HR4E Group J: Survey Implementation

Douglas Roffel, Christopher Garrett, Abhishek Gupta, Howard Tjong, Alejandro Aguilar

May 25, 2014

Abstract

Our group worked with the HR4E ('Health Records For Everyone) organization, who initially tasked us to "Use the HR4E database structure to create and deploy a 'non-trivial medical questionnaire"'. After careful consideration, we instead shifted our efforts to rewriting and rethinking the major aspects of the codebase in Ruby on Rails, instead of utilizing the existing Python code. Despite the time necessary to recreate the entire questionnaire system in Rails, we still managed to design a fully functional system that allows for non-trivial questionnaire creation, in which we recreated the BRCA1 questionnaire in.

Switching to PostgreSQL

After examining the existing HR4E Python codebase, we realized that the database schema was extremely complicated, and the dependency on that schema made writing the rest of the application difficult. By changing the database dependency to PostgreSQL, we allowed ourselves to directly store arbitrary key:value pair mappings (related to a Python 'dictionary') in a datatype called an hstore. Using hstores let us simplify the 5 database tables in the old model (ResponseChoice, ResponseSelection, Response, Submission, Respondent) into a single table in the new model, simply named 'Response'.

Tautology of Questionnaire Logic and Presentation

Questionnaire editing and creation was a disjointed flow between creating questions, creating pages, creating branching logic, and additionally adding HTML code to each of those steps for presentation. This complexity stemmed from a model where the steps for creating questions, pages, transitions, and branching logic, were all separately compartmentalized processes. Additionally, the user modified presentation logic (embedded HTML) in the same places they put data, which both should have a separation of concerns.

Our Implementation:

We simplified 11 database tables (ResponseChoice, Question, PageQuestion, Page, PageAttributes, PageAnalysis, QuestionnairePage, QuestionnaireAttributes, ProjectQuestionnaire) into a single table, named 'Form'. A form has two states, one where it gets edited, and one where where it gets presented (taking the questionnaire). This simplification was achieved by the realization that a Form is the data that it represents: Questions are contained in sections and branches, and branches observe the questions they are logically linked to. We designed a custom syntax using the templating language Handlebars, for users to simplify the form creation process. By designing this syntax, we have built-in extensibility for future offshoots of the HR4E project to write code on top of, like a WYSIWYG editor.

Conclusion

Our team went above and beyond the requirements of this project, to better address the needs of HR4E. We hope that our system is enough of an improvement to be adopted, and that future work on the project can be dealt with in a less complicated way.