DALHOUSIE UNIVERSITY

CSCI 5409 - Cloud Computing

Final Report

DalBooks

Group 22 - Dumbledore's army

Tasnim Khan (B00882598)

Siddharth Kharwar (B00897211)

Ridham Kathiriya (B00893712)

Table of Contents

Project Overview

Here, our fundamental aim was to develop an effective book exchange system with the concept of providing credibility points. Those points can be earned and used within the system. For example, if the user provides a book to this system, then the user will get some credibility points and those points can be used any time to get the book from the system. Moreover, each user will get some free credibility points weekly so that some users can fulfill their need for books without providing any books to the system and the users will also receive sign-up bonus credibility points. [1]
This will have various features like:

- User management,
- Book management,
- Point system
- Review ratings and sentiment analysis

Here users can post their books in a barter system with images and descriptions. But that user can not receive points immediately rather the user receives a point if another user receives that book successfully. [1]

Overall, we have built an effective book barter system using the concept of credibility points and cloud-based services.

Gitlab Repositories Link: **https://git.cs.dal.ca/courses/2022-summer/csci4145-5409/group-22**

Deployment Link: **http://dalbooksfinal20-env.eba-xdwkz6ih.us-east-1.elasticbeanstalk.com/**

Final Architecture

As shown in the below diagram, we deployed our application on Elastic Beanstalk in form of the docker container. We used Amazon Cognito for authorization and authentication. In the backend, we are using various AWS services (Amazon Comprehend, AWS Lambda, Amazon DynamoDB, Amazon API Gateway) and the front-end is developed using ReactJS. Here we are using Firebase Storage and making the multi-cloud architecture. [1]

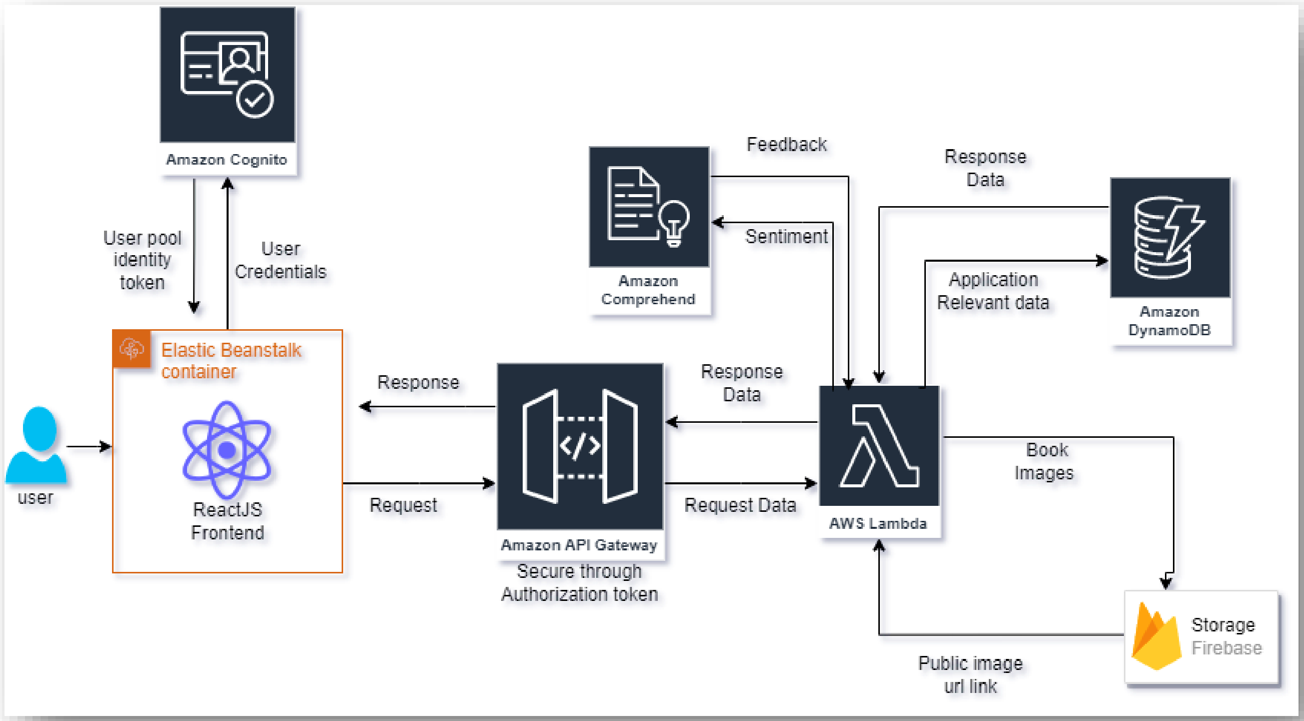


Figure 1: The final architecture of the application[1]

Table 1: Decided AWS service from each category

Category	Services
Compute	1. **AWS Elastic Beanstalk & Docker** 2. **AWS Lambda**
Storage	1. **Firebase Storage** 2. **AWS DynamoDB**
Network	1. **AWS API Gateway**
General	1. **Amazon Cognito** 2. **Amazon Comprehend**

How do all the cloud mechanisms fit together to deliver your application?

