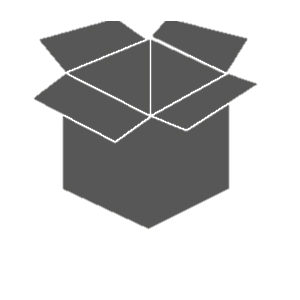
Doug



Audit System

Redevelopment Proposal

# Audit Records

## Redevelopment Proposal

|  |  |
| --- | --- |
| Who Uses It? | What they want |
| Small Companies | Setup Audits |
|  | Take audit |
|  | Produce report |
|  | Find where problems as highlighted by audit |
| Large Companies | Setup Audits |
|  | Get staff/other companies to fill out. On site possible |
|  | Produce report |
|  | Email/Print report |
|  | See Problems |

### Whats wrong at the moment.

* Pressing enter does strange things.
* Lack of help/support when building an audit. Makes very little sense as to why you are doing anything.
* Answer type field is confusing at best.
* The home page has far too many sections on it with little or no actual labeling as to what does what.
* Scoring is restricted to 'Yes'
* System too rigid, makes developing new parts hard and takes a lot of time.

### Proposed fixes.

* Separate the audit homepage into sections (Audit Control panel, Taken Audits, Usable Audits)
* Work out why pressing enter in a field doesn't control the closest button but does something else entirely, and fix it.
* Rework the answer type field to be more understandable. Make more pre-made question/answer types with predefined scores in an attempt to reduce the need for such a field. Using JavaScript to swap in and out each so the end user actually knows what is happening.
* The whole scoring system need's rethinking in order to make it more useful and understandable (I had to get Gareth to talk me through the current audit system and even he struggled to explain some aspects, especially how the scoring works, if we don't understand it non-tech's must really struggle).
* Completely separate Controller/Model (Back-end stuff) from the View (Front-end). So changing the look and feel is simply a case of modifying/adding a new template never touching any data/business logic. Making new looks easier to implement and far less likely to break anything.

### Proposed Enhancements

* Save as you type for questionnaires
* Search-able library of premade questions with score for creating own audits.
* Take audit without having to log in.
* System back-end needs making more generic so we can build “new” products simply by giving it a new front-end (Legionella, Gap analysis, KPI/Employee appraisals etc).
* Normalize data in order to cut down on repetitive actions for end user.
* Once base system has been worked on and in flawless. Fork off and develop it to a full Employee appraisals system.
* Write the entire system as a standalone API.

# How it should work.

### Setup Audit

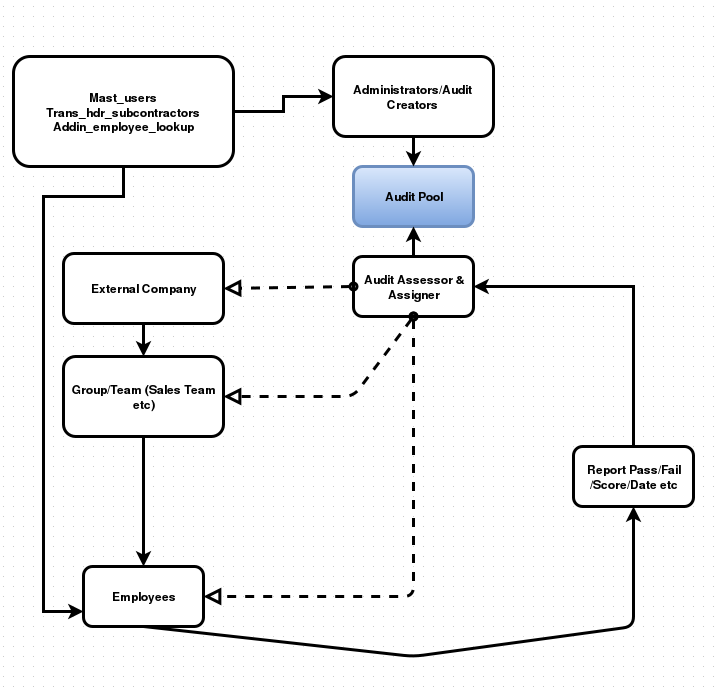
**Instructions** - Complete instructions, easy to use with. Make full use of JavaScript to create a nice flowing easy to use UI so the need for written instruction is less apparent, but provide as much instruction as needed. Use tool-tips where appropriate.

