Configuration Manual

MSc Research Project
MSc in Data Analytics

Jack Dunne
Student ID: x21133760

School of Computing
National College of Ireland

Supervisor: Anh Duong Trinh

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: ……. …Jack Dunne……………………………………………………………………………

Student ID: ……x21133760………………………………………………………………………………..……

Programme:…… MSc in Data Analytics…………………… **Year:** …………2023………..

Module: …………MSc Research Project……………………………………………………..………

Lecturer: ………Anh Duong Trinh…………………………………………………………….………

Submission Due Date: ……………14/08/2023……………………………………………………………..………

Project Title: Optimising Scheduling for Computed Tomography Imaging in a Healthcare Setting Using Discrete Event Simulation

Word Count: ……………8558………………… **Page Count:** ………………19……….…….………

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Jack Dunne

Date: …14/08/2023…………………………………………………………………………………

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Jack Dunne
Student ID: x21133760

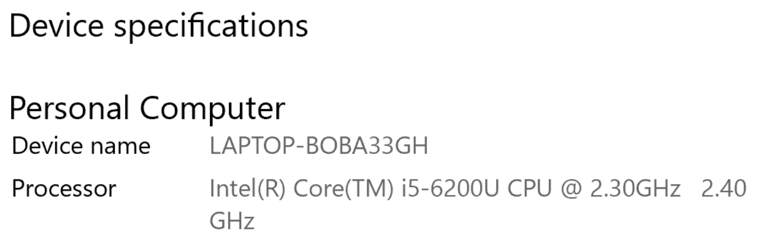
1 Introduction

This configuration manual contains the necessary details in order to run/execute the project ‘Optimising Scheduling for Computed Tomography Imaging in a Healthcare Setting Using Discrete Event Simulation’.

It includes system requirements such as software and hardware specifications, library versions and it explains the necessary code.