The whole architecture of this application is shown in the figure 1. Our software is remarkably flexible, accessible, and scalable. We used AWS Elastic Beanstalk to host the application to handle more requests concurrently while maintaining performance. It supports dynamic load sharing and can automatically handle the traffic. Data about users is kept in the DynamoDB database. The Book Image Uploading method stores information in a Firebase storage and returns the public image access URL, which may then be placed in DynamoDB. Feedback on books is analyzed for sentiment using Amazon Comprehend. The user pool, which holds the user's credentials and other necessary user data, will be kept in AWS Cognito.[1]

Where is data stored?

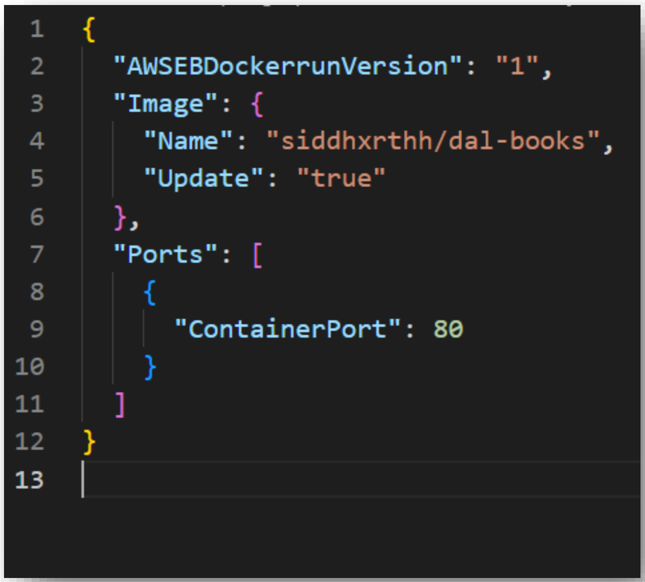
Here we are using the multi-cloud approach for storing the data. We are using DynamoDB and Firebase Storage. Firebase storage stores the data of book images while DynamoDB stores all other information of books and user information. We are also storing the book’s public image URL to DynamoDB.[1]

What programming languages did you use (and why) and what parts of your application required code?

We tried to use JavaScript everywhere as all of the team members had prior experience in it.
For the frontend, we use the ReactJS Library, as it was open source and something we had recently learnt in another course. It helped is create a responsive frontend with ease. The library allows is to use JavaScript while building the web app.
For the backend, we used Lambda functions with NodeJS.[2]

How is your system deployed to the cloud?

We deployed our front-end of the application to AWS Elastic Beanstalk with Docker. Firstly, we Developed a Docker Image of our front-end and put it on the docker hub. Afterward, we simply developed Dockerrun.aws.json as shown in the below image. Here Nginx uses default port 80. Then we used this file on elastic beanstalk to deploy the application. The back end is deployed using cloud Formation.[4]



```json
{
  "AWSEBDockerrunVersion": "1",
  "Image": {
    "Name": "siddhxrthh/dal-books",
    "Update": "true"
  },
  "Ports": [
    {
      "ContainerPort": 80
    }
  ]
}
```

Figure 2: Dockerrun.aws.json file

Proposed Architecture vs Final Architecture

If your final architecture differs from your original project proposal, explain why that is. What did you learn that made you change technologies or approaches?

The final architecture was almost the same as the proposed architecture with only 2 changes:

1. Amazon Comprehend (ML) – We used the sentiment analysis service provided by AWS instead of Amazon Batch as we were more excited to learn and implement ML in the review feature of our application. We wanted to give our users the analyzed review where they could take a glimpse and understand the overall sentiment of the reviews of the book and then decide if they want to read it.
2. Firebase Storage – During the end of our project, we wanted to explore Multi-Cloud Architecture. But since most of our feature was already implemented, we decided to stop using S3 Bucket and try to integrate Firebase Storage to store our media files. We also tried to add ML service provided by GCP, Vertex AI – Auto ML, but it was very expensive to implement in our project.

How would your application evolve if you were to continue development? What features might you add next and which cloud mechanisms would you use to implement those features?

The features and mechanism that we plan to implement after the semester are:

1. Multi-Cloud Architecture – We would like to explore multi-cloud platform properly and replicate every service from AWS to GCP for higher availability and also to understand the complexity that comes up with multi-cloud architecture
2. We would like to train a ML Recommendation Model where we could provide our users recommendations of the books based on their borrowing history.

Data Security

We have kept all our data in Dynamo DB. To access these, we have created lambda functions which are accessible using API Gateway. All the API endpoints require an authorization token to access the endpoints. We get this authorizer's token from Cognito User Identity Pool which is set to expire after thirty mins. This way we must update the JWT token every thirty minutes and keep the API endpoints secure from unauthorized access.

Vulnerability

While querying the API data from the backend, our payload data is not encrypted. This is one of the security vulnerabilities that we wish to work on and update our application after this semester.
The AWS services communicate with each other and are only exposed to outside attacks via the API Gateway, which requires the JWT Token.

