Name: Jaime Valencia
Program: CLOUD DEVELOPER
Project: Capstone **- Newsletter Publisher Serverless**
Github: https://github.com/vaduinc/cloud-developer

This application allows users to sign in to a Web application using Auth0 third party provider. Users can create and subscribe to newsletters. Each newsletter can be published only by its owner/creator. Publishing consists of uploading a PDF file to the system and sending it by email to all its subscribers.

The Newsletter Publisher system is composed of two main parts; front-end application developed using Reactjs and back-end using serverless technology.

The technical details of the system (architecture, technologies used, models, etc) and how to use it are described in the next pages.

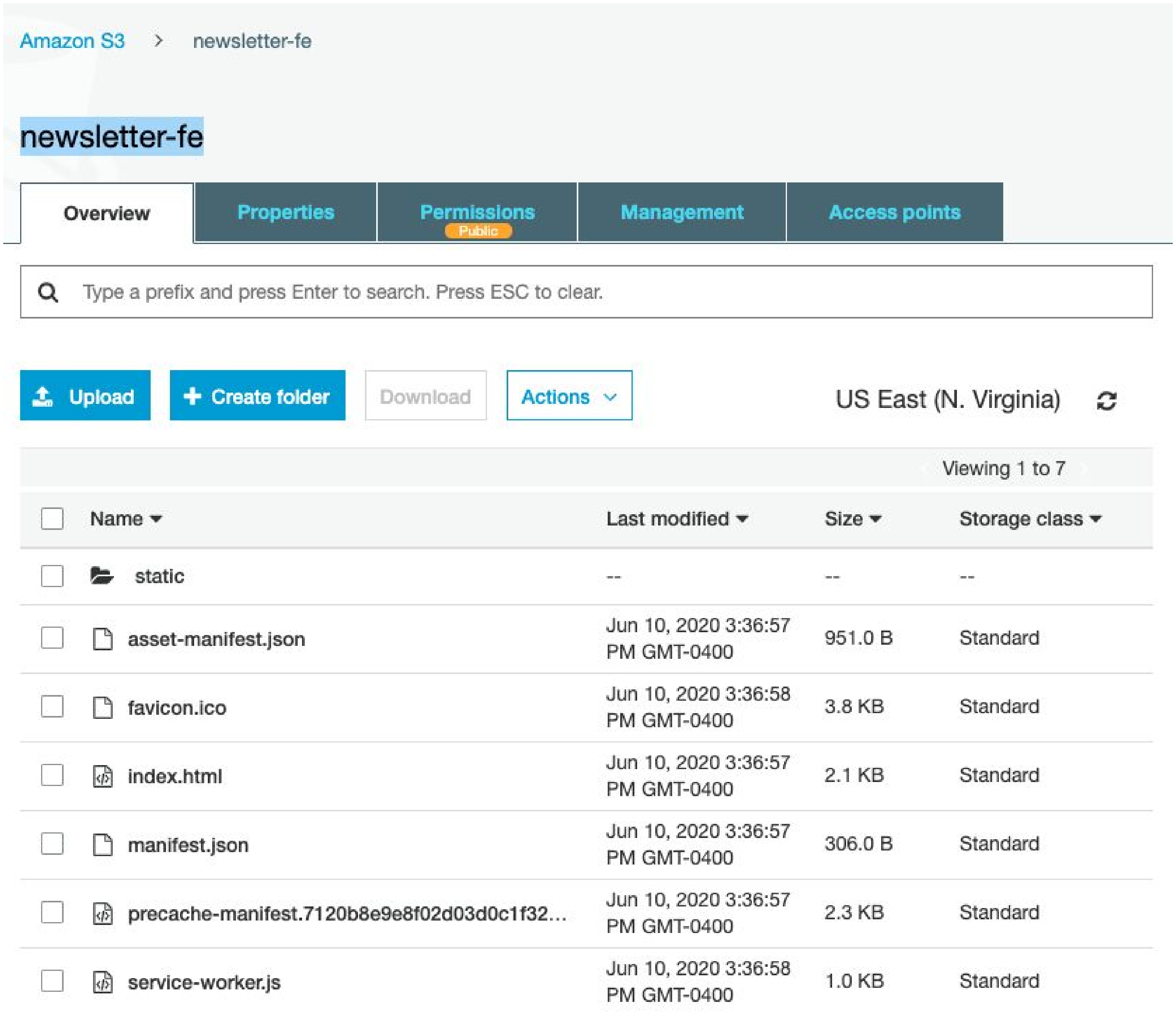
Note:

- Think of the system like each user can create their own digital magazine and anyone else in the system can subscribe to it. Each publication (PDF) can be accessed in the system and also would be sent by email.
- The system is supposed to be used for PDF files but users can upload any type of file.
- A Postman file is provided to execute the API back-end calls: ***Udacity-JV-Capstone.postman_collection.json***
- Users will receive email notifications from my personal email account; jaime.vadu@gmail.com This is the account setup in aws SES service. If you are testing the application and want to test the publish functionality to received newsletters make sure you check in the spam folder of the account you register in the profile.

Architecture

Front-End

the ui was created using Reactjs. It was based on the existing code from the project #4 serverless. It is deployed on a AWS S3 bucket - ***newsletter-fe*** - see next picture



Name	Last modified	Size	Storage class
static	--	--	--
asset-manifest.json	Jun 10, 2020 3:36:57 PM GMT-0400	951.0 B	Standard
favicon.ico	Jun 10, 2020 3:36:58 PM GMT-0400	3.8 KB	Standard
index.html	Jun 10, 2020 3:36:57 PM GMT-0400	2.1 KB	Standard
manifest.json	Jun 10, 2020 3:36:57 PM GMT-0400	306.0 B	Standard
precache-manifest.7120b8e9e8f02d03d0c1f32...	Jun 10, 2020 3:36:57 PM GMT-0400	2.3 KB	Standard
service-worker.js	Jun 10, 2020 3:36:58 PM GMT-0400	1.0 KB	Standard

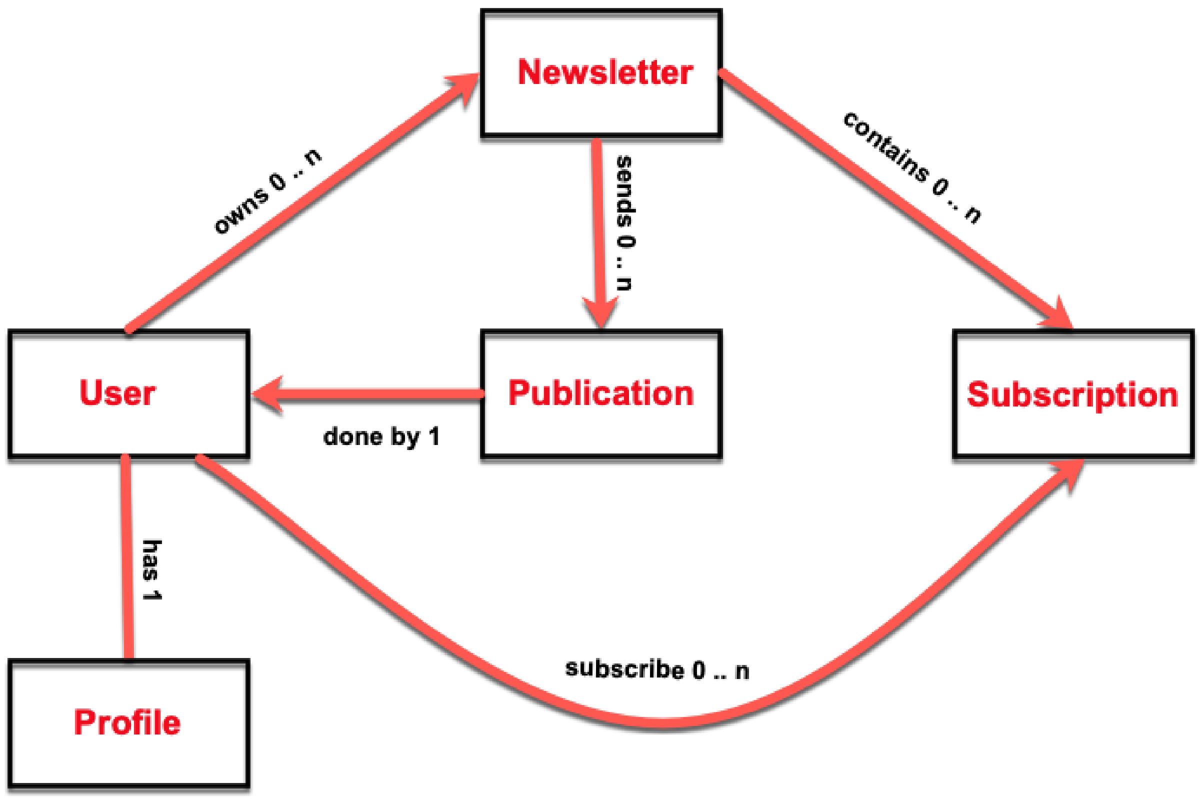
An Auth0 account was created and integrated with the application to provide the registration and login-in to the Web application.

