Purposes:

1. Allow employers to assign work hours for employees.
2. Inform employees of when they (and others) are working.
3. Allow employees to exchange working hours amongst each other.
4. (Potentially) Encourage employees to be present for scheduled hours.
5. (Potentially) Enable employees to declare when they are willing to work.
6. (Potentially) Enable multiple employers to collaborate on the same schedule.

Concepts:

1. Shift (open shifts)
2. Position (“job” or “role”)
3. Schedule (“week”)
4. Offer (relationship with claim: Trade)
5. Claim
6. Employee
7. Employer
8. Availability
9. Reminder

Design Challenge:

- Employer as a subset of employee -- differentiation; is it just a permission; are employers employees?
- Employer power vs. Employee schedule security -- can my shifts change at the last minute
- Not enough people for a given shift -- or a person tries to trade a shift but isn’t successful
- How many companies can an employee work for -- only one; implementation would be too difficult and deciding what schedule to show
- Delete future shifts. Delete linked (by parent) shifts rather than just deleting shifts that occur at that time. This is like the iCal behavior.
- How to represent time separate from date - object? number? schema?

Risks

1. black market trading
    a. system is inaccurate
    b. employer reassignment
    c. simplify process A LOT <better UI>
2. high initial time commitment barrier to getting started
    a. employer creation, employee initiation should be VERY easy
    b. possibly have a lot

Optional Extra Features

1. Google Calendar Export
2. Mark whenever a person doesn’t show up
3. Remind employees of their shifts via automated emails

Minute Crucial Details:

If you have some idea related to this in the middle of the night -- write it down!

1. Time stamps on everything!
2. Ensure traded shifts have not already passed.
3. Stop people from claiming more than 40 hours (or whatever limit you decide)
4. Assignment vs. free-for-all claim
5. Writing locks the collecting. Can partially-written data be read.
6. Easy to make a lot of recurring or one-time events (this is a preference or different use case).