EasyShift Final Report

Final report for CSCI 483. Written by Travis MacDonald and Prahar Ijner.

Introduction

Task scheduling is a process that occurs in most companies on a regular basis. For businesses that employ both part-time and full-time workers, a good schedule can result in better employee satisfaction, customer satisfaction, and minimize company costs. However, manually producing and maintaining physical schedules is a needlessly difficult process that can be simplified using software. For this reason, we aim to create a software solution that automates and facilitates the scheduling process.

Background

Though the scheduling process may appear trivial upon first glance, a closer look should reveal its laborious nature. Consider how this might look for a small grocery store.

The first step in creating a schedule is to consider all of the parameters involved. There are global parameters that apply to all employees (e.g. regular hours of operation), recurrent parameters that apply to individual employees (e.g. some employee is unavailable on Wednesday evenings), and single-occurrence parameters that apply to individuals (e.g. some employee is unavailable on February 1st).

It's important to note that parameters also vary in nature. There are strict parameters that must be upheld, and there are soft parameters that are not necessarily satisfiable. Regular hours of operation, for example, are strict because employees are guaranteed to not work outside of these hours. Requested time off however, is a soft parameter because it cannot be guaranteed. For example, if all employees request the same day off, but the store requires that at least one employee be present, then some employee's request cannot be fulfilled.

Once the parameters are in place, the second step is to actually develop the schedule. There are many approaches to do this manually: develop a schedule one employee at a time, or perhaps develop a schedule one day at a time. Regardless of the method, it's important to note that conflicts will arise and revisions must be made; it's an iterative process. It's also worth noting that the resulting schedule may contain errors (e.g. scheduling an employee during their vacation).

The third – and perhaps most important step – is the maintenance of the schedule. Of course plans change, and employees may request time off after a schedule has been posted. Unfortunately, a single change might cascade into further conflicts, requiring more time and effort from the scheduler. Likewise, changing an already posted schedule requires that the manager notifies all employees involved so that they are up-to-date with the latest schedule.

All in all there is a needless amount of overhead for an intuitively simple task.

Our aim is to create a software scheduling solution that overcomes the main downfalls mentioned above. EasyShift will allow all employees to operate on a shared schedule with position-based access control. We also plan to have future implementations include a feature where schedules can be automatically generated based on given parameters.

EasyShift will address the parameter organization problem by implementing transparent communication. With manual scheduling, managers might forget about an employee's requested time off. With EasyShift, all parameters are visible on the schedule (e.g. requested time off appears greyed out on the timeline), meaning that the manager isn't tasked with manually maintaining all of the latest parameters. Ultimately, this reduces errors when making schedules.

The issue of schedule generation can be solved through the use of scheduling algorithms. Such algorithms have been implemented in operating systems for over 20 years, and although they can't automatically create schedules, they can surely be modified to carry-out this task. This means managers do not have to spend as much time making schedules, therefore reducing company costs.

Finally EasyShift aims to minimize the difficulty of schedule maintenance. It's clear that revising digital data is much easier than its physical counterpart. On top of this, changes to the schedule could also trigger notifications to all employees involved, meaning that managers are not required to manually contact all employees after a change is made. This ensures that employees are always up-to-date with the latest version of the schedule.

Additional efforts by authors

In addition to concepts covered in the course, we researched and implemented the following to produce the web application.

- Database implementation and connection
- Error handling
- Cascading style sheet
- Container management
- UI Components
  - DatePicker
  - Timeline
  - Card
  - Dialog
  - TabMenu
  - Ajax - partial rendering and processing

Website design and workflow

Home page (landing page)

The home page is the first page the user sees. There isn't any particularity interesting functionality here. Rather it acts as a simple entrance to the website whereby users can either login or register. The UI is consistent with the application's color theme.

The only real functionality on this page is the navigation. Users can navigate to the login and register pages by either using the home page buttons, or the PrimeFaces' `<p:tabMenu>` bar at the top. It's worth noting that these three pages share xhtml code for this component code through the `<ui:include>` tag.