**Who Needs to Use It?** - In small businesses the main portal user will likely be the person who creates and answers audits. However in larger companies the chances are a group of people may create audits, another group may assign audits to certain people/groups and then the groups of people will need to answer the audits. Then returning the results back to the stage above.  
 **Emailing –** Audit results should be emailed to both the Audit taker and the Audit assessor. In PDF form, acting as an instant review. Also all audits should be save within the software it's self. The person who complete the audit should always be able to see all audit's they have taken. Audit assigners or admins should be able to see all audit's taken by everyone within their organization. Filterable by Group/Team/Individual/Location etc.

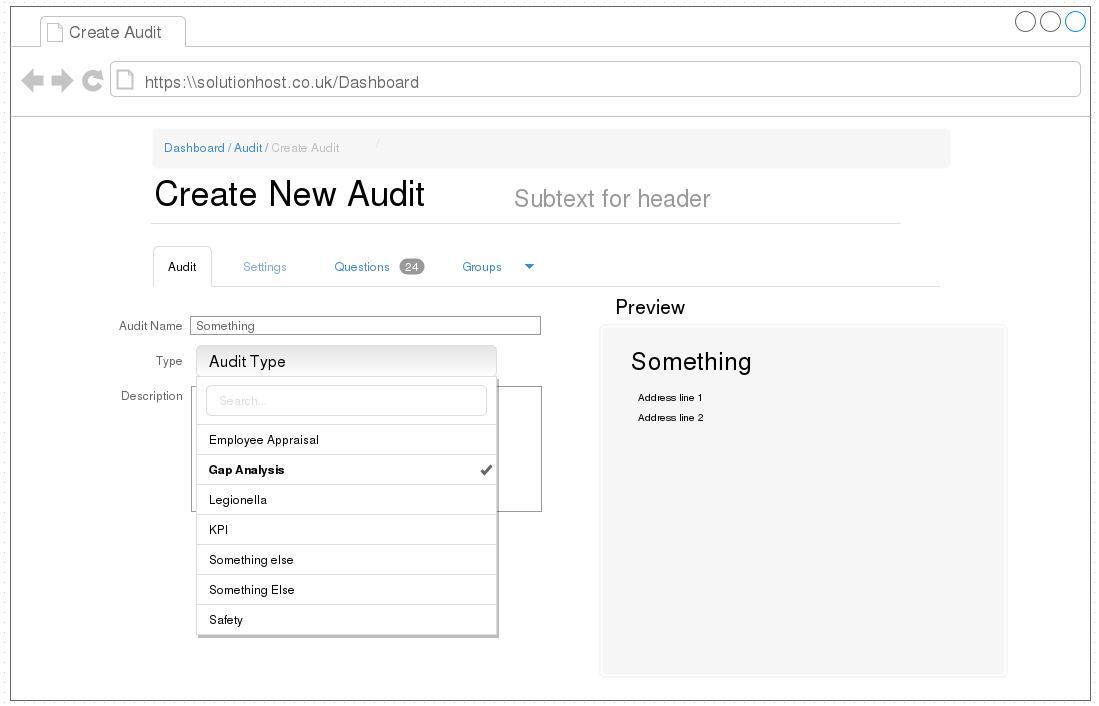
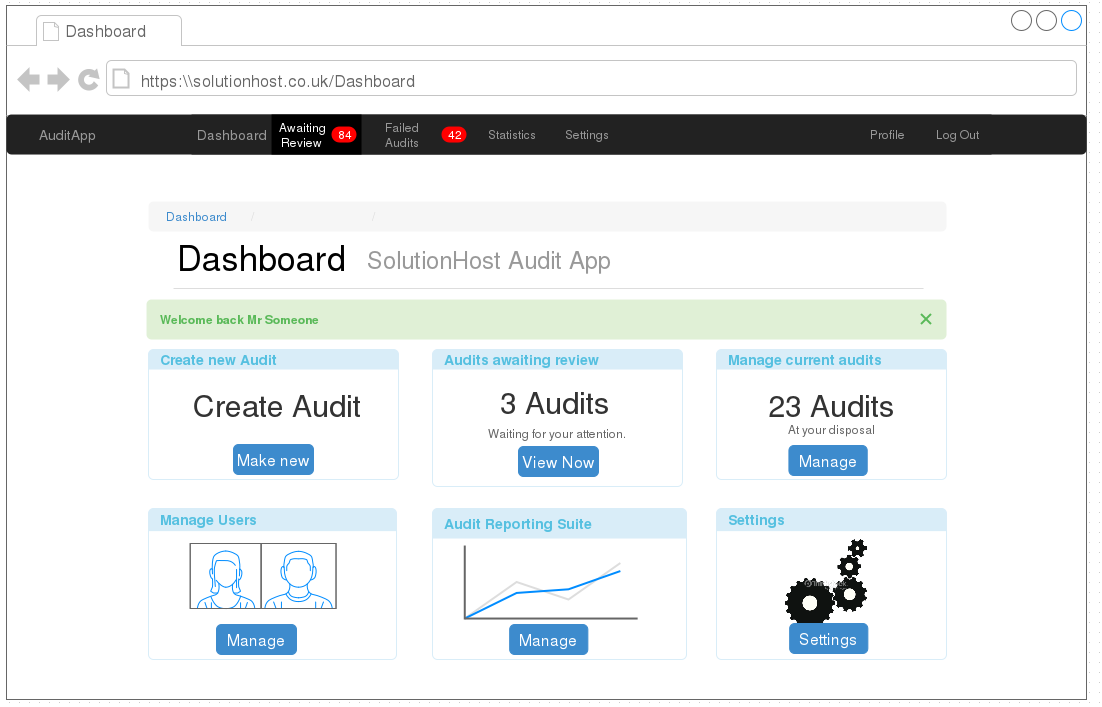
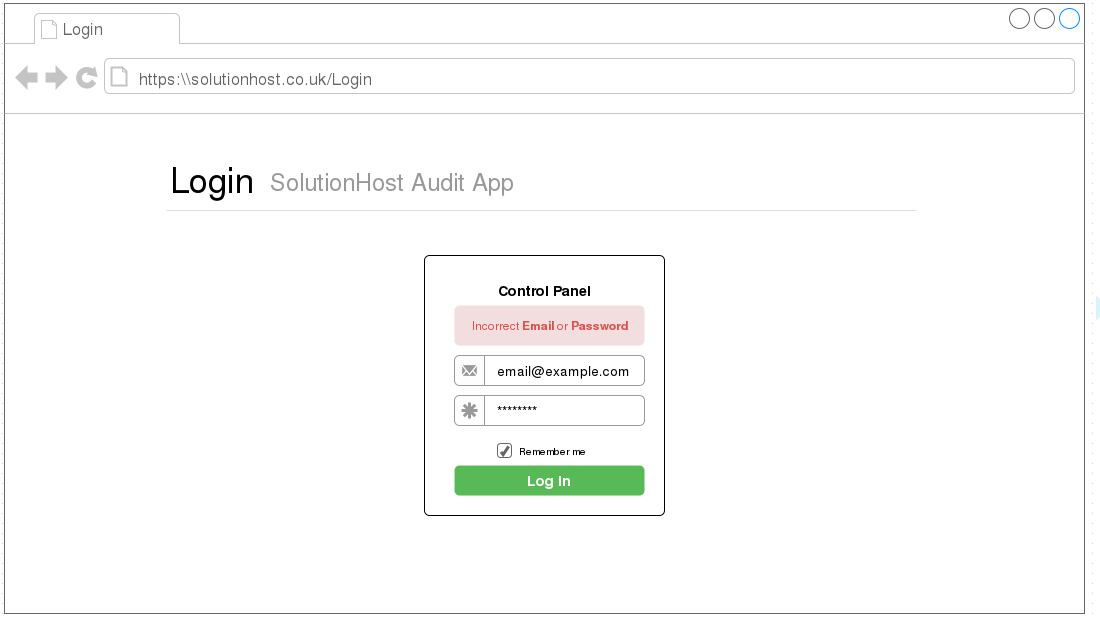
### Reports

The dashboard should display a record of all past Failed/Successful audits for quick viewing. A wide range of graphs should be available for the end user to change as needed. This would be more of a front end job than back-end. The API should process the data in the back-end and spit out everything needed to create pretty graphs on the front. Not too different to what Tom is working on at the moment with his SAP app stuff.

