The Cafe23 Management System

Cafe23 is the brainchild of Alex and Mark and currently consists of seven stores that sell premium hot beverages and baked goods in the Greater Pittsburgh area as well as one in Morgantown, WV and two in Youngstown, OH. (Of course, Alex and Mark want to expand the number of stores in their franchise as circumstances allow.). These entrepreneurs know that a key to successfully managing these stores will be their ability to track key data for each store and help manage employees and payrolls for the cafe. They want a web-based application to help them in this regard and have recruited you to build this application due to your remarkable skills and low rates (i.e., pro bono). They provide you with the application specifications listed below.

The first thing the application must do is give an administrator (i.e., Alex and Mark) the ability to log into the application and create new stores or edit information about existing ones. Currently, the entrepreneurs are content to simply track the store's location – both its full address (street, city, state and zip code) and its phone number. Of course, each store does have a name that is used to reference it internally; for example, the cafe located in the Carnegie Mellon University Center is usually referenced as the "CMU" shop. Tracking that name in the database is also important.

The cafe employs a number of people at each store and while people can be transferred from store to store, at any one time each person only has one assignment. There are two ways an employee gets an assignment: the first is to be transferred to a new store and the second is to move up to a different pay grade. Even if the employee remains at the same store, if he/she moves up to a higher pay grade, the employee is given a new assignment. When an employee is given a new assignment, the start date is recorded for the new assignment and the end date is recorded for the previous assignment. The employee's current assignment can be found by finding the assignment for that person that has no end date.

An administrator can create employees and assignments. Once an employee is assigned to a store, he/she can be given work shifts at this store and the store manager can record those shifts in the system. A standard shift lasts 3 hours, but it can go longer; since the time actually worked is needed for payroll, store managers or employees must record the start and end times of each employee's shift. In addition to the information necessary to process payroll, Alex and Mark want to have a record of the job or jobs the employee performed on that shift (e.g., making drinks, baking treats, mopping floors, cash register work). Right now, there are a limited number of jobs an employee can do on a particular shift, but the boys expect that as the cafe grows that the number of jobs will grow as well and want to make sure the system can handle this growth.

When a manager creates a shift for an employee, he/she supervises, the shift is initially marked as 'pending'. When the employee "clocks in", their shift begins and is marked 'started'. At the conclusion of the shift, when the employee "clocks out", then the shift changes to 'finished'. When shifts are pending, the times given are not considered actual times, but when a shift changes to 'started', the actual start time is recorded; likewise, when the shift is 'finished' the actual end time is recorded.

As for employees themselves, the system would need to have at minimum the employee's name, Social Security Number, date of birth (the state of Pennsylvania has special work rules for any employee under 18), and phone number. The SSN and phone number data should be stripped of non-digit characters in the database but should be presented to the user in a human-friendly format with appropriate dashes. If an employee logs into the system, they should be able to see not only a summary of their personal data[1], but they can see a log of the shifts that they have worked, details for each shift (i.e., what jobs they worked) as well as a list of upcoming shifts they are scheduled to work in the next two weeks.

It should be noted that there are three types of "employees" at Cafe23. The first are administrators – for now this title is limited to Alex and Mark, although they intend to add more administrators as the franchise grows and overtakes Starbucks as the dominant beverage retail chain. Administrators have unlimited power in the system to make needed changes. The second type of employee is the store manager. Managers cannot create new employees, but they can edit employee information that might be in error. They can also create and record shift information. All administrators and managers must be users and can log into the system with their email address and a password. The final type of employee we will call an "employee" – these employees do the day-to-day work at the cafe: barista duty making drinks, baking treats to sell to customers, mopping floors, cleaning tables, running the cash register, and the like. Of course, all users can reset their passwords if they are forgotten.[2]

In terms of managers, they can add new, upcoming shifts to an employee that works at the store the manager is currently assigned to. In addition, after a shift is completed, managers are expected to record a list of jobs that the employee worked during the shift.

As for employees, the most important thing they can do is “punch the clock”; that is to say, if they are logged in, they can mark when they started their shift as well as mark when they ended the shift. The actual times may not (and often are not) the same as the expected times that managers enter for the shift initially. For instance, a manager might have had an employee marked for working today from 1-4pm, but employee could have been tardy getting to work and not starting until 1:10pm and may be forced to stay until 6:58pm due to the fact that a fellow employee is sick and needs to have someone cover his/her shift. Before the shift begins, the start/end times displayed are the expected times the manager enters, but for payroll purposes, it is more important to have the actual start and finish times.

Regarding payroll, an important aspect of the system is to help with managing payroll for each of the stores. Keeping track of accurate records is the first step, of course, but it would be nice if the system could calculate the pay due each worker at each store, for a specified period of time. One thing that complicates matters is the requirement that for payroll purposes, shift starts need to be rounded down to the nearest 15-minute interval while end times need to be rounded up to the next 15-minute interval. For example, if someone starts at 10:12am, the start time must adjusted to 10:00am and if the shift ends at 2:47pm, then the time rounds to 3:00pm. In the database Alex and Mark want to track the actual times the shift starts and ends, but when calculating payroll, they need to round the times to meet requirements. Thankfully, their dad has already created a gem for you called `time_date_helpers` that will be helpful with the rounding issues and is available at GitHub at

[1] Although for security purposes, it should be noted that employees can only edit their password and phone number – any other changes must reported to an administrator or manager for correction.
[2] Passwords must be reset rather than resent because all passwords must be encrypted in the database.

https://github.com/profh/time_date_helpers. What you will need to focus on is building the functionality to calculate the pay amounts based on the time worked and the current pay grade (and pay rate currently associated with that grade) of each employee.

In terms of calculating payrolls, it is important that an administrator can generate the payroll record for all the employees at a given store for a given time period. For now, we will assume if the system can print out a written record of each employee's earning during that period, that it can be transferred to a fully working payroll manager.

When each type of user logs in, that person should see information relevant to him/her and have commonly used functions readily at hand. For example, when a regular employee logs in, he/she should see an option to clock in or clock out, depending on the shift status, if they have a shift on that day. In addition, an employee would want to see a list of upcoming and recently completed shifts. They would also want easy access to their latest payroll report to see what they've earned in the past pay period. When managers and administrators log in, they would also want an appropriate dashboard given their task, but the exact design of these dashboards will be left to you. As a final note, prior to logging in, any user or guest who comes to the site will simply see a welcome and a list of stores in the Cafe23 franchise.

Given there are so many employees spread out among many stores, there does need to be an easy way for administrators and managers to search and find employees. One key difference is that manager searches would be constrained to those employees under their supervision, whereas administrator searches would be global. In addition, if a search only returns one record, then the system should just display that record, whereas if there are multiple records matching the search query parameters, a list of results should be returned, and the user can then choose the most appropriate record to view.

As for destroying records, Alex and Mark want to maintain good historical records, so rather than delete a record, they would prefer to mark employees, stores, and jobs as inactive. Inactive records would not appear in dropdown lists for making new assignments or shifts, but still be accessible in the database if an audit were needed. Shifts can be deleted, but only if they are pending and assignments can be deleted, but only if there are no started or finished shifts associated with the assignment.

The application needs to be built with Ruby on Rails, version 7.0.4, and using Ruby version 3.1.2. Alex and Mark have heard from their dad the importance of unit testing and all models must have 100 percent unit test coverage. There will be other tests, such as integration tests using cucumber, that will be provided later in the semester, and it is expected that the application should be able to pass all tests prior to being deployed at heroku.com or a similar service.

Additional requirements will be given in the second half of the semester.