ClinicFlow

Design Document

Maxim Vasiliev #400043983
Susie Yu #000955758
Karl Knopf #001437217
Weilin Hu #001150873
Yunfeng Li #001335650

January 12 2017

Contents

1 Introduction

1.1 Purpose

The Josef Brant Hospital pre-operative clinic (the client) schedules upwards of 50 patients per day. The appointment times are digitized, yet manually chosen by clinic staff. Once at the clinic, each patient undergoes a varying set of procedures with different durations. While staff have a good feeling of how to schedule patients, mistakes and inefficiencies often occur considering the numerous constraints and temporal variation of events. The client has approached us to explore the potential of optimizing the scheduling process. This would help staff both foresee potential scheduling errors and free themselves up to do other work, maximizing profit and minimizing surplus capacity.

1.2 Description

This project aims to produce a tool which allows parties in the preoperative clinic sector to optimize the scheduling of patients amongst different procedures at the clinic, as well as by arrival time and date. Demand for the solution stems from a lack of relevant products, and the reliance of clinic staff on intuitive and error prone manual scheduling. With this tool, prior patient temporal data will be fed in and used to build a model of all the variables involved. A simulation engine will then be created to produce an optimized scheduling of patients under inputted constraints. This would allow clinic staff to reduce scheduling errors, as well as conserve resources by automating the scheduling process.

1.3 Scope

A desktop application or web-interface application can allow users to insert necessary data, such historical patient procedure durations, doctor and nurse shift hours, and other constraints such as break allotments or soft constraints such as employee shift end times. Based on provided data and inputs, the system will generate patient schedules for both arrival time into the clinic, and between procedures within the clinic. This tool will be designed with the intention to be implementable in multiple health care institutions for automating and managing patient scheduling.

2 Overview

2.1 Scheduled Deliverable and Development Time

The final deadline for the project is mid April 2017. The detailed deliverable and their respective deadlines are listed below:

- Requirements Document - revision 0: October 12, 2016
- Proof of Concept Plan: October 26, 2016
- Test Plan - Revision 0: November 2, 2016
- Proof of Concept Demonstration: November 21, 2016
- Design Document - Revision 0: January, 11, 2016
- Demonstration - Revision 0: February 13, 2017
- User's Guide - Revision 0: March 1, 2017
- Test Plan - Revision 0: March 22, 2017
- Final Demonstration: Mid-April, 2017
- Final Documentation: April 5, 2017

2.2 Revision history

Table 1: Revision history

Date	Comment
Jan 11, 2017	First draft

3 Overall Structure

The product will be delivered in a web application format. This maximizes its accessibility and allows it to be device agnostic. The core simulation component is written in Python 3, and makes use of the discrete event simulation libraries provided as part of the SimPy package. To maintain consistency in development language, we opted to use the Django framework for the web interface to our application. Data to be inputted and generated will be stored in a MongoDB database, and will use the provided Python MongoDB Drivers to interface between the view and data.

4 User Interface Design

4.1 Navigation flow

The first page presented to the user is the login page. The user must login as a manager or a authentic viewer before further operation. After logging in, the user will be redirected to the main menu page. The main menu page can direct user to the patient information page, clinic information page, generate schedule page and simulate schedule page. The user who log in as a viewer can only view patient information and generated schedules. Meanwhile the manager account would have viewer capabilities, in addition to being able to add/change patient information, add or change clinic attribute data, run simulations on specific patient data, or generate schedules based on patient data.

Each page has a navigation bar which let user transfer to different page. Each page has a logout button, thus the user can quit the system at any time.

4.2 Login

The user has to log in with the correct account and password before using the system. The login page has straight forward design which allows users to easily log in. This page contains internal validation to protect the system. This page can distinguish the user as a manager or a viewer, and gives the user corresponding authority.