Failed audit's should generate a red flag which appear in the notification bar until resolved. As should all audits waiting review (Yellow Flag) and just for completeness a green flag for successful accepted audits which haven't been view yet.

Storing data and allowing it to be put into graphs will help users identify reoccurring problem areas or common problems people are contending with. Plus everybody loves a graph.

# Wire-frames



# Proposed technologies to be used.

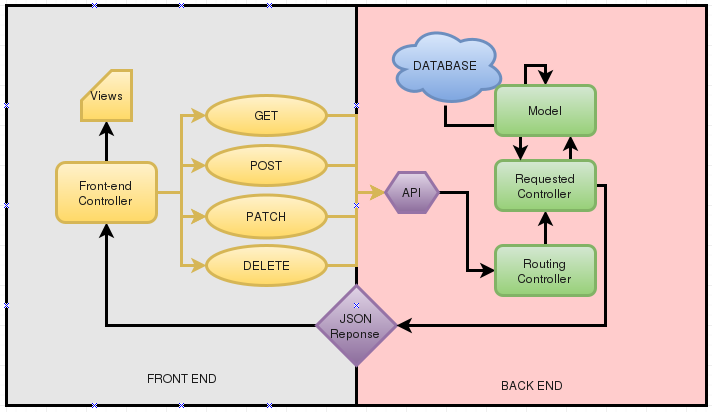
**Mysql**: Continue to use the current database (MySQL) however create new tables which are more generic so can be reused for different apps. I propose normalizing the data completely in order to have an easier to work with table structure. This will either mean migrating current data to the new structure OR allowing old customers the choice of using the new setup or the old version, offering new customers only the new setup, similar to how Dan's new PDF layout works.

**Our own ORM (Object-Relational Mapping):** In my spare time I have been modifying a simple ORM created by w[ww.notorm.com](http://www.notorm.com/) this will allow us to communicate with the Database in an object fashion, mapping out all fields in the Database once and reusing the same code repeatedly. This will help make out code more testable, faster and less error prone. It will also mean we no longer have to write SQL, instead using PHP to generate the SQL for us.

**Zurb Foundation for Apps**: Zurb Foundation is like Bootstrap but primarily designed to work well with apps. It run's on an AngularJS back-end and comes with pretty routing built in. The JavaScript thrown in as default makes creating apps quick, easy and beautiful. I also propose dropping the idea of using Bootstrap in the future to use Foundation instead. Bootstrap is great, however the Zurb Foundation group offer a more fully fledged Front-end website framework and also a great framework for emails. Using all 3 of these together will create more continuity across all Apps/Websites/Emails and more reusable styles. I think the apps version of foundation will be more suitable for this task as the Audit app is intended to be used more on Mobile and Tablet than anywhere else.

# How I intend on making it work

I intend on separating the Front-end and Back-end completely. The back-end should process requests from the front and produce a JSON response to be consumed by the front. I will do this by creating a simple Model & Controller interface on the back-end which responds to Restful requests (Get, Post, Put, Patch and Delete). Rather than have a view this should simply send back JSON to the front-end which will then be processed by the front end controllers and turned into a view. The front end will be created in JavaScript and HTML with the help of Angular and Foundation for Apps. The back end will be pure PHP and SQL. This separation of front and back should make the application more future proof by allowing us make changes to the front without affecting any of the business logic and similarly will allow us to add more features to the back-end without affecting the front.#

