Hassan Amjad SPFx Showcase Project

Note: To jump straight to the code snippets go to Page 7.

Project: CNDS (Compliance Notice Distribution System)

Summary

The need for this application arose due to COVID-19. The Compliance department of my client wanted the ability to send out “Return to Workplace” surveys to their various offices around the world and then analyze responses submitted by employees. Originally, they were sending out surveys manually via email to every office but were finding it hard to keep track of all the responses. This application aimed to solve this problem by giving the ability to design surveys and send them out to large distribution lists with just a single click of a button. Once the survey goes out, the application automatically starts capturing the responses from recipients as they come in. The Compliance department can then analyze the captured submissions to decide which offices can be re-opened.

For the sake of brevity, I will focus on the front-end modules of the application.

Technology Stack

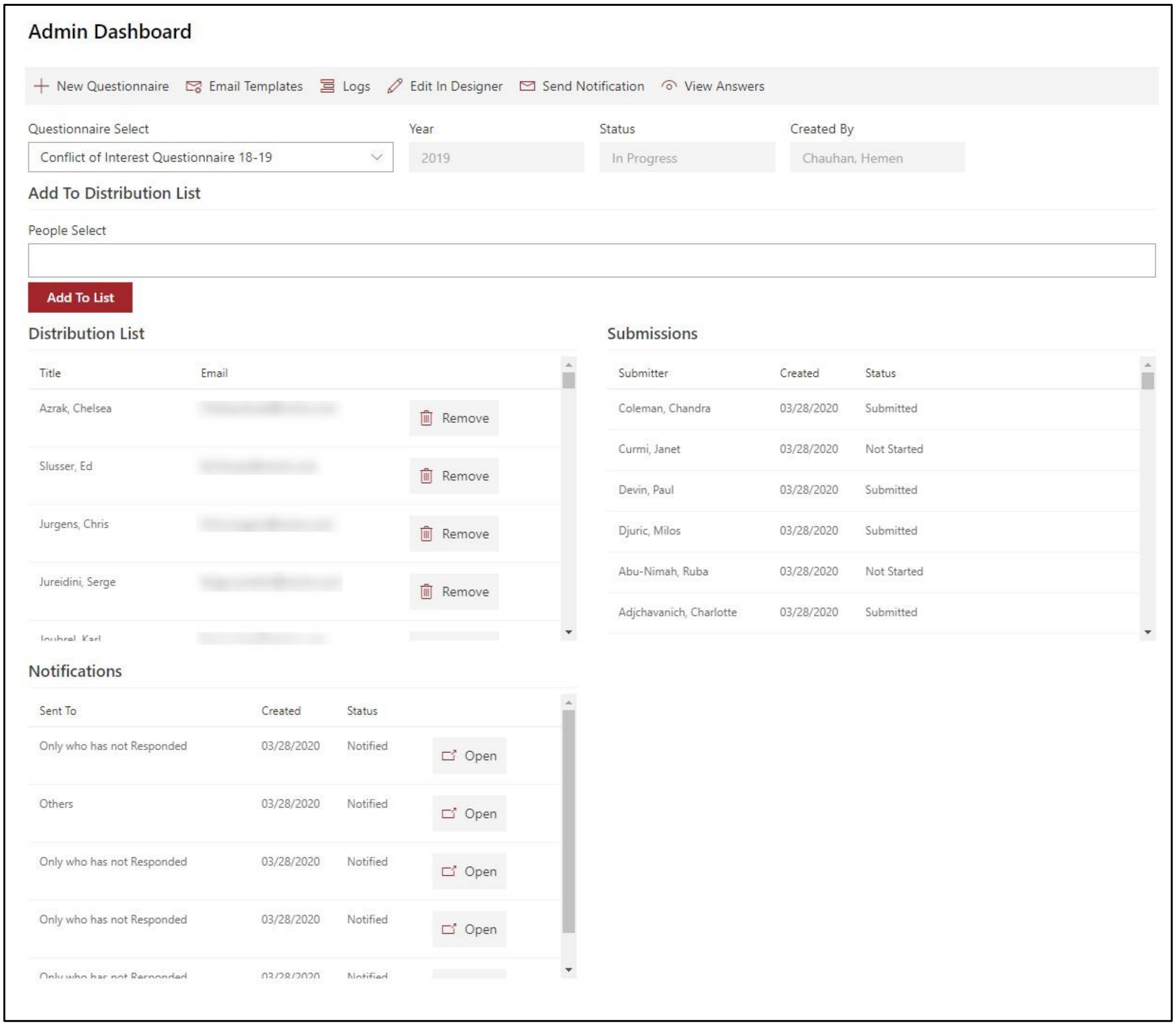
The following technologies were used to develop application front-end.

