Survey App Design

Saturday, April 14, 2018 8:01 AM

Architecture

Front-end: React/Mineral hosted in S3.
Back-end: Serverless on AWS (API Gateway -> DynamoDB).

Dev Env for Client

Install latest nodejs

Create a CDSurvey dir.
Open a command window in the CDSurvey dir.

- Install node modules
  - npm install --save react
  - npm install --save react-router-dom
  - npm install --save mineral-ui
  - npm install --save mineral-ui-icons
  - npm install --save bad-words
- Make sure you have Mineral 0.28 or higher (lower will work in browsers other than IE - 0.28 includes the fix so it will work in IE too)
- Download the /src and /public directories and root level files from the cdsurvey GIT repo /stephenjtyler2/cdsurvey into the CDSurvey dir.
- You should now have this file structure
  - /CDSurvey
    - /src
    - /public
    - /node-modules
- Start server
  - npm start to the run locally in dev mode.
- Browser should open automatically and load the home page but if not: http://localhost:3000

config.js contains the URL of the API Gateway endpoint exposing the server side functionality, whether the app is running in Broken mode or not, the BuildNum that will be displayed and the questions.
Example:

```
{
	"ServerURL" : "https://7yi3oi9x2a.execute-api.us-east-1.amazonaws.com/CDSurveyProdStage",
	"Broken" : "false",
	"BuildNum" : "41",
	"Questions" :
	[
		{
			"question": "How many days per week do Engineers spend picking their noses?",
			"answers" : [
				"Eight.",
				"It can vary signficantly from engineer to engineer.",
				"None.   They have built machines for that.",
				"Nobody ... err ... nose"
```

```
            ],
            "correctAnswer": "1"
        },
        {
            "question": "What is the most amusing anagram of Stephen Feloney?",
            "answers" : [
                "Elf Hoe Spent Yen",
                "Feel Those Penny",
                "Pee Hefty Nelson",
                "Tense Phoney Elf"
            ],
            "correctAnswer": "4"

        },
        {
            "question": "Fill in the blank: Shift ___",
            "answers" : [
                "Left",
                "Right",
                "Up",
                "Happens"
            ],
            "correctAnswer": "4"
        },
        {
            "question": "What should the fourth question be?",
            "answers" : [
                "What sport would be the funniest to add a mandatory amount of alcohol to?",
                "What would be the coolest animal to scale up to the size of a horse?",
                "How many chickens would it take to kill an elephant?",
                "What’s the best type of cheese?"
            ],
            "correctAnswer": "2"
        }
    ]
}
```

Deployment

Build the app: in the CDSurvey directory execute: *npm run build*. This will create a /build directory containing a minimized version of the application. Only the contents of /build need to be uploaded to S3, not the rest of the CDSurvey dir. The src lives in GIT, not S3. S3 is hosting the "compiled" version of the code.

When uploading the /build dir to S3 you must:
- make all files public
- on the config.js and _broken and _fixed variants, set Metadata as follows:
  - cache-control : no-cache, no-store, max-age=0, must-revalidate
  - Expires: Fri, 01 Jan 1990 00:00:00 GMT

The .js files for the components (the various pages) are under src/routes. Src/routes/Index.js contains the main page layout and route path matching instructions. The code to render each page is under /routes e.g. /routes/Admin.js has the code for the Admin page. /routes/index.js has the common header and the path <-> component mappings for the content.

DynamoDB Config

Simple table storing 16 counters.

- TableName: CDSurveyResponses
- PartitionKey: questionId (number)
- SortKey: responseId (number)
- Attribute: counterValue (number)

You MUST manually create 16 entries for questionId 1-4 and responseId 1-4. All with counterValue 0. Be careful with spelling and capitalization and data types of the attributes. The app will not work if you get any of this wrong.

At the start of survey, admin ui will issue an update to zero all the counters.

During survey on user hitting Next/Finish the client will make an AJAX call to API GW to increment the relevant counters in DynamoDB.

Not storing individual responses (i.e. one row per userId, questionId, responseId) because I don't actually need that info for any reasons, and I need to optimize for select count(*) where question ID = n in order to update the stats pages for admin and user UI which is a lot more expensive than a simple SCAN of a 16 row table.

Also at the moment not storing the names that the participants enter as there is no requirement to do anything with them. That may change.

IAM Config

Create a role called CDSurveyAPIToDynamoDBRole with the policies:

- AmazonAPIGatewayPushToCloudWatchLogs
- AmazonDynamoDBFullAccess

API Gateway Config

Create API called CDSurvey

GET / returns all 16 rows of the table as JSON.

a. Call with empty body.
b. Integration Request -
    i. DynamoDB service call
        1) role: arn of the CDSurveyAPIToDynamoDBRole
        2) HTTP Method: POST
        3) Action: Scan
    ii. body mapping is { "TableName" : "CDSurveyResponses" }

PUT / increments the counter for a given questionId/responseId

a. Call with body:
    i. {"questionId":"1", "responseId":"1"}
b. Integration Request

   i. role: arn of the CDSurveyAPIToDynamoDBRole
   ii. HTTP Method: POST
   iii. Action: UpdateItem
   iv. body mapping is

```
{
   "TableName": "CDSurveyResponses",
   "Key": {
            "questionId": {
        "N": "$input.path('$.questionId')"
        },
      "responseId": {
         "N": "$input.path('$.responseId')"
      }
   },
   "UpdateExpression": "set counterValue = counterValue + :inc",
   "ExpressionAttributeValues": {
      ":inc": {"N": "1"}
   }
}
```

POST / sets the counter to zero

a. Call with body:
   i. {"questionId":"1",   "responseId":"1"}
b. Integration request
   i. role: arn of the CDSurveyAPIToDynamoDBRole
   ii. HTTP Method: POST
   iii. Action: UpdateItem
   iv. body mapping is

```
{
   "TableName": "CDSurveyResponses",
   "Key": {
            "questionId": {
        "N": "$input.path('$.questionId')"
        },
      "responseId": {
         "N": "$input.path('$.responseId')"
      }
   },
   "UpdateExpression": "set counterValue = :counterValue",
   "ExpressionAttributeValues": {
      ":counterValue": {"N": "0"}
   }
}
```

Enable CORS on the API and deploy it to a stage.

Note: the API URL, and edit the config.js in the client app with the API URL.

Note: I tried using BatchWriteItem API to reset all counters in one call but could not get it to work - weird errors from Dynamo API, so the client from the Admin page instead makes 16 calls on the POST call above to zero the counters.