Some ideas about a software design.

Terminology.

- "Owner" refers to a company or division running their own deployment of the Software, with data independent of any other deployments.
- "Developer" means an individual who is either a programmer with write access to the source code, or a database admin working for an Owner.
- "User" means an individual who has authenticated access to a deployment.
- "Staff" means a User who works for an Owner.
- "Client" refers to a User who is a customer of an Owner.

Requirements.

What features must the service exhibit?

- **Platforms**: Access to the service must be available via a variety of recent-to-modern, general business computing devices. (i.e. desktop, laptop, tablet and smartphones as of 2013).
- **Mobility**: Reasonable efforts at providing access to the service for all Users must be made, regardless of their physical location.
- **Secure**: All data must require authentication and authorization to access, and non-safe requests must come with sufficient confirmation that they were initiated by user intent.
- **Access Controls**: Resources and sets of resources must have variable authorization permissions.
- **Admin Powers**: Read and write authorization to individual resources and sets of resources must be separate.
- **Unbreakable**: The system must not be able to be put into a "broken" state by Users.
- **No Clobber**: A User should not be allowed to overwrite another user's changes unknowingly.
- **Off-line**: A User's scheduled events for the current day must be readable off-line.
- **Rapid Bootstrap**: New Staff must be able to get up to speed on the internal side of the system quickly.
- **Owner Agnostic**: Nothing specific to one Owner should be present in a new, clean deployment for another Owner.
- **Brandable**: An Owner must be able to configure the system to exhibit their branding and corporate specifics.
- **Safe Data**: A frequent and reliable back-up of all data must be easily scheduled.

Qualities.

What other features would we *like* the end product to exhibit?

- Fast developer turn-around for bug fixes.
- Fast prototyping for new features.
- Error-free implementation of new features.
- Regression-free changes.
- Attractive Client UI.
- Efficient Staff UI.
- Widest device and OS support reasonably achievable.
- All accountability/blame data (who did what, when?) should be involatile.
- All other data should be editable by a user with sufficient authorization, without needing a Developer.
- Data back-ups should be creatable on demand by a sufficiently authorized user.
- Data back-ups should be restorable on demand by a sufficiently authorized user.

Non-Qualities.

What features are *not important* and will allow flexibility of design?

- Programming language choice
- UI monolingualism
- We do not need a generic resource import or export system.