- SPFx
- Node.js
- JavaScript (ES5, ES6)
- TypeScript
- Webpack 4
- Formik
- Yup
- Redux
- React 16

Module: Admin Dashboard

This is where all the surveys can be managed by Administrators. Its main feature are as follows:

- Add or delete people from distribution list.
- View submission status of each recipient.
- View notifications that have either been sent or are waiting to be sent.
- Monitor status of the notifications sent out to distribution lists.



Admin Dashboard

Questionnaire Select: Conflict of Interest Questionnaire 18-19

Year: 2019

Status: In Progress

Created By: Chauhan, Hemen

Add To Distribution List

People Select

Distribution List

Title	Email	
Azrak, Chelsea	[redacted]	Remove
Slusser, Ed	[redacted]	Remove
Jurgens, Chris	[redacted]	Remove
Jureidini, Serge	[redacted]	Remove
Joubrel, Karl	[redacted]	

Submissions

Submitter	Created	Status
Coleman, Chandra	03/28/2020	Submitted
Curmi, Janet	03/28/2020	Not Started
Devin, Paul	03/28/2020	Submitted
Djuric, Milos	03/28/2020	Submitted
Abu-Nimah, Ruba	03/28/2020	Not Started
Adjchavanich, Charlotte	03/28/2020	Submitted

Notifications

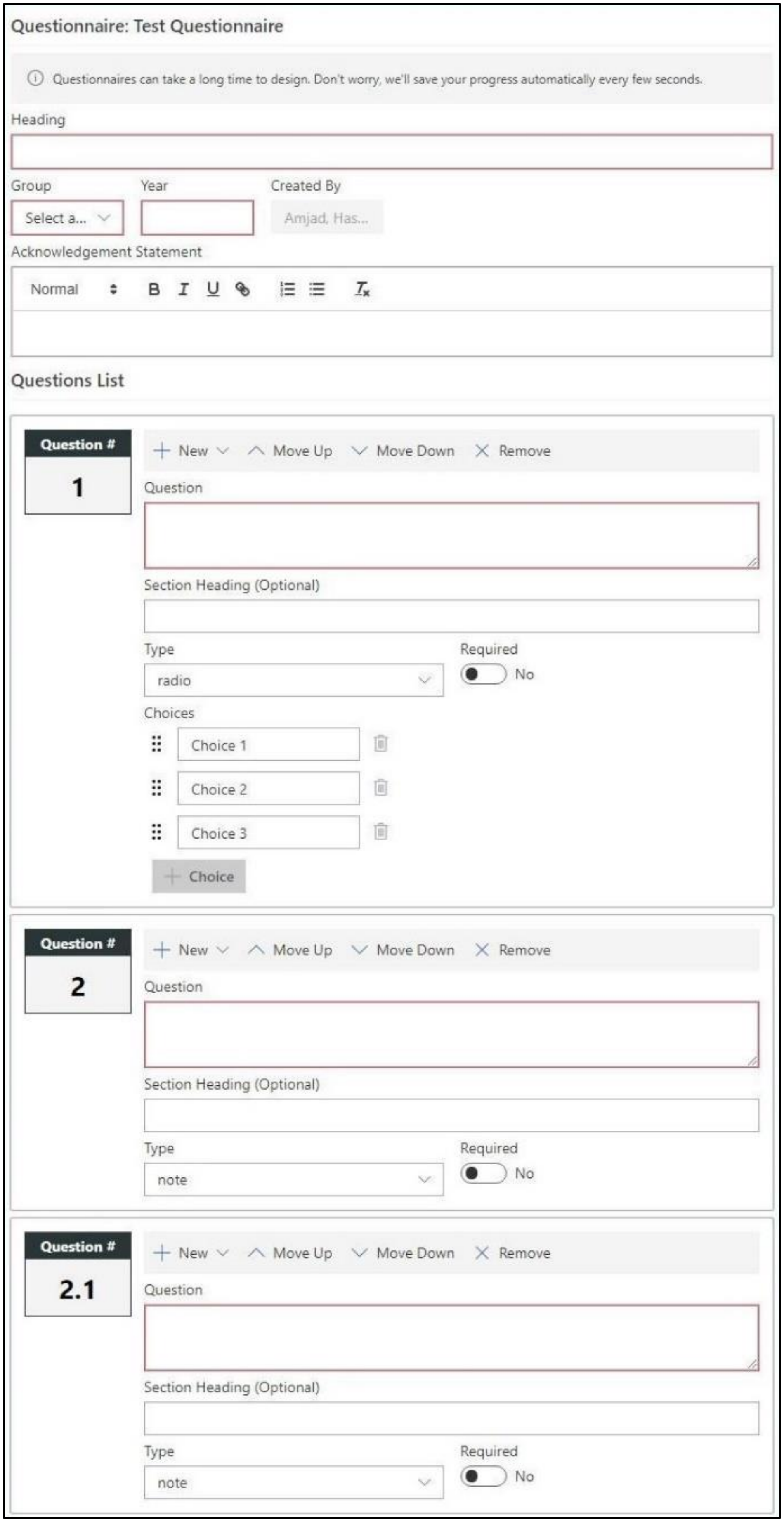
Sent To	Created	Status	
Only who has not Responded	03/28/2020	Notified	Open
Others	03/28/2020	Notified	Open
Only who has not Responded	03/28/2020	Notified	Open
Only who has not Responded	03/28/2020	Notified	Open
Only who has not Responded	03/28/2020	Notified	

