You need to create a system for a pizza ordering app. This system should be designed with the following specs in mind:

- I want to be able to query all the pizza types and variants from the database via an API

- Each order must have an email address associated to it, so it can notify the customer of any status change of the order. I want be able to query all the orders that were placed using a certain email.

- I must be notified via email each time the status of my order changes (e.g. order received, order being prepared, order being delivered, etc.)

- A receipt has to be generated and stored in S3 for each order (don't get fancy with it, a .txt will do). Once generated and saved, a copy of the file has to be sent to the customer's provided email.

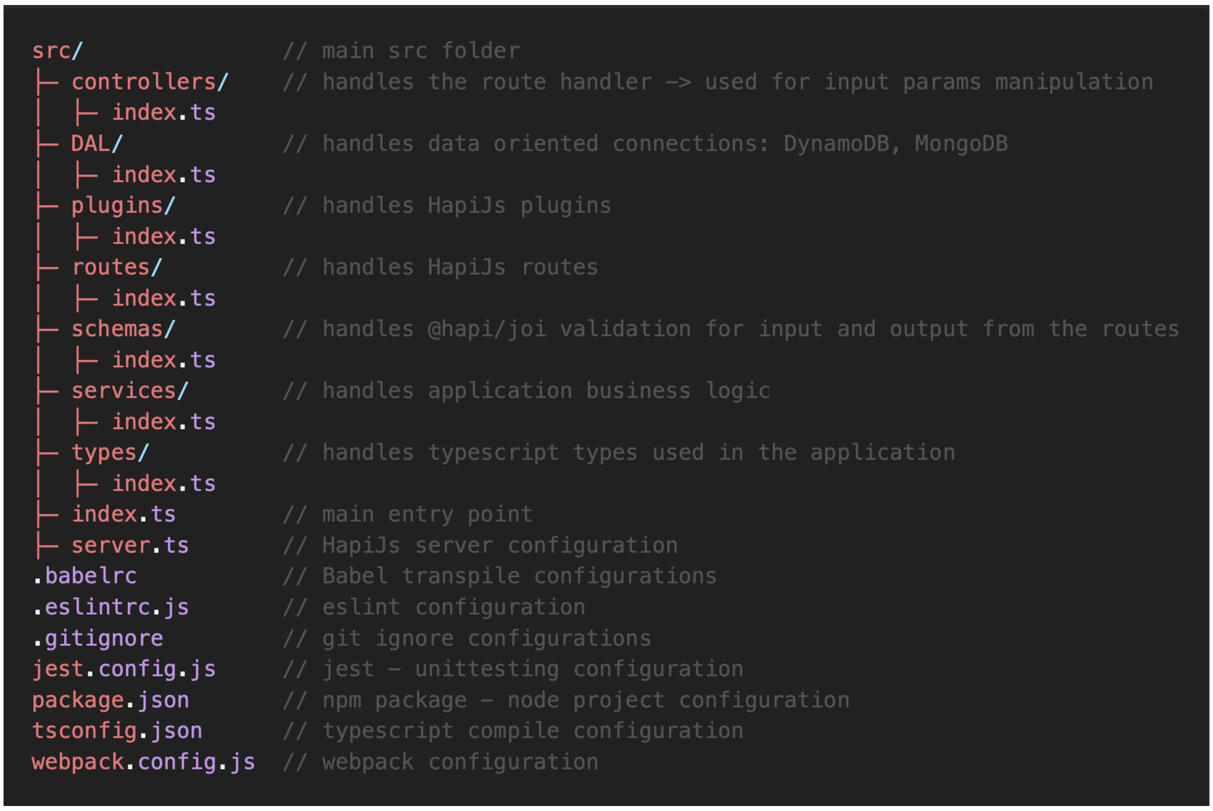
Tech stack you should use:
TypeScript (4+), node.js (12+), hapi.js, DynamoDB, DynamoDB Streams, S3, SQS, SNS, Lambda, Serverless Framework, SES

Notes:

- There's no need for a UI interface. Placing orders and querying the system can be done via Postman / cURL or any other similar tool of your choice.
- The REST API you're going to build has to be wrapped into a Lambda (https://www.serverless.com/blog/serverless-express-rest-api)
- Try sticking to the file structure shown below

Useful links:
https://docs.aws.amazon.com/sns/latest/dg/sns-getting-started.html
https://www.dynamodbguide.com/
https://www.serverless.com/
https://hapi.dev/tutorials/?lang=en_US
https://docs.aws.amazon.com/sdk-for-javascript/v3/developer-guide/getting-started-nodejs.html

https://github.com/prisma-labs/serverless-plugin-typescript
https://www.npmjs.com/package/serverless-offline



```
src/                // main src folder
├─ controllers/     // handles the route handler -> used for input params manipulation
│  ├─ index.ts
├─ DAL/             // handles data oriented connections: DynamoDB, MongoDB
│  ├─ index.ts
├─ plugins/         // handles HapiJs plugins
│  ├─ index.ts
├─ routes/          // handles HapiJs routes
│  ├─ index.ts
├─ schemas/         // handles @hapi/joi validation for input and output from the routes
│  ├─ index.ts
├─ services/        // handles application business logic
│  ├─ index.ts
├─ types/           // handles typescript types used in the application
│  ├─ index.ts
├─ index.ts         // main entry point
├─ server.ts        // HapiJs server configuration
.babelrc            // Babel transpile configurations
.eslintrc.js        // eslint configuration
.gitignore          // git ignore configurations
jest.config.js      // jest - unittesting configuration
package.json        // npm package - node project configuration
tsconfig.json       // typescript compile configuration
webpack.config.js   // webpack configuration
```