Back-End

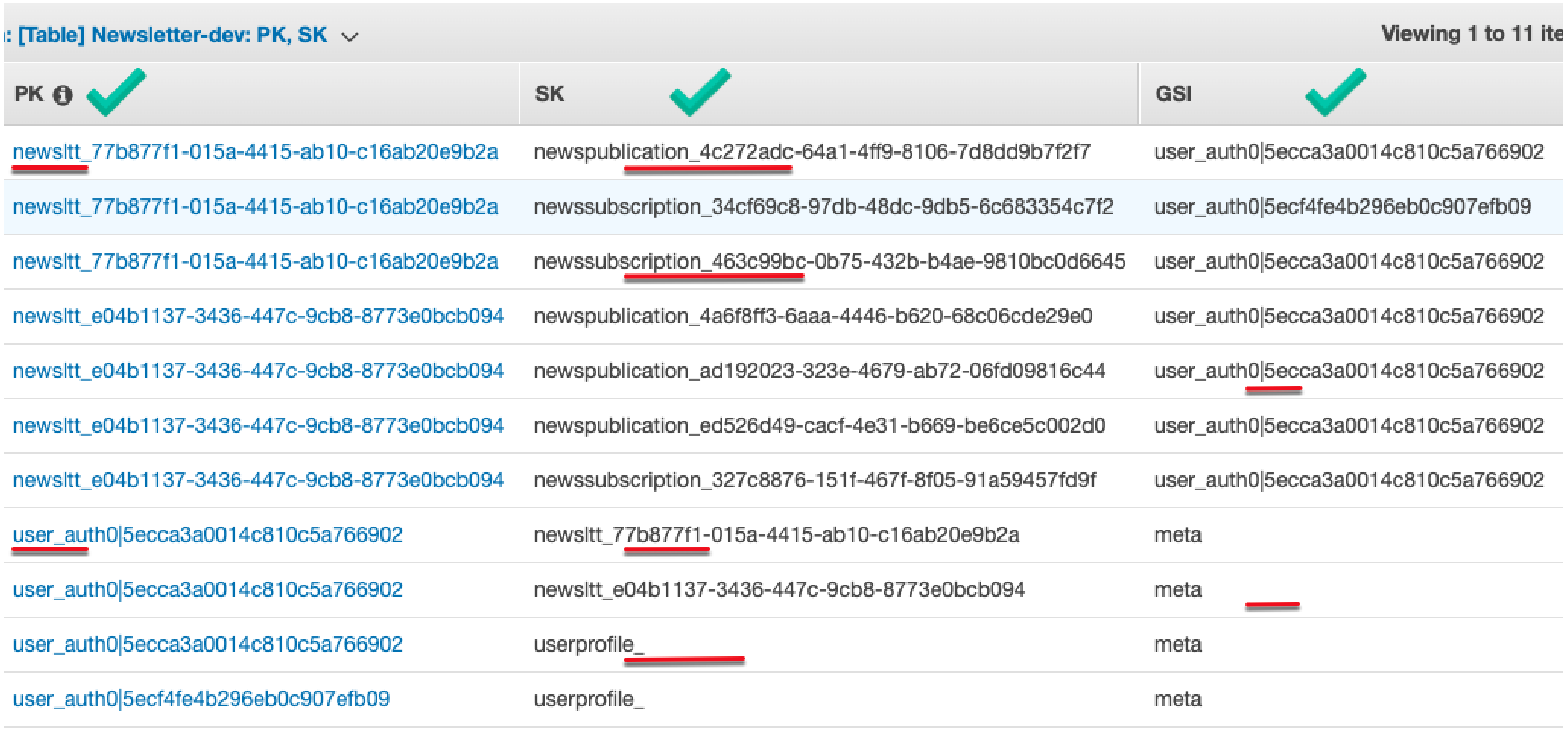
It is composed of several technologies and pieces Api-Gateway, DynamoDB, Lambda (Nodejs), IAM, CloudFormation, AWS-X-ray, SES, SNS, etc. All of these resources are built/created using serverless framework with ***serverless.yaml*** definition file.

Model and Database

Following describes the entity-model that explains the relationships between the different entities in the application/system.



To represent this mode in the persistent layer a table was created - ***Newsletter-dev*** - in DynamoDB. It is the only table in the application and with the combination of these three special properties/columns; Hash key, Sort Key, Global Secondary Index, and some prefix labels and some additional columns depending on the entity. The system can model any one-one, one-many, many-many and many-one relationship between the above entities. Following videos (https://youtu.be/ZvASCMxOFIU) explain the inner workings on how you can represent any of the above relations using only one table and some labels. The end result in the next picture.



