The STEM Academy Data Solution

Design Document

Shannon Ernst, Kyle Nichols, Javier Franco

2 December 2016

CS 461 Fall 2016

Group 48

Abstract

The STEM Academy Data Solution will generate and distribute surveys, collect the responses of these surveys, and generate reports based on these responses. The design of these components are key. Through examination of each component through an interface and logic standpoint, the developer team has created a design to satisfy the project requirements.

1 FRONTSPIECE

1.1 Date of issue and status

2 December 2016 In-progress

1.2 Issuing organization

The STEM Academy Data Solution, CS Capstone Group 48, Oregon State University

1.3 Authorship

Shannon Ernst, Javier Franco, Kyle Nichols

2 INTRODUCTION

2.1 Purpose

This document outlines the design concepts and algorithms for the STEM Academy Data Solution. It is to be used as a guide in how the product will be developed during the year.

2.2 Scope

The solution presented in this document is for survey generation, survey distribution, data collection, database management, and report generation for the STEM Academy Data Solution. The STEM Academy Data Solution will be used by STEM Academy to generate surveys, store the responses of those surveys, and view reports on the surveys. The design document will be used to describe our processes and solutions as a reference while developing the product. This document does not, however, include designs for stretch goals. It instead focuses only on the necessary requirements and goals of the product.

2.3 Summary

The document following describes the main components of the product. The first section covers question generation, covering the interface from the user perspective and the process of storing and displaying questions. This is followed by survey generation, which covers the user interface design and the logic of how surveys are collected in a JSON format. The next section covers database management, from how it is organized and created to how the other components will interact with it. Next is a section on the search form, which also covers the user interface design and how the search forms are displayed. Lastly, report generation is covered, detaling the design of the user view and how the report will be displayed.

3 REFERENCES

REFERENCES

[1] Lau, Dmitri. "10 Reasons Why You Should Use AngularJS." *sitepoint. N.p., 5 Sept. 2013. Web. 26 Nov. 2016.*

[2] C, Ravindra T. "Connecting to Database Using AngularJS." Code Project. N.p., 25 Mar. 2016. Web. 26 Nov. 2016.

4 BODY

4.1 Identified stakeholders and design concerns

The STEM Academy is the primary customer of this product. Their team is non-technical and require a fully functional survey generation and distribution center that will collect responses and generate summary reports based on the responses. The STEM Academy is the primary user of the software. Their camp participants will be taking the surveys. The developers are the capstone group who will be delivering the project based on the following design.

4.2 Question Generation

4.2.1 Question Generation Interface Viewpoint

The user of the Question Generation Interface is the STEM Academy administrator who will be making surveys. Every survey has questions. The interface for the question is within the interface of the survey. The user should be able to easily add, delete and reorder questions in the survey. The add, delete, and reorder functionality is in the survey scope while the results of those actions affect the question. The following view contians what the user will see in regards to question generation.

4.2.2 Question Generation Interface View

Upon entering a survey to edit or create, the user can add, delete or reorder questions in the survey. A user will be able to add a question by clicking an add question button. A question box will then appear in the editable area of the survey. This question box will have a delete question functionality which can be triggered by clicking the ”x” in the upper right hand corner of the question box. This ”x” will be there so long as the survey is in edit mode. If the delete button is clicked, the question will be removed from the survey and the survey json. A save button will be located in the bottom right hand corner of the question box. If the save button is clicked, then the information will be saved to an appropriate json format for both a question and the survey at large. There will be two buttons to the immediate left of the question: one an up arrow, the other a down arrow. The up arrow shall be staked on top of the down arrow. Clicking one of these buttons will move the question either before the previous question in the case of the up arrow, or after the question following its current position in the case of the down arrow. The clicking of either of these arrows will result in the change of the ordering of the questions in the survey json as well as the physical movement of the question in the editable area of the survey. The question box will have a text entry area which will take the question text of the question.