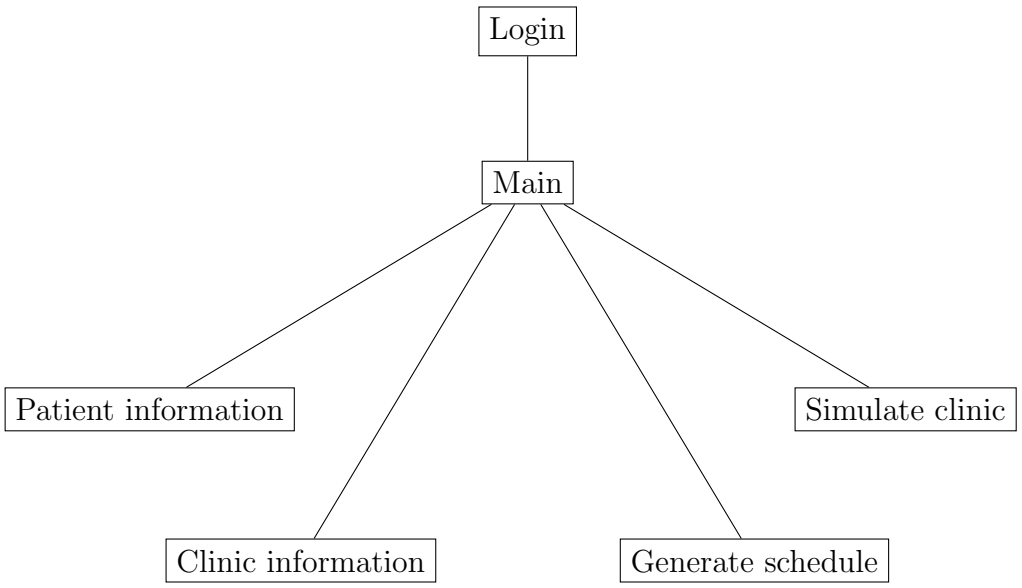

Figure 1: Simulation Engine Uses Diagram

4.3 Main Menu

This page outlines general information such as today's reservation schedule, today's patients in list, and patients need schedule for next day which can give the user a quick overview. This page can guide the user to patient information,clinic information, clinic simulation, and schedule generation pages.

4.4 Patient Information

The manager account user can insert patient specific information such as name/id, appointment time, and required procedures into the system's database. There is a list of empty fields for the user to fill up. The fields for appointment time and procedures are vital and must be entered in order to add the patient to the system, while other fields may be optional . The save button will validate the inputs before saving the patient data into database. The validation function will check the inputs to ensure the data consistency. If errors occur, the system will retain the inputs and inform the user of the source of error. Bulk patient data import is also supported through import

of standardized csv files.

Both manager and view accounts are able to also see information on all patients current in the system. This will be presented both as list and timetable formats. Search and filter functionality will be included to allow users to target a particular subset of patients. Modification of existing records is also allowed, but only be the manager account.

4.5 Clinic Information

This page will give an overview of the attributes that model the clinic. This includes employee numbers, break times, starting times, etc. Both manager and view accounts will have access to this page, but only manager accounts will be able to edit the information therein and save it to the database. Multiple profiles for clinic information are also supported. This allows the user to simulate the clinic under models and compare the results.

4.6 Simulate Clinic

This page will be the interface to the simulation engine. The user will be able to run a simulation of the clinic given the data provided in the patient and clinic information sections of the application. The results of the current simulation and its summary statistics, as well as those of previous simulations, will be displayed here. The user is able to choose the day, and clinic data profile to use in the simulation. This page will also allow users to modify patient or clinic attributes (such as number of nurses available) to quickly see their effects on simulated clinic operation.

While managers and viewers will both be able to see past results, only managers have the authority to run simulations.