: [Table] Newsletter-dev: PK, SK

Viewing 1 to 11 ite

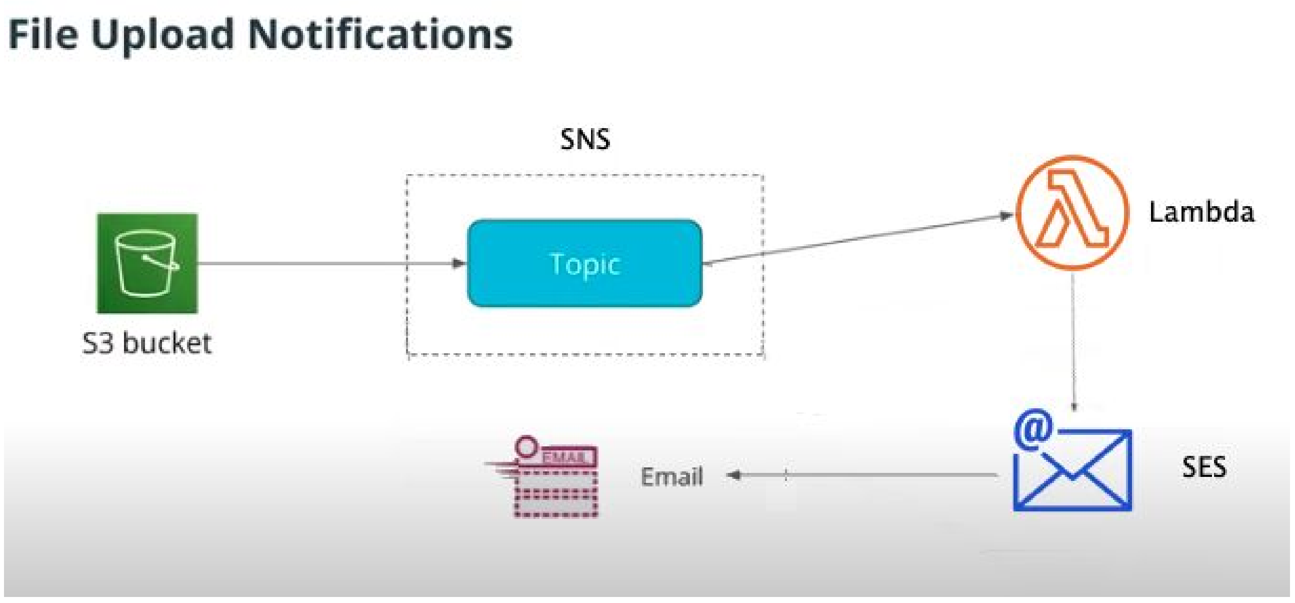
PK ✓	SK ✓	GSI ✓
newsltt_77b877f1-015a-4415-ab10-c16ab20e9b2a	newspublication_4c272adc-64a1-4ff9-8106-7d8dd9b7f2f7	user_auth0|5ecca3a0014c810c5a766902
newsltt_77b877f1-015a-4415-ab10-c16ab20e9b2a	newssubscription_34cf69c8-97db-48dc-9db5-6c683354c7f2	user_auth0|5ecf4fe4b296eb0c907efb09
newsltt_77b877f1-015a-4415-ab10-c16ab20e9b2a	newssubscription_463c99bc-0b75-432b-b4ae-9810bc0d6645	user_auth0|5ecca3a0014c810c5a766902
newsltt_e04b1137-3436-447c-9cb8-8773e0bcb094	newspublication_4a6f8ff3-6aaa-4446-b620-68c06cde29e0	user_auth0|5ecca3a0014c810c5a766902
newsltt_e04b1137-3436-447c-9cb8-8773e0bcb094	newspublication_ad192023-323e-4679-ab72-06fd09816c44	user_auth0|5ecca3a0014c810c5a766902
newsltt_e04b1137-3436-447c-9cb8-8773e0bcb094	newspublication_ed526d49-cacf-4e31-b669-be6ce5c002d0	user_auth0|5ecca3a0014c810c5a766902
newsltt_e04b1137-3436-447c-9cb8-8773e0bcb094	newssubscription_327c8876-151f-467f-8f05-91a59457fd9f	user_auth0|5ecca3a0014c810c5a766902
user_auth0|5ecca3a0014c810c5a766902	newsltt_77b877f1-015a-4415-ab10-c16ab20e9b2a	meta
user_auth0|5ecca3a0014c810c5a766902	newsltt_e04b1137-3436-447c-9cb8-8773e0bcb094	meta
user_auth0|5ecca3a0014c810c5a766902	userprofile_	meta
user_auth0|5ecf4fe4b296eb0c907efb09	userprofile_	meta

See the following TABLE definition from the *serverless.yml* file definition.

```
  NewsletterTable:
    Type: AWS::DynamoDB::Table
    Properties:
      AttributeDefinitions:
        - AttributeName: PK
          AttributeType: S
        - AttributeName: SK
          AttributeType: S
        - AttributeName: GSI
          AttributeType: S
      KeySchema:
        - AttributeName: PK
          KeyType: HASH
        - AttributeName: SK
          KeyType: RANGE
      BillingMode: PAY_PER_REQUEST
      TableName: ${self:provider.environment.NEWSLETTER_TABLE}
      GlobalSecondaryIndexes:
        - IndexName: ${self:provider.environment.INDEX_NAME}
          KeySchema:
            - AttributeName: GSI
              KeyType: HASH
            - AttributeName: SK
              KeyType: RANGE
          Projection:
            ProjectionType: ALL
```

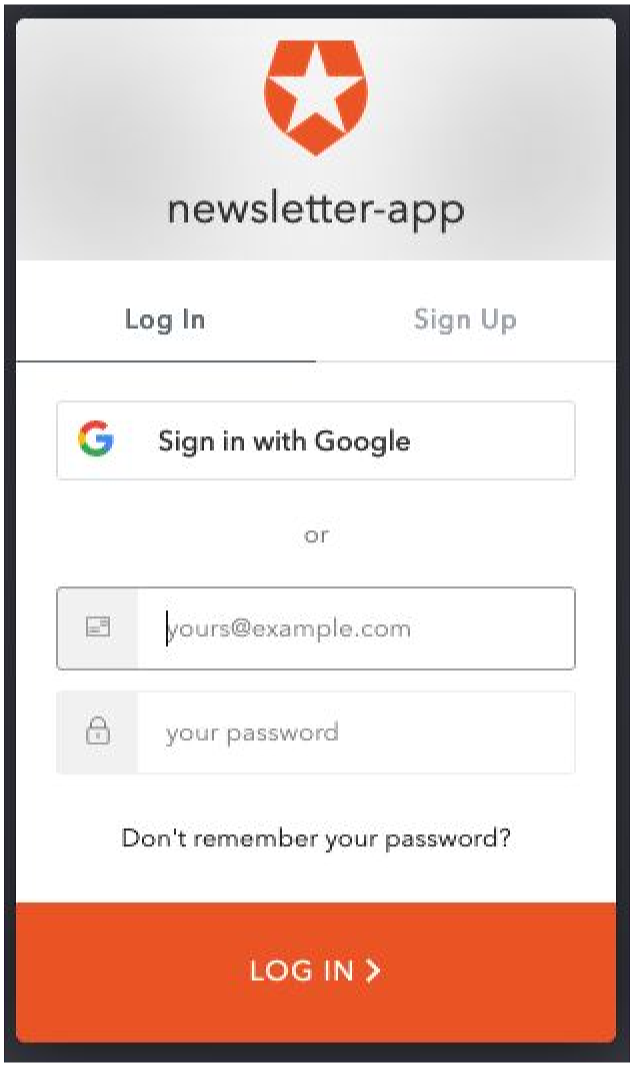
S3 bucket - SNS notification - SES email

When a file is uploaded to the S3 bucket, it sends a notification to a SNS topic which in turns sends a notification to a lambda function (SendUploadNotification) that will process the message calling the SES service that sends emails to all the subscribers of the newsletter with a link to the uploaded PDF/file. See next picture to see all the parts and the flow of uploading files and sending emails to users.