2 System Configuration

2.1 Hardware



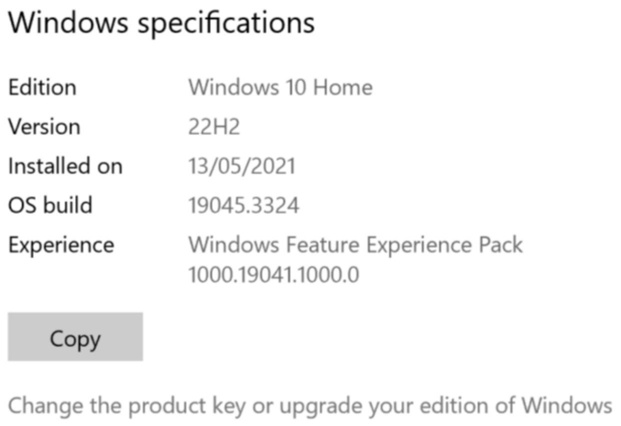
Device specifications

Personal Computer

Device name LAPTOP-BOBA33GH

Processor Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz 2.40 GHz

Figure 1. Device specifications



Windows specifications

Edition Windows 10 Home

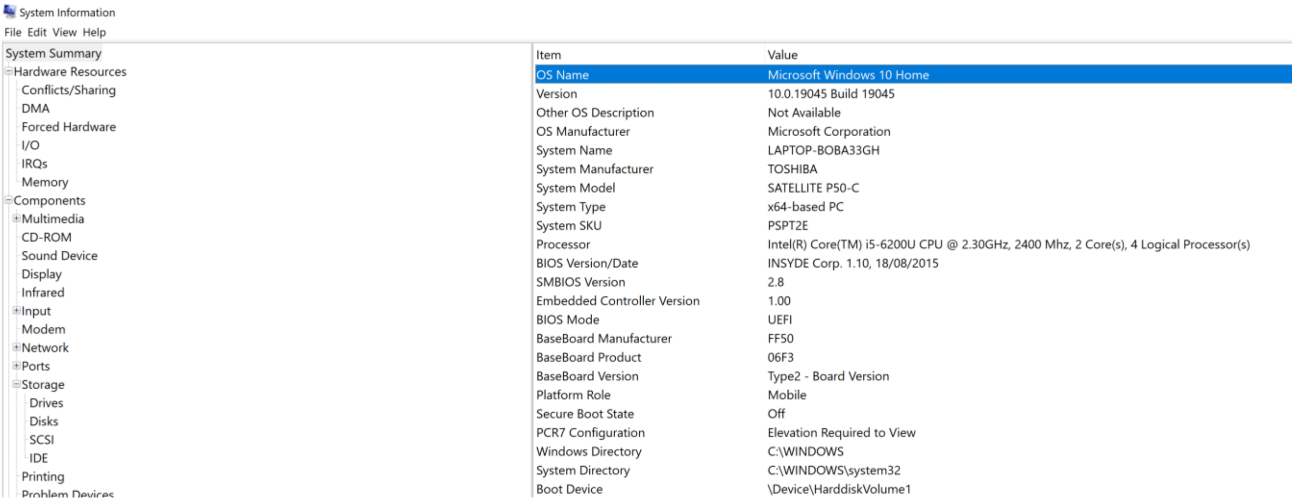
Version 22H2

Installed on 13/05/2021

OS build 19045.3324

Experience Windows Feature Experience Pack 1000.19041.1000.0

Figure 2. Device specifications

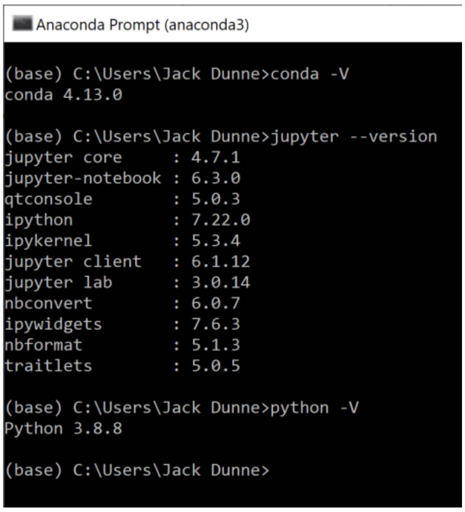


Item	Value
OS Name	Microsoft Windows 10 Home
Version	10.0.19045 Build 19045
Other OS Description	Not Available
OS Manufacturer	Microsoft Corporation
System Name	LAPTOP-BOBA33GH
System Manufacturer	TOSHIBA
System Model	SATELLITE P50-C
System Type	x64-based PC
System SKU	PSPT2E
Processor	Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz, 2400 Mhz, 2 Core(s), 4 Logical Processor(s)
BIOS Version/Date	INSYDE Corp. 1.10, 18/08/2015
SMBIOS Version	2.8
Embedded Controller Version	1.00
BIOS Mode	UEFI
BaseBoard Manufacturer	FF50
BaseBoard Product	06F3
BaseBoard Version	Type2 - Board Version
Platform Role	Mobile
Secure Boot State	Off
PCR7 Configuration	Elevation Required to View
Windows Directory	C:\WINDOWS
System Directory	C:\WINDOWS\system32
Boot Device	\Device\HarddiskVolume1

Figure 3. Device specifications

2.2 Software

To execute or run the code implemented to complete the project, the following applications used are :



```
(base) C:\Users\Jack Dunne>conda -V
conda 4.13.0

(base) C:\Users\Jack Dunne>jupyter --version
jupyter core     : 4.7.1
jupyter-notebook : 6.3.0
qtconsole        : 5.0.3
ipython          : 7.22.0
ipykernel        : 5.3.4
jupyter client   : 6.1.12
jupyter lab      : 3.0.14
nbconvert        : 6.0.7
ipywidgets       : 7.6.3
nbformat         : 5.1.3
traitlets        : 5.0.5

(base) C:\Users\Jack Dunne>python -V
Python 3.8.8

(base) C:\Users\Jack Dunne>
```

Figure 4. Software versions

The following Python libraries are required to be installed:

- simpy - 4.0.1
- pandas - 1.3.4
- numpy - 1.21.4
- matplotlib - 3.5.0
- seaborn - 0.12

3 Coding of the files

There is only 1 python file required. It is called ‘radiology_dept.ipynb’ and is in the form of a Jupyter notebook file.

4 Execution of the code

1. Clearing the kernel and running the code
2. Must change the paths to the files to whatever location you are creating them/saving them on your computer.