There will also be four buttons where the user can select which type of question it will be: text entry, multiple choice - single answer, multiple choice - multi-answer, or matrix. Upon selecting one of these buttons it will change the appearance of the potential answer section. The default of the answer section will be text entry which will not provide a spot for the user to put in answers. Upon selecting either multiple choice options, the answer area will be transformed into one text entry available with an accompanying delete button, represented by an ”x”, to its immediate right and an add answer button to its south east corner. Upon clicking add answer, another text entry line will appear. Upon clicking the ”x” delete button, the line will be removed. The matrix option will work similarly only this time, multiple questions will be prompted for instead of answers. The answers will be selected via a drop down menu of common matrix style answers.

4.2.3 Question Generation Logic Viewpoint

The developer is concerned with what happens behind the scenes when a question is generated. In order for this section to be implemented the add question button must be active in a survey. The delete and ordering function will be implemented after the question generation logic has been implemented. This view examines the logic of creating the question so that the user can see it in their survey creation. The developer will need to be aware of templates and being able to dynamically changes views.

4.2.4 Question Generation Logic View

Questions are going to be stored in a json format. The format is as follows:

{

id: alphanumeric randomized string unique to this question

type: a string signifying the question type (textEntry, multi-s, multi-m, matrix)

qtext: the question text string

answers: an array of potential answer strings

}

The type will be used to select which html template to use for the question in generating the survey. The html templates will be premade with their own services to correctly construct the survey with the included templates. The qtext will be substituted in each template so it will appear as the question text. Answers will appear according the question type. If it is a text entry, a default text area of four rows and fifty columns will be provided. If it is a mutliple choice, the answers will be produced using a loop to repeat the html structure with a different value according to the answer text. If it is a matrix, a table will be generate with one of the standard scales spanning the top, each radio button beneath it having the appropriate value. The questions will align on the left in rows with the radio buttons to their right. The questions fields will be populated via a loop. In AngularJS, ng-repeat will be used to adjust the values in the templates. You can also insert html to the screen via DOM manipullation. A mixture of these methods will be used.

4.3 Survey Generation

4.3.1 Survey Generation Interface Viewpoint

The user of the Survey Generation Interface is the STEM Academy administrator who will be making surveys. Every survey has questions. The user is non technical so editing and creation needs to be intuitive. This should be the first thing created in the course of the project.

4.3.2 Survey Generation Interface View

Upon selecting create survey or selecting an existing survey to edit, the user will be taken to the survey creation page. This page will consist of a text entry line where the title of the survey will go. There will also be an add question button, preview button, save button and distribute button to the bottom right of the title entry. Upon clicking the add question button, a question will be added to the editable area between the survey title and the buttons. The questions will fill the area and once the max view has been achieved will begin to scroll. Upon clicking the preview button, a window will open with the survey in the form that survey takers will see. This is technically a live survey but will be prepopulated with ids which will exclude these entries from the database. The window can be closed using the ”x” in the upper right hand corner. Clicking the save button will save the current survey, whether complete or not. The distribute button will

produce a link to the survey immediately beneath the buttons for the user to copy and paste and distribute in whatever medium they see fit. They can exit a survey by clicking the home button in the upper left hand corner of the screen.

4.3.3 Survey Generation Logic Viewpoint

The developer is concerned with what happens behind the scenes when a survey is generated. In order for this section to be tested the database will need to be set up otherwise it will the surveys will not be able to be saved.

4.3.4 Survey Generation Logic View

The survey will be saved as a json object. The json structure is as follow:

{

id: alphanumeric randomized string unique to the survey

title: string representing the title of the survey

questions: an array of question objects

}

The survey will be generated for the user based on this json. There will be a survey template which will include the title as a variable. The question templates will be generated through a mixture of ng-repeat and DOM manipulation. The questions will be produced in order of the appearance in the array in the json. A submit button will be included. Surveys will be a single page. Upon submit, the form will be posted to the database. Responses will be sent as json. This survey may either be a preview or the real thing. If it is a preview, it will have a response id with the value of test which will exclude it from the database.

