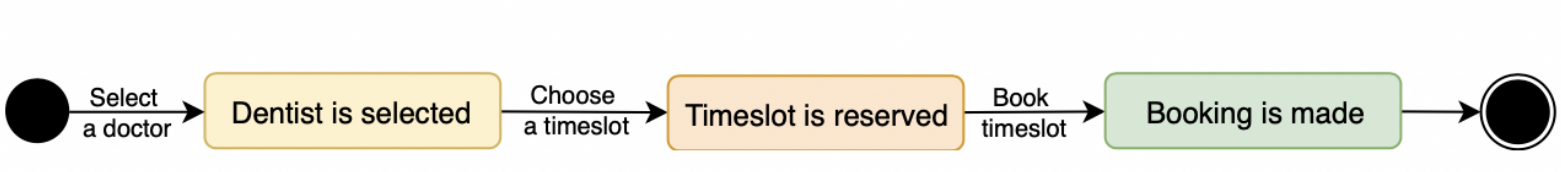
Assignment 2: Building a dental clinic system using a chatbot and dockerized services

Business Scenario

In this assignment, you implement a Chatbot that rely on a set of REST API based services to perform booking in a dental clinic.

First, let us imagine that the appointment booking operation in the dental clinic is based on the following workflow (modelled as state machines).



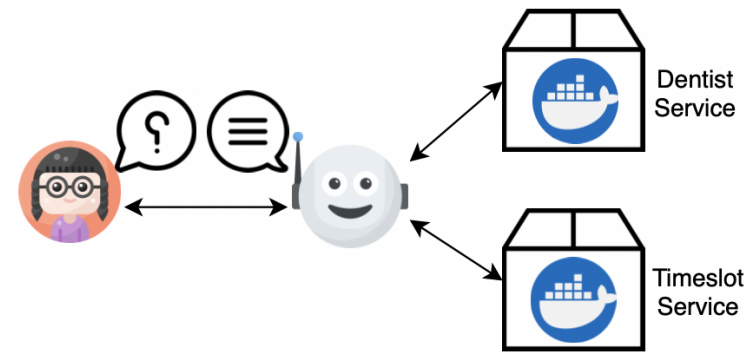
The scenario here is the patient contact the Chatbot to request for a booking. If the patient needs to specify the doctor or ask for the list of doctors available. After the selection of the dentist the patient needs to specify the preferred timeslot the chatbot checks if the timeslot is available and if not, it provides list of available timeslots. The patient will select one of the available timeslots and the booking is made.

The assumption is that we have at least three dentists. They are available from 9AM-5PM every day. Each dentist by nature has a timetable.

The specification are *deliberately left open* to allow students to take follow their own strategy to achieve the required outcome. You are allowed to use any of three approaches (Rule, ML, Flow) to build your chatbot.

API Implementation (5 Marks)

System overview



Icons made by www.flaticon.com

The diagram above shows the high level components of the system you are building. The *Dentist* and *Timeslot* services are to be deployed in a docker container and the chatbot should communicate with the services through REST API calls.

To summarise the main functionalities:

- Get available dentists
- Get dentist information
- Get available timeslots for each dentist
- Reserve timeslot
- Cancelling appointment

The following describes the API you should implement.

The resources and their URI patterns:

There are two types of resources to be managed: Dentist and Timeslot. It is up to you to design these resources and their URI patterns. Indeed, this will be your first task. Go through the scenario and make sure that you have every piece of data needed to service the scenario is covered.

Some suggested information for each resource:

- Dentist: name, location, specialization (e.g., Paediatric Dentistry, Orthodontics, Oral Surgery…etc.)
- Timeslots: 1 hour timeslots from 9:00 AM to 5:00 PM, status flag (reserved, available)

Managing resources:

It is important that all HTTP methods implemented by you satisfy the safety and idempotency properties of REST operations. It is also important that the design of the API allow stateless communications.

It is part of your design task to choose an appropriate HTTP method to implement the following management operations.

Persisting resources:

Ideally, you would like to store these data in a persisting data source. Since our dental clinic system have separate services (Dentist, Timeslot) for different functionalities, you need to make sure these services are isolated (all the communications/data exchanges between them are via REST APIs). The form and type of data is up to the students to decide.

RESTful APIs are stateless:

Your implementation of APIs should show that your APIs are stateless and all communications are self-contained with the messages.

Chatbot Interaction (13 Marks)

There is no special requirement which chatbot or messaging platform to use but here are some general guidelines you need to follow.

- The bot should be able to respond to basic *greetings.*
- The bot asks the client for the preferred doctor and provide information about the doctor.
- The bot can list all the available doctors in the clinic and the client can choose.
- The bot can check if the selected timeslot is already reserved and suggest another timeslot.
- The bot can provide a list of available timeslots for the selected doctor.
- The bot can confirm the booking and summarize at the end.
- The bot can cancel the booking if the client requested it and ask for confirmation.
- Interaction between user and chatbot is happening in an interface. Thus, a front-end (even **very basic** one) is required. From end-users perspective, swagger and/or terminal interfaces are not something that they would expect to use. Please note that you are **NOT allowed** to use any messaging app (Facebook Messenger, Telegram, WeChat, Slack, etc) as front-end, you need to make your own front-end even **very basic** one (e.g. simple html page).