Figure 1- Admin Dashboard - UI

Module: Survey Builder

The module allowed the application-admins to design and build surveys from scratch. Its main features are as follows:

- Configure question response types:
  - Dropdown
  - Radio
  - Text
  - Note
  - Static
- Define list of choices to display to the end user for questions whose response type is Radio or Dropdown.
  Note: Order of choices can be changed through drag and drop functionality.
- Configure follow up questions up to nth level based on answer of parent question.
- Re-order sibling questions.



Questionnaire: Test Questionnaire

ⓘ Questionnaires can take a long time to design. Don't worry, we'll save your progress automatically every few seconds.

Heading:

Group: Select a...

Year:

Created By: Amjad, Has...

Acknowledgement Statement: Normal

Questions List

Question # 1

Question:

Section Heading (Optional):

Type: radio

Required: No

Choices:

- Choice 1
- Choice 2
- Choice 3

Question # 2

Question:

Section Heading (Optional):

Type: note

Required: No

Question # 2.1

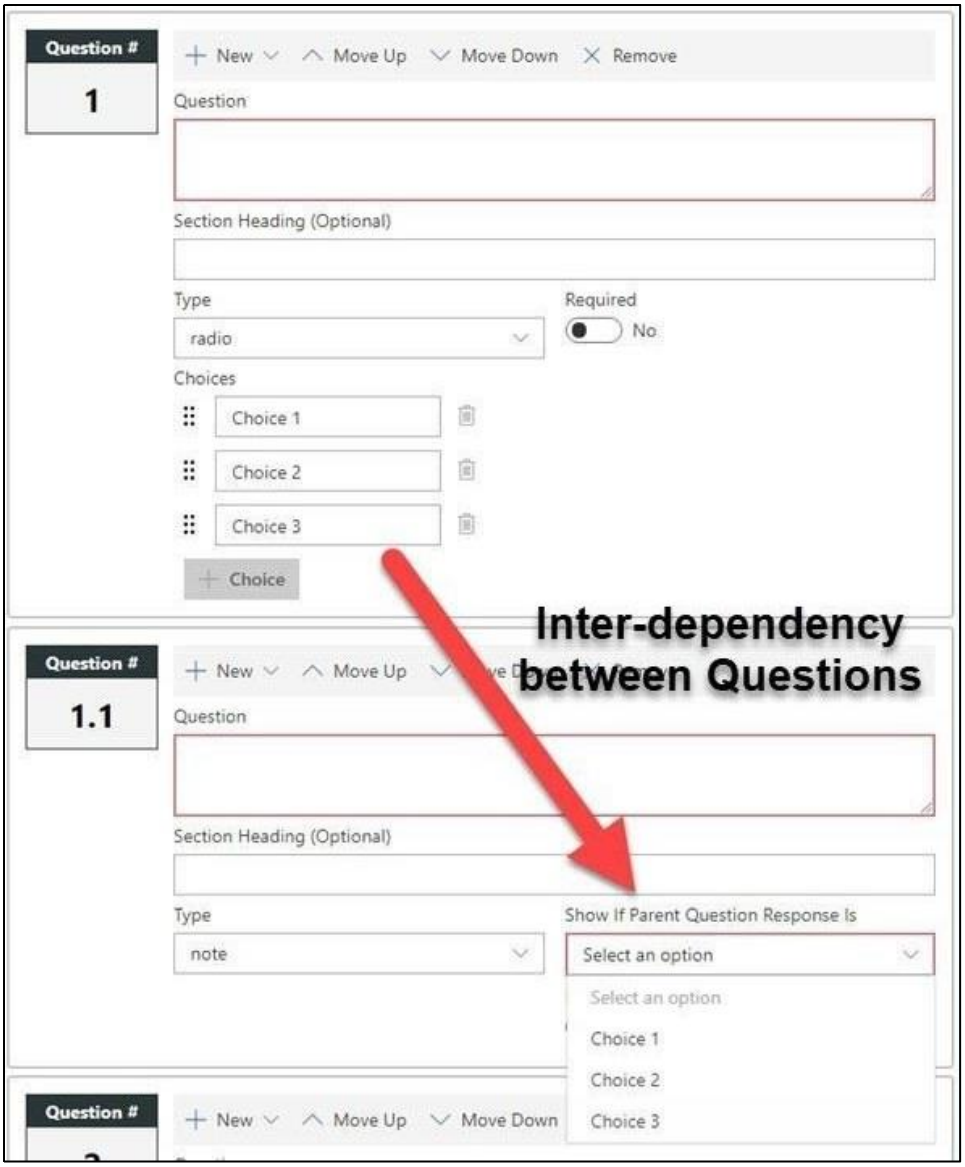
Question:

Section Heading (Optional):

Type: note

Required: No

Figure 2 - Questionnaire Designer - UI


Question #
1
New
Move Up
Move Down
Remove
Question
Section Heading (Optional)
Type
Required
Choices
Choice
Question #
1.1
New
Move Up
Question
Section Heading (Optional)
Type
Show If Parent Question Response Is
Select an option
Select an option
Choice 1
Choice 2
Choice 3
Question #
New
Move Up
Move Down

Type: radio

Required: No

Choices: Choice 1, Choice 2, Choice 3

Type: note

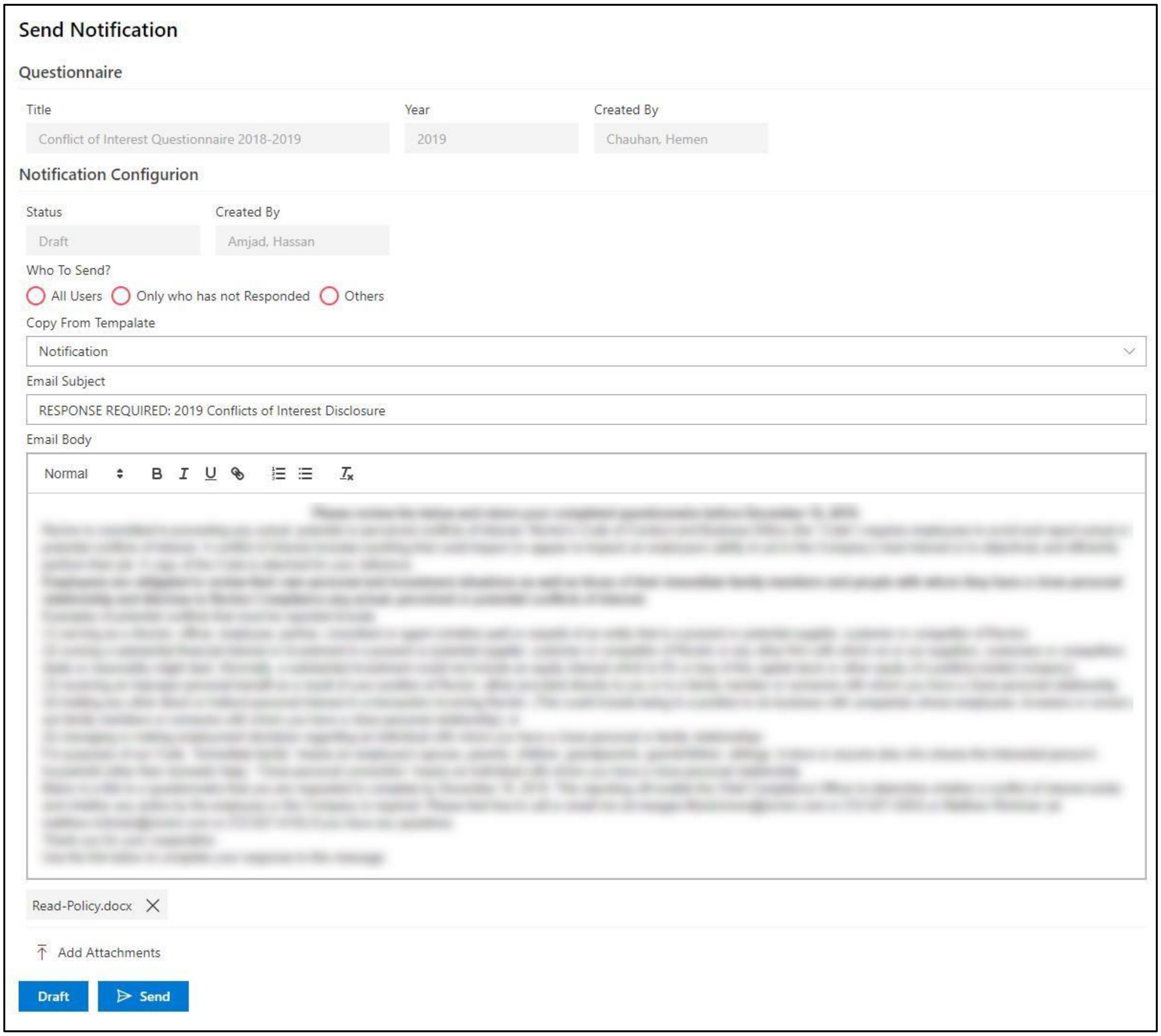
Inter-dependency between Questions

Figure 3 - Configuring Follow Up Questions - UI

Module: Send Notification

This module is used to send out customized notifications to the distribution list. Its main features are as follows:

- Copy email subject and body from pre-defined set of email template.
- Customize email subject and body.
- Customize source HTML of email body.
- Configure email attachments.
- Option to save the notification as draft or send it immediately.



