Electronic Scheduling Tool for Surgery Clinic

PROJECT PLAN

sdmay18-26
UnityPoint
Srikanta Tirthapura
Joel May - System Admin
Repo and Schedule Manager - Matthew Burket
Point of Contact & Meeting Facilitator - Ryder Schoon
System Architect & Chief Engineer - Matthew Cavalier
Meeting Scribe - Maddie Andreassen
Report Manager - Matthew Burket
Testing Manager - Luke Sternhagen

sdmay18-26@iastate.edu
https://sdmay18-26.sd.ece.iastate.edu/

Revised: 22 September 2017

Contents

1 Introduction 2

1.1 Project statement 2

1.2 purpose 2

1.3 Goals 2

2 Deliverables 2

3 Design 3

3.1 Previous work/literature 3

3.2 Proposed System Block diagram 3

3.3 Assessment of Proposed methods 3

3.4 Validation 3

4 Project Requirements/Specifications 4

4.1 functional 4

4.2 Non-functional 4

5 Challenges 4

6 Timeline 5

6.1 First Semester 5

6.2 Second Semester 5

7 Conclusions 6

8 References 6

9 Appendices 7

NOTE: This template is a work in progress. When in doubt, please consult the project plan assignment document and associated grading rubric.

1 Introduction

1.1 Project statement

We are developing an electronic scheduling tool for UnityPoint's weight loss clinic in West Des Moines. This needs to take the available staff and rooms and schedule them as efficiently as possible for patient visits, as well as take in a set of user-defined constraints on when and how things are scheduled.

1.2 purpose

The purpose of this project is to develop a more efficient scheduling system that can save staff time, and patient waiting time which has proven to be problematic for the clinic.

1.3 Goals

Explain what you hope to accomplish through this particular senior design project. What would you like to achieve? Enlist as many goals as you can envision.

We would like to create a web application with the following features as goals

- Retrieve information from EPIC
- Schedule a patient time slot
- Assign rooms to time slots
- Assign Employees to time slots
- Input scheduling constraints
- Schedule non-patient busy time for employees where they cannot be scheduled

The following are our stretch goals for our project's features

- Update the information in EPIC
- Set employee pairing preferences
- Set regular employee schedules
- Appointment cancellation notifications
- Automated client notification
- Responsive design

2 Deliverables

We are expected to deliver an internal web application that connects to UnityPoint's EPIC Electronic Medical Records (EMR) system and can be used to determine available appointment times for patients.

3 Design

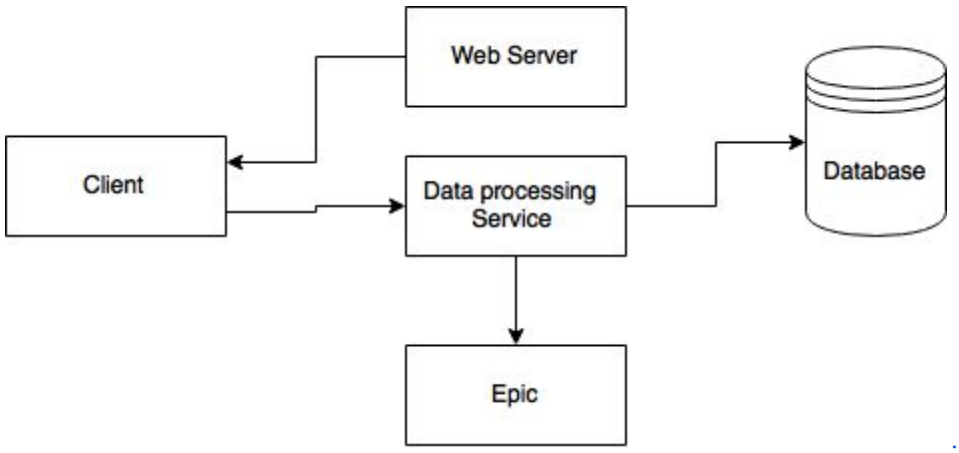
3.1 Previous work/literature

At the moment there do exist several products perform scheduling tasks. One such product is the Iowa State Course Schedule planner (see references), which allows a student to input courses and get back possible course schedules for a semester.