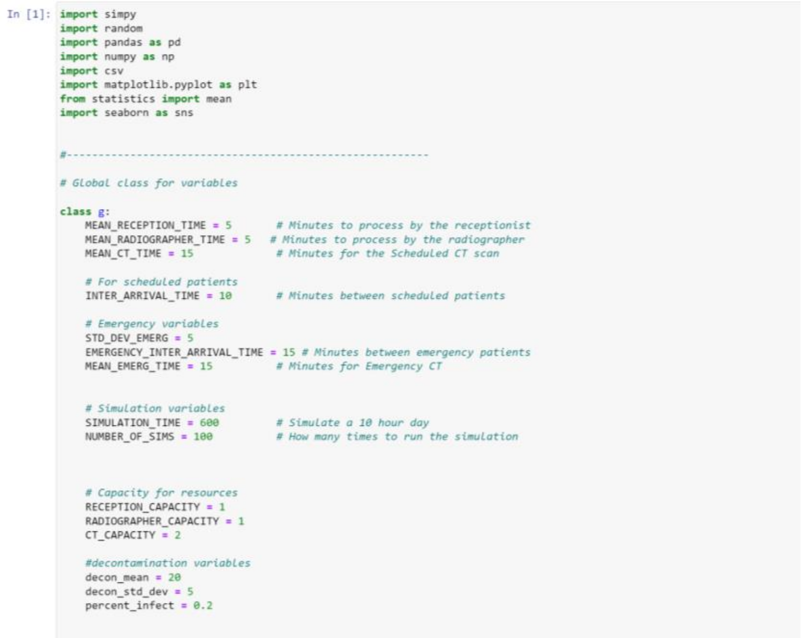
5 Data sets used

3 .csv files created and used during the programming

- CT_simulation_run.csv
- trial_CT_scan_sim.csv
- CT_single_run.csv

6 Code snippets

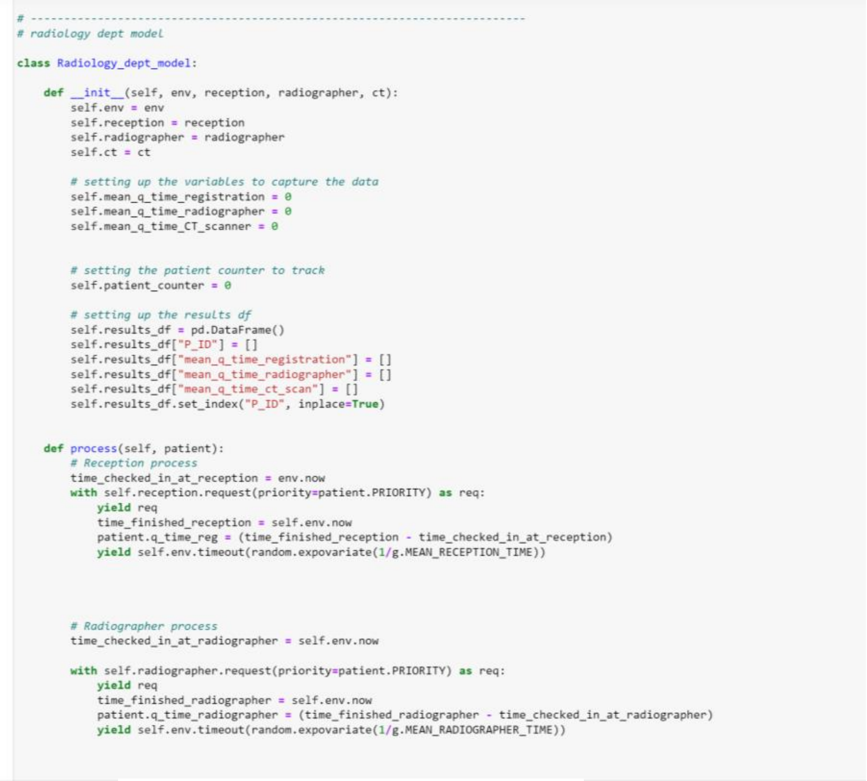
The following are some important code pieces to be aware of:



```
In [1]: import simpy
        import random
        import pandas as pd
        import numpy as np
        import csv
        import matplotlib.pyplot as plt
        from statistics import mean
        import seaborn as sns


        #--------------------------------------------------------

        # Global class for variables

        class g:
            MEAN_RECEPTION_TIME = 5       # Minutes to process by the receptionist
            MEAN_RADIOGRAPHER_TIME = 5   # Minutes to process by the radiographer
            MEAN_CT_TIME = 15             # Minutes for the Scheduled CT scan

            # For scheduled patients
            INTER_ARRIVAL_TIME = 10       # Minutes between scheduled patients

            # Emergency variables
            STD_DEV_EMERG = 5
            EMERGENCY_INTER_ARRIVAL_TIME = 15 # Minutes between emergency patients
            MEAN_EMERG_TIME = 15          # Minutes for Emergency CT


            # Simulation variables
            SIMULATION_TIME = 600         # Simulate a 10 hour day
            NUMBER_OF_SIMS = 100          # How many times to run the simulation



            # Capacity for resources
            RECEPTION_CAPACITY = 1
            RADIOGRAPHER_CAPACITY = 1
            CT_CAPACITY = 2

            #decontamination variables
            decon_mean = 20
            decon_std_dev = 5
            percent_infect = 0.2
```

