ReShift

By

Alex Romero

Christian Mata

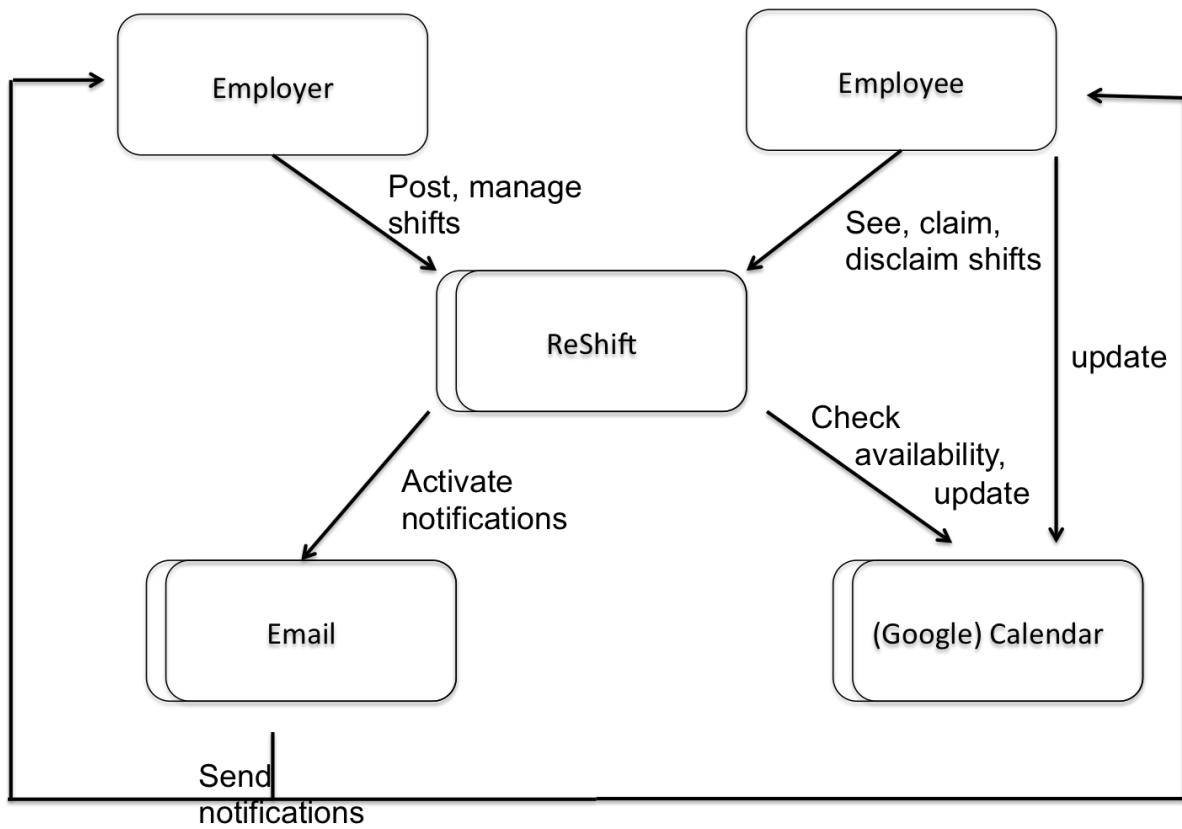## Sheldon Trotman

I. Motivation:

ReShift is a web app that allows various campus jobs to efficiently handle changes in shift availability. The app creates a workplace that have employees and managers. Each employee has his/her usual shift, but if the employee cannot make his/her shift because of a meeting or class, they can offer the shift up for fellow employees to pick up and work. The web app makes it more efficient to offer and pick up hours. It supports creation of workplaces, offering and picking up of shifts, notifications of available shifts and calendar integration.

Purposes

- Ease the administration's job for posting and managing shifts

- Enable reminders for shifts that a user has signed up for as well as notifications of upcoming non-claimed shifts

- Facilitate organization and updates of a pool of employees for a first come-first serve, shifts-based job

- Ease individual's time administration and payroll processes when working at a shift-based job

II. Context Diagram:

Context Diagram

```
   ┌──────────────┐                    ┌──────────────┐
   │   Employer   │                    │   Employee   │
   └──────────────┘                    └──────────────┘
         │                                    │      │
         │ Post, manage                       │ See, claim,
         │ shifts                             │ disclaim shifts    update
         ▼                                    ▼      │
      ┌──────────────┐                               │
      │   ReShift    │                               │
      └──────────────┘                               │
         │                  Check                    │
         │ Activate         availability,            │
         │ notifications    update                   │
         ▼                         ▼                 ▼
   ┌──────────────┐          ┌──────────────────────┐
   │    Email     │          │  (Google) Calendar   │
   └──────────────┘          └──────────────────────┘
         │
      Send
      notifications
```

III. Concepts:

- **Personal accounts (employer/employee)** – organization of both the people posting the available shifts and people claiming them; tracking of individual worked hours

- **Email notifications** – reminders for employees and notification of non-claimed shifts

- **Workplace** – Forum with a set of users and shifts to be claimed with real time updates

- **Calendar** – individual's time administration

- **Shift** – This is the set start time and duration for a slot of time an employee can work.

- **Posting** – Users that cannot work the specified shift that they are supposed to have the option to "post" the shift. This notifies all other employees that the shift is now available to be covered by someone else.