4.4 Database Management

4.4.1 Database organization Structure Viewpoint

The database will be organized around the core elements of surveys, questions, and responders. These will compromise the three core tables in the database, with other tables functioning as relationships or specializations of these core tables. All tables contain unique IDs as their primary key, and each ID is a non-negative integer.

The Survey table will contain the survey_id as its primary key to identify each individual Survey. Each Survey will also contain a string attribute for the title and a string attribute for the program it is used in.

The Question table represents any individual kind of question. It's attributes include a unique question_id to serve as its primary key and the question's text as a string attribute. The question text will be generated through the survey generation process and be stored here when the Question is created.

Surveys and Questions have a many-to-many relationship: Questions can exist in many different Surveys and Surveys contain many different questions. To create this relationship in our database, a Contains table will exist. The Contains table's primary key will be a combination of a survey_id and a question_id to correspond a questions use in a survey. The question_id and survey_id individually can be used in other Contains entries, but since surveys don't have the same question multiple times, the combination will be unique for each table entry.

The Responder table will store information on the individual responders. This includes a unique responder_id for the primary key, two string attributes for first_name and last_name, the birthdate as a date attribute, and a non-negative integer attribute for the age derived from the birthdate and the current date. The age can be updated or refreshed when necessary. The Responder table has two specializations: a Student and a Parent. A Parent does not contain any extra

information, but a Student contains a parent1_id and a parent2_id to correspond to both of its parents. The specialization is to allow for more accurate and specified queries of students and parents separately or together. As specializations, they also contain a matching responder_id to their corresponding Responder table entry.

To save responses, a Response table will exist. This functions as the many-to-many relationship between Responders and Questions, but also stores the answer provided for the question. The Response table has three specializations for each of the possible types of questions: Multiple_Choice, Matrix, and Text. Multiple_Choice entries contain a character attribute for the answer, Matrix entries use a non-negative integer attribute, and Text entries use a string attribute. Each of these specializations also contain the response_id for the corresponding entry.

The below diagram outlines the relational database schema described above.

