doctor lemontracy$ cd ..
LemondeMacBook-Pro:z5147976_TingfengLin lemontracy$ cd Time/
LemondeMacBook-Pro:Time lemontracy$ docker build -t time .
```

3. bash

```
Removing intermediate container 4bb2e712e1f1
--> 3c1288cad195
Step 8/9 : WORKDIR /TimeslotService
--> Running in ece1c44b68c3
Removing intermediate container ece1c44b68c3
--> 8d384150e3a9
Step 9/9 : CMD python3 __init__.py
--> Running in bdc48d5e73bb
Removing intermediate container bdc48d5e73bb
--> e7b25ca7df9b
Successfully built e7b25ca7df9b
Successfully tagged time:latest
lemondeMBP:Time lemontracy$
```

- For Chatbot Service, [cd ..](#), then [cd chatbot-app](#) and type command [docker build -t chatbot .](#) to create an image (*Notes: There is a comma at the end of the command*). After building images, it will show ‘Successfully built XXXX’. This means image is created perfectly.

3. bash

```
lemondeMBP:Time lemontracy$ cd ..
lemondeMBP:z5147976_TingfengLin lemontracy$ cd chatbot-app/
lemondeMBP:chatbot-app lemontracy$ docker build -t chatbot .
```

3. bash

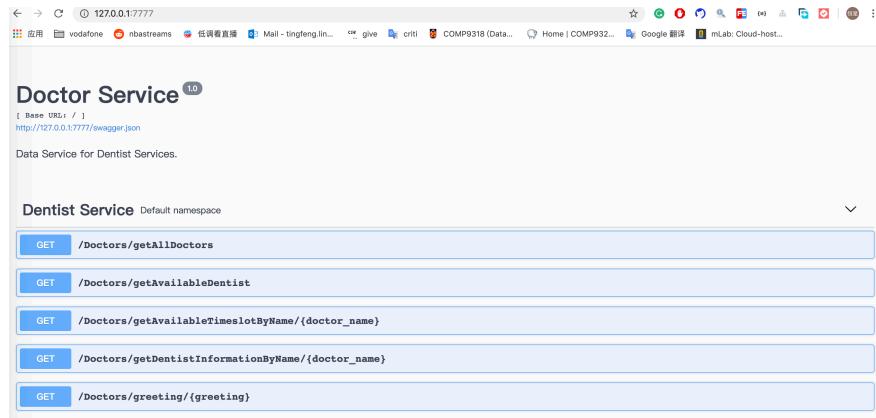
```
--> 5f5797b2060d
Step 8/9 : WORKDIR /comp9322chatbot
--> Running in d84018f6cbf0
Removing intermediate container d84018f6cbf0
--> 58c9be97365a
Step 9/9 : CMD python3 app.py
--> Running in f3143746c564
Removing intermediate container f3143746c564
--> a762103ce961
Successfully built a762103ce961
Successfully tagged chatbot:latest
lemondeMBP:chatbot-app lemontracy$
```

Step 4: Running Doctor image

- Next, type command [docker run -p 7777:5000 -t doctor](#) to run the image. It might take you 1-2 second to start this Doctor Service.

```
lemondeMBP:Doctor lemontracy$ docker run -p 7777:5000 -t doctor
* Serving Flask app "__init__" (lazy loading)
* Environment: production
WARNING: Do not use the development server in a production environment.
Use a production WSGI server instead.
```

- After that, typing <http://127.0.0.1:7777/> in browser, then you will see the swagger interface as shown below

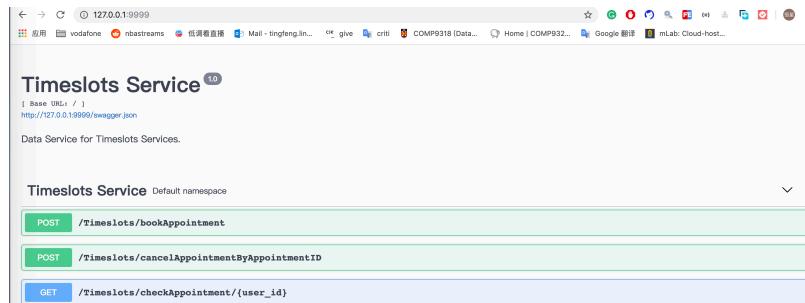


Step 5: Open a new Terminal and run Time images

- Open a new Terminal application then type command `docker run -p 9999:5000 -t time` to run the image. It might take you 1-2 second to start this Time Service.