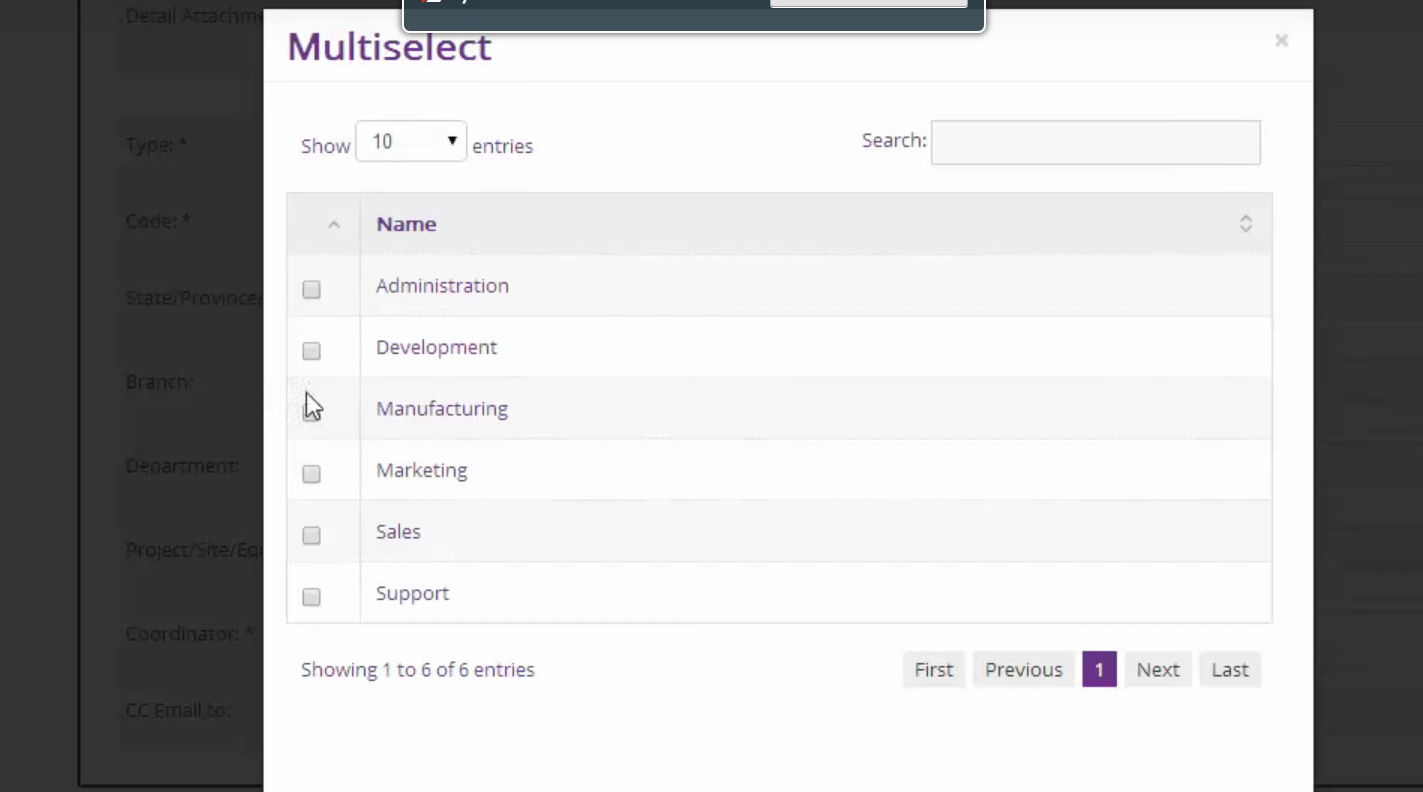


# Current Competition

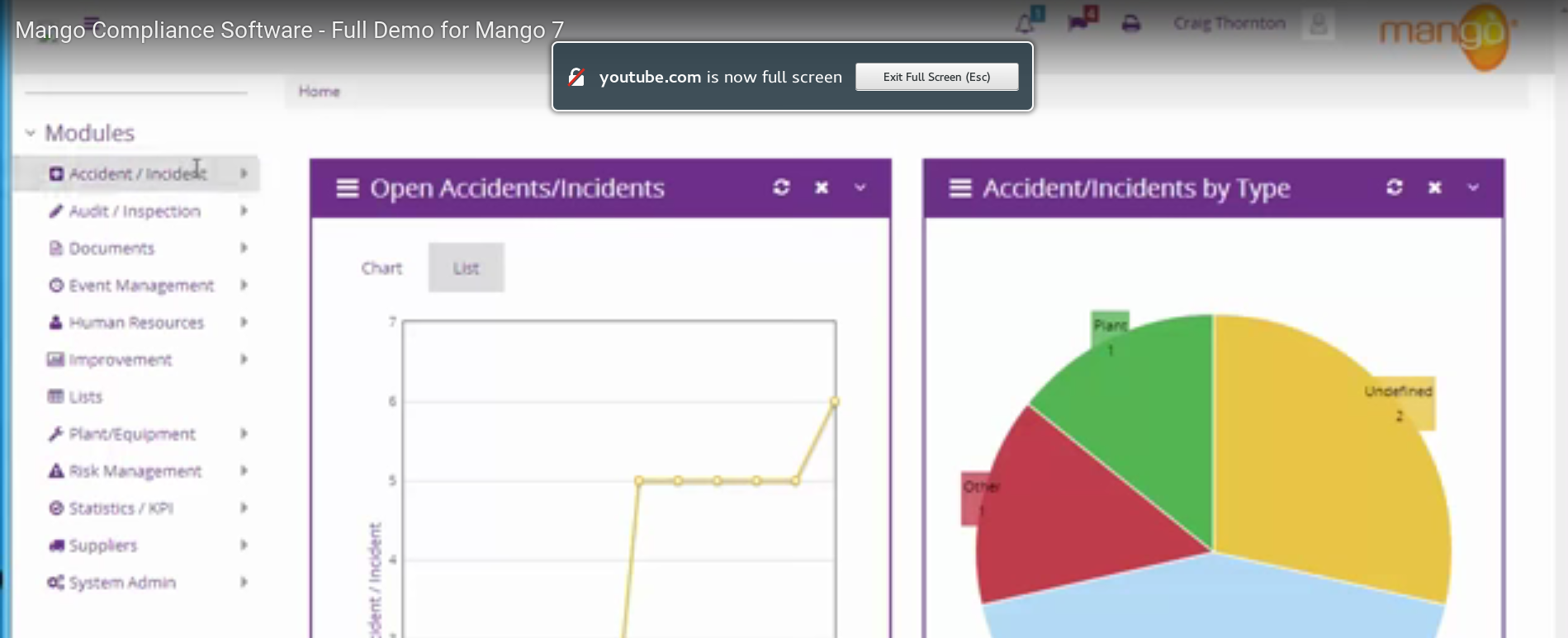
Currently there are a few similar services/applications out there which would probably act as our most direct competition. Finding price's is quite difficult because they work like we do and force you to request a call back before they show you any £ signs. Looking at their websites I'm fairly confident they are all similarly priced though as the few I could get a price for.

## [Www.mangolive.com](http://Www.mangolive.com/) (£250 per month)

Mango Live appears to be a slightly prettier version of what we currently offer. They offer very little automation when it comes to filling in audits, it seems a little awkward to work with. However one trick they use which I think would be quite a good asset to