

ReShift

By

Alex Romero

Christian Mata

Sheldon Trotman

Project 3 Phase 2
Group Design Document

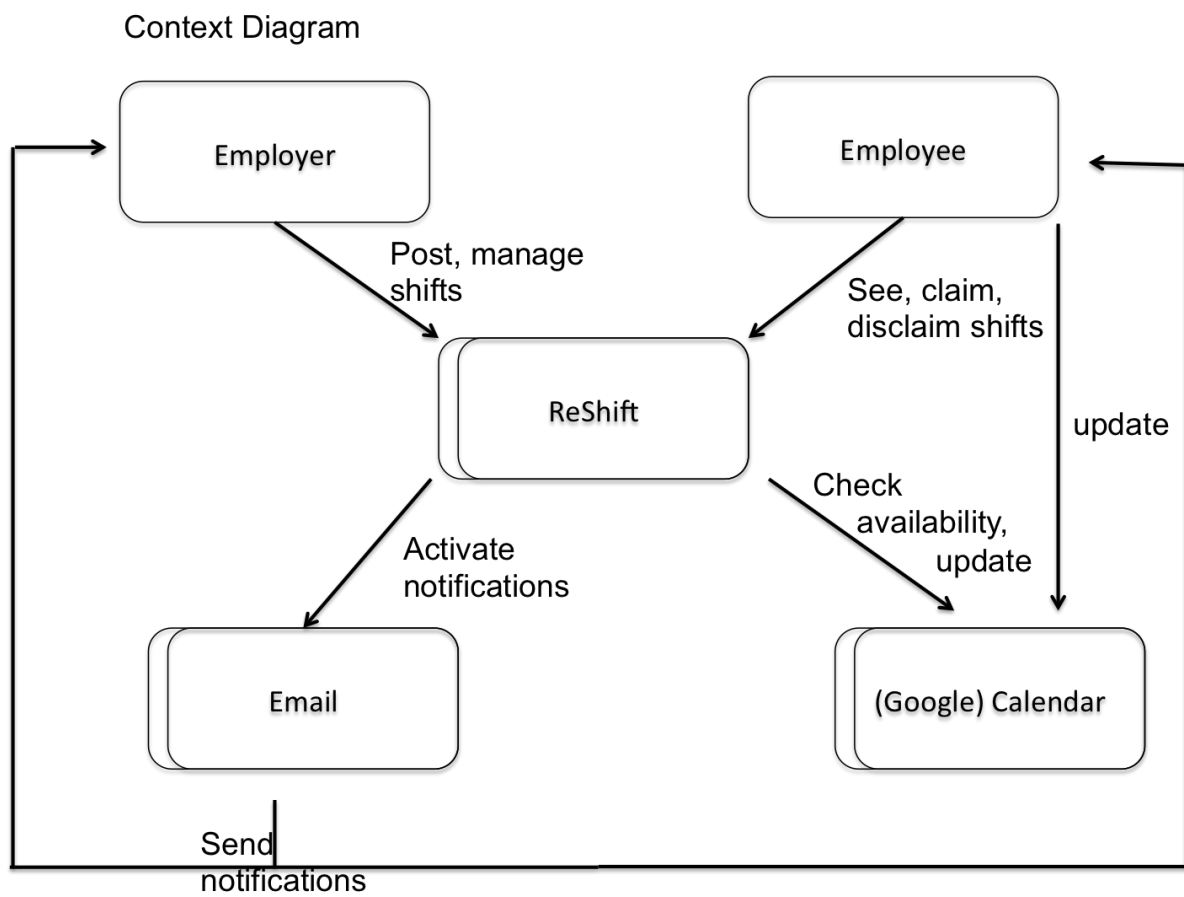
I. Motivation:

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

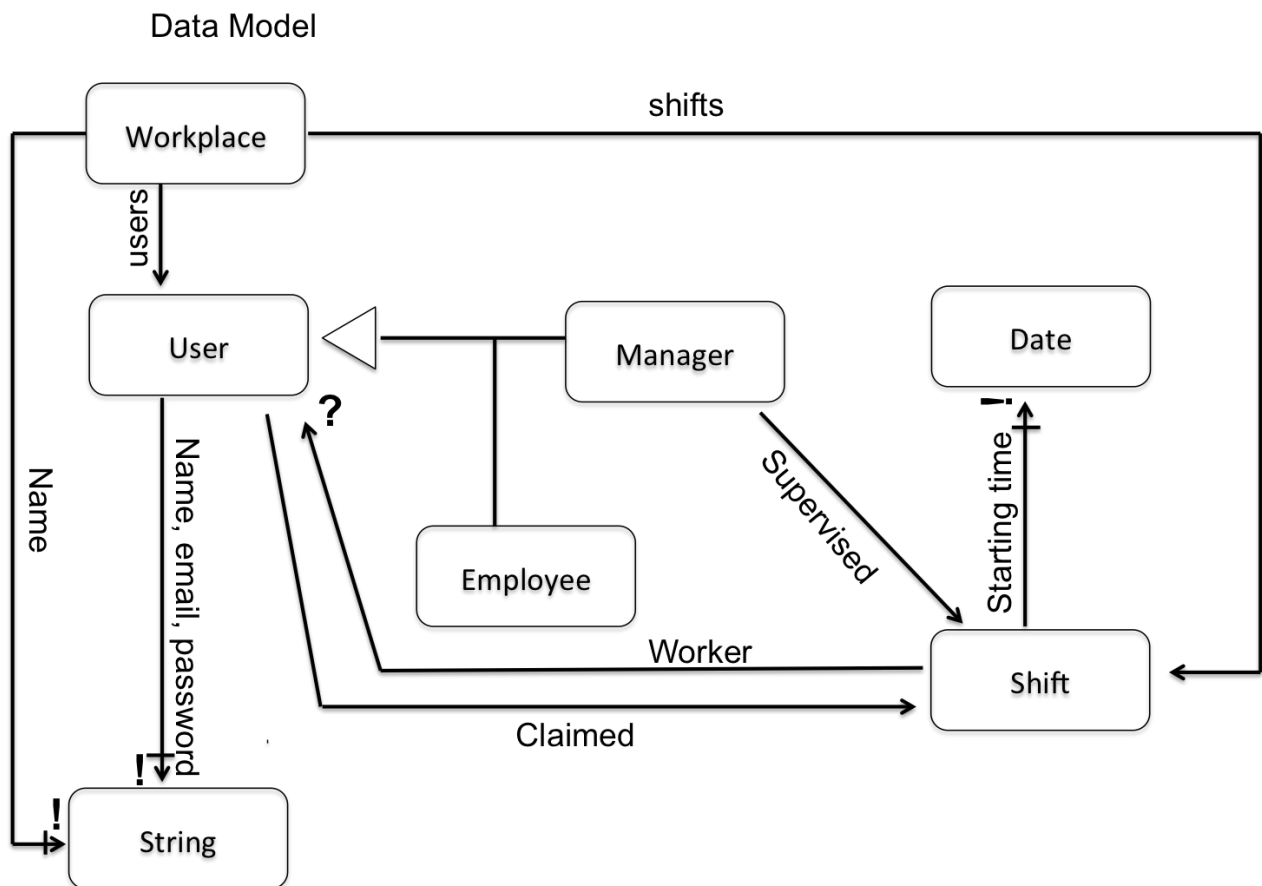
II. Context Diagram:



III. Concepts:

- **Manager and 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.

IV. Data Model:



Additional constraints:

- Shift will have a starting time & a duration integer value

** This has changed please refer to the data design.

V. Design Challenges:

Testing:

What is the preferred testing method? As required for 3.2, we needed to automate some queries to our API with tests.

Potential Solutions:

— Use QUnit, jQuery & asserts to automate the testing and handle the test running. This is the recommended method shown in the example API documentation.

— Create a script that simply used AJAX calls and compare the expected to the results. Instead of including QUnit and having to go through the headache of CORS issues, we could host the script and execute it when a view is shown.

Essentially, we would have to make our own testrunner.

Solution: We chose to go with QUnit. Although it had considerable headaches and pains including it into our code, the `asyncTest` functionality allowed us to work with AJAX calls that return asynchronously.

Employee, User, Manager:

How to relate User, Manager and Employees? While implementing the API, we ran into issues of what would be the optimal way to organize the User, Manager and Employee models.

— User as superclass of Manager and Employee with Mongoose Extends. Here we would let each model inherit from the User and take all its fields, which include the email and password.

