**COSC2674/2755 (PIoT) Semester 2, 2018**
**Assignment 2 Specification (*Building a Smart Office*)**

| | |
|---|---|
| Marks allocated: | 100 (worth 25% of the total score) |
| Deadline: | 11:59 pm Melbourne time (Sunday 07th October 2018) |
| Submit via: | Canvas |
| Work mode: | In in a group of 2-4 (*individual submissions discouraged*); Register your groups via https://goo.gl/VzUhF6 |
| Submission format: | **.zip** (*No other formats will be accepted*) |
| Face to Face demo: | Week 12 (Monday 08th October– Friday 12th October 2018) |

# READ THIS FIRST

The real-life projects that you will face in Industry never come with crystal clear, direct list of instructions in a linear manner. In fact, the reality is far from that, the project requirements often come in bits and pieces from often a confused client who thinks that they know everything.
It is the job of requirement engineers to elicit the requirements. Business Analysts then spend good amount of time clarifying these requirements and creating more sensible, doable and negotiable list of deliverables.

When you read the specifications for this assignment, you will realise that some of the ones may have multiple ways of implementing them (*just like in real life software development*). So instead of blaming it on the specifications, clarify the requirement(s) either

- via emailing to the lecturer or the head tutor (*not your tutors*) or,
- posting in discussion board for assignment 2

Do not start this assignment late, you have *five weeks* to complete it which is more than enough time to do well and make sure that you use this time judiciously. Starting work at the last minute will only lead to poor outcome(s).

There are certain specifications which will push you out of the comfort zone. This has been done on purpose. There are certain parts of the assignment where you will need to do self-research as you will not find answers in lectures, tute/labs.

If you do a good job of this assignment, you can choose to add it as a part of portfolio for future employers. You are being prepared for potential employability prospects.

# 1 Scenario

Your team has been contacted by a medical office in Melbourne to advise them on an office automation project to automate the medical appointments process known as **MAPS** (*medical appointment system*), usually handled by medical receptionists.

You have been tasked create a website for three types of users: patients, doctors and medical clerks and a series of inter-connected IoT devices in the doctor's office and the front desk. (Please make use of Flask, Jinja2 templating and Bootstrap to build your web interface).

For this assignment, you will be making extensive use of the Google Calendar API, and Google Assistant SDK for devices to work with your Raspberry Pi. You can watch this YouTube video about the SDK. These links are also outlined as below, just in case if the actual links do not work in this PDF document:

Google Calendar API: https://developers.google.com/calendar/v3/reference/
Google Assistant SDK for devices: https://developers.google.com/assistant/sdk/
YouTube about the SDK: https://www.youtube.com/watch?v=FBXRwu6hgy8

All of data is now saved in cloud database(s).

# 2 Important

You must adhere to the following requirements:

a. Only Raspberry Pi model 3 should be used

b. You must use Python 3.5 or >3.5 to complete the tasks. Older versions must not be used.

c. You must use a version control system of some sorts such as *GitHub, Bitbucket,* etc. A private repository is to be used ONLY.

d. You must stick to the standard *style guide for your Python code:* (https://www.python.org/dev/peps/pep-0008/)

e. You must attend a **25 minutes** demo session to get the assignment 1 marked during week 6 (Oct 08- 12, 2018). A schedule and a booking document will be published soon. You must submit the assignment prior to demo. No submission → No demo → No marks.

# 3 Detail(s) of MAPS

## Scheduling an appointment

**For patients:** Your cloud database should keep track of the patient list and allow new patients to register on your website (*Refer to any online medical appointment systems for inspirations*). After registering, a patient is allowed to book an appointment with any available doctor. Each patient will have a list of medical records.

**For doctors:** Your website should allow the doctor to specify their availability on a weekly basis on their personal calendar. You will also save these appointments separately on your cloud database for aggregation and reporting purposes.

**For medical clerks:** Your website should show a list of doctors' appointments that had already been made for that week. The medical clerk may add or remove any appointments.

---

## On the day of appointment (*this feature is only needed for part D – see page 5, specification xi*)

Your client will make use of an IoT device at the reception to check in. This device will keep track of the patient arrivals. It will also interact with another IoT device in each of the doctor's office. **Two Raspberry Pis will be needed for this assignment**.

So, in a nutshell, one of the devices (known as *Reception Pi*) will monitor the arrival of clients (via facial recognition or entering the details) and each of the remaining device (known as *Advisor Pi*) will listen to the doctor's voice commands for instructions and prompt them to attend to new patient at appropriate time.

We will be using Google Cloud Platform for this assignment to integrate Google Assistant to your Raspberry Pi.

