Sports Injury Tasmania

SIT Admin Application

Design and Specifications

Executive Summary

The following application design specifications are for the new Sports Injury Tasmania (SIT) administrative application, referred to as SIT Admin.

SIT currently operate with an appointment application which is utilised as is a tool for administrative staff to create patient appointments' cards with appointment details such as date, time and Doctor. The SIT Appointment application uses data from two files patients.json and staff.json.

However, the SIT Appointment application lacks the ability for staff to enter new patients or to delete a current patient from the patient list. This has highlighted the need for a new application named SIT Admin to be created.

The following specifications are for the SIT Admin application which will allow the administrative staff of SIT to enter and or delete patients from the patients.json file.

The specification will also allow for the alphabetically sorted printout of the patient names.

Table of Contents

Table of Figures

Design and Specifications

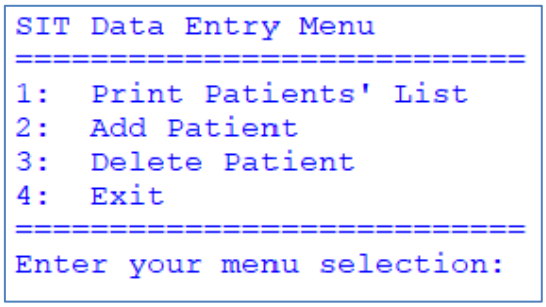
Important:

Ensure both User Interface and Code Architecture sections are read before design and implementation of algorithm.

User Interface

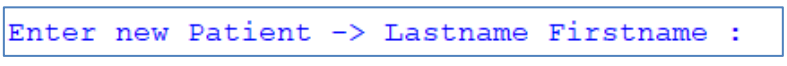
- Admin staff must be able to enter or delete patients and close the application using input prompts:
    1: Print out of alphabetically sorted Patients' List.
    2: Entry for new Patient.
    3: Entry for delete Patient.
    4: Close Menu
- The application must have a menu allowing for the selection of each entry as outlined in points 1, 2, 3 and 4 above. The menu must match Figure 1.
- The menu must re-appear after each selection is made and completed except for selection 4: Exit.
- The exit option must close the menu and may also close the application.

Figure 1 - Menu Layout



- For data entry in selection 2: Add Patient, the input message must match
- Figure 2 below. Reminding the user to enter patient names with lastname first and firstname second, separated with a space.

Figure 2 - Example of Data Entry



- Each data entry in menu items 2 and 3 must update to the patient list.
- Printout of Patients' list must have the layout as displayed in Figure 3 below. All names to be entered as `lastname firstname`.
- Patients' list must be printed out in alphabetical order.

Figure 3 - Printout format for Patients' and Staff lists

```
SIT current patients:

1 Brokentoe Susan
2 Kaputknee Matt
3 Neckache Annette
4 Soreback Leigh
5 Sorefoot Jo
6 Tornligament Alex

Total number of registered patients is 6
```

- The application must be designed with Python3 for the shell environment.
- Closing the application must save the updated patient list to the patient.json file.

Code Architecture

Below are the requirements for code expression and code protocols to be used in the application.

Programmer's Identification

- The author must identify their name as the author and version and date of submission at the top of the code using the variables:

  `__author__`

  `__version__`

  `__date__`

See Figure 4 below:

Figure 4 - File Structure of Application's Code

```
'''Docstring
Documentation'''
__author__ = '  '
__version__ = '  '
__date__ = '  '

import Python3 Modules

def all_functions(might have parameters):
    '''Function documentation here'''
    code here
    # With hash possible comments

Might have more functions here

Main code here
# With possible hash comments

```

Documentation and Layout

- The author must use documentation in their code with the use of docstrings and / or hash comments. Docstrings are only to be used at the start of the program and at the start of functions. Hash comments are throughout the code. See Figure 4 above.
- All importing of any library modules must be done after the `__date__` declaration.

- All functions must be declared after the `import` statement.
- The main program code must be the last section of the code.

Data Format

Patients' list must be in `json` format and imported and manipulated in the code as a `list` variable.

Required Code Expressions and Constructs

The application's code must have the following constructs and expressions:

- At least one library import.
- At least one user designed function that will be called on in the main program.
- Read and write to `json` files.
- One or more `Loop` iterations.

Data and Code Verifications

The completed application must be presented:

- Free of errors and bugs.
- With data integrity, i.e., data .json files are properly updated by the application.