make use of is their select lists. Instead of requiring you to scroll through a never ending list of option the options pop up on top of the screen like this:



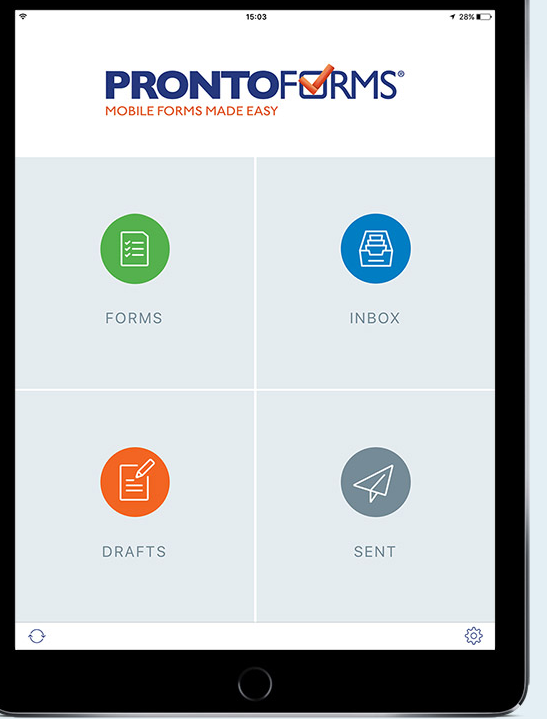
I also quite like their Dashboard layout, and their notification section in the toolbar at the top of the page:



It appear they also offer what we call the employee manager in with their software. However they seem to have missed a trick by not offering a KPI or Appraisal system. Looking at their source-code I can actually see they used Hub Spot point and click template designer to create the front-end of their system which uses Bootstrap for the main look and feel. Their website is Mobile friendly but they do not offer any sort of native Mobile app. They do however state they offer full ISO compliance.