```
--> 9bd11ac162be
Successfully built 9bd11ac162be
Successfully tagged time:latest
lemondeMacBook-Pro:Time lemontracy$ docker run -p 9999:5000 -t time
* Serving Flask app "__init__" (lazy loading)
* Environment: production
  WARNING: Do not use the development server in a production environment.
  Use a production WSGI server instead.
* Debug mode: on
* Running on http://0.0.0.0:9999/ (Press CTRL+C to quit)
```

- After that, typing <http://127.0.0.1:9999/> in browser, then you will see the swagger interface as shown below

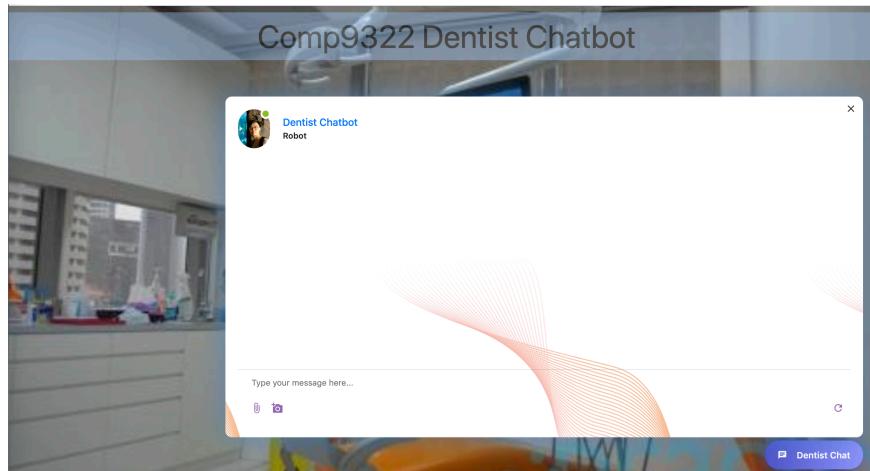


Step 6: Open a new Terminal and run Chatbot images

- Open a new Terminal application then type command `docker run -p 5002:5000 -t chatbot` to run the image. It might take you 1-2 second to start this Time Service.

```
3. docker
lemondeMBP:Doctor lemontracy$ docker run -p 5002:5000 -t chatbot
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: Do not use the development server in a production environment.
  Use a production WSGI server instead.
```

- After that, typing <http://127.0.0.1:5002/> in browser to access to the application.



3. Restful API introduction

In Doctor Service, there are five APIs as shown below.

■ /Doctor/getAllDoctor

Executing this API by clicking button, it will show information of each doctor.

```

200
Response body
{
  "content": [
    {
      "doctor_name": "James",
      "location": "R303",
      "specialization": "Periodontics"
    },
    {
      "doctor_name": "Tony",
      "location": "R301",
      "specialization": "Orthodontics"
    },
    {
      "doctor_name": "Jenny",
      "location": "R302",
      "specialization": "Oral Surgery"
    },
    {
      "doctor_name": "Rick",
      "location": "R304",
      "specialization": "Dental Public Health"
    }
  ]
}
!data": "<p>Doctor list is shown below</p><table class='table table-striped table-bordered table-hover table-condensed'><tr><th>Doctor name</th><th>Office</th><th>Specialization</th></tr><tr><td>James</td><td>R303</td><td>Periodontics</td></tr><tr><td>Tony</td><td>R301</td><td>Orthodontics</td></tr><tr><td>Jenny</td><td>R302</td><td>Oral Surgery</td></tr><tr><td>Rick</td><td>R304</td><td>Dental Public Health</td></tr></table><br><p>You can enter 'doctor name + information' to get doctor's information (eg. James information)</p>"

```

■ /Doctor/getAvailableDentist

Clicking the execution button, it will give you all dentists who still have a timeslot to be booked.

If there has a doctor whose schedule is full. This doctor will not appear.

Code Details

200 Response body