Figure 5. Global class for variables

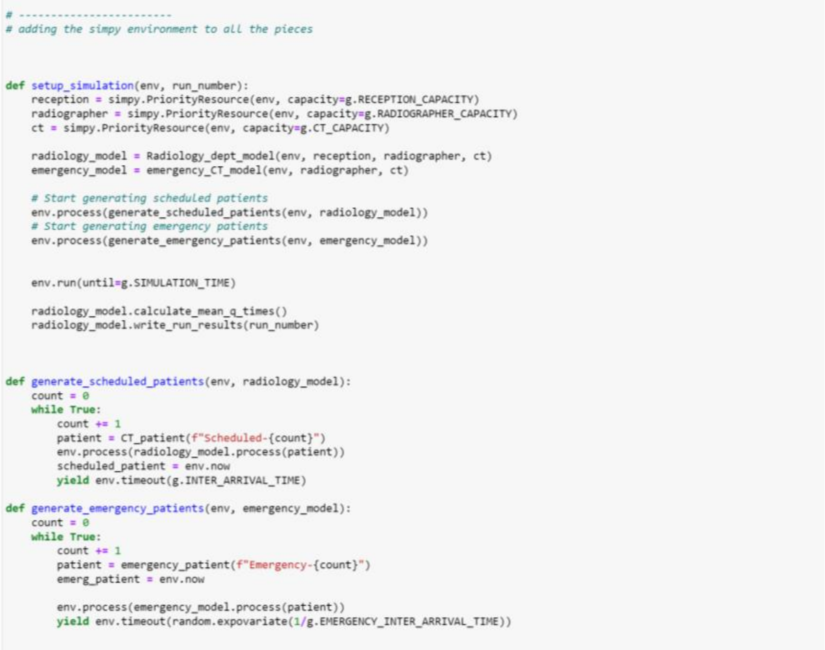
Figure 5 shows the global class for variables which is used throughout the code.



```python
# ---------------------------------------------------------------------------
# radiology dept model

class Radiology_dept_model:

    def __init__(self, env, reception, radiographer, ct):
        self.env = env
        self.reception = reception
        self.radiographer = radiographer
        self.ct = ct

        # setting up the variables to capture the data
        self.mean_q_time_registration = 0
        self.mean_q_time_radiographer = 0
        self.mean_q_time_CT_scanner = 0


        # setting the patient counter to track
        self.patient_counter = 0

        # setting up the results df
        self.results_df = pd.DataFrame()
        self.results_df["P_ID"] = []
        self.results_df["mean_q_time_registration"] = []
        self.results_df["mean_q_time_radiographer"] = []
        self.results_df["mean_q_time_ct_scan"] = []
        self.results_df.set_index("P_ID", inplace=True)


    def process(self, patient):
        # Reception process
        time_checked_in_at_reception = env.now
        with self.reception.request(priority=patient.PRIORITY) as req:
            yield req
            time_finished_reception = self.env.now
            patient.q_time_reg = (time_finished_reception - time_checked_in_at_reception)
            yield self.env.timeout(random.expovariate(1/g.MEAN_RECEPTION_TIME))



        # Radiographer process
        time_checked_in_at_radiographer = self.env.now

        with self.radiographer.request(priority=patient.PRIORITY) as req:
            yield req
            time_finished_radiographer = self.env.now
            patient.q_time_radiographer = (time_finished_radiographer - time_checked_in_at_radiographer)
            yield self.env.timeout(random.expovariate(1/g.MEAN_RADIOGRAPHER_TIME))
```