Whenever a doctor asks to see the next patient, their *Advisor Pi* interact with *Reception Pi* to check:

**If patient has arrived (at scheduled time)**:

+ Pull up the medical record for the patient
+ And invite them to the doctor's office

**Otherwise**:

+ Inform the doctor that they are free until the next appointment

You can decide how you want the medical records to be shown to the doctor. Your marks will be based on how well you display the medical records to the doctor.

At the end of every appointment, the doctor will dictate the medical note to *Advisor Pi*, which will then save the information as a medical record on the cloud.

# 4 Tasks

**Note**: This time the tasks are divided into *four* parts: A, B, C and D:

You MUST
- complete all the part A tasks before you proceed to part B →
- all of part B tasks before you proceed to part C →
- all the part C tasks before you proceed on to part D.

**Part A - Web Interface using Flask [30 marks]**

Your team will need to

i. (20 marks) create at least three separate web pages for doctors, patients and medical clerks. Each of the user will have a different URL to access their page. You need not implement complete login features for these users.

A patient can make and delete an appointment for a doctor using MAPS.

Doctor will use MAPS to pull up patient history, add patient notes and diagnoses.

Medical clerk can add, delete appointments on behalf of patients. They are also responsible for keeping an eye on doctor appointments.

The pages must be professionally styled using Bootstrap. You will be marked on the design, professional look & appeal and user friendly of these pages.

Please note that no actual implementation of booking is required for this part. You are setting up the web pages for the further parts.

ii. (5 marks) Your team are expected to set up private github/bitbucket repositories and work on separate branches individually. Everyone will have to explain what they have done in each branch during the demonstration.

iii. (5 marks) Complete documentation of the project using PyDoc and Sphinx. You can read an excellent tutorial at
https://projects.raspberrypi.org/en/projects/documenting-your-code/

**Part B – APIs (Patient and Doctor) [30 marks]**

You will now implement

iv. (5 marks) The patient registration and booking feature. All the data must be saved to Cloud.

v. (5 marks) all of the Doctor's page(s): pulling up patient data, making notes and diagnoses during the appointment. All the data must be saved to Cloud.

vi. (10 marks) Create separate API(s) to interact with the cloud. The patient and doctor pages should not directly talk to the Cloud.

vii. (5 marks) Since the interface is web, all the user inputs must be validated

viii. (5 marks) Complete documentation using PyDoc and Sphinx

*It is your responsibilities to make sure that you do not exceed the free tier limit on the Google Cloud Platform.

**Part C – API (Medical Clerk) [10 marks]**

You will now implement

ix. (4 marks) All of the medical clerk features: add/delete appointment and list of the doctor appointments. Once again use an API to talk to cloud.

x. (3 marks) Add an extra feature where the clerk can generate a visual representation of the doctors appointment(s) (*number of patients seen*) week wise.

xi. (3 marks) Complete documentation using PyDoc and Sphinx

**Part D - Advanced Implementation [30 marks]**

xii. (10 marks) Your team will be tasked to create custom actions and traits on Google Assistant SDK to interact with the doctors and patients. This involves

a. implementing the interaction between Reception and Advisor Rapsberry Pi s
b. Doctor dictating medical notes to Advisor Pi

xiii. (15 marks) Your team will choose ONE advanced implementation. Some of the suggestions are (**you may come up with your own implementation**):

facial recognition, deep learning for scheduling, Integration with Arduino, etc.

YOU WILL NEED TO GET THIS approved by the lecturer or the head tutor (*not your tutors*) to receive marks in this section.

xiv. (5 marks) Complete documentation using PyDoc and Sphinx

# 5 Late submission and Extension

a. A penalty of 10% per day of the total marks will apply for each day late, including both weekend and weekdays.

b. After five days, you will receive a zero for the whole assignment.

c. Extension requests should only be emailed to the lecturer (shekhar.kalra@rmit.edu.au)

d. Look out for the Assignment feedback sessions – these will be held during week(s) 9-11. In the meantime, please bring questions/doubts to weekly consultation sessions held by the head tutor on Fridays and/or post in the discussion board Assignment 2 folder.

# 6 Plagiarism

All assignments will be checked with plagiarism-detection software; any student found to have plagiarised would be subject to disciplinary action. Plagiarism includes

- submitting work that is not your own or submitting text that is not your own
- allowing others to copy your work via email, printouts, social media etc.
- posting assignment questions (in full or partial) on external technical forums
- copying work from/of previous/current semester students
- sending or passing your work to your friends
- posting assignment questions on technical forums to get them solved

A disciplinary action can lead to

- a meeting with the disciplinary committee
- a score of zero for the assignment
- a permanent record of copying in your personal university records and/or
- expulsion from the university, in some severe cases

All plagiarism will be penalised. There are no exceptions and no excuses. You have been warned.