Login page

The login page is responsible for sending users to the core scheduling functionality. The UI components on this page are pretty simple, consisting mainly of `commandButton`s, `inputText`s, and `password`s. The input fields contain floating placeholders that are provided from PrimeFaces by using the class `ui-float-label`. Basically the placeholder (e.g. "Username") is located inside the component until the user interacts with it, where it then *floats* above the component.

The functionality from this page uses database queries for user verification. When the user clicks the login button, the system will either verify them and navigate to the dashboard page, or show them an error and keep them on the login page. The only successful verification case is when the username exists in the database, and the password matches the one in the database. If either of these conditions fails (including the user leaving the fields blank), then the user is kept on the login page and an error message is shown (e.g. "Incorrect Password").

Registration page

The registration page is similar to the login page with a bit of added complexity. On this page, users are required to enter their email, username, and a password (with a confirmation copy).

A registration attempt is successful when all of the following conditions hold: the username doesn't already exist in the database; the email doesn't already exist in the database; both passwords match; the email

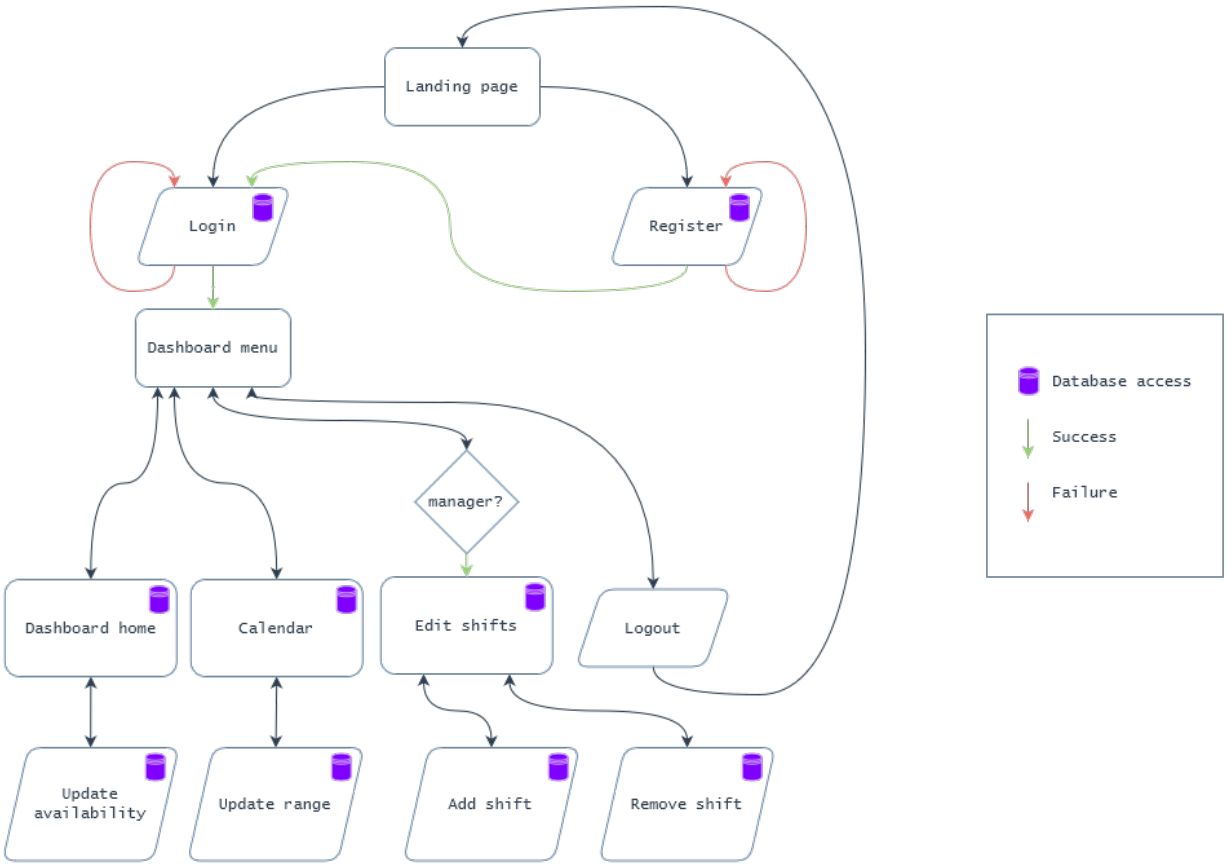

Figure 1: Implemented workflow

address is valid (i.e. regex that verifies a name and domain, with an '@' symbol as a partitioning character).

