**ECE 458 Resource Manager: Evolution 3**
**By Davis Gossage, Emmanuel Shiferaw, Sam Ginsburg and Allan Kiplagat**

## Retrospective Analysis

In previous evolution write-ups, we discussed the packaging system as one of the main benefits for choosing Meteor, and it has continued to be helpful. One of the most important aspects of our design so far has been the use of the alanning:roles package, which simply maintains a key-value store of Users to sets of 'permission' strings that can be used for authentication. We used this to implement the viewing and reservation permissions requirements in evolution 2, and continued to make use of it during this evolution.

Another aspect we previously discussed was Meteor's two-way data binding and the simplicity that offered us, both in speed of development and amount of code to comprehend/maintain. During this evolution, we did encounter a few cases where the code we wrote didn't quite fit with the Meteor 'publish-subscribe' model of client-server communication. This happened for various reasons, one of which was our use of publicly exposed methods for the API requirement in evolution 2 - at times, we would write a method that ran on the server and returned the correct results for the client, only to realize that it would not be "Reactive" in the UI. Fixing these problems, however, was not difficult, as it usually just meant moving the server method functionality into a client-side 'reactive query'.

## Analysis of Current Design

"In the large", our design has changed very little from evolution 2 to 3. For the 'compound reservations' feature, the 'resource_id' field in the Reservations schema simply changed from a single string to an array. This required some changes in our server methods and client-side code, but significantly less than there would have been if we were using a relational database, which would have required adding another table to keep the data normalized, and more complex logic in mapping information in those tables to our 'resource' and 'reservation' javascript objects. The calendar page is now reached by selecting one or more resources and then clicking 'Create Reservation'. This takes the user to the calendar page which shows reservations associated with any combination of resources that the user selected. For example, asking to see reservations associated with resources A and B would show any reservations involving resource A, resource B, or both. The calendar shows reservations you own in blue, others in green, and incomplete reservations as semi-transparent.

For restricted resources and oversubscriptions, there were similar additions to the schema. The most important one was the addition of a new 'permission' field (Roles package) on reservations that describes who has the ability to approve reservations on a restricted resource. Although the evolution 3 requirements stated that a single user or more (per resource) designated as a 'manager' should be able to approve reservations on restricted

resources, we felt that it would fit our existing model better to allow for grouping of users under permission tags rather than explicitly choosing users for permissions, as we already had done for viewing and reserving. This requires the administrator/resource manager to give every resource that they wish to be restricted one or more ‘approve_permission’ string. The administrator/user manager must then add this same string tag to the permissions on every user that they wish to give approval capabilities on reservations holding said resource.

In our previous write-up, we stated “We are very dependent on the roles package and should there be new requirements that are unsupported, we will have to do a lot of work to customize the package or build our own”. This evolution did have some new requirements related to this package, but it turned out that it was flexible enough that we did not have to actually edit the package.

We also added a new page strictly for viewing reservations. It lists their details and allows for users to *extend* reservations by creating a new one starting at the previous reservation’s end time. It also allows for creation of arbitrarily long reservations, which wasn’t necessarily possible just from the calendar in previous evolutions.

**Contributions to this evolution:**
Davis: compound reservations
Emmanuel: restricted resources
Sam: approval/denying, new overall UI
Allan: edit reservations