Through a simple google search you will find a lot of different scheduling software available for enterprise use, however, they don't meet the unique criteria that clinic needs.

The problem at hand also reminds the team of the job-shop scheduling problem discussed in Com S 311 which is an optimization problem generally regarding what our client wants, a scheduler that assigns jobs to resources at particular times. Due to jobs already being scheduled and the general approach that clinics use for scheduling, we believe the best solution for this problem would be a FIFO approach to this problem.

3.2 Proposed System Block diagram



In this architecture, we have five major components to work with. Firstly, we have the Web Server which has a singular purpose of serving the web page to any browser that requests it. Then we have the client itself, which will be the web page that the user can interact with. Finally, we have the Data processing Service that acts as a linchpin bringing together the data from the Epic system and the database that stores information needed for scheduling, along with providing predicted schedules to the client.

3.3 Assessment of Proposed methods

For this problem there is a basic but tried and true method that we can follow when developing the system. This method is the client-server method of implementation. What this means is that we build a client (think website, desktop application) that the end user will see and interact with, and and server that is stored somewhere away from the user that handles all of the processing, storage, and communication amongst all of the client applications.

The reason that this is a stronger candidate for how we implement the system is that it allows us to be flexible. We can start designing the system earlier and get the server-side processes figured out prior to planning out the end user implementation. This allows us to leave the possibility open for a website or a desktop application for the end user. This also allows us to be more confident that our system has more security as all interactions involving multiple users can happen away from a single user.

Another option that we have is to focus on building a website, that does all of the processing in the client (browser), and a server that simply delivers the page to the end user. However, this method is less than ideal as it gives the end user access to API's (see references) which could lead to security issues. Along with that, performing computations or logic would be limited by the specific machine and the web browser that the user is interacting with.

Among these two options, we know that the server client method would be ideal for our system, and that is what we will move forward with.

3.4 Validation

In order to confirm a quality solution, we will implement testing procedures ranging from specific units of code all the way to manual testing. Luckily, the base problem that we are trying to solve does have a fairly clear cut requirements. The main indicator that we will look for is whether the system produces good suggestions: arranging appointments with various medical professionals as closely as possible while taking up the least time possible and avoiding down time for staff.

4 Project Requirements/Specifications

4.1 FUNCTIONAL

Application that helps the client with scheduling - By the end of this project the client should receive an application that helps them with scheduling appointments for patients and employees.

EPIC Information Retrieval - Application should be able to retrieve information from the client's current EPIC scheduling software in order to suggest potential appointment times.

Scheduling Algorithm - The client would like an algorithm that can efficiently schedule multiple appointments for people with as little waiting for the patient and employee as possible. The algorithm must also be able to adapt to added or removed doctors/nurses or patients canceling an appointment.

Security - The software may contain sensitive information such as the patients or employees contact info. This data should not be accessible outside of the hospital and should be encrypted and password protected.

4.2 NON-FUNCTIONAL

Load in under 5 seconds - This software needs to be able to determine the best times for a patient to schedule an appointment. It needs to be able to generate multiple options for appointments so that the patient and the scheduler can quickly determine the next appointment and get back to their work.

EPIC Information Update- Application should be able to update the EPIC scheduling software. It should be able to add/remove appointments from the schedule

Flexibility - The client has requested that this software be as flexible as possible. They can't predict their future needs so this software needs to be able to accommodate the addition or removal of rooms to schedule in, nurses and doctors being out for the day, and patients cancelling. The client suggested that this software may also be useful for other areas of the hospital so it should be applicable to those areas as well.

