



## Technical assessment

### Notes:

1. Feel free to switch out any parts of the Stack you are not fully comfortable with. E.g., Angular instead of React, Postgres instead of MongoDB, etc
2. Focus on your areas of strength. E.g., If you are a wizard at HTML/CSS, then create the best looking UI you can with a lesser focus on the backend. If you are an API and backend expert, design and implement the DB with perfect indexes and APIs 100% REST compliant.
3. Include a README.md file with these details as the bare minimum. Feel free to add more data if you want to.
  - a. What has been completed
  - b. What is not completed
  - c. Deployment steps
  - d. Known bugs
4. You can email a zipped file for the final work or invite us to your private Github repo.

### Task


Create a Single Page App (SPA) or Mobile App that uses to

- Create a UI that allows users to create and edit survey questions as below

#### Step 1: Listing questions

### Custom questions

You must add at least one question to launch a survey.

Question	Question settings ⓘ
<div><p>No Data</p></div>	

Add a question

### Custom questions

You must add at least one question to launch a survey.

Question	Question settings ⓘ
Assignment	
Custom • Candidate	
Assignment question 2	
Custom • Candidate	

Add a question

## Step 2: Add a question

Settings

Summary

Custom questions

You must add at least one question to launch a survey.

Question

Add a question

Add a custom question

×

Question

Answers

Answer 1

Edit Remove

Answer 2

Answer 3

Add new answer

Save changes

## Step 3: Edit an added question

Same as add

Data saving can be via a real or mock API call

## Validations

1. Question text cannot be empty
2. A question must have a minimum of two answers and a maximum of 5 answers
3. Appropriate loaders where required



## Technical assessment

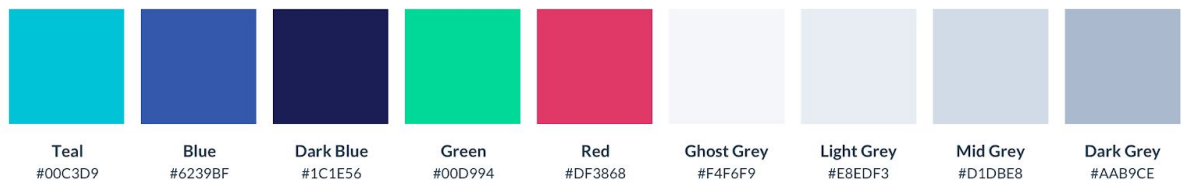
### Evaluation criteria

- Frontend
  - Clean state management and immutability
  - Adherence to HTML5 structures and React/Redux standards
  - Separation of concerns with components, selectors, reducers, etc.
  - Understanding and implementation of implied requirements from the screen. (E.g., Editing an answer while adding question)
  - ***UI doesn't have to be pixel perfect or match the CSS shown.***
- Backend
  - REST compliance
    - Use of appropriate HTTP verbs for requests
    - Use of appropriate HTTP error codes for responses
    - Providing linked objects in a clean manner
    - Statelessness
  - Entity design choices and considerations for extensibility and performance
- Considerations in terms of code quality, adherence to S.O.L.I.D and DRY design principles illustrated clearly with in-code documentation.
- **100% Completion is not a criteria. We expect what is completed to be completed cleanly.**

### BONUSES

- **Getting the UI pixel perfect. Refer to colour palette below. We use Roboto and Lato fonts. Feel free to substitute any icons in the screens with whatever is relevant.**

#### Colors



- Getting the code production ready
- UI components developed using Storybook
- Live API documentation
- ***Anything else you want to show off! :-)***