Send Notification

Questionnaire

Title: Conflict of Interest Questionnaire 2018-2019

Year: 2019

Created By: Chauhan, Hemen

Notification Configurion

Status: Draft

Created By: Amjad, Hassan

Who To Send?
○ All Users ○ Only who has not Responded ○ Others

Copy From Tempalate: Notification

Email Subject: RESPONSE REQUIRED: 2019 Conflicts of Interest Disclosure

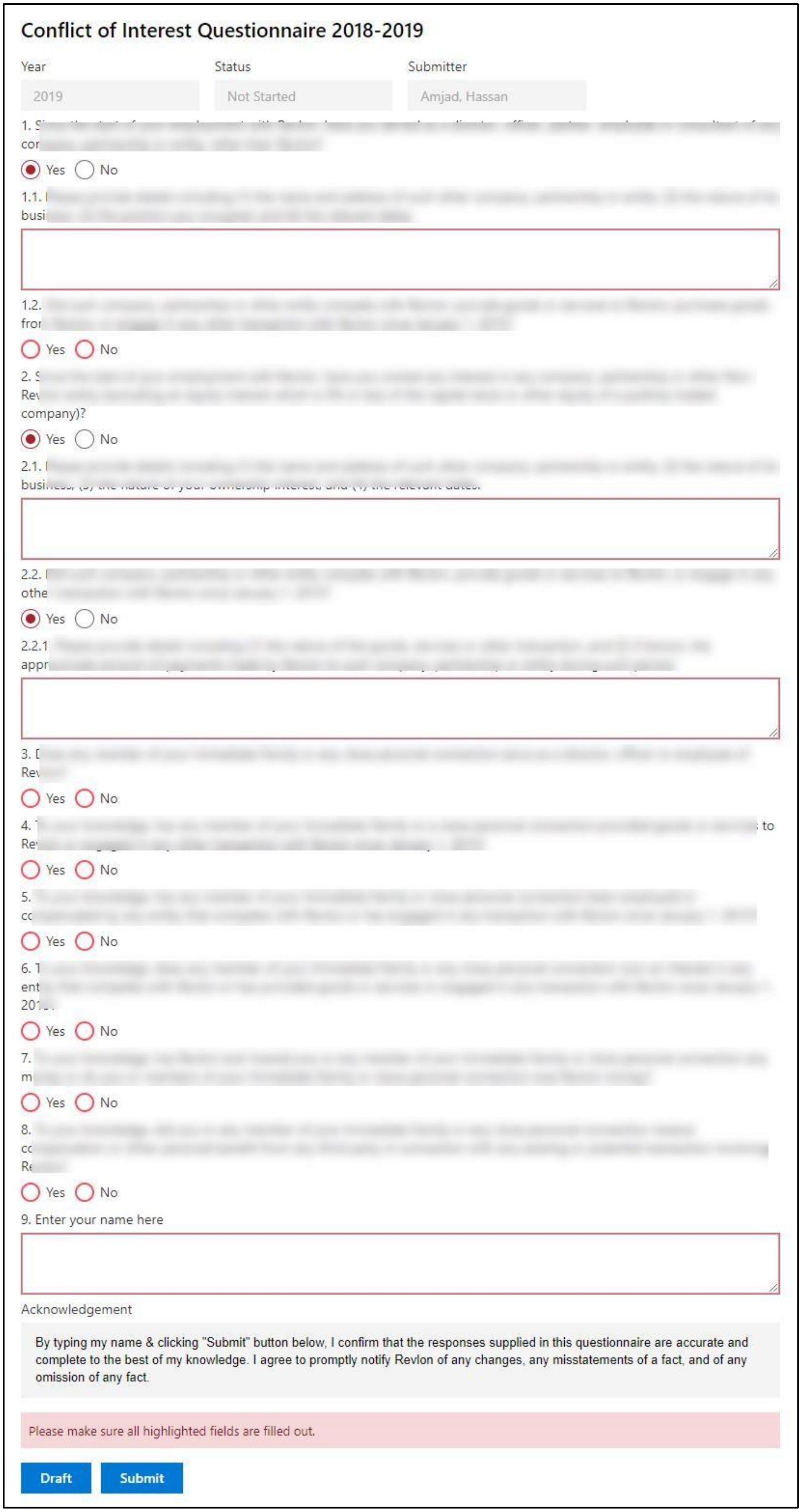
Email Body

Read-Policy.docx

Figure 4 - Send Notification Form - UI

Module: Survey

The is the actual survey that gets generated by the ‘Survey Builder’ module and is meant to be filled out and submitted by people in the distribution list. In terms of implementation, the trickiest part of this module was maintaining the ‘display’ property of each question based on the answer of its immediate parent question and also ensuring that the ‘display’ property of the question is ultimately determined by the answer of the top most parent in the hierarchy, not by its immediate parent. All the configuration data for this module is generated by the ‘Survey Builder’ module as stated earlier.



