# Login TA

## Data Layer

Using the Repository pattern and interfaces in the data layer gives a level of abstraction over the storage mechanisms.

The abstraction goes from

1. xml file in FakeXmlDbContext
2. database-like implementation in IFakeDbContext
3. Repository witch is using instance of IFakeDbContext in FakeRepository
4. Screen for any storage mechanism in IFakeRepository

The client has no knowledge about the storage mechanism that it is using and no access to the context.

## Cache

The caching is implemented with build in method in the storage context. Cache happens every time when the data is called (first the data is searched in the cache and only if it is not presenting there it is get from the storage). Following the requirements the data is loaded in the cache manually on Application\_Start. Also a CacheDependency to a file is added, to automatically remove the data from the cache if anything changes.

The idea behind the fact that objects are saved in the cache with the name of their type is that the caching is seen as an internal operation of the context and is trying to act like “saving a table into the cahce”, so there is impossible two collection of objects to be saved with the same name since every type is treated like one table in the database-like storage.

## Dependency Injection

For the purposes of easier testing the application uses dependency injection.

## Unit testing

A sample set of unit test is implemented. Its purpose is mostly to show how the application can be tested rather than fully test it for every possible scenario.

## Client site

The login and register actions use ajax for submitting the user data forms, the page reload occurs only on success – after redirection.

The validation is in custom.js and it is the same for both actions. On !Model.IsValid it accepts an object with properties that correspondent to the input fields in the validated form, and collection of errors for every one of the fields. Properties and fields relay on kind of a “Conventions over configurations” expecting that person who is using the function will have inputs with id-s the same as the properties from the object that is receiving from – server (typically the case if @Html.EditorForModel() is used).

The user information is available for everyone since no other requirements were given.