Project Proposal: Web-Based Scheduling Application for New Haven Reads

I. Introduction

The purpose of this project is to create an application for a local non-profit tutoring organization called New Haven Reads. The application will focus on improving the way that the staff interacts with the schedules of tutor-student appointments. It will allow a more efficient work-flow at New Haven Reads, freeing staff members to work on tasks other than those related to the schedules. It will address scheduling on two fronts: first, it will be a tool for more efficiently creating, editing, and maintaining the schedules; second, it will be a tool for more efficiently using the schedules.

II. Background: New Haven Reads

The New Haven Reads Free Community Book Bank was founded by Christine Alexander in 2001 as a place to collect books and distribute them, free of charge, to the community. New Haven Reads offers free books to anyone who wants them, and currently gives out about 100,000 each year to individuals, teachers, and other non-profit organizations.

A few years after its start, New Haven Reads began offering free one-on-one tutoring to local children. The tutoring program has since grown from fewer than 10 children tutored each week to more than 450. This tutoring is made possible through the work of over 350 volunteer tutors. Tutoring takes place at three locations, and will soon expand to five.

III. Scheduling at New Haven Reads Now

Schedules of tutors-student appointments are maintained by the scheduling coordinator at New Haven Reads. These schedules are stored as Excel spreadsheets on a network drive at the main location of New Haven Reads, printed weekly, and distributed to the three locations. Thus, any changes made mid-week are usually written in by hand, particularly at the two secondary locations that do not have daily access to the Excel spreadsheets. These hand-written changes often get confusing.

During the week, the schedules are used by coordinators, tutors, and students to identify which tutors work with which students. It is also used to check-in tutors and students when they arrive, to indicate when students and tutors will not be coming (or did not come), and as a convenient place to leave notes about permanent changes to the schedule. Furthermore, to cope with students and tutors who do not come, lots of handwritten changes are often made, as tutors are moved from a child who did not come to one whose tutor is absent.

At the end of the week, these schedules are returned to the main location. There, the scheduling coordinator makes any changes that need to be made to the Excel spreadsheets, students and tutors who have consistently not come for their appointments are called, and a staff member or volunteer counts various statistics. The paper schedules are then filed into a binder.

IV. Goals of the Application

The primary goal of this application is to move this Excel-and-paper work-flow into a web browser.
The data will be moved to a database, where there is much less worry of version issues. Staff members will be able to access and edit the schedules from anywhere with an Internet connection. Coordinators will be able to make short-term changes to the day's or week's schedule, as well as check-in students and tutors as they arrive. And all data relevant to the schedules will be automatically stored and easily accessible in a useful format.

V. Essentials of Functionality: Daily Use

The application's main purpose will be to replace the paper schedule used by coordinators on a daily basis. After a log-in prompt, the main screen of the application will display the day's schedule of tutor-student appointments at a particular tutoring location. There will be tabs to easily switch between days and between locations, as well as some visual indication of which is the current day and the current location, so that coordinators do not accidentally use the wrong schedule.

The schedule shown in the browser, at least in its first iteration, will mimic the current paper schedule. It will be shown as a table, with each row corresponding to a tutor-student appointment. Each row will contain the tutor's and student's names and phone numbers, as well as any relevant notes. The rows will be separated by start-time and then ordered by student first name. These decisions have been made by New Haven Reads regarding their paper schedules, and are being inherited for this project.

Each cell of the table will have a white background by default. Next to each name there will be two small, round buttons. One button (perhaps a check-mark) will be pressed when a student or tutor checks in. Pressing this button will turn the person's name and phone number cells green, indicating that he or she is present. When checked-in, the check-in button will be replaced by an undo button, used to indicate that a mistake was made, and that the student is not present. The second button next to each person's name (perhaps a C/O or an X) will be pressed to indicate that the tutor or student has (been) called out. Pressing this button will turn the person's name and phone number cells red, indicating that he or she will not be coming for the appointment. After it is pressed, it too will be replaced by an undo button. There will also be a button that allows notes to be added about why the person did not come. These notes may or may not be visible on this screen.

Coordinators must also be able to change tutoring assignments in the short-term. They must be able to move tutors from one student to another. Perhaps most elegantly, this can be accomplished by drag-and-drop. A less elegant solution will be accepted, however, if implementing drag-and-drop turns out to be excessively complex.

The application will record in the database all relevant data. This includes which students and tutors were present on which days and which tutors worked with which students (which is of particular interest when a tutor works with a student other than the one he or she was scheduled to work with).

VI. Essentials of Functionality: Editing Schedules

There will be a separate screen devoted to editing the permanent weekly schedule. This screen will have a design that is similar to the daily-use screen, but it will not have the check-in and call-out buttons. Instead, there will be an edit button that will allow the coordinator to permanently change the tutor and/or student for a particular appointment.

There are several non-essential features associated with this page. In order of importance, the first is that the application refuse to allow the coordinator to enter the name of a student or tutor who is not in
the relevant table in the database. To aid with this requirement, a second feature is that the application suggest names from the relevant table, based on what has already been typed. A third non-essential feature is, as in the daily schedule, the ability to drag-and-drop tutors and/or students between appointments.

VII. Non-Essential Features

The following are other features to be added if there is time remaining after the essential features have been completed.

First is integration with, or perhaps replacement of the Excel-based “database” of tutor, student, and donor information. With this data moved into the proposed application, the application will be able to display an information page for each person on a schedule. This seems to be the correct long-term solution, but is not essential to getting the scheduling application running and increases productivity.

Second is the ability for tutors, parents, and older students to call out from the Internet. The requirements for this feature are both an interface for the person who wishes to call out, and the implementation of behavior on the scheduling screens. Presumably the person must be colored red on the relevant daily schedule, but there may also need to be some sort of notification if the call out happens on the day of the appointment so that the coordinator is immediately made aware of the situation.

VII. Technical Overview

The application will be written in a high-level language, with the aid of a web framework technology. Python is the most likely candidate for the language, because of personal preference and familiarity, as well as the availability of frameworks. Google App Engine is a promising candidate for a web framework, as it also offers free web hosting for applications that fall below particular quotas. It is expected that this application will fall below those quotas, as it will only be used by members of the New Haven Reads staff. Google App Engine can also integrate with Django, another web framework for Python. The competing alternative to Google App Engine and Django is Turbogears. The former choice is currently preferred due to greater ease of installation and availability of documentation and support.

Javascript will likely be added to the front-end of the application, in order to enhance visual effect and facilitate AJAX communication between the client and the database. The Javascript library jQuery will likely be used.

The database will likely be an SQLite database, as it comes with the Python language and the volume of relevant data is sufficiently small.

VIII. Deliverables

Expected deliverables are those required for completion of CPSC 490:

- a paper about the project
- a set of web pages explaining the project
- the code for the application
- anonymized data to be used with the application