Usability - This software needs to be easy to use for someone who has never used the software before. For example if a substitute was ever needed for the day they should be able to learn the software as quickly as possible. At most it should take no more than 5 minutes to become proficient in the software with help from an experienced user.

Notifications - System should be able to alert patients if their appointment is delayed or cancelled.

BackUp - Should backup scheduling data once a day on a separate system.

4.3 Standards

HIPPA - This software must be HIPPA compliant. We are working with sensitive data so it needs to be secured.

JavaScript Style Guide - We will select a style guide for JavaScript when we start development.

Shared Repository Model - Developers must request a merge request before merging with master

UML Standard - We will use this standard for use case, architecture, deployment diagrams.

Continuous Delivery - As a team we should be committed to keeping Master branch bug free and ready to deploy at any time.

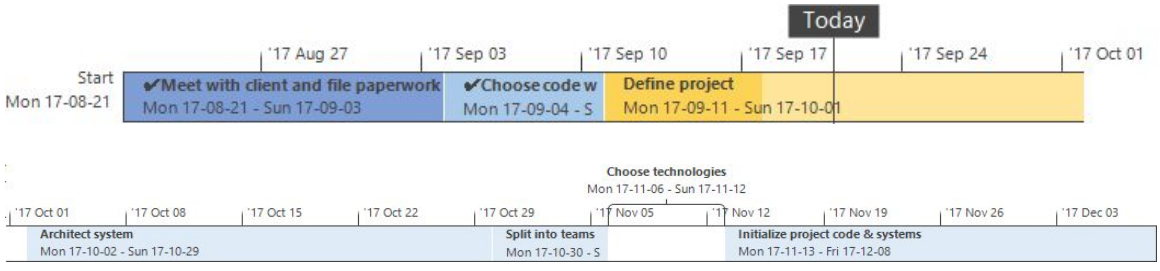
5 Challenges

A large part of our project is a scheduling algorithm. Only two of our group members have completed COM S 311, Iowa State's Undergraduate algorithms course. This could possibly cause issues with design of scheduling algorithm due the team's lack of formal knowledge of scheduling algorithms.

One of the goals of the project is to integrate with UnityPoint's Electronic Medical Records (EMR) system, Epic. Since we don't have any knowledge of Epic we do not know how much work we will need to put into integrating our code with Epic. Integrating with Epic is land of unknowns and therefore could be challenge.

6 Timeline

6.1 First Semester



Past:

- 2 weeks: Meet client and file paperwork.
- 1 week: Choose general code workflow and standards.

Current:

- 3 weeks: Define project and technological constraints. Get project requirements.

Future (~10 weeks):

- 4 weeks: Architect system.
- 1 week: Split into teams: front-end, back-end, and algorithm.
- 1 week: Choose frameworks and languages/platforms. Choose style guides.
- 3 weeks: Initialize project code and systems (CI, test framework).

6.2 Second Semester

- Continue work on code
    - 8-10 weeks
- Write tests
    - 8-10 weeks
    - Done along side with the coding
- Validate solution with client
    - 1-2 weeks
- Integrate feedback into application
    - 2-4 weeks
- Write Documentation
    - 1-2 weeks

When the project has been architected, we can start making a better the plan for the second semester.

7 Conclusions

Goals

- Retrieve information from EPIC
- Schedule a patient time slot
- Assign rooms to time slots
- Assign Employees to time slots
- Input scheduling constraints
- Schedule non-patient busy time for employees where they cannot be scheduled

Stretch goals

- Update the information in EPIC
- Set employee pairing preferences
- Set regular employee schedules
- Appointment cancellation notifications
- Automated client notification
- Responsive design

To sum things up, the requirements for this project could use tidying up and our understanding of the technologies limits our ability to further define part of our plan. While we still have plenty of options to sort through and plenty more questions to ask, what we do know is definitely enough to keep us busy with both research and planning.

8 References

Iowa State Course Schedule planner: http://classes.iastate.edu/home

9 Appendices