Reproduce the Application in Private Cloud

To reproduce this application in private cloud, our organization needs to provide the following:

1. Server for Deploymentz
2. Server for NodeJS backend
3. Sentiment Analysis Model – This was provided by AWS, but in private cloud, we have to train and keep updating our own model.
4. Data server for DB
5. Storage for Media

Which Service is the Most Expensive in our Application?

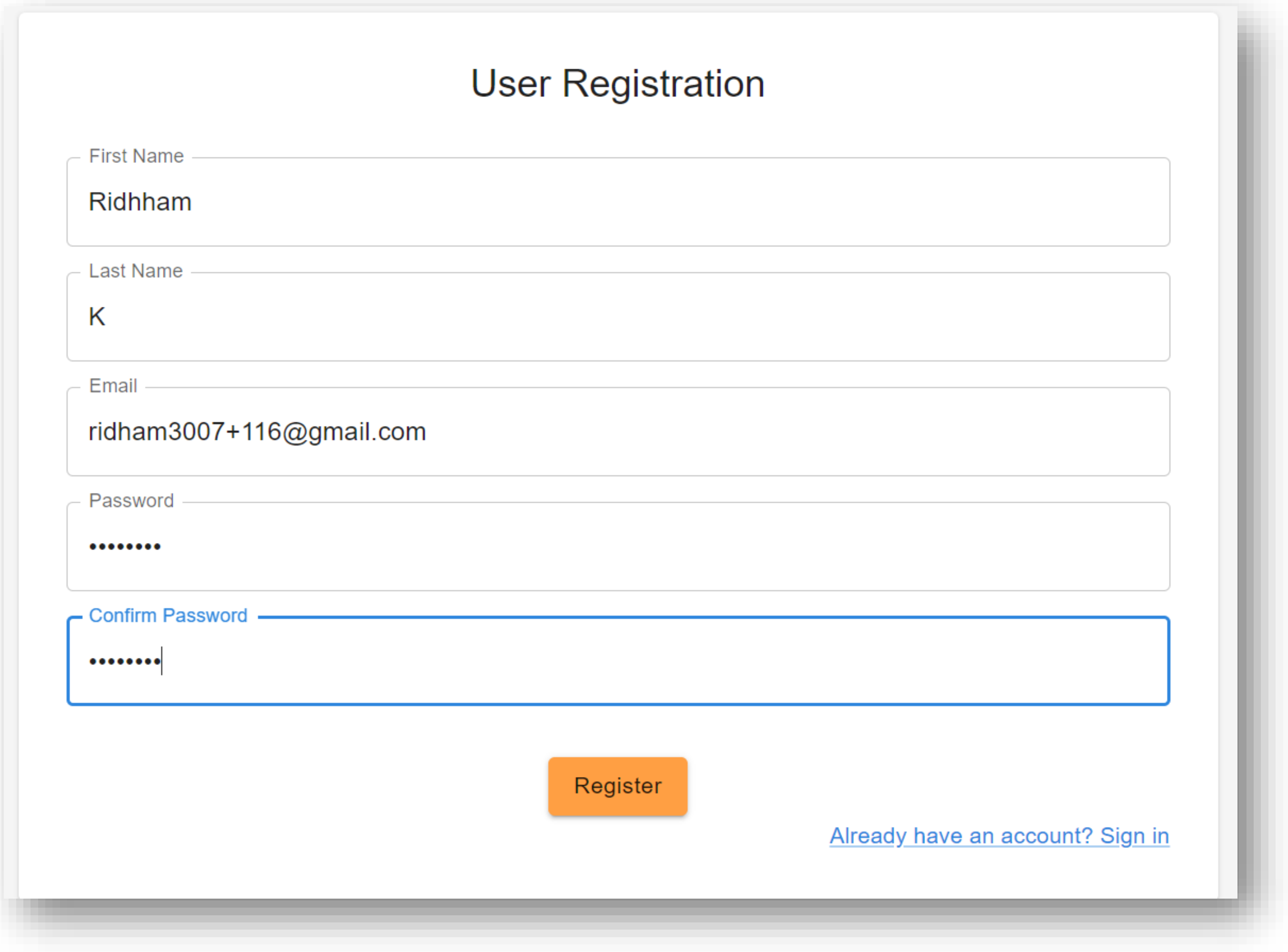
We plan to integrate ML Recommender System so that our application can use the user's history and recommend the books in the future. GCP Vertex AI – Auto ML's Tabular Classification is an expensive service, but it can help us build our service with ease. This service can prove to be the most expensive feature that our application can offer, and we might want to add a monitor to that.

Along with ML, we host our application in Elastic Beanstalk, which is an expensive hosting service. We would like to monitor that, especially during the launching of our application when we would not have too many users and do not need the Cloud Provider's Service to automate load balancing and elasticity.

Screenshots

Screenshots of the application

1. User Authorization and Authentication:



User Registration

First Name: Ridhham