```
{
  "content": [
    {
      "doctor_name": "James",
      "information": "Diagnosing and treating diseases of gum tissue and bones supporting teeth",
      "location": "R303",
      "specialization": "Periodontics"
    },
    {
      "doctor_name": "Zoey",
      "information": "Diagnosing, intercepting and correcting dental and facial irregularities",
      "location": "R301",
      "specialization": "Orthodontics"
    },
    {
      "doctor_name": "Denny",
      "information": "Diagnosing and surgically treating disease and injuries of mouth, oral and maxillofacial region",
      "location": "R302",
      "specialization": "Oral Surgery"
    },
    {
      "doctor_name": "Knick",
      "information": "Preventing and controlling dental disease through organized community efforts",
      "location": "R304",
      "specialization": "Dental Public Health"
    }
  ],
  "data": "<p>Available doctor list is shown below</p><table class='table table-striped table-bordered table-hover table-condensed'><tr><th>Doctor name</th></tr><tr><td>James</td></tr><tr><td>Zoey</td></tr><tr><td>Denny</td></tr><tr><td>Knick</td></tr></table><div style='text-align: right; margin-top: -20px;'><a href='http://0.0.0.0:7777/Doctors/bookAppointment?doctor_id=5564cma4bleille9b4fd88e9fe648756&start_time=10:00&end_time=10:00'>Book Appointment</a></div>"}

```

[Download](#)

■ /Doctors/getAvailableTimeslotByName/{doctor_name}

In this API, it requires a user to type doctor's name. In this situation, you can firstly execute *getAllDoctor* API to get a doctor's name then type it into the input area. I am going to type James who is a doctor in the hospital.

Code Details

200 Response body

```
{
  "content": [
    {
      "doctor_id": "5564cma4bleille9b4fd88e9fe648756",
      "information": "Diagnosing and treating diseases of gum tissue and bones supporting teeth",
      "location": "R303",
      "specialization": "Periodontics",
      "timetable": "9,10,11,12,13,14,15,16"
    }
  ],
  "data": "<p>James</p><p>Information: Diagnosing and treating diseases of gum tissue and bones supporting teeth</p><p>Location: R303</p><p>Specialization: Periodontics</p><table class='table table-striped table-bordered table-hover table-condensed'><tr><th>Start time</th><th>End time</th><th>Book appointment</th></tr><tr><td>10:00</td><td>10:00</td><td><a href='http://0.0.0.0:7777/Doctors/bookAppointment?doctor_id=5564cma4bleille9b4fd88e9fe648756&start_time=10:00&end_time=10:00'>Click to book</a></td></tr><tr><td>11:00</td><td>11:00</td><td><a href='http://0.0.0.0:7777/Doctors/bookAppointment?doctor_id=5564cma4bleille9b4fd88e9fe648756&start_time=11:00&end_time=11:00'>Click to book</a></td></tr><tr><td>12:00</td><td>12:00</td><td><a href='http://0.0.0.0:7777/Doctors/bookAppointment?doctor_id=5564cma4bleille9b4fd88e9fe648756&start_time=12:00&end_time=12:00'>Click to book</a></td></tr><tr><td>13:00</td><td>13:00</td><td><a href='http://0.0.0.0:7777/Doctors/bookAppointment?doctor_id=5564cma4bleille9b4fd88e9fe648756&start_time=13:00&end_time=13:00'>Click to book</a></td></tr><tr><td>14:00</td><td>14:00</td><td><a href='http://0.0.0.0:7777/Doctors/bookAppointment?doctor_id=5564cma4bleille9b4fd88e9fe648756&start_time=14:00&end_time=14:00'>Click to book</a></td></tr><tr><td>15:00</td><td>15:00</td><td><a href='http://0.0.0.0:7777/Doctors/bookAppointment?doctor_id=5564cma4bleille9b4fd88e9fe648756&start_time=15:00&end_time=15:00'>Click to book</a></td></tr><tr><td>16:00</td><td>16:00</td><td><a href='http://0.0.0.0:7777/Doctors/bookAppointment?doctor_id=5564cma4bleille9b4fd88e9fe648756&start_time=16:00&end_time=16:00'>Click to book</a></td></tr><tr><td>17:00</td><td>17:00</td><td><a href='http://0.0.0.0:7777/Doctors/bookAppointment?doctor_id=5564cma4bleille9b4fd88e9fe648756&start_time=17:00&end_time=17:00'>Click to book</a></td></tr><tr><td colspan='3'>Please select one following timeslot and click the button to book your appointment</td></tr></table><div style='text-align: right; margin-top: -20px;'><a href='http://0.0.0.0:7777/Doctors/bookAppointment?doctor_id=5564cma4bleille9b4fd88e9fe648756'>Book Appointment</a></div>"}

```

[Download](#)

■ /Doctors/getDentistInformationByName/{doctor_name}

This API also needs user to input a doctor's name. I type James into the input area.

Request URL
<http://0.0.0.0:7777/Doctors/getDentistInformationByName/James>

Server response

Code Details

200 Response body

