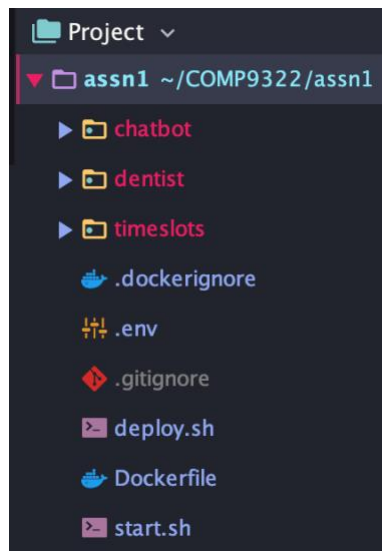


# COMP9322 Assignment1 Document

Author: Tianpeng Chen z5176343 (tianpeng.chen@student.unsw.edu.au)

## Project structure

### Backend



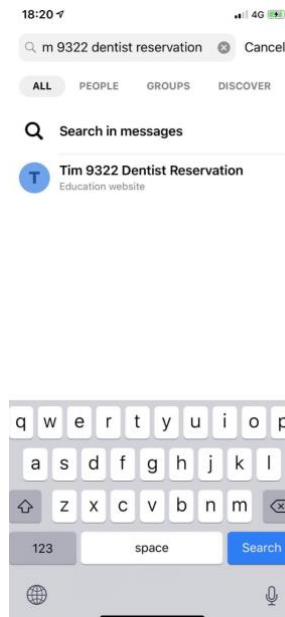
- chatbot  
contains the source code and docker file for the Chatfuel chatbot webhook server
- dentist  
contains the source code and docker file for the **dentist** microservice
- timeslots  
contains the source code and docker file for the **timeslot** microservice

Under each sub-folder, I also wrote a deploy script to help me deploying my app.

In addition, as I have all my services deployed on *Heroku*, a docker image containers *Heroku-cli* is also provided. You can run *start.sh* to build and run the image, which will automatically deploy all the three services to *Heroku*.

## Frontend

I used Chatfuel to build a chatbot which is accessible on Facebook Messenger. You can search “Tim 9322 Dentist Reservation” to start chatting with the bot.



## API Documentation

API documentation and tests are available with *swagger*. You can test these APIs through:

- timeslot microservice

<https://tim-9322-timeslot.herokuapp.com/v1/help>

- dentist

<https://tim-9322-dentist.herokuapp.com/v1/help>

## Deployment

All the services are deployed to *Heroku*. A start script “*start.sh*” is provided to build and run a *Heroku-cli* enabled docker image (which also contains an API key to access my Heroku

apps), to deploy all the three apps to *Heroku*. If you want to deploy the services, just run “\$ ./start.sh” in the project root.

## Bonus

As considering how to identify users, I found that Chatfuel provides a “*chatfuel user id*” which can help me identify each user. After that, I added a jwt-encryption to my webhook server, which encrypts the current user\_id into a JWT, and send it to the **timeslot** microservice.

These two endpoints requires a JWT authorization in **timeslot**:

/timeslots/{id}/reserve

PUT /timeslots/{id}/reserve To make an appointment	
Parameters	
Name	Description
<b>id</b> <span>✚ required</span>	ID of the timeslot
integer (path)	
<b>doc_id</b> <span>✚ required</span>	ID of the dentist
integer (query)	
<b>Authorization</b> <span>✚ required</span>	JWT of the patient
string (header)	

After a successful API call, the user JWT will be stored in the database.

```
rs-d019916:PRIMARY> db.collection.find({"_id":90);
```

```
{ "_id": 90, "doctor_id": 1, "date": "Friday", "time": "17", "status": "reserved", "comments": "", "reserved_by": "eyJ0eXA0IjkiKVJvYXVzLnR5cyJlc2VyZXZlkiJoImIjN2c3MzE3MTYxNDQwNSJ9.BtLZWCR_6cvrhXRyUeH7_Pht2iaUyzMcGst5uk" }
```

`/timeslots/{id}/cancel`

PUT /timeslots/{id}/cancel To cancel an appointment	
Parameters	
Name	Description
<b>id</b> * required integer (path)	ID of the timeslot
<b>doc_id</b> * required integer (query)	ID of the dentist
<b>Authorization</b> * required string (header)	JWT of the patient

If someone called this API to cancel a reservation, it will require a user JWT and check against the one stored in the database, to ensure only the creator of the booking has the right to cancel it.

Thank you!