— Create a pointer from the Manager & Employee model instance to the User.

Rather than extending, we'd create a field in each of the Manager and Employee schemas that would have an ObjectId for the User that it was referring to.

Solution: We chose to use the pointer/reference idea. This is because it created a degree of separation between the email and password and Employee instances.

We figured it would be more secure to only have Employee fields passed around.

The extends functionality simply did not do enough to keep the two separated.

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:

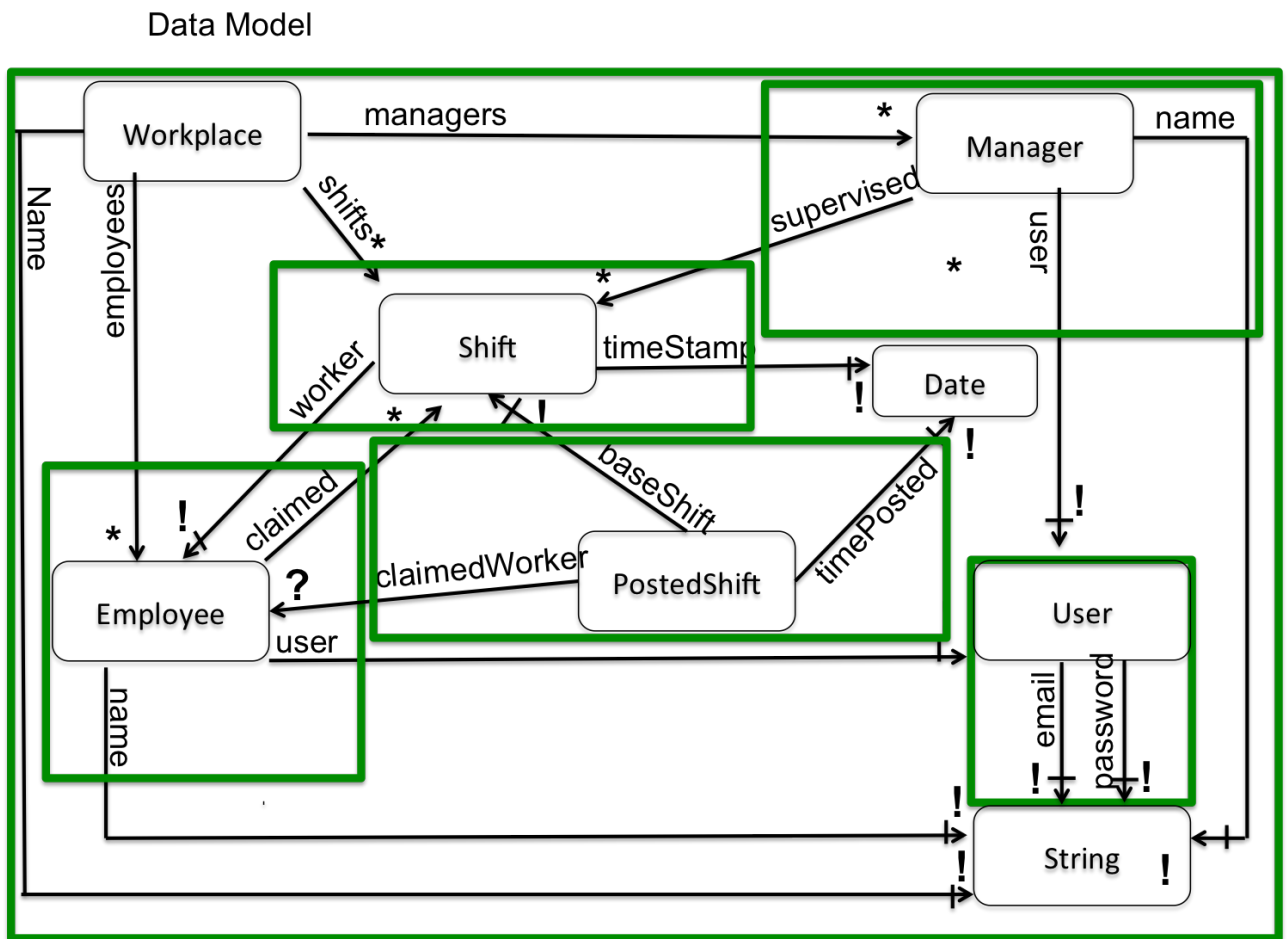
- *Allow managers to have fields of workers* -- Build the data model to include the worker fields and easily allow the manager to claim shifts if desired.
- *Force manager to create a worker account* -- Simplifies the logic for the manager model.