Conflict of Interest Questionnaire 2018-2019

Year	Status	Submitter
2019	Not Started	Amjad, Hassan

1. [redacted]

● Yes ○ No

1.1. [redacted]

1.2. [redacted]

○ Yes ○ No

2. [redacted] company)?

● Yes ○ No

2.1. [redacted]

2.2. [redacted]

● Yes ○ No

2.2.1 [redacted]

3. [redacted]

○ Yes ○ No

4. [redacted] to [redacted]

○ Yes ○ No

5. [redacted]

○ Yes ○ No

6. [redacted]

○ Yes ○ No

7. [redacted]

○ Yes ○ No

8. [redacted]

○ Yes ○ No

9. Enter your name here

Acknowledgement

By typing my name & clicking "Submit" button below, I confirm that the responses supplied in this questionnaire are accurate and complete to the best of my knowledge. I agree to promptly notify Revlon of any changes, any misstatements of a fact, and of any omission of any fact.

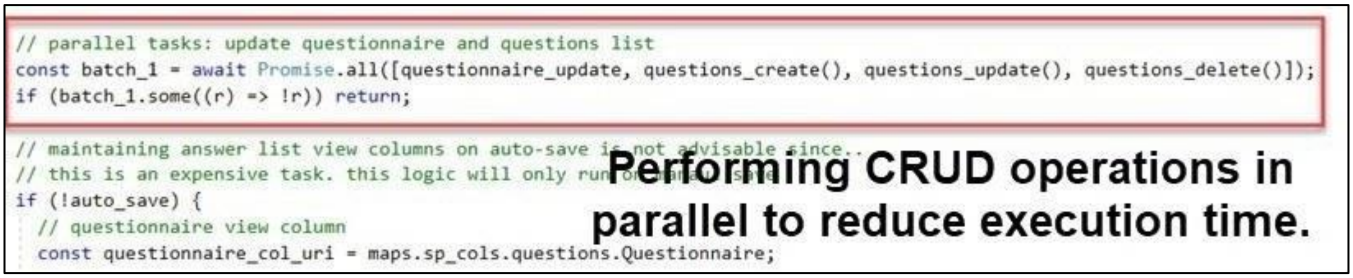
Please make sure all highlighted fields are filled out.