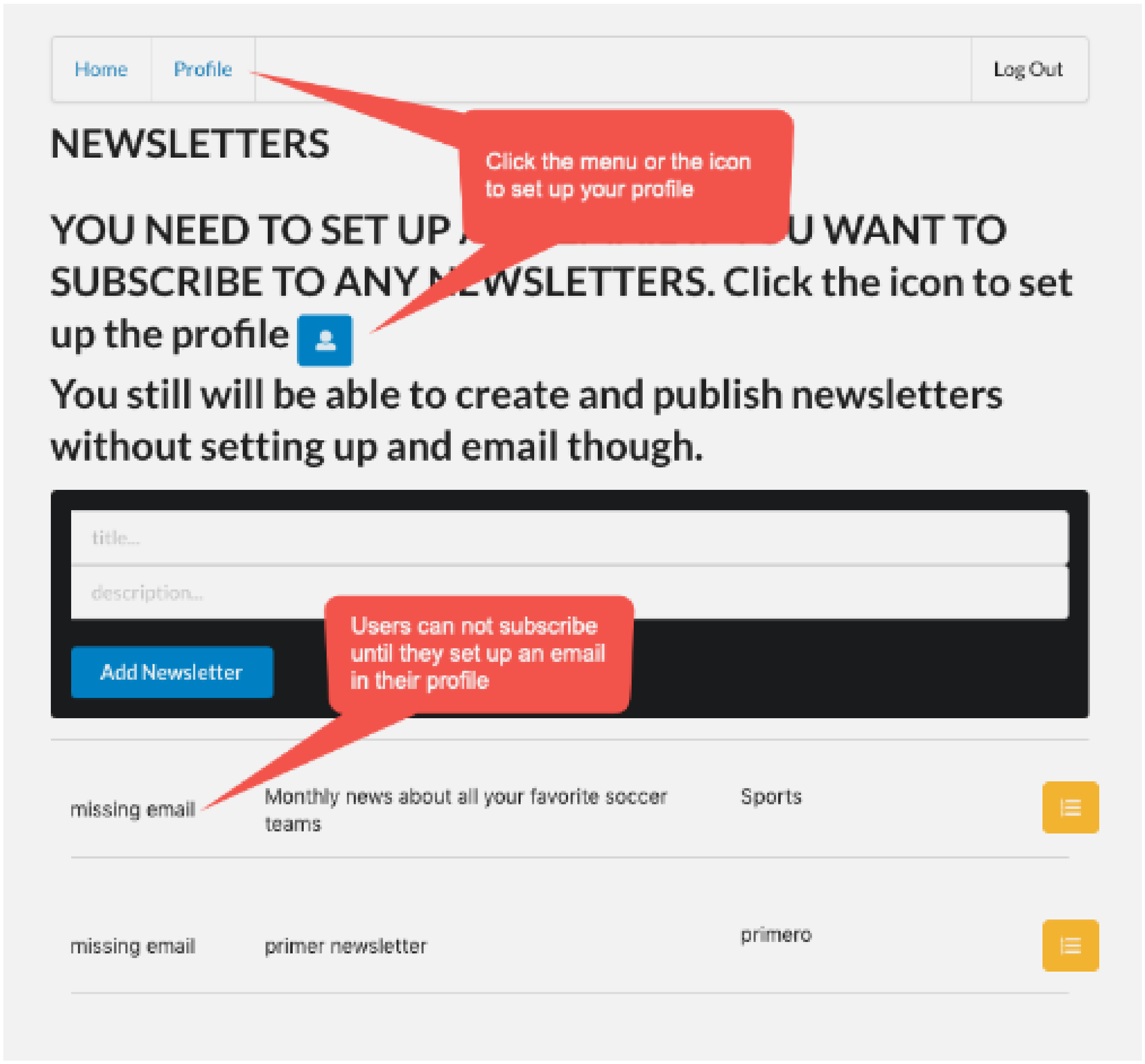
How it works

Users can access the web application using the following link: http://newsletter-fe.s3-website-us-east-1.amazonaws.com. User will be able to register and sign-in using the standard Auth0 page. See next picture.



The first time users login into the application, they will be asked to set up their profile; first name, last name and email where the subscribed newsletter should be sent.

Next picture shows the page and message displayed to the user when he/she logs in for the first time.


Home
Profile
Log Out
Click the menu or the icon to set up your profile
Users can not subscribe until they set up an email in their profile
title...
description...
Add Newsletter

NEWSLETTERS

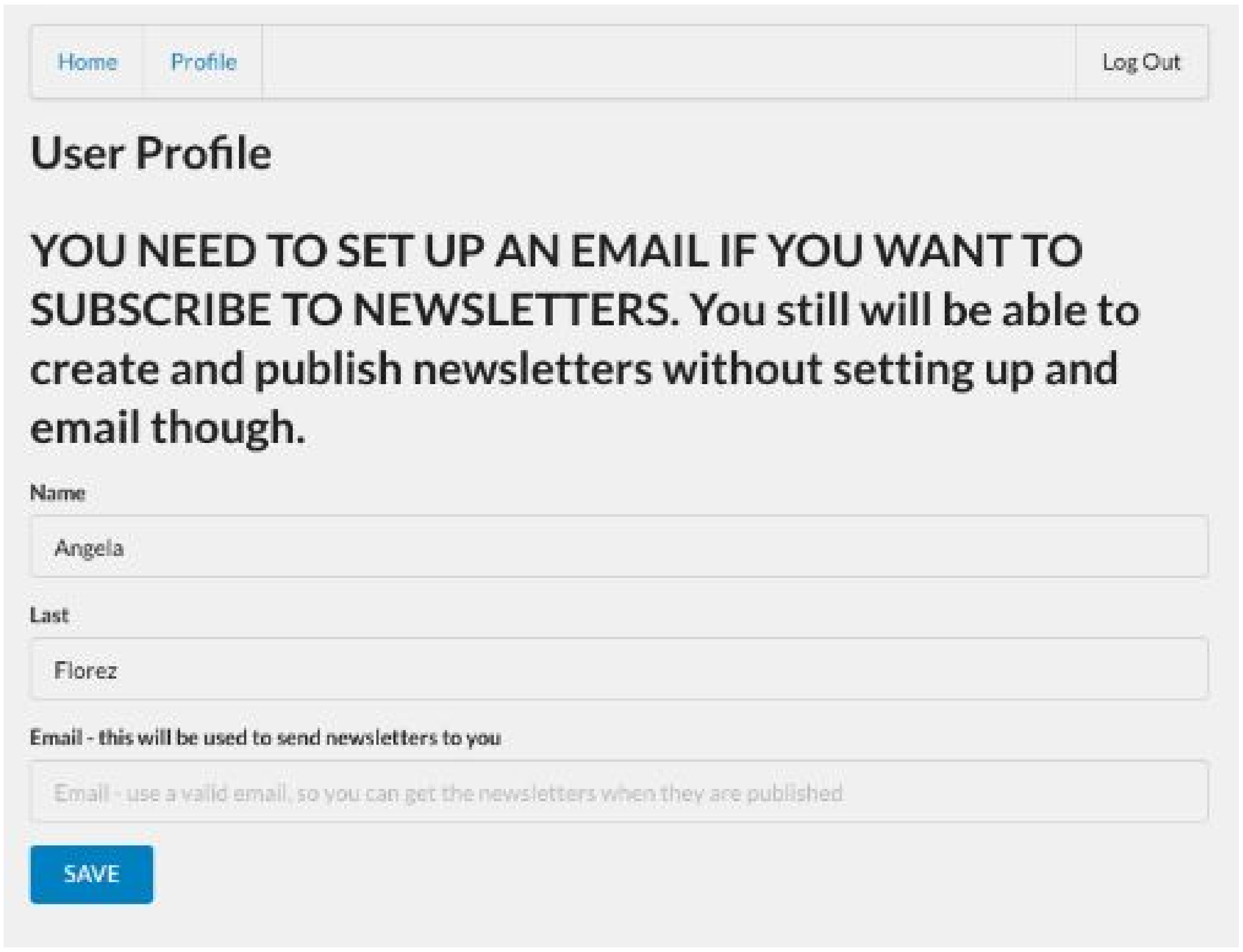
YOU NEED TO SET UP [illegible] U WANT TO SUBSCRIBE TO ANY [illegible]EWSLETTERS. Click the icon to set up the profile

You still will be able to create and publish newsletters without setting up and email though.

missing email	Monthly news about all your favorite soccer teams	Sports
missing email	primer newsletter	primero

Following picture shows the profile page where the user can save his/her name, last name and email.

