Issues (Framework, Shifter.Managers.Domain, Shifter.Managers.Web):

* Framework
  + Static Persistence Helper??
  + Email Manager needs an interface.
  + Review PasswordEncryptor to use Dependency Inversion or Double dispatch to remove coupling to specific implementations.
* Shifter.Managers.Domain
  + Why can’t we use AutoMapper with Fluent NHibernate?
  + Decouple ORM and Domain.
  + Domain Model lacks value Objects.
  + Extract Services into client API.
  + Implement WCF services for scaling.
  + Review Domain Model to identify context boundaries and Aggregate Roots. Domain model is very anaemic.
  + Remove magic strings i.e. within Shift.GetAssignedToHint().
  + Generalise services.
  + Too many services need to move logic into the domain objects. This will happen once aggregate roots are clearly defined.
  + Lack of null checking.
  + Too much logic in predicate builder… Look at generic alternative to match based on template object. I’m fairly certain Fluent NHibernate can do this.
  + Delete mocks in Domain!! We use mocking & stubbing frameworks therefore hard coded mocks server no purpose.
* Shifter.Persistence
  + No Transaction Management at a database level i.e. sharing transaction across a unit of work. Example: I save a waiter which passes, then save a manager and it fails, how do I rollback the waiter transaction given these two transactions happen in the same unit of work?
  + Validate method should be caching rules. These will not change after the application is initialised.
* Shifter.Managers.Web
  + Could we look at possibly using Spring.Net instead of Unity? Personally I prefer design-time configuration. Spring.Net also has many other utilities out-the-box which includes client proxy generation for WCF. This ensures loosely coupled dependencies. It provides functionality for MVC in terms of controller injection and NHibernate mappings. Download @ <http://www.springsource.com/download/community?project=Spring.NET> and checkout the sample projects when you have time.
  + In line strings need to be moved to resources so that we’re able to deal with localisation seamlessly.
  + Generalise Actions, there’s too much duplication.
  + We should extract common components from the two separate web applications (if they even need to be separate).

Unit Testing:

* We need a comprehensive test suite. The more guys who start developing on this, the higher the chance of us breaking code. On this note, with everyone’s permission I’d like to setup a CI build on TFS. The CI build will trigger every time you check in your code. It firsts compiles all projects then runs the relevant unit tests. Should either of these fail, the person responsible will receive an email and be required to fix the build (other punishment can be decided on at a later stage i.e. buy everyone else a beer ☺). At a later stage we can start getting fancy and start running code analysis on the build artefacts and look into continuous deployment.

Environments:

* We need to set up the relevant environments. Obviously the Dev environment will happen locally on our machines. We need to setup a UAT/Staging environment will need to be hosted. We need this environment to ensure integration between all our changes is still functional. As I said last time, I already have hosting space for this, we’d just need a host name i.e. staging.shifter.com. Do you guys perhaps have something like this registered?