When a registration is successful the user is navigated to the login page and their credentials are added to the database. Similar to the login page, an invalid registration results in the user staying on the page with an error message.

Dashboard-menu

Dashboard home

Dashboard calendar

Dashboard edit-events

Logout

Styling

To ensure consistent design patterns and colors throughout the web pages, we defined a style sheet (see `project/src/main/webapp/styles.css`). This tool significantly aided us in improving the visual appeal of the application and setting parameters to distinguish calendar events, i.e., available and scheduled events.

Database

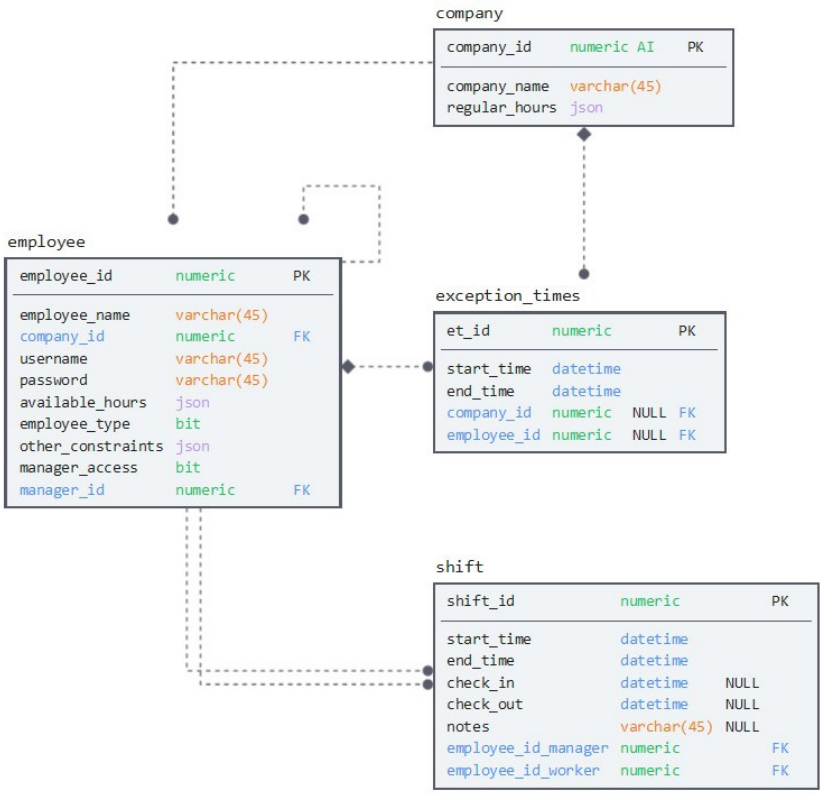

Figure 2: Database schema

Future steps

Our approach to building EasyShift is to start with the core functionality and expand outward from there. Technically speaking, the timeline is the basis for our project; every feature aims to enhance its serviceability.

For core functionality, we have implemented a timeline that is intractable: managers can edit, remove and add shifts, and users can view and edit their respective availability. We also have user verification and registration, that is, users can create and access their own scheduling information..

For next steps we plan to implement the ability to users to create and edit companies. This would allow managers of a store (e.g. Sobey's bakery) to create a schedule for their department. Ideally this means they could choose which employees could access the schedule.

Later on, we believe a parameter based schedule generator is a great feature for a lot of companies. This would allow managers to create schedules with a click of a button. This feature could also include some sort of metric that shows the number of parameters that could not be satisfied, as well as generate multiple schedules and let the manager pick the best one.

Some other ideal features include a payroll system that tracks actual worked hours; a notification system that notifies all employees of changes that affects them; and a remote database so all employees are operating on the same instance.