Figure 6. radiology department class

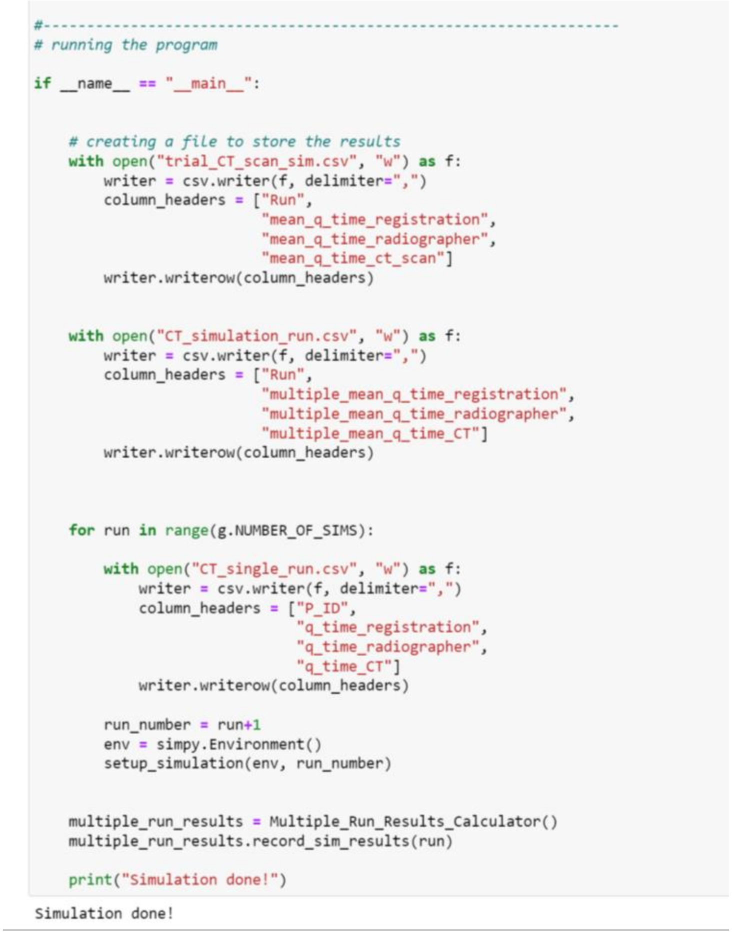
Figure 6 defines the radiology class which the entities will run through.



```python
# -----------------------
# adding the simpy environment to all the pieces


def setup_simulation(env, run_number):
    reception = simpy.PriorityResource(env, capacity=g.RECEPTION_CAPACITY)
    radiographer = simpy.PriorityResource(env, capacity=g.RADIOGRAPHER_CAPACITY)
    ct = simpy.PriorityResource(env, capacity=g.CT_CAPACITY)

    radiology_model = Radiology_dept_model(env, reception, radiographer, ct)
    emergency_model = emergency_CT_model(env, radiographer, ct)

    # Start generating scheduled patients
    env.process(generate_scheduled_patients(env, radiology_model))
    # Start generating emergency patients
    env.process(generate_emergency_patients(env, emergency_model))


    env.run(until=g.SIMULATION_TIME)

    radiology_model.calculate_mean_q_times()
    radiology_model.write_run_results(run_number)


def generate_scheduled_patients(env, radiology_model):
    count = 0
    while True:
        count += 1
        patient = CT_patient(f"Scheduled-{count}")
        env.process(radiology_model.process(patient))
        scheduled_patient = env.now
        yield env.timeout(g.INTER_ARRIVAL_TIME)

def generate_emergency_patients(env, emergency_model):
    count = 0
    while True:
        count += 1
        patient = emergency_patient(f"Emergency-{count}")
        emerg_patient = env.now

        env.process(emergency_model.process(patient))
        yield env.timeout(random.expovariate(1/g.EMERGENCY_INTER_ARRIVAL_TIME))
```

Figure 7. simulation setup function

Figure 7 shows the simulation setup function which starts the simpuy processes running.



```
#-----------------------------------------------------------------
# running the program

if __name__ == "__main__":


    # creating a file to store the results
    with open("trial_CT_scan_sim.csv", "w") as f:
        writer = csv.writer(f, delimiter=",")
        column_headers = ["Run",
                          "mean_q_time_registration",
                          "mean_q_time_radiographer",
                          "mean_q_time_ct_scan"]
        writer.writerow(column_headers)


    with open("CT_simulation_run.csv", "w") as f:
        writer = csv.writer(f, delimiter=",")
        column_headers = ["Run",
                          "multiple_mean_q_time_registration",
                          "multiple_mean_q_time_radiographer",
                          "multiple_mean_q_time_CT"]
        writer.writerow(column_headers)



    for run in range(g.NUMBER_OF_SIMS):

        with open("CT_single_run.csv", "w") as f:
            writer = csv.writer(f, delimiter=",")
            column_headers = ["P_ID",
                              "q_time_registration",
                              "q_time_radiographer",
                              "q_time_CT"]
            writer.writerow(column_headers)

        run_number = run+1
        env = simpy.Environment()
        setup_simulation(env, run_number)


    multiple_run_results = Multiple_Run_Results_Calculator()
    multiple_run_results.record_sim_results(run)

    print("Simulation done!")
```

```
Simulation done!
```

Figure 8. running the function

Figure 8 demonstrates the part where the code runs altogether, bringing in the functions and the classes.