4.7 Generate Schedule

This page allows users to generate optimized patient arrival schedules based on patient and clinic information data. Users can select the day and clinic profile to generate schedules for. Top results will be displayed with their summary statistics in an easily comparable format. Past generated schedules are also viewable, and sorted by the date. The schedule will clearly present

the arrive time and patients' first visit room. Manager has the permission to delete or update existing schedules. The manager account user can add a patient's reservation into the existed schedule. The backend system will simulate the procedure and time expense of the patient inside clinic. The system also will alert the use to any potential issues such as a patient having to wait too long.

While managers and viewers will both be able to view past generated schedules, only managers have the authority to generate new schedules.

5 System Architecture

5.1 Design Description

The simulation engine is the core component set which is responsible for running the simulation of the clinic. The patients and their associated data are the entities that are tracked by the simulation, the clinic workers are the available resources that can service the patients, while the clinic environment acts as the constraints under which the patients flow through the system. This lends itself to a natural tree like organization structure. The modules thus have low coupling and high cohesion, with details pertinent to their operation being compartmentalized. *SimulationEngine* and Simulation run the simulation events, *PatientSchedule*, *HealthCareSchedule*, and *Clinic* maintain the objects used in the simulation, while *Patient*, *HealthCareWorker*, and *ClinicModule* keep track of the data if individual units in the simulation.

5.2 Simpy

Simpy is a Process-based discrete-event simulation framework which works with Python 3. It is well known and has a solid support base. The framework is relatively easy to work with and allows for a variety of event simulation types which are useful to this project.

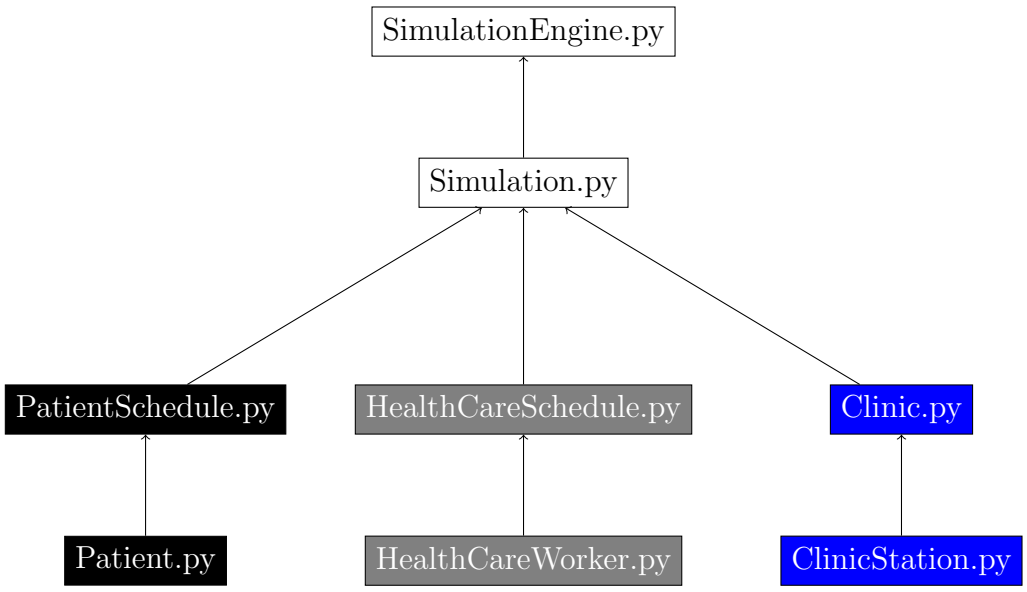

Figure 1: Simulation Engine Uses Diagram

5.3 SimulationEngine.py

- Acts as a top level for the program
- Handles File I/O for simulation
- Uses Simulation.py to run a simulation
- May be merged with Simulation.py for lower coupling

5.4 Simulation.py

- Manages the simulation of the clinic
- Uses SimPy
- Has methods for workerRun and patientRun to create "threads" for the objects
- Also contains the simulation resources