Figure 5 - Questionnaire Form - UI

Project Code Highlights

Batched API Operations

Batched non-dependent API calls to reduce overall processing time.

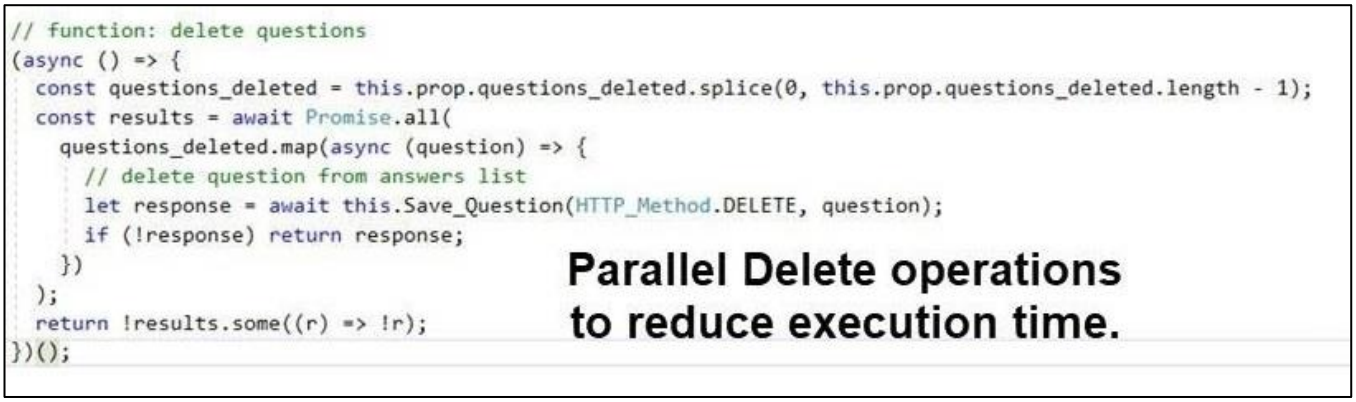


```
// parallel tasks: update questionnaire and questions list
const batch_1 = await Promise.all([questionnaire_update, questions_create(), questions_update(), questions_delete()]);
if (batch_1.some((r) => !r)) return;

// maintaining answer list view columns on auto-save is not advisable since..
// this is an expensive task. this logic will only run on manual save
if (!auto_save) {
  // questionnaire view column
  const questionnaire_col_uri = maps.sp_cols.questions.Questionnaire;
```

Performing CRUD operations in parallel to reduce execution time.

Figure 6 - Batched API Calls – Code Snippet



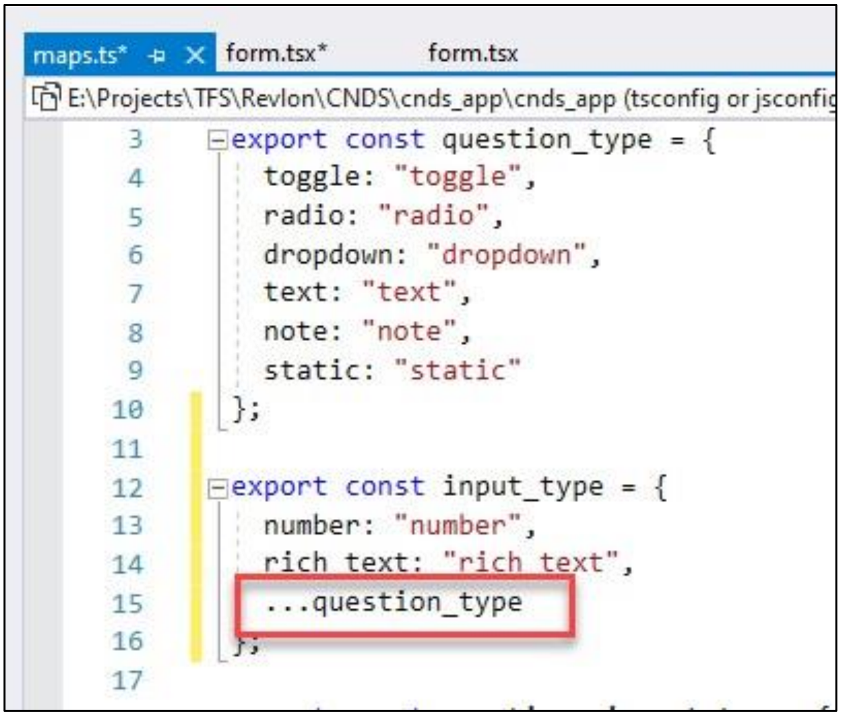
```
// function: delete questions
(async () => {
  const questions_deleted = this.prop.questions_deleted.splice(0, this.prop.questions_deleted.length - 1);
  const results = await Promise.all(
    questions_deleted.map(async (question) => {
      // delete question from answers list
      let response = await this.Save_Question(HTTP_Method.DELETE, question);
      if (!response) return response;
    })
  );
  return !results.some((r) => !r);
})();
```