```
{
  "content": [
    {
      "doctor_id": "5564cma4bleille9b4fd88e9fe648756",
      "doctor_name": "James",
      "gender": "Male",
      "information": "Diagnosing and treating diseases of gum tissue and bones supporting teeth",
      "location": "R303",
      "specialization": "Periodontics",
      "timetable": "9,10,11,12,13,14,15,16"
    }
  ],
  "data": "<p>This is Doctor James introduction:</p><p>Diagnosing and treating diseases of gum tissue and bones supporting teeth</p>"}

```

[Download](#)

■ /Doctors/greeting/{greeting}

I put a common greeting function in Doctor Service. Typing the common greeting in the input area then you will get feedback.

Request URL
<http://0.0.0.0:7777/Doctors/greeting/hi>

Server response

Code	Details
200	Response body <pre>{ "data": "hey" }</pre> <p>Download</p> Response headers <pre>access-control-allow-origin: * content-length: 15 content-type: application/json date: Tue, 02 Apr 2019 13:24:52 GMT server: Werkzeug/0.15.1 Python/3.6.8</pre>

Responses

In Time Service, there are three APIs as shown below.

Timeslots Service Default namespace

POST	/Timeslots/bookAppointment
POST	/Timeslots/cancelAppointmentByAppointmentID
GET	/Timeslots/checkAppointment/{user_id}

■ /Timeslots/bookAppointment

There are several values should be passed into this POST request.

200 Response body

```
{
  "data": "Appointment has been booked successfully<br>Your appointment id is 606f677e554b11e981390242ac110003<br>Please type these questions to continue your service<ul><li>show dentist list</li><li>list available dentists</li><li>check my appointments</li><li>list available timeslot of + (doctor name)</li><li>(doctor name) + information</li></ul>"
}
```

[Download](#)

Response headers

```
content-length: 33
content-type: application/json
date: Tue, 02 Apr 2019 13:29:44 GMT
server: Werkzeug/0.15.1 Python/3.6.8
```

Responses

■ /Timeslots/cancelAppointmentByAppointmentID

In this API, the appointment ID should be passed into the API.

Code Details

200	Response body
	<pre>{ "data": "Appointment has been canceled successfully.
Please type these questions to continue your serviceshow dentist listlist available dentists check my appointmentslist available timeslot of + (doctor name)(doctor name) + information"</pre> <p>Download</p>

Response headers

```
content-length: 301
content-type: application/json
date: Tue, 02 Apr 2019 13:34:59 GMT
server: Werkzeug/0.15.1 Python/3.6.8
```

Responses

■ /Timeslots/checkAppointment/{user_id}

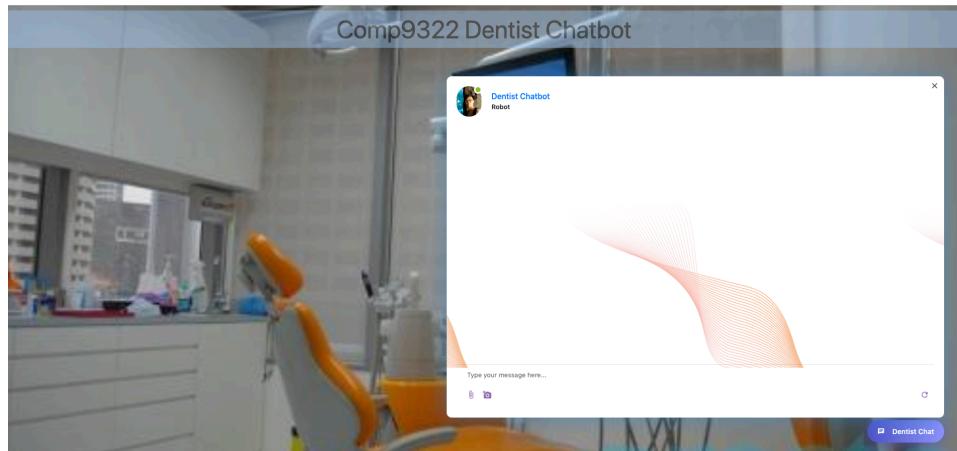
This is a GET API, a user has to enter their user id. In order to get the specific user id, you have to log in to my database to get the ID. There is no API can get the user ID.

200 Response body

