Analysis of the Circle

Overview

The quiz/survey app has potential as part of a wider system. It could be used to get feedback on courses and training materials as well as the system itself. In it’s current form it gives a good way to feedback the answers, and has the flexibility to produce many more reports, including aggregations for higher level reporting if necessary.

One thing I think would add to the app is the ability to allow the user to respond with comments. These would require a little extending of the data to accommodate this additional payload. Additionally the processing of the comments into useable reports would require some work (you don't want to trawl through 1000’s of comments by hand), but I think using Natural Language Processing with Machine Learning would allow comments to be processed into ‘Good or Bad’ which would get better over time from a feedback loop. This would be an interesting project.

Using a file based system would not be my first choice (though I understand this was part of the development brief) (I have come from a SQL, noSQL document DB background). I think the bottlenecks currently will end up being around processing the “single thread” FIFO queue as it requires the output files only be used by one process at a time as in the example report file entries can be added from several surveys which could clash resolving in last save wins effectively loosing data that may be being added from a different thread. With a database such as DynamoDB as instead of a single file the data could be processed into individual records allowing multiple simultaneous posts to the database without loss of data.

There may be a solution to the single thread issue with thefiles but I would have to look in further and make sure it’s something that could be used to work with other report outputs.

An advantage of using a file based system is AWS has come good reporting tools that can be hooked into S3 file based systems to visualise data in graph form. Some of these tools don’t currently work directly with DynamoDB and would require data to be extracted into something that it does work them (ironically probably files in S3) so a file based or hybrid DB/file based system may be the right solution.

Issues

Front-end

Issues

- Inconsistent use of Bootstrap throughout the web application
- Accessibility/Invalid issues HTML (e.g. missing labels for input)
- Using Context for state management is OK but is quickly outgrown. I have looked at the Context code and feel there are some issues around the re rendering of the questions, this may just be how Context works, but I don’t have any experience using it (I have always started all my projects with Redux state management)
- No way to go back to the previous question (this will become more annoying as quizzes longer and the user makes mistakes or rethink their answers)
- No error handling when storing answers via API
- Quiz hard coded into application
- Source formatting is all over the place with some files using 2 spaces and others 4 there also seems to be a mixture of CRLF and LF line ends, which can cause issues when developing across mixed environments, e.g. Windows and Linux

Actions Taken

1. Fixed the inconsistencies across the app, making use of Bootstrap/reactstrap. I did not put any effort into restyling the front end just cleaned it up so it can be styled.
2. Fixed missing labels, ids and invalid HTML. To aid this I installed Axe into the browser to check the accessibility. The only issue left is the colour contrast on buttons which on the Bootstrap default is only 3.9:1 but WCAG 2 AA requires it to be at least 4.5:1. This is often an issue with branded websites as well as many companies do not have the amount of contrast in their colour schemes needed to satisfy WCAG 2 AA and understandably don't want to change their brand colours.
3. Moved state management from Context to Redux. While doing this I also streamlined and removed some of the processing that was taking place when loading the questions and saving the answers.
4. Added a ‘back’ action that pops the last element of the answers array and sets current question to popped answer question id. This allows the user to go back and re answer questions differently. This is not such a big issue on the supplied quiz but could be tiresome in longer quizzes if users miss-click etc.
5. Added simple error handling to the saving of the answers. In the original there was no indication when any error had occurred when when saving the quiz, added a simple Saving… text and error Alert giving the user some feedback, I have not added any way to retry, but this could be added as a button that attempts the call again.

6. Added loading of questions from file. This will allow a new quiz to be uploaded without the rebuilding and deploying of a new website. It is load from the API with some work at the back-end.
7. Installed and set-up eslint to help with consistency of code formatting and to prompt where best practices are not being adhered to. This has been set up to auto format the code on save (in VS Code).

Back-end

Issues

- listenCapture
  - Both GET and POST are allowed, this causes issues as GET has no Body which ends up with the function creating an empty JSON raw document.
- listenQuestionTrigger
  - The system doesn't check the key is of the expected format, this could cause issues if someone uploads a random file into the S3 bucket.
- listenDataStore
  - Bug found. When saving code to report documents, array is initialised with a number of empty elements which cause empty elements to be written to the report document. Single queue processing could cause a bottleneck but this looks to be the best way to get the correct data without adding much more complexity

Actions Taken

- listenCapture
  - Used GET method on API call to load Quizzes from server. Added code to respond to GET request with quiz from new quizzes bucket, CloudFormation files changed to add new bucket and give access to ListenCapture Lambda
- listenQuestionTrigger
  - No action taken at this time
- listenDataStore
  - Fixed the bug

Installation notes

There are two additional for installation (in the root README.MD but repeated here)

Once the stack has been created upload the 0.json file from the quizzes folder into the quizzes bucket root

If this file is omitted an error message will be displayed when the client is run