NOTE: the page left out the validation of setting up an email when saving the profile to test that the back-end replies back with an error (request validation was implemented in the back-end as well)



User Profile

YOU NEED TO SET UP AN EMAIL IF YOU WANT TO SUBSCRIBE TO NEWSLETTERS. You still will be able to create and publish newsletters without setting up and email though.

Name

Angela

Last

Florez

Email - this will be used to send newsletters to you

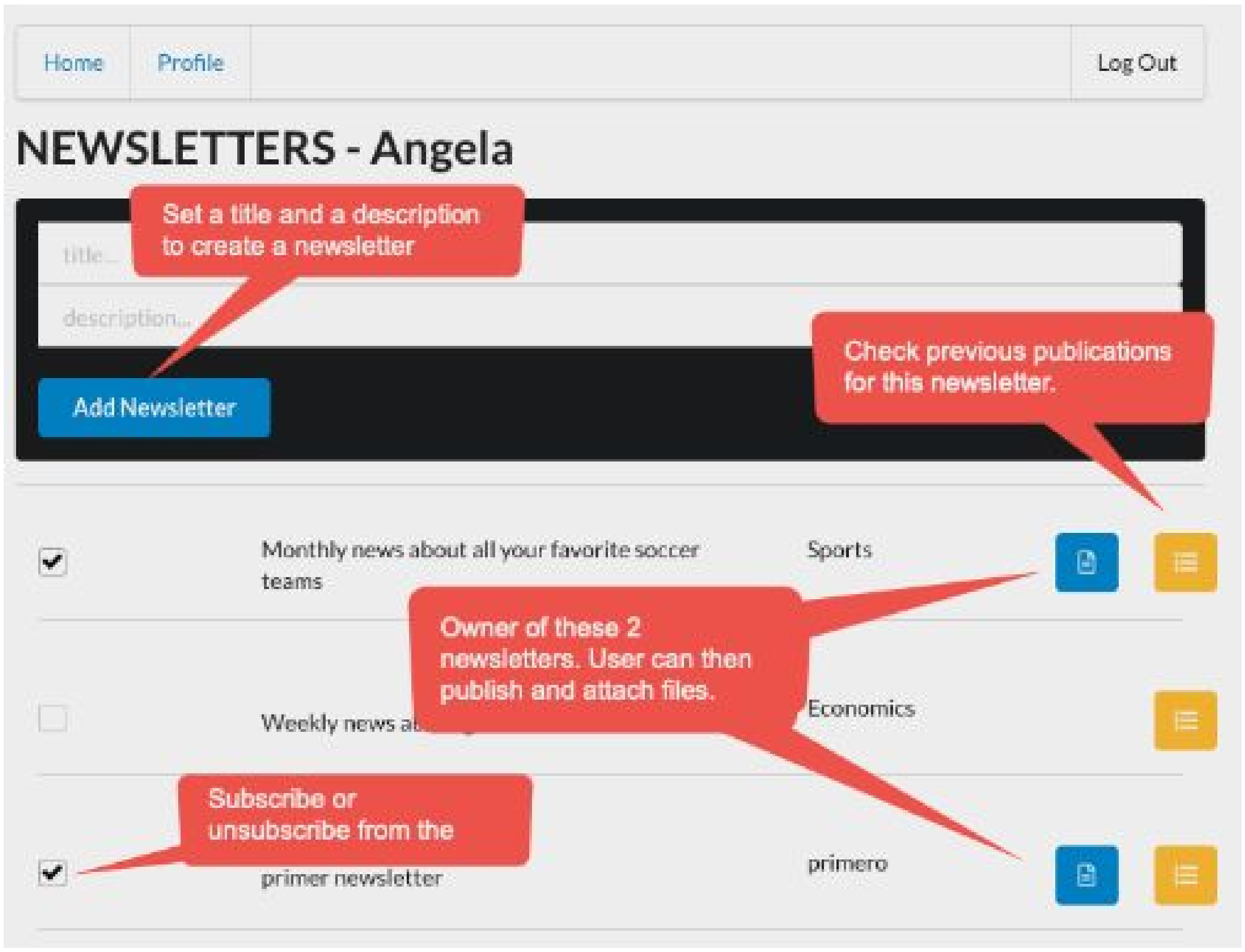
Email - use a valid email, so you can get the newsletters when they are published

Once the user sets up his/her profile, the application will allow him/her to subscribe to any newsletter in the system - a checkbox will be displayed in the newsletter row -

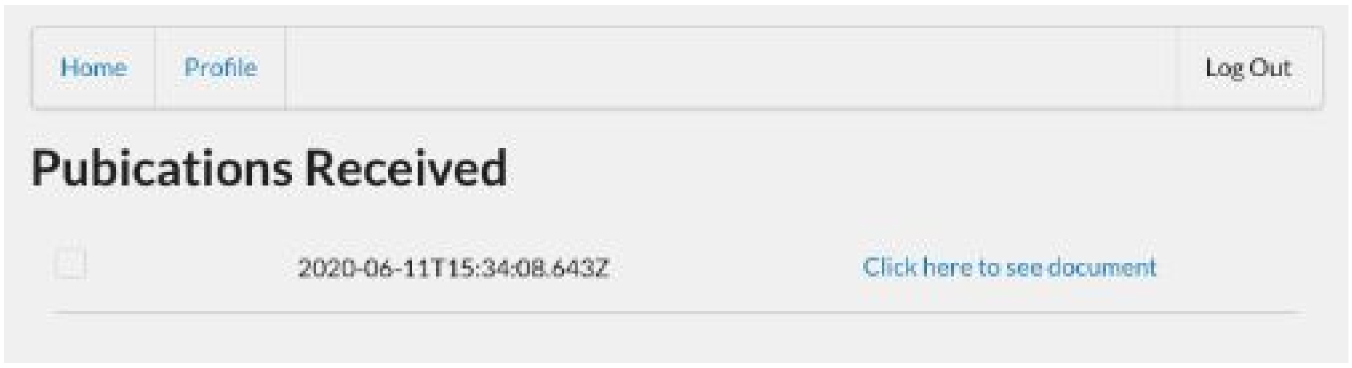
Users will be able to create their newsletters as well, so any user can subscribe as well. Only newsletters owners are allowed to publish their newsletters.

Users can also check which publications have been sent from the newsletters they are subscribed to and then can download the publication/attachments if they want to.

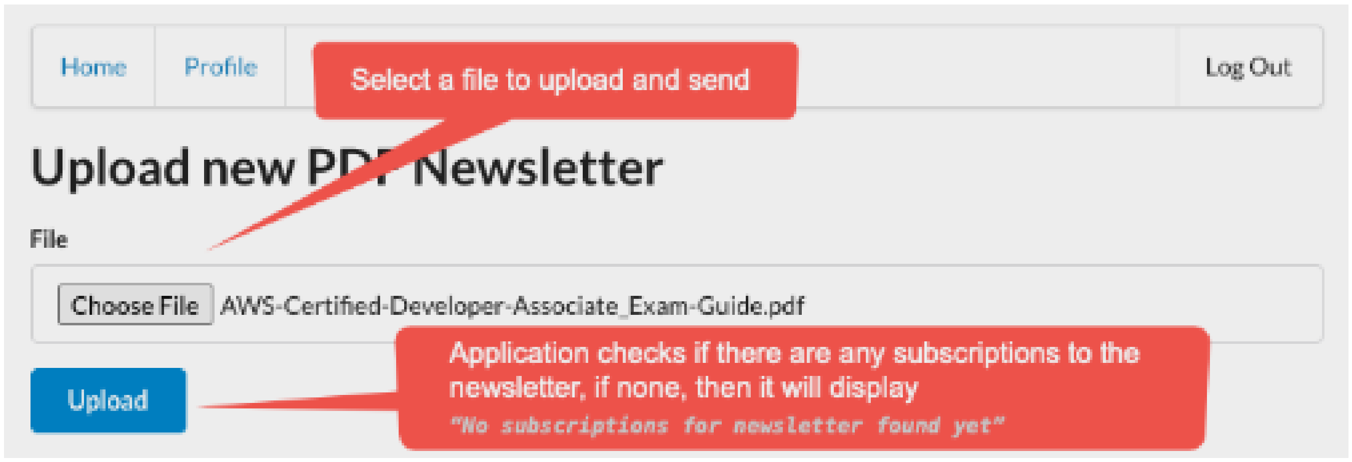
See next picture that depicts the functionality of the main page of the application;