```
{
  "content": [
    {
      "appointment_id": "313b4fcfa554b11e981390242ac110003",
      "appointment_time": "9",
      "doctor_id": "string",
      "doctor_name": "string",
      "location": "string"
    }
  ],
  "data": "<p>Appointment list is shown below</p><table class='table table-striped table-bordered table-hover table-condensed'><tr><th>Doctor name</th><th>Office</th><th>Specialisation</th><th>Time</th><th>Operation</th></tr><tr><td>string</td><td>string</td><td>string</td><td>9</td><td>313b4fcfa554b11e981390242ac110003</td><td><button class='btn btn-sm-primary' onclick='(313b4fcfa554b11e981390242ac110003 , 9 , string).click()'; click to cancel</button></td></tr></table>"
```

[Download](#)

4. Chatbot operation introduction

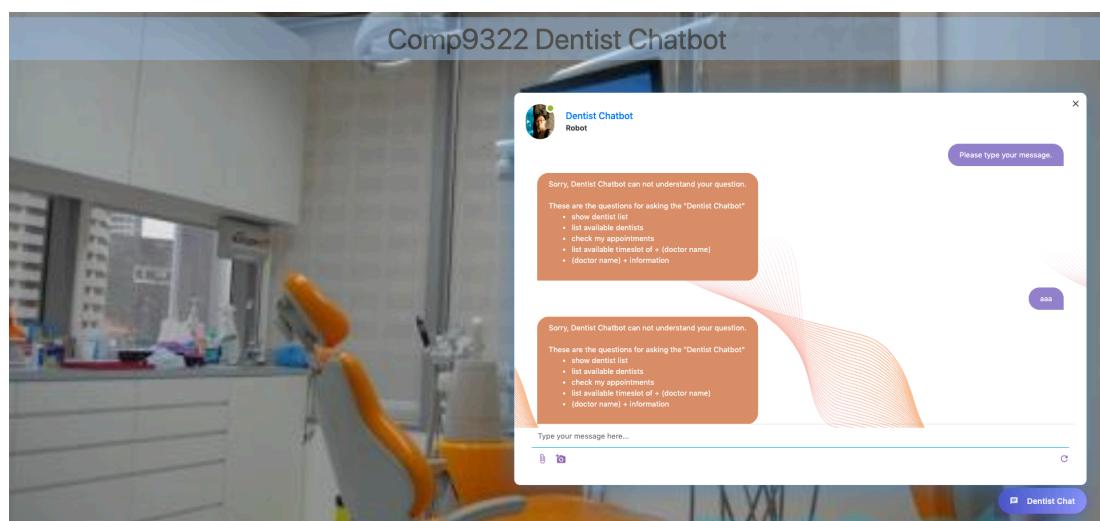


This is the interface of chatbot. The user needs to type question in the input area and press *Enter* to send the message. Due to the network environment, it might take you 2-3 second to send the message and get the response.

The chatbot can identify some specific or related sentences such as

- ◆ show dentist list
- ◆ list available dentists
- ◆ check my appointments
- ◆ list available timeslot of + (doctor name)
- ◆ (doctor name) + information

When you type these questions or other similar questions, chatbot might give you related response. However, when a user types a question that chatbot cannot identify or press enter without any question, it will response some hints and require the user to type a new question again.



5. Bonus part

In assignment requirement, it mention that display the timeslot graphically. I have completed this task. These are the screenshots.