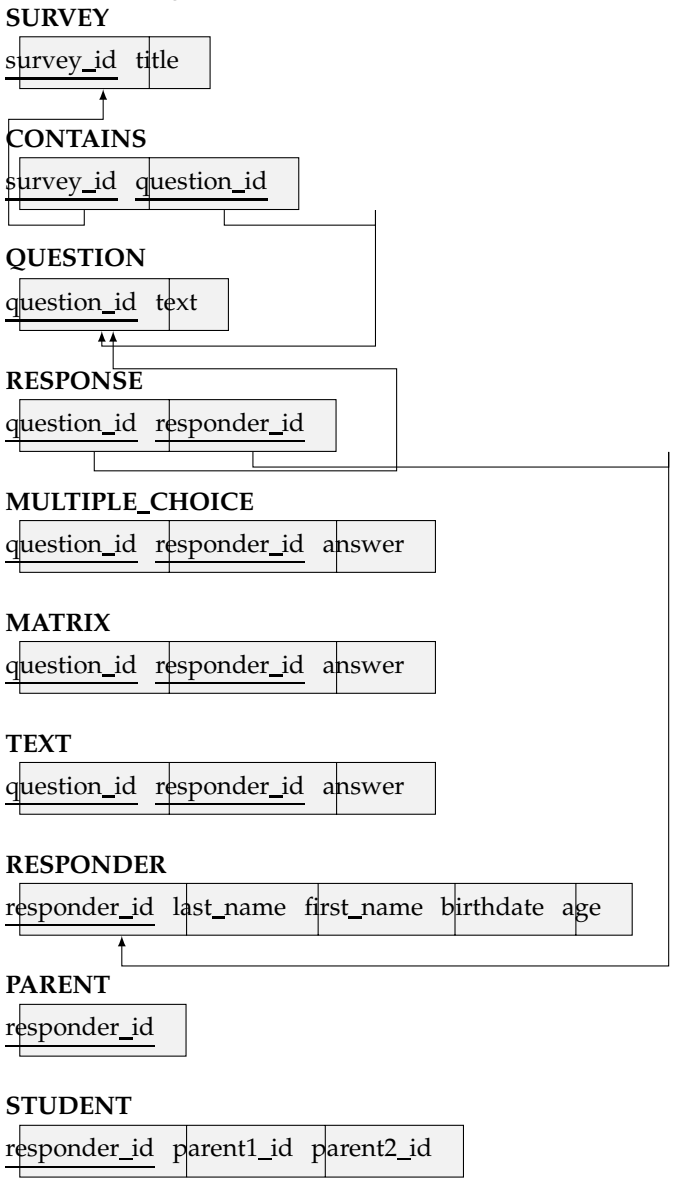

Fig. 1: The relational database schema

4.4.2 Database Creation Logical Viewpoint

Each table will be created with the following SQL syntax. In particular, tables will use AUTO INCREMENT to automatically increment and create IDs. They will also be done in the following order because some tables reference

others.

```
CREATE TABLE Survey
(
 survey_id int NOT NULL AUTO_INCREMENT
 title varchar(255)
 program varchar(255)
 PRIMARY KEY (survey_id)
)

CREATE TABLE Question
(
 question_id int NOT NULL AUTO_INCREMENT
 text varchar(1020) NOT NULL
 PRIMARY KEY (question_id)
)

CREATE TABLE Contains
(
 survey_id int NOT NULL
 question_id int NOT NULL
 PRIMARY KEY (survey_id, question_id)
 FOREIGN KEY (survey_id) REFERENCES Survey(survey_id)
 FOREIGN KEY (question_id) REFERENCES Question(question_id)
)

CREATE TABLE Responder
(
 responder_id int NOT NULL AUTO_INCREMENT
 first_name varchar(255) NOT NULL
 last_name varchar(255) NOT NULL
 birthdate date
 age int
 PRIMARY KEY (responder_id)
)

CREATE TABLE Parent
(
 responder_id int NOT NULL
 PRIMARY KEY (responder_id)
```

```
 FOREIGN KEY (responder_id) REFERENCES Responder(responder_id)
)

CREATE TABLE Student
(
 responder_id int NOT NULL
 parent1_id int
 parent2_id int
 PRIMARY KEY (responder_id)
 FOREIGN KEY (responder_id) REFERENCES Responder(responder_id)
 FOREIGN KEY (parent1_id) REFERENCES Parent(responder_id)
 FOREIGN KEY (parent2_id) REFERENCES Parent(responder_id)
)

CREATE TABLE Response
(
 question_id int NOT NULL
 responder_id int NOT NULL
 PRIMARY KEY (question_id, responder_id)
 FOREIGN KEY (question_id) REFERENCES Question(question_id)
 FOREIGN KEY (responder_id) REFERENCES Responder(responder_id)
)

CREATE TABLE Multiple_Choice
(
 question_id int NOT NULL
 responder_id int NOT NULL
 answer char(1)
 PRIMARY KEY (question_id, responder_id)
 FOREIGN KEY (question_id) REFERENCES Question(question_id)
 FOREIGN KEY (responder_id) REFERENCES Responder(responder_id)
)

CREATE TABLE Matrix
(
 question_id int NOT NULL
 responder_id int NOT NULL
 answer int
 PRIMARY KEY (question_id, responder_id)
```

```
 FOREIGN KEY (question_id) REFERENCES Question(question_id)
 FOREIGN KEY (responder_id) REFERENCES Responder(responder_id)
)

CREATE TABLE Text
(
 question_id int NOT NULL
 responder_id int NOT NULL
 answer varchar(1020)
 PRIMARY KEY (question_id, responder_id)
 FOREIGN KEY (question_id) REFERENCES Question(question_id)
 FOREIGN KEY (responder_id) REFERENCES Responder(responder_id)
)
```

4.4.3 Database Interaction Algorithm Viewpoint

To allow for interaction between the survey and report generation, an API will need to be created to work with both. The API will be written in PHP since PHP provides functionality to generate SQL queries with a database using `mysqli` objects and the `query()` function.

