Audit System

Redevelopment Proposal

Doug

Audit Records
Redevelopment Proposal

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.

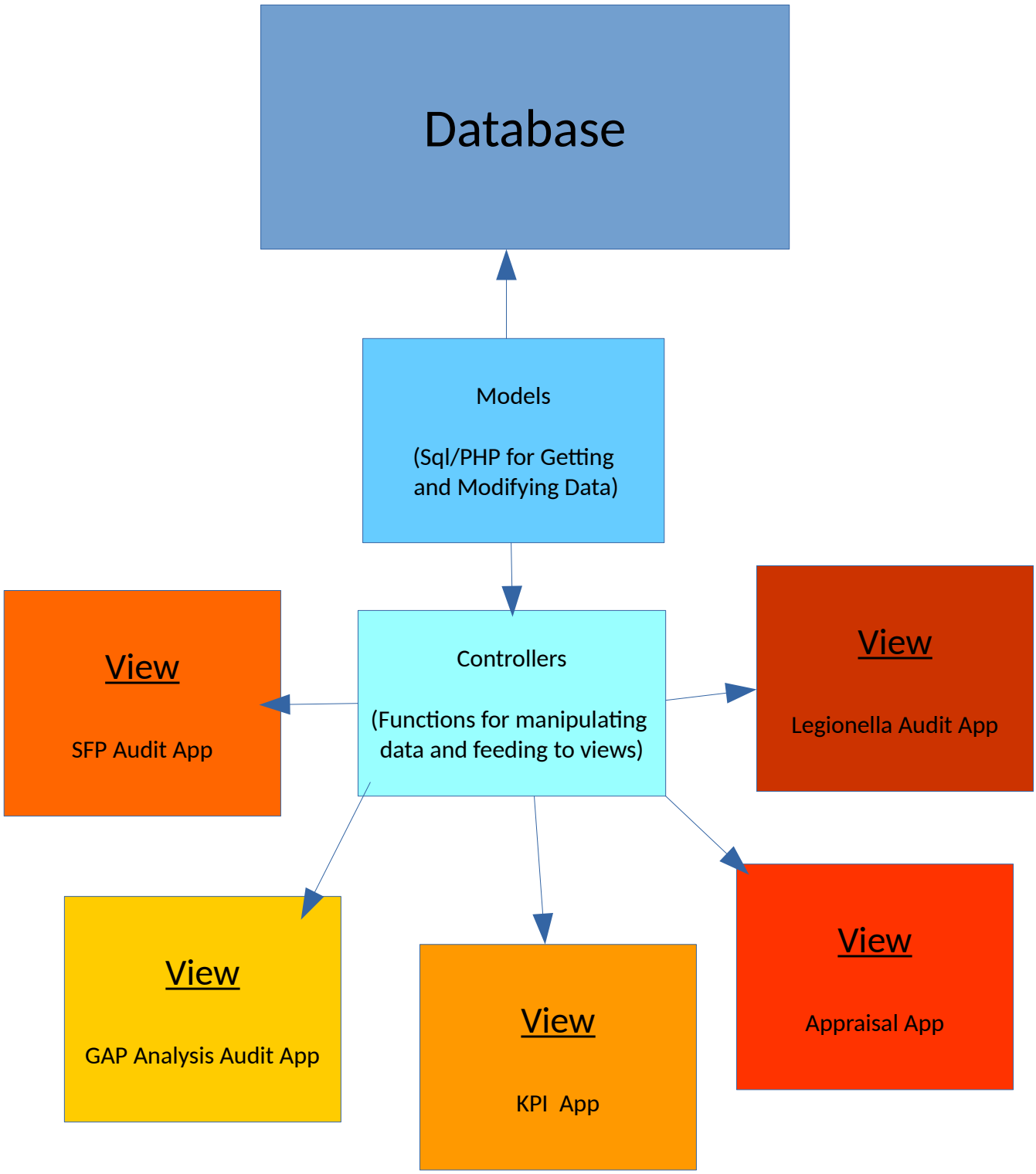
Proposed fixes.

- 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 premade 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 (Backend stuff) from the View (Frontend). 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.

Lots of different looking applications, all using the same central controllers and database tables.



Extra Bonus Points (Long term, may be a big Job)

Build a Mobile app using Ruby language. I suggest Ruby because there are compilers* which can compile from Ruby to native Android (Java) ~~and iOS (Swift/Objective C)~~*2. Its also not very different to PHP as far as syntax goes so learning it isn't too hard if you know PHP (I think I have pretty good grasp of it already).

Why:

- 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)

This would essentially require turning the current audit system into a PHP Api, with all REST (get, put, post, delete) actions setup. Then building a Ruby application which connects to the API and building the wrapper to allow all the needed features.

Pro:

Would look great, work great and probably be quite sell-able, as a separate application outside of the SFP.

Con:

Expensive to build.
Would require a big re think of how the current system works.

* - ~~Compiler - http://www.rubymotion.com/~~ Apple are Nazi's *2
* - ~~Compiler - 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 RubyMotion. You also need a Mac to compile to an iOs native app (ObjectiveC code).