Home
Profile
Log Out
NEWSLETTERS - Angela
title...
description...
Add Newsletter
Set a title and a description to create a newsletter
Check previous publications for this newsletter.
Monthly news about all your favorite soccer teams
Sports
Owner of these 2 newsletters. User can then publish and attach files.
Weekly news a
Economics
Subscribe or unsubscribe from the
primer newsletter
primero

If a user clicks the ***publications received*** icon (yellow) in any of the newsletters rows. The user will get a list of the publications related to it and a link to download the file will be provided. Following picture depicts a newsletter with only one publication. The left check button informs the user whether this newsletter was sent by email or not.



To create and send a newsletter publication to all its subscribers the user needs to click on the blue icon in the newsletter list in the main page. The user will get to the following page;



Upload new PDF Newsletter

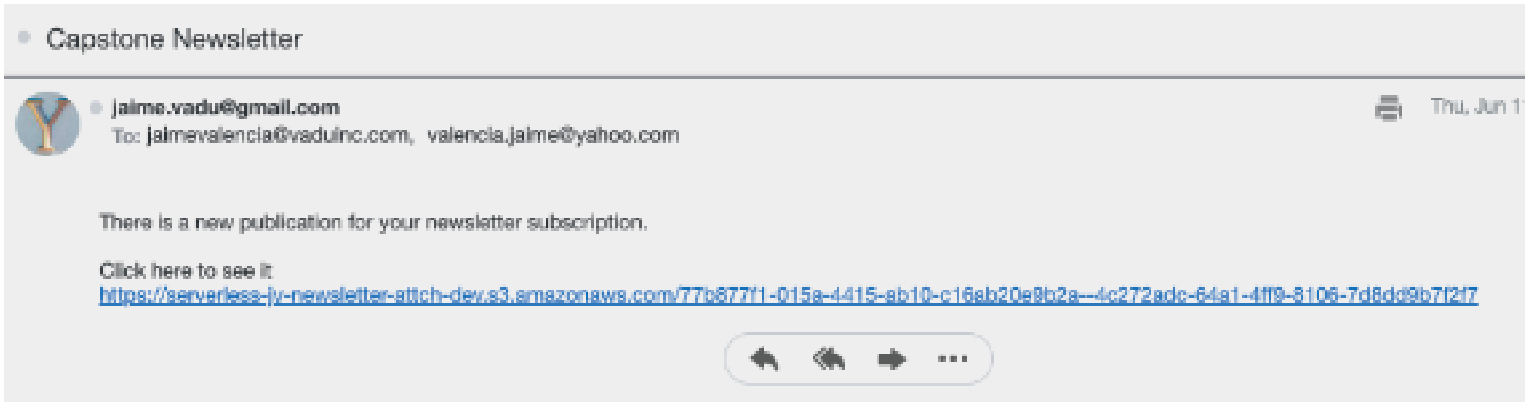
File

AWS-Certified-Developer-Associate_Exam-Guide.pdf

Application checks if there are any subscriptions to the newsletter, if none, then it will display *"No subscriptions for newsletter found yet"*

This page allows uploading a file to AWS S3. Once the file is uploaded then the S3 sends a notification to a SNS topic which in turns sends a notification to a lambda function (SendUploadNotification) that will process the message sending an email to all the subscribers of the newsletter with a link to the uploaded PDF/file.

See next picture of an email sent/received by one of the subscribers.



Capstone Newsletter

jaime.vadu@gmail.com

To: jaimevalencia@vaduinc.com, valencia.jaime@yahoo.com

Thu, Jun 1

There is a new publication for your newsletter subscription.

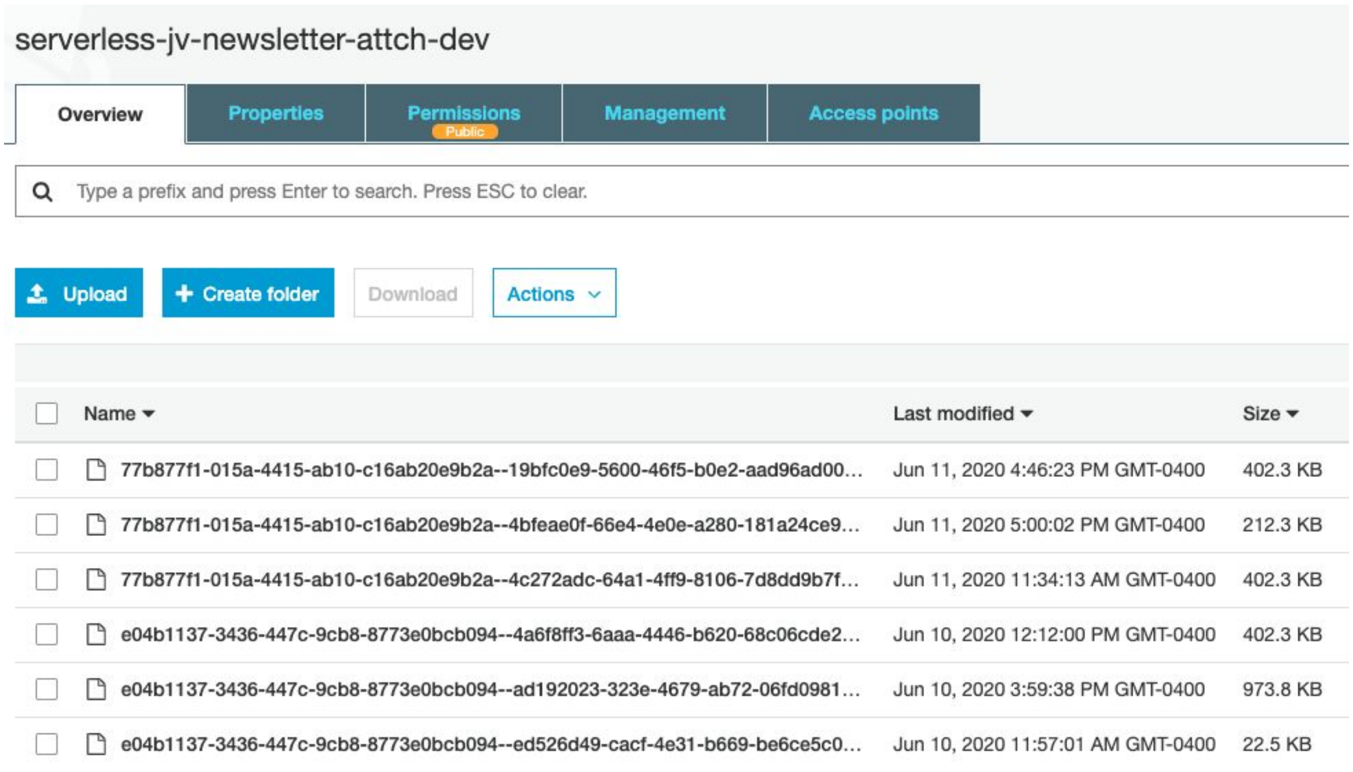
Click here to see it
https://serverless-jv-newsletter-attch-dev.s3.amazonaws.com/77b877f1-015a-4415-ab10-c16ab20e8b2a--4c272adc-64a1-4ff9-8106-7d8dd9b7f2f7