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.

VI. Data Model:



From our previous data model our main changes are the following (plus justification):

****Note:** Most justifications of changes in design are included in design challenges

- Workplace has two lists: employees and managers

- Ease of access to each kind of user
 - Separates types of users for authorization
- Employee and Manager are now independent objects, not subclasses, that refer to a user.
 - Avoided passing around the password of each user
 - Separated properly the actions for each one
- Employee has a list of claimed shifts
 - Keep track of the amount of hours worked
- Shifts have a fixed worker
 - Shifts are meant to be assigned to a specific worker for a long duration of time. This does not change. The user however, is allowed to post it in case it cannot do his regular shift in a given week.
- Posted Shift is a new class that points to its original shift
 - Enables claiming and disclaiming of shifts without changing the original owner of the shift
 - Mutable type that points to an immutable type
 - Allows us to keep track of when the shift was posted and who claimed it so that it can return to its default worker

VII. API Specificaion:

Models:

- users
- managers
- workplaces
- shifts

Schemas:

- User
 - name : String
 - email : String
 - password : String
- Employee
 - user : {type: ObjectId, ref: users}]
 - name : String
 - claimed : Array[{type : ObjectId, ref: shifts}]

- Managers
 - user : {type: ObjectId, ref: users}
 - name : String
 - supervised : Array[{type : ObjectId, ref: shifts}] **changed shifts -> workplace
- Workplaces
 - name : String
 - users : [{type : ObjectId, ref: users}]
 - shifts : [{type : ObjectId, ref: shifts}]
- Shifts
 - timestamp : Date
 - duration : Number
 - worker : {type : ObjectId, ref : users}
- PostedShifts
 - baseShift : {type : ObjectId, ref : shifts}
 - claimed : {type : Boolean, default : false}
 - claimedWorker : {type : ObjectId, ref : users}
 - timePosted : Date // date of the time the user wants to post

Response Format:

Description		
This is the response model for all requests		
Model		
Key	Format	Description
status	"ok" "error"	Indicates if the request was successful
response	String	
	Error specification	
	Key	Format
	data	Mixed
		Object with all generated values or null if method is DELETE

Workplaces

Workplace Creation		
Create a new Workplace		
Method	POST	
Route	/workplaces	
Response Content Format	Workplace object	
Parameters		
Key	Required	Description
name	TRUE	Username of this user
Errors		
Code	Response content	
500	An unknown error occurred	

Workplaces	
Get a list of all workplaces	
Method	GET
Route	/workplaces
Response Content Format	List of Workplace objects
Errors	
Code	Response content
404	Resource not found

Workplace	
Get a specific workplace	
Method	GET
Route	/workplaces/:id
Response Content Format	Workplace Object
Errors	
Code	Response content
404	Resource not found

Workplace Deletion	
Deletes a specific workplace	
Method	DELETE
Route	/workplaces/:id
Response Content Format	null
Errors	
Code	Response content
500	An unknown error occurred

Workplace's Workers Update - Employee	
Adds an employee to this workplace's workers	
Method	PUT
Route	/workplaces/:id/employees/:eid
Response Content Format	Workplace Object
Errors	
Code	Response content
404	Resource not found

Workplace's Workers Update - Manager	
Adds an employee to this workplace's workers	
Method	PUT
Route	/workplaces/:id/managers/:mid
Response Content Format	Workplace Object
Errors	
Code	Response content
404	Resource not found

Workplace's Shifts Update	
Adds a shift to this workplace's shift	
Method	PUT
Route	/workplaces/:id/shifts/:sid
Response Content Format	Workplace Object
Errors	
Code	Response content
404	Resource not found

Employees

Employee Creation		
Create a new Employee		
Method	POST	
Route	/employees	
Response Content Format	Employee object	
Parameters		
Key	Required	Description
email	TRUE	Unique email of this user
password	TRUE	Secret password of this user
name	TRUE	Username of this user
Errors		
Code	Response content	
500	An unknown error occurred	

Employees	
Get a list of all employees	
Method	GET
Route	/employees
Response Content Format	List of Employee objects
Errors	
Code	Response content
404	Resource not found