- Uses HealthCareSchedule, PatientSchedule and Clinic

5.5 Clinic.py

- Contains the information about the clinic
- Constructor method reads in clinic data from file
- Uses ClinicStation

5.6 ClinicStation.py

- Contains information about an individual station in the clinic
- Contains methods for activating and deactivating a station
- Contains a method for getting a random number, generated from that particular station randomness

5.7 HealthCareSchedule.py

- Contains information about the provider's schedule in the clinic
- Contains methods for generating a schedule and for reading a schedule from a file
- Uses HealthCareWorker.py

5.8 HealthCareWorker.py

- Contains information about a particular provider in the clinic
- Contains methods to change the worker's scheduled times, and which station they are at
- Has a toString method for ease of use

5.9 PatientSchedule.py

- Contains information about the patient's schedule in the clinic
- Contains methods for generating a schedule and for reading a schedule from a file
- Uses Patient.py

5.10 Patient.py

- Contains information about a patient in the clinic
- Has methods to adjust the patient's schedule, and help keep track of their actions throughout the clinic
- Has a toString method for ease of use

6 Database Structure

6.1 Implementation Details

MongoDB will be used as a database to store all input data related to modeling the clinic, as well as output data generated from simulation. The database will also be used to recall previously stored data. MongoDB does not have a rigid structure, but its simplicity is well suited for this project. All data sets will be stored in a corresponding MongoDB collection. All data has a key attribute to improve searching efficiency and avoid duplication.

6.2 Data format

The **patient** collection contains all required patient information. This includes patient name/id, appointment date, time, as well as the list of services required by the patient. This information will be used to simulate clinic operations on a per day basis.

Table 2: Patient Table

Attribute Name	Sample Value
Patient Name	Jack Square
Reservation Date	2017-01-01
Reservation Time	8:15
Procedure	Interview, Bloodwork, x-ray

The **clinic** collection contains all attributes of the clinic we are modeling. This data will act as constraints on the clinic simulation, and can be modified by staff to accommodate operational changes.

Table 3: Clinic Table

Attribute Name	Sample Value
Nurse Number	3
Available Interview Room	3
Clinic Open Time	8:00
Clinic Close Time	18:00
Reception Close Time	15:00

The **result** collection contains all results from the simulations once they complete. This data will be used directly by the staff to help in deciding patient flow within the clinic during operation. This data will also help inform staff of attribute or scheduling effects on clinic operations.

Table 4: Results Table

Attribute Name	Sample Value
Day	2017-01-01
Simulation Run #	1
AVG duration bloodwork	20 minutes
AVG patient wait time	1 hr
Final patient end time	17:00

6.3 Sanitation

Simulation engine relies on the format and accuracy of data. The patterns of data must match the pattern inside simulation engine to ensure the simulation engine runs properly. Upon insertion or modification of data, a validation process verifies the inputs or changes before storing into database. Invalid inputs or changes would be rejected and the system would inform the reason for rejection. If no error in input or changes is detected, the system should allow the updates.

7 Web Application Design

7.1 Frameworks

The Django framework is used in this project. It provides the users with a way to access the simulation engine, which includes import data, running the simulation and utilizing other functions, and presents the simulation results in a concise and clear manner. At the same time, It helps to organize the backend structure of the project, even though it also supports the front end user interface. The following is a breakdown of the framework as it applies to our project.

- models.py: integrates the simulation engine. It also deals with data flow from database to the simulation engine, and sends the corresponding outputs of the engine to the views.py. Other modules may be required to interact with the MongoDB as a database interface.
- views.py, static folder and templates folder: when the users send requests to the presetting URL address, it acts as a router and directs the users to pages that are generated from the template html in the templates folder. Since the static template HTML will be merged into the simulation results by a dynamic mechanism, the users can view the simulation results and operate the simulation engine instantly.
- test.py: for the later testing cases, it ensures the backend works correctly and meets the design requirements.
- urls.py: stores the settings of the URL address for users to visit.