**Mail exercise to**

[ragesh.rajanpilllai@tcs.com](mailto:ragesh.rajanpilllai@tcs.com)

Ragesh: This is a good exercise, one of the better ones I have been involved with.

Had issue with rails 7 ruby environment, switched to rails 8 for exercise. Will go back and debug at a later date. Not resolving a 8.0 component in the rails 7 environment. Using ASDP, exactly like another project I have.

**Current Status**

The bulk of the function is implemented, with a less than optimized UI. See release notes for exact status. There are a few big issues, including survey fails to submit (fails silently).

Will continue to debug.

**Release Notes**

TBD

**CODE**

See GITHUB for the code.

**www.github.com/**[**fgermanojr/survey\_management\_system**](https://github.com/fgermanojr/survey_management_system)

Will switch public to private after our exercise is done.

**Design questions**

1. Who creates users? Admin vs self-registration. CHOICE: Admin sets up user.
2. How is the first Admin get set up? CHOICE: Rails console is used to create the first user (a Admin). UI is then used to create other Admins and regular users. The Admin that creates a user notifies the new user of his account.
3. User setup with encrypted password: obviously bcrypt gem.
4. What rails technologies should be used. CHOICE: use what I know. Traditional rails views/controllers. Use tables for layout; later use bootstrap. After this exercise I plan to repeat using turbo/stimulus/tailwind
5. How best to handle in number of questions and number of options in views: fixed number or flexible number, latter is preferred. CHOICE: limited fixed number with add question/add option button to add one more. Plan to use rails request to return full page with added row. I traded a network request for simplicity.
6. Future: This would be a clear choice for a ajax request to replace the table division or better, directly add the row, with JS. JS choices: pure JS w ajax, jquery (big library to add for one function), use rails returning JS (preferred choice; this is where RAILS returns JS, that runs to insert the row). Used this at EmrBEAR (rails 5, 6)

**Future Future Enhancements**

1. What if we don’t want to complete the survey all at once? The presenter design pattern can be used. This is where code figures out where you are in the survey and starts you where you left off.
2. Responses are simple text; one could see a richer set of datatypes: checkbox, 1 of m, n of m, Boolean.
3. The entire survey was created and taken on one large form. Could Show one question on its own page so you would progress thru the survey in order.

**Models**

1. **User**

first\_name, last\_name, email, is\_admin Boolean Added email for use in user management. Tied login to email.

Has\_many surveys

1. **Survey**   
   title string, description text

Has\_many questions

1. **Question**  
    question\_text string

Has many options

1. **Option**

answer\_text text  
belongs\_to Question

1. **UserSurveyTaken**

Belongs\_to user, survey

Has\_many option\_respones

1. **OptionResponse**

Belongs\_to question, options, and user\_survey\_taken

**Schema Diagram**

**A diagram of a survey

AI-generated content may be incorrect.**

**Screen Shots**

A screen shot of a login form

AI-generated content may be incorrect.

A close-up of a website

AI-generated content may be incorrect.

A screenshot of a survey

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a survey

AI-generated content may be incorrect.

A close-up of a white background

AI-generated content may be incorrect.