4.4.4 Interacting with the survey generation

Since the survey generation will be done using JSON, the API will need to get the POST information as the JSON object and parse it for a SQL INSERT. It will do so using the following design:

1) Generate a string `sql_query` for the SQL query setting up the INSERT query with 'INSERT INTO ' and adding the name of the table from the POST information.
2) Because each table has different sets of attributes, a switch statement will be used to check the table name and add the necessary string to `sql_query` (this switch statement will take the value from the POST and convert it all to lowercase).
   a) The Survey table will add '(title, program)'.
   b) The Contains table will add '(survey_id, question_id)'.
   c) The Question table will add '(text)'.
   d) The Response table will add '(question_id, responder_id)'.
   e) The Multiple_Choice, Matrix, and Text tables will add '(question_id, responder_id, answer)'.
   f) The Responder table will add '(last_name, first_name, birthdate, age)'
   g) The Student table will add '(responder_id, parent1_id, parent2_id)'.
   h) The Parent table will add '(responder_id)'.
3) The string 'VALUES ' will be appended to `sql_query`.
4) A while loop will wrap around the following process to account for multiple entries being added to the table.
   a) There will be another switch structure to determine the number of values that will be inserted based on the table to be inserted into:

i) The Question, and Parent tables will read one value and add '(*value1*),' to `sql_query` accordingly.
ii) The Survey, Contains and Reponse tables will read two values and add '(*value1*, *value2*),' to `sql_query`.
iii) The Multiple_Choice, Matrix, Text, and Student tables will read three values and add '(*value1*, *value2*, *value3*),' to `sql_query`.
iv) The Responder table will read four values and add '(*value1*, *value2*, *value3*, *value4*, *value5*),' to `sql_query`.

5) After the loop, the last character of `sql_query` should be a comma, and it will be replaced with a semicolon.
6) After the string has been completed, a mysqli object `conn` will be created that connects to the server the database is stored on.
7) A call to `conn`'s function `query()`, passing `sql_query` as the argument query string. The result will be stored in a variable called `result`.
8) There will be a check on if the value of `result` is true.
   a) If it is true, then display a success message.
   b) If it is not true, then display an error message.
9) Lastly, `connection` will be closed.

4.4.5 Interacting with report generation

1) The API will receive the SQL query as a string and store it as `sql_query`.
2) A mysqli object `conn` will be created that connects to the server the database is stored on.
3) A call to `conn`'s function `query()`, passing `sql_query` as the argument query string. The result will be stored in a variable called `result`.
4) There will be a check on if the value of `result` is true.
   a) If it is true, then display a success message.
   b) If it is not true, then display an error message.
5) `connection` will be closed.
6) Lastly, `result` will be echoed.

4.5 Report Generation

4.5.1 Search Form Interface Viewpoint

The type of user that will be generating reports will be a STEM Academy administrative user. The user will be able to generate reports using a user freindly interface that uses a search form for generating reports. Below the search form there will be a report form that contains the query results of the user. When a user has finished generating a report they will be able to save their report and also have the option to print their report.

4.5.2 Search Form Interface View

The user interface for report generation will be a search form that uses text input boxes for querying information from the database. The fields that will be shown on the webpage include survey questions, responses, student info, club, program, and student demographic information. When the user arrives at the report generation page for the first time there will be one input text box for each field by default. Each input field will have its corresponding name written above incase the user deletes all of the input fields for the search form. This will allow the user to distinguish which

add button corresponds to each field. The add button for each field will be located below the most recently created text input box. The add button will allow administrative users to add additional information for their query. Each input field will have a button on the right hand side for deleting the input field which will be indicated by ”x” symbol and this will allow a user to delete any input fields that are no longer necessary for their query.