Project Rubric

1. Functionality

a. The application allows users to create, update, delete Newsletters, profiles, subscriptions, publication, etc.

b. The application allows users to upload a file: see next picture of S3 with uploaded files using the FE application.


serverless-jv-newsletter-attch-dev
Overview
Properties
Permissions
Public
Management
Access points
Type a prefix and press Enter to search. Press ESC to clear.
Upload
Create folder
Download
Actions

Name	Last modified	Size
77b877f1-015a-4415-ab10-c16ab20e9b2a--19bfc0e9-5600-46f5-b0e2-aad96ad00…	Jun 11, 2020 4:46:23 PM GMT-0400	402.3 KB
77b877f1-015a-4415-ab10-c16ab20e9b2a--4bfeae0f-66e4-4e0e-a280-181a24ce9…	Jun 11, 2020 5:00:02 PM GMT-0400	212.3 KB
77b877f1-015a-4415-ab10-c16ab20e9b2a--4c272adc-64a1-4ff9-8106-7d8dd9b7f…	Jun 11, 2020 11:34:13 AM GMT-0400	402.3 KB
e04b1137-3436-447c-9cb8-8773e0bcb094--4a6f8ff3-6aaa-4446-b620-68c06cde2…	Jun 10, 2020 12:12:00 PM GMT-0400	402.3 KB
e04b1137-3436-447c-9cb8-8773e0bcb094--ad192023-323e-4679-ab72-06fd0981…	Jun 10, 2020 3:59:38 PM GMT-0400	973.8 KB
e04b1137-3436-447c-9cb8-8773e0bcb094--ed526d49-cacf-4e31-b669-be6ce5c0…	Jun 10, 2020 11:57:01 AM GMT-0400	22.5 KB

c. The application only displays items for a logged in user: User can only publish newsletters created by him/her. They can only update their own profile, and they can only see the publications sent to them.

d. Authentication is implemented and does not allow unauthenticated access: JWT token is required to access the API end-point and Auth0 was set to login into the FE

2. Code Base

a. The code is split into multiple layers separating business logic from I/O related code:

- Data Layer
- Business Layer
- Utilis
- Model

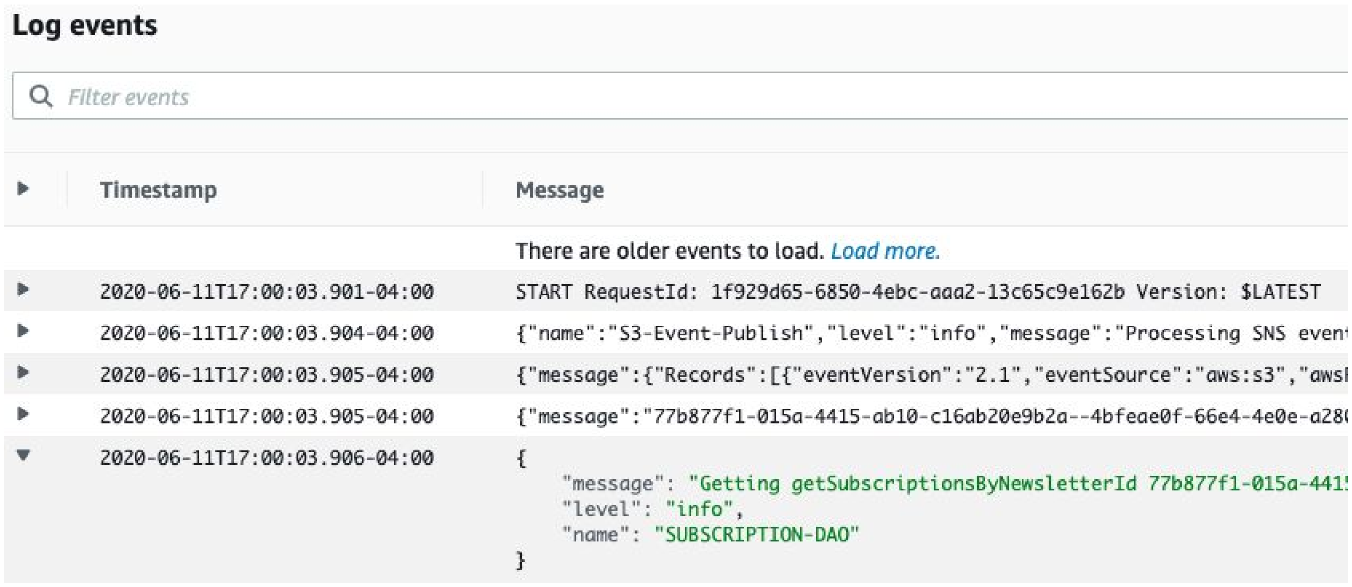
b. Code is implemented using async/await and Promises without using callbacks: *nodejs12.x* was used in the YAML file. There are no callbacks in the code and async/await was used in most of the functions.

3. Best Practices

a. All resources in the application are defined in the "serverless.yml" file: All resources needed by an application are defined in the "serverless.yml". See file in the code.

b. Each function has its own set of permissions: See "serverless.yml" file in the code

c. Application has sufficient monitoring. It has a sufficient amount of log statements and X-Ray was used/implemented

