Project 4 - Develop and Deploy Serverless Applications - My Submission

GitHub Repository	https://github.com/rebhartell/udacity-aws-cloud-developer-project-4

My Submission documentation	This document, images of the UI, AWS, and Postman, and additional material.
	
	• My_Submission
	  ◦ 000_My_Submission.pdf
	  ◦ 001_UI_Auth_Login.png
	  ◦ 002_UI_User_1_GetTodos.png
	  ◦ 003_UI_User_2_GetTodos.png
	  ◦ 004_UI_CreateNewTodo.png
	  ◦ 005_UI_New_Todo.png
	  ◦ 006_UI_Upload_New_Image.png
	  ◦ 007_UI_Todo_with_Attachment.png
	  ◦ 008_UI_Update_Todo_as_done.png
	  ◦ 100_AWS_API_Gateway.png
	  ◦ 110_AWS_Lambda.png
	  ◦ 120_AWS_DynamoDb.png
	  ◦ 130_AWS_S3_Attachments.png
	  ◦ 140_AWS_Cloudwatch_Log_Groups.png
	  ◦ 141_AWS_Cloudwatch_Auth_log.png
	  ◦ 142_AWS_Cloudwatch_GenerateUploadUrl_log.png
	  ◦ 150_AWS_X-Ray_Service_Map.png
	  ◦ 200_Postman_Collection_GetAllTodos.png
	  ◦ 201_Postman_Collection_Runner.png
	  ◦ 300_Serverless_Deploy_Output.txt

UI Guest Account	guest@test.com Password+1

1) Functionality

CRITERIA	MEETS SPECIFICATIONS	COMMENTS
The application allows users to create, update, delete TODO items	A user of the web application can use the interface to create, delete and complete a TODO item.	Done
The application allows users to upload a file.	A user of the web interface can click on a "pencil" button, then select and upload a file. A file should appear in the list of TODO items on the home page.	Done
The application only displays TODO items for a logged in user.	If you log out from a current user and log in as a different user, the application should not show TODO items created by the first account.	Done
Authentication is implemented and does not allow unauthenticated access.	A user needs to authenticate in order to use an application.	Done

2) Code Base

CRITERIA	MEETS SPECIFICATIONS	COMMENTS
The code is split into multiple layers separating business logic from I/O related code.	Code of Lambda functions is split into multiple files/classes. The business logic of an application is separated from code for database access, file storage, and code related to AWS Lambda.	Done
Code is implemented using async/await and Promises without using callbacks.	To get results of asynchronous operations, a student is using async/await constructs instead of passing callbacks.	Done

3) Best Practices

CRITERIA	MEETS SPECIFICATIONS	COMMENTS
All resources in the application are defined in the "serverless.yml" file	All resources needed by an application are defined in the "serverless.yml". A developer does not need to create them	Done

	manually using AWS console.	
Each function has its own set of permissions.	Instead of defining all permissions under **provider/iamRoleStatements**, permissions are defined per function in the **functions** section of the "serverless.yml".	Done
Application has sufficient monitoring.	Application has at least some of the following: • Distributed tracing is enabled • It has a sufficient amount of log statements • It generates application level metrics	Done • Winston logger • AWS X-Ray See images
HTTP requests are validated	Incoming HTTP requests are validated either in Lambda handlers or using request validation in API Gateway. The latter can be done either using the **serverless-reqvalidator-plugin** or by providing request schemas in function definitions.	Done Schemas in function definitions. MinLength and Pattern checks used Could not get path variable validation working nor media type rejection (seems to change JSON objects to text)

4) Architecture

CRITERIA	MEETS SPECIFICATIONS	COMMENTS
Data is stored in a table with a composite key.	1:M (1 to many) relationship between users and TODO items is modeled using a DynamoDB table that has a composite key with both partition and sort keys. Should be defined similar to this:	Done Partition Id: userId Sort Key: todoId Index Partition Id: userId Sort Key: dueDate
Scan operation is not used to read data from a database.	TODO items are fetched using the "query()" method and not "scan()" method (which is less efficient on large datasets)	Done

Suggestions to Make Your Project Stand Out!

1. Fetch a certificate from Auth0 instead of hard coding it in an authorizer.
    1. **Done**
2. Implement pagination support to work around a DynamoDB limitation that allows up to 1MB of data using a query method.
    1. **NOT Done**
3. Add your own domain name to the service.
    1. **NOT Done**
4. Add an ability to sort TODOs by due date or priority (this will require adding new indexes).
    1. **Partially Done - Index added to sort by dueDate**
5. Implement a new endpoint that allows sending full-text search requests to Elasticsearch (this would require copying data from DynamoDB to Elasticsearch as we did in lesson 4).
    1. **NOT Done**
6. Postman tests using variables from test to test and a large suite of tests for regression testing
    1. **Done**