- **Covering** – In contrast to "post", "covering" allows the user to claim the given shift. Users can claim as many shifts as are available.

IV. Data Model:

Data Model
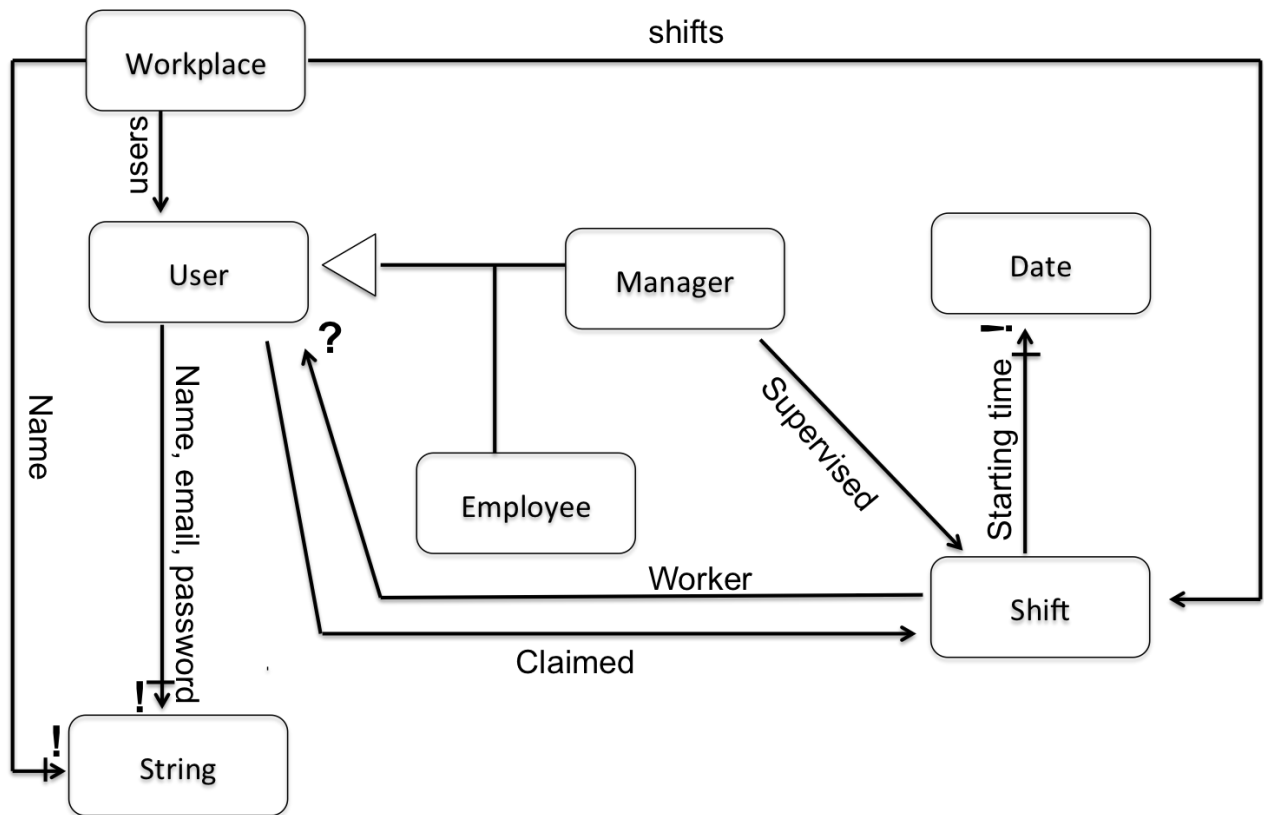
Additional constraints:
- Shift will have a starting time & a duration integer value

V. Design Challenges:

How to store shift durations? In the Shift object, the object needs to keep track of the amount of time that must be worked. In this, the Shift object has a few different options that can be used.

- *Starting and ending time stamps --* Consistent data type. Native data type operations are implemented so computing difference in time should be trivial.

- *Starting time and duration as a number --* This would use a time string as the starting time and then have a variable duration value. The variable duration value would allow the user to work for portions of hours (e.g. 15 minutes) and be able to save the point values quickly. Also this would lend itself well to computing the total hours worked and amount earned if desired.

- *Creating a new object "Time slot" with a fixed unit of time --* The idea here would be to create new objects that would store a fixed unit of time. This would allowed the Time Slot object to be given to each shift as a field and then stored and accessed later.

Are managers allowed to be workers? For situations, the manager may be considered a manager or a manager may have to work a shift.

Potential Solutions:

- *Allow managers to have fields of workers* -- Build the data model to include the worker fields and easily allow the manager to claim shifts if desired.

- *Force manager to create a worker account* -- Simplifies the logic for the manager model.

Are workers allowed to post a shift that they said they'd cover? This is the situation that a worker said that they would cover a shift for another worker, but later determined that they would not be able to cover it.

Potential Solutions:

- *Treat the shift as before* -- Simply allow the shift to be able to go to posts as before. This would allow the app to have flexibility in handling shifts until someone can pick up the hours.

Force the shift coverer to keep it -- This would be a way to maintain the integrity of covering. Rather than being able to post a shift you said you'd cover, the app would enforce it and whatever punishment would go to the coverer.