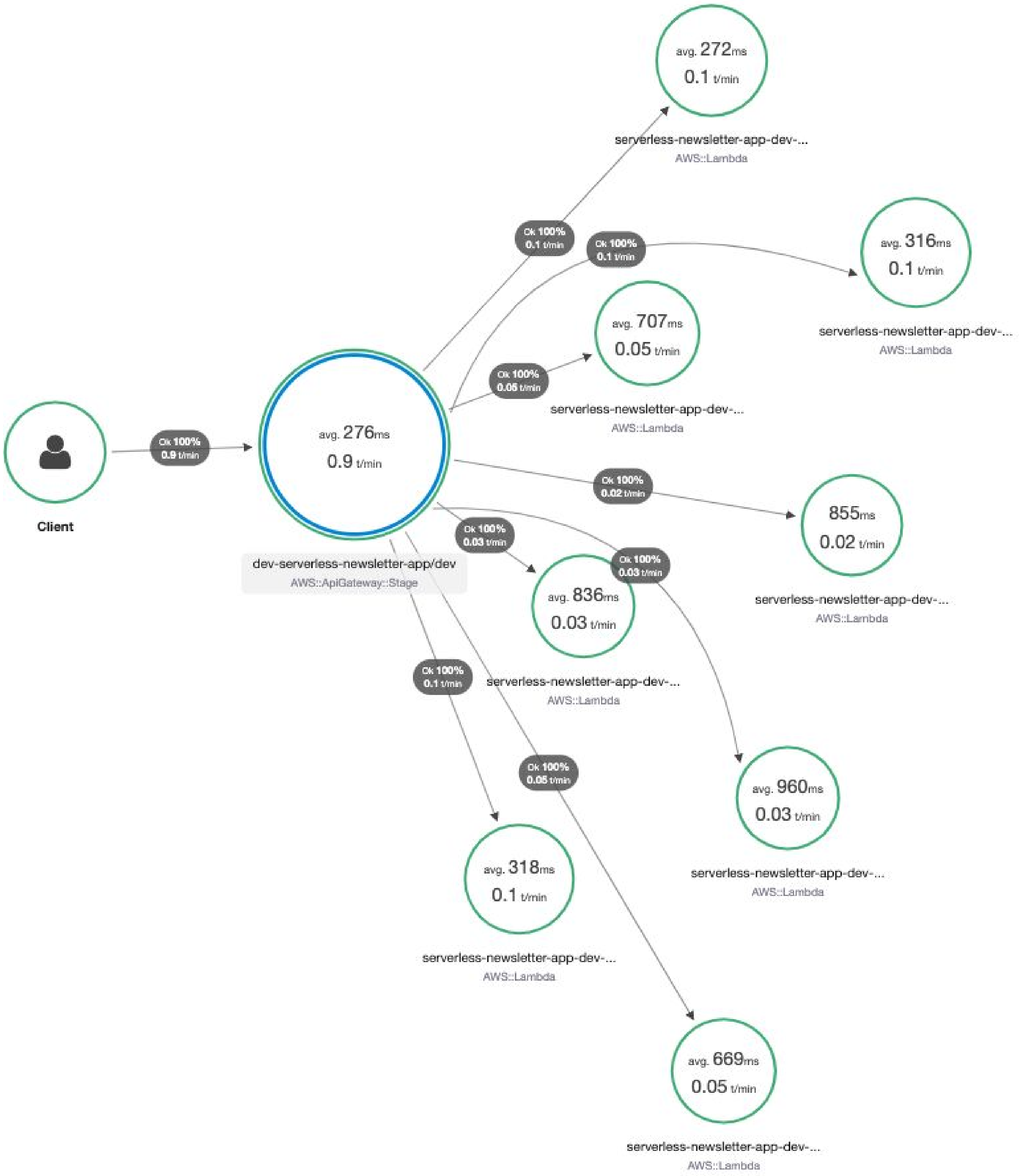


Log events

Filter events

	Timestamp	Message
		There are older events to load. *Load more.*
▶	2020-06-11T17:00:03.901-04:00	START RequestId: 1f929d65-6850-4ebc-aaa2-13c65c9e162b Version: $LATEST
▶	2020-06-11T17:00:03.904-04:00	{"name":"S3-Event-Publish","level":"info","message":"Processing SNS event
▶	2020-06-11T17:00:03.905-04:00	{"message":{"Records":[{"eventVersion":"2.1","eventSource":"aws:s3","awsR
▶	2020-06-11T17:00:03.905-04:00	{"message":"77b877f1-015a-4415-ab10-c16ab20e9b2a--4bfeae0f-66e4-4e0e-a28
▼	2020-06-11T17:00:03.906-04:00	see below

```
{
    "message": "Getting getSubscriptionsByNewsletterId 77b877f1-015a-4415
    "level": "info",
    "name": "SUBSCRIPTION-DAO"
}
```


Client
Ok 100% 0.9 t/min
avg. 276ms 0.9 t/min
dev-serverless-newsletter-app/dev
AWS::ApiGateway::Stage
Ok 100% 0.1 t/min
avg. 272ms 0.1 t/min
serverless-newsletter-app-dev-...
AWS::Lambda
Ok 100% 0.1 t/min
avg. 316ms 0.1 t/min
serverless-newsletter-app-dev-...
AWS::Lambda
Ok 100% 0.05 t/min
avg. 707ms 0.05 t/min
serverless-newsletter-app-dev-...
AWS::Lambda
Ok 100% 0.02 t/min
855ms 0.02 t/min
serverless-newsletter-app-dev-...
AWS::Lambda
Ok 100% 0.03 t/min
avg. 836ms 0.03 t/min
serverless-newsletter-app-dev-...
AWS::Lambda
Ok 100% 0.03 t/min
avg. 960ms 0.03 t/min
serverless-newsletter-app-dev-...
AWS::Lambda
Ok 100% 0.1 t/min
avg. 318ms 0.1 t/min
serverless-newsletter-app-dev-...
AWS::Lambda
Ok 100% 0.05 t/min
avg. 669ms 0.05 t/min
serverless-newsletter-app-dev-...
AWS::Lambda

d. HTTP requests are validated. See next validation schemas created for requests:

create-newsletter-request.json

```
{
 "$schema": "http://json-schema.org/draft-04/schema#",
 "title": "save newsletter",
 "type": "object",
 "properties": {
   "shortDesc": { "$ref": "#/definitions/non-empty-string" },
   "longDesc": { "$ref": "#/definitions/non-empty-string" }
 },
 "anyOf": [
   { "required": ["shortDesc"] },
   { "required": ["longDesc"] }
 ],
 "definitions": {
     "non-empty-string": {
         "type": "string",
         "minLength": 1
     }
 },
 "additionalProperties": false
}
```

subscribe-to-newsletter-request.json

```
{
   "$schema": "http://json-schema.org/draft-04/schema#",
   "title": "subscribe 2 newsletter",
   "type": "object",
   "properties": {
     "newsletterId": { "$ref": "#/definitions/non-empty-string" },
     "enrolled": {
       "type": "boolean"
     },
     "subscriptionId": {
       "type": "string"
     }
   },
   "anyOf": [
     { "required": ["newsletterId"] }
   ],
   "definitions": {
       "non-empty-string": {
           "type": "string",
           "minLength": 1
       }
   },
   "required": [
     "enrolled"
   ],
   "additionalProperties": false
}
```

publish-newsletter-request.json

```json
{
 "$schema": "http://json-schema.org/draft-04/schema#",
 "title": "publish newsletter",
 "type": "object",
 "properties": {
   "newsletterId": { "$ref": "#/definitions/non-empty-string" }
 },
 "anyOf": [
   { "required": ["newsletterId"] }
 ],
 "definitions": {
     "non-empty-string": {
         "type": "string",
         "minLength": 1
     }
 },
 "additionalProperties": false
}
```

save-profile-request.json

```json
{
   "$schema": "http://json-schema.org/draft-04/schema#",
   "title": "save profile",
   "type": "object",
   "properties": {
     "name": { "$ref": "#/definitions/non-empty-string" },
     "last": { "$ref": "#/definitions/non-empty-string" },
     "email": { "$ref": "#/definitions/non-empty-string" }
   },
   "anyOf": [
     { "required": ["name"] },
     { "required": ["last"] },
     { "required": ["email"] }
   ],
   "definitions": {
       "non-empty-string": {
           "type": "string",
           "minLength": 1
       }
   },
   "additionalProperties": false
}
```

4. Architecture

a. Data is stored in a table with a composite key. See the following TABLE definition from the *serverless.yml* file definition.

```yaml
  NewsletterTable:
    Type: AWS::DynamoDB::Table
    Properties:
      AttributeDefinitions:
        - AttributeName: PK
          AttributeType: S
        - AttributeName: SK
          AttributeType: S
        - AttributeName: GSI
          AttributeType: S
      KeySchema:
        - AttributeName: PK
          KeyType: HASH
        - AttributeName: SK
          KeyType: RANGE
      BillingMode: PAY_PER_REQUEST
      TableName: ${self:provider.environment.NEWSLETTER_TABLE}
      GlobalSecondaryIndexes:
        - IndexName: ${self:provider.environment.INDEX_NAME}
          KeySchema:
            - AttributeName: GSI
              KeyType: HASH
            - AttributeName: SK
              KeyType: RANGE
          Projection:
            ProjectionType: ALL
```

b. Scan operation is not used to read data from a database. Only query() calls were used to fetch data.