Scheduling Application

Team 8: George Wanjiru, Barend Venter, Jonathan Hyry, Ryan Soushek

Tentative Application Name: iPlan

Problem

Our team could not figure out when the best times were to meet for optimal productivity. We knew each other's schedules, and with those we can determine the best time to meet by comparing how they match up. It would be nice if a program could do this for us, hence iPlan.

Other scheduling applications are not automated when it comes to dividing and allocating the available times of team members, based off team member schedules. We think this problem is unique.

End Users

Our target category of end users is any team or group that runs on a schedule.

Typical Tasks

User tasks include the following:

- Adjust a probability
- Add a new team member
- Edit existing team members
- Select team members who will be meeting
- Adjust number of hours to meet a week
- Find meeting times
- View meeting times
- Save meeting times
- Save a team member

The probability setting in the program would attempt to foresee absences based on attendance trends collected over time. In some situations it will be user adjustable. A user will be able to adjust the number of hours per week that the team would like to meet, and will be able to select which team members will be available for meeting times across the span of a week. The user will be able to then generate meeting times, view them, find meeting times, and save the meeting times, visible in the UI at the time of viewing, to a file.

There will be a detailed form for adding new team members to the program. The user

will access this form, enter data, and click a button to save the new team member. The most important component of the team member objects/data will be the scheduling data; this data is critical to the operation of the program. A user will be able to edit existing team members and save team members to a file.

Program Design

In designing a scheduling program, a significant portion of that design time will go to the calendar view that shows a calculated schedule. Our program will handle the specified scheduling tasks and nothing more; we may add more functionality if we have extra time, but our main design focus will be implementing a well-designed scheduling application for teams. It doesn't seem that any of the features we are proposing will need to be faked. We will address UI challenges with respect to a perspective of team management. The program needs to be tailored to the applicable operations in general. The UI must be graphically designed to meet the needs of management and team members.

Possible difficulties

We have anticipated the following difficulties:

- Sharing Plans across multiple computers, for multiple team members
- Informing team members of new meeting times
- Making it easy to use
    - Maintain efficiency through that process
- Displaying a calender with associated probabilities
- File format for saving and sharing student schedules

In determining a schedule, the user should be able to share it with other users that also have the application on their computers. We need to solve this problem by implementing some way to share it over a network. In informing team members of a newly generated schedule, we propose that a capability to automatically send emails to all team members would be a useful piece of the program. This presents implementation difficulties in figuring out what email system to use to send automatically generated emails, and how to integrate that solution into our application. These networking features may be faked on the prototype. We will need to build some sort of canvas/grid to present calendar data in a user-friendly manner. This is a main program challenge; significant attention will be paid to this problem and any underlying algorithms necessary to generate calendar data. We will have the application save data in its own file format. This will create difficulty in the implementation of team member files and calendar data, in ensuring they are organized and able to be serialized in such a manner as to keep the file size as small as possible. We propose building and retaining ease of use for the specified end user audience, while maintaining efficiency in completing tasks within the application.