## [www.prontoforms.com](http://www.printoforms.com/) (£20 per month, per user plus extras)

Pronto Forms offer the same as Mango Live, with the added benefits of a native mobile app. The mobile app appears to offer the same services as you can do on their web version plus the ability to draw directly on top of photo's taken as supporting evidence to audits.



The mobile app also offers offline support, calendar integration, GPS tracking and automatic time-stamps. As an added way of monetizing their product they also sell usage of their own API. I can't find a specific price for their API service but they do charge $500 just for “API integration consulting session”. Their customer base includes BP, HP, Unilever, Toshiba, Crowley and Rentokil. I personally think we could build a system which at least rivals their service if not better it. They also state that their customers have successfully integrated their API with SAP. Which may mean we could sell our product to Dassian ;-) ?

## [www.medgate.com](http://www.medgate.com/) (£10,000 year)

They offer the same as the other 2 products I've mentioned but with far more pre-made ready to use audits. Their prices start at £10k a year and some of their customers include NASA, Glaxo Smith Kline and Rio Tinto. From what I can tell from their website their software is the entire Portal on steroids. Maybe a target in the future?

# Extra Bonus Points.

Build a Mobile/Tablet native app using Ruby language. Ruby is not too different to PHP and is essentially just another scripting language like PHP. However Ruby can be compiled to native Android (Java) and iOS (Swift/Objective-C) applications. Saving us from having to learn or employee someone who knows both JAVA and Objective-C which are both quite low level complicated languages. Do the job once instead of twice.

## Why a Mobile app?

* Make use of SqlLite3 to store data from audit on phone so audits are usable even without internet connection. Post data when connection is restored.
* Make use of native mobile calendars to alert of pending/out of date audits.
* Better support for mobile camera than the basic browser support (Cropping, image effect etc)
* Make use of built in GPS for detecting location when completing Audit.
* Could be extended to create a checking in/out system (Employee KPI system, sign into app to sign in at work, I think a customer suggested this before so they could track their Sub Contractors and see if they were actually working when they said they were)

## How a API helps.

If we were to create a Restful API centered back-end which took care of handling all requests we could easily hook into it only requiring a couple of view files requesting data, and posting data. This would require very little actual programming and mainly only require a couple of loops in the view files to iterate over the JSON response given from the API.

The main work involved in this would be utilizing the native database support offered by mobiles for Databases and GPS etc. Which could be separated into a second development plan. Essentially focusing on developing a native mobile app for iOS and Android first then building these apps into more advanced software.

\* - ~~Compiler -~~ [~~http://www.rubymotion.com/~~](http://www.rubymotion.com/) (Apple and Android) \*2

\* - ~~Compiler -~~ [~~http://ruboto.org/~~](http://ruboto.org/) Only Android

\* – Wrapper - <http://rhomobile.com/> Creates wrapper apps for Android, Windows and iOs. But also can make use of native features (eg Local mobile SqlLite database).

\*2 - You need to have a Mac to install Ruby-motion. You also need a Mac to compile to an iOs native app (Objective C code). Apple are Nazi's..

# Install process.

**Node JS**

Yum install npm -y

**Composer**

cd /tmp

curl -sS https://getcomposer.org/installer | php

mv composer.phar /usr/local/bin/composer

**Foundation**

npm install -g foundation-cli

**Ruby/Bundler**

sudo yum install ruby

sudo yum install gcc g++ make automake autoconf curl-devel openssl-devel zlib-devel httpd-devel apr-devel apr-util-devel sqlite-devel

sudo yum install ruby-rdoc ruby-devel

sudo yum install rubygems