Parallel Delete operations to reduce execution time.

Figure 7 - Batched Delete API Operations - Code Snippet

ES6 Spread Syntax

Use of ES6 spread syntax to extend or combine map variables.


maps.ts*
form.tsx*
form.tsx
E:\Projects\TFS\Revlon\CNDS\cnds_app\cnds_app (tsconfig or jsconfig

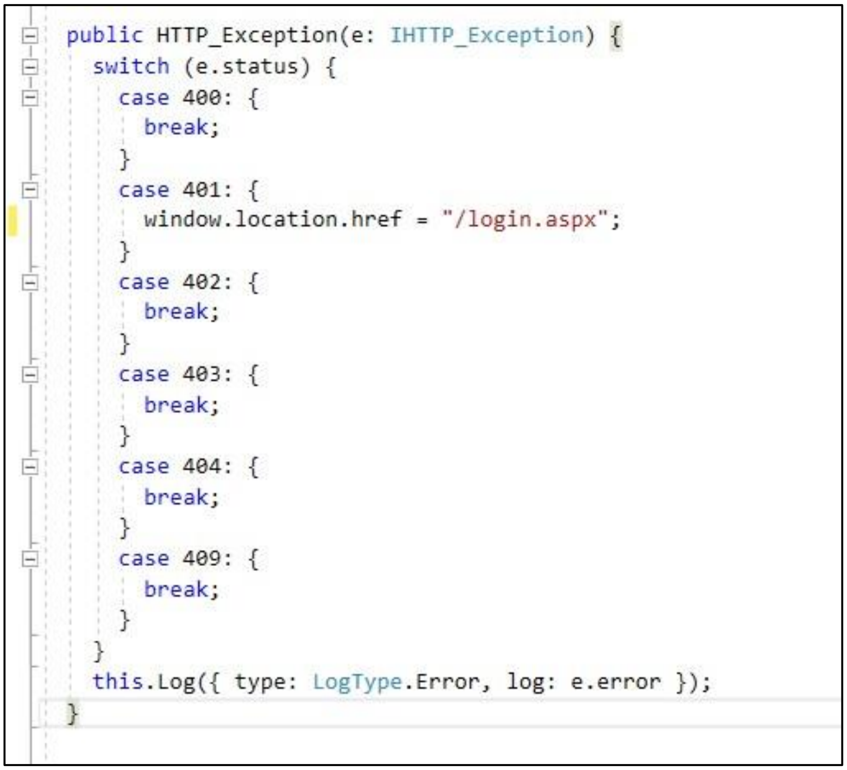
```
export const question_type = {
  toggle: "toggle",
  radio: "radio",
  dropdown: "dropdown",
  text: "text",
  note: "note",
  static: "static"
};

export const input_type = {
  number: "number",
  rich_text: "rich_text",
  ...question_type
};
```

Figure 8 - ES6 Spread Operator - Code Snippet

Handling HTTP Errors Efficiently

Clever use of switch statement for handling HTTP error codes.



```
public HTTP_Exception(e: IHTTP_Exception) {
  switch (e.status) {
    case 400: {
      break;
    }
    case 401: {
      window.location.href = "/login.aspx";
    }
    case 402: {
      break;
    }
    case 403: {
      break;
    }
    case 404: {
      break;
    }
    case 409: {
      break;
    }
  }
  this.Log({ type: LogType.Error, log: e.error });
}
```

Figure 9 - HTTP Error Handling - Code Snippet