Once an administrative user has finished inputting their query information into the input fields they will be able to submit their query by clicking on a submit button that will be located on the bottom right hand side of the search form. Below the search form there will an empty report form if the user decided to create a new report. If the user decided to edit a saved report the report form will display the query results of their saved report. Each query result shown in the report will have a text box above it, so that an administrative user is able to add a caption. If a user no longer needs a query in their report they will be able to delete a query in the report form by clicking on a delete button which will be indicated by ”x” symbol. The delete button for each query will be located on the right side. When an administrative user has finished querying information for their report form they will be able to save their report by clicking on a save button that will be located at the bottom right hand side of the report form. Administrative users will be able to print out their generated reports or any saved reports by clicking on a print button that will be located on the top right hand side on the report form.

4.5.3 Search Form Logic Viewpoint

This viewpoint examines the logic for the tools that will be used to develope the report generation webpage. It also explains the tools that will be used for saving and printing a report.

4.5.4 Search Form Logic View

For creating the report generation form we will be using AngularJS, HTML, and PHP. The reason for using AngularJS is that it will enable us to create a dynamic search form which will allow the user to add additional input fields for their query. If we did not use AngularJS the form will be static and if an administrative user were to need more input fields for their query they would have to notify a developer to add the input fields manually to the HTML page which is inefficient. AngularJS requires an HTML for defining the application’s user interface [1]. The buttons for deleting and adding new input fields will have an ng-click AngularJS directive which will call their correspoding function for deleting and adding a new input field. A controller will be used to modify the HTML expressions and the $scope service will be used to detect any changes in the form[2].

In order to save the report a table will be created in the database that only stores reports. By creating a table for storing reports it will permanently save them unless they are deleted from the database. PHP will be used for saving reports into the database and this will be done by using the INSERT INTO statement. For printing the report form CSS and Javascript will be used. CSS will be used to create two style sheets, so that when the print button is clicked it does not display the entire webpage as the content that is going to be printed. The first style sheet will be used to display the contents of the report generation webpage. The second style sheet will be used to exclude content on the webpage that is not related to the search form.

For displaying the print dialogue box the line of code href="javascript:window.print()" will be added to the print button. The print dialogue box will allow adminstrative users to choose the orientation, number of copies to be printed, margins, page range, etc. When the submit button is clicked PHP will be used to query the information from the database using the SELECT statement. Any input text box fields left empty will be ignored when the submit button is clicked. The results of the query will be inserted into an empty form that will be located below the search form for making queries. The form will be created using the HTML <form> tag. The <text area> element that has the type text will be used for inserting the query information, title, and caption for a query into a report form.

4.5.5 Report Form Logic Viewpoint

This viewpoint examines the logic for creating the saved report form that is displayed below the search form.

4.5.6 Report Form Logic View

The reports generated by administrative users will be saved as a JSON object. The structure that the report form will have is the following:

{
id: random numerical value to identify the report
title: string that contains the title of the report
queries: an array of query objects

}

The queries made by administrative users will be saved as JSON objects. The following is the structure of a query:

{
id: a unique random numerical identifier for a query
caption: an explanation of the query result: the text result of the query generated by an administrative user
}

An administrative user's saved report form will be generated using the JSON object for a report and a form template. The title of the report will be displayed in the form template using the title attribute of the JSON report object. The queries will be displayed in the form template using the attribute queries which is an array of query objects. Each query result will be displayed in the form by using the result attribute of the query object. Furthermore, the corresponding caption of a query result will be displayed by using the caption attribute of a query object.

5 SIGNATURES

By signing below, I agree to the ideas and concepts presented in this document and the list of deliverables.

Client Signature	Date
Team Member 1 Signature	Date
Team Member 2 Signature	Date
Team Member 3 Signature	Date