Employee	
Get a specific employee	
Method	GET
Route	/employees/:id
Response Content Format	Employee Object
Errors	
Code	Response content
404	Resource not found

Employee Deletion	
Deletes a specific employee	
Method	DELETE
Route	/employees/:id
Response Content Format	null
Errors	
Code	Response content
500	An unknown error occurred

Employee's Claimed Shifts Update	
Deletes a specific employee	
Method	PUT
Route	/employees/:id/shifts/:id
Response Content Format	Employee Object
Errors	
Code	Response content
404	Resource not found

Managers

Manager Creation		
Create a new Manager		
Method	POST	
Route	/managers	
Response Content Format	Manager object	
Parameters		
Key	Required	Description
email	TRUE	Unique email of this user
password	TRUE	Secret password of this user
name	TRUE	Username of this user
Errors		
Code	Response content	
500	An unknown error occurred	

Managers	
Get a list of all managers	
Method	GET
Route	/managers
Response Content Format	List of Manager objects
Errors	
Code	Response content
404	Resource not found

Manager	
Get a specific manager	
Method	GET
Route	/managers/:id
Response Content Format	Manager Object
Errors	
Code	Response content
404	Resource not found

Manager Deletion	
Deletes a specific manager	
Method	DELETE
Route	/managers/:id
Response Content Format	null
Errors	
Code	Response content
500	An unknown error occurred

Manager's Supervised Shifts Update	
Adds a shift to this manager's supervised shifts	
Method	PUT
Route	/managers/:id/shifts/:supervised
Response Content Format	Manager Object
Errors	
Code	Response content
404	Resource not found

Shifts / Posted Shifts

Shift Creation		
Create a new Shift		
Method	POST	
Route	/shifts	
Response Content Format	Shift Object	
Parameters		
Key	Required	Description
timestamp	TRUE	Starting time of the shift
duration	TRUE	Duration in minutes of the shift
worker	TRUE	Employee that this shift belongs to
Errors		
Code	Response content	
500	An unknown error occurred	

Shifts	
Get a list of all shifts	
Method	GET
Route	/shifts
Response Content Format	List of Shift objects
Errors	
Code	Response content
404	Resource not found

Shift	
Get a specific shift	
Method	GET
Route	/shifts/:id
Response Content Format	Shift Object
Errors	
Code	Response content
404	Resource not found

Shift Deletion	
Deletes a specific shift	
Method	DELETE
Route	/shifts/:id
Response Content Format	null
Errors	
Code	Response content
500	An unknown error occurred

Posted Shift Claimed Update	
Changes the shift to being available for claiming	
Method	PUT
Route	/shifts/posted/:id/disclaim
Response Content Format	Posted Shift Object
Errors	
Code	Response content
404	Resource not found

Posted Shift Creation		
Create a new Posted Shift		
Method	POST	
Route	/shifts/:id/disclaim	
Response Content Format	Posted Shift Object	
Parameters		
Key	Required	Description
baseShift	TRUE	Original shift that is being posted
claimedWorker	FALSE	Who the shift belongs to in the moment (defaults to whoever posted the shift)
timePosted	TRUE	Date when the shift was posted
claimed	FALSE	Indicates if the shift is available to be claimed (defaults to TRUE)
Errors		
Code	Response content	
500	An unknown error occurred	

Posted Shifts	
Get a list of all posted shifts	
Method	GET
Route	/shifts/posted
Response Content Format	List of Posetd Shift objects
Errors	
Code	Response content
404	Resource not found

Posted Shift	
Get a specific posted shift	
Method	GET
Route	/shifts/posted/:id
Response Content Format	Posted Shift Object
Errors	
Code	Response content
404	Resource not found

Posted Shift Deletion	
Deletes a specific shift	
Method	DELETE
Route	/shifts/posted/:id
Response Content Format	null
Errors	
Code	Response content
500	An unknown error occurred

Posted Shift's Claimed Worker Update		
Changes the claimed worker of this Posted Shift		
Method	PUT	
Route	/shifts/posted/:id/claim	
Response Content Format	Posted Shift Object	
Parameters		
Key	Required	Description
worker	TRUE	User that is claiming the shift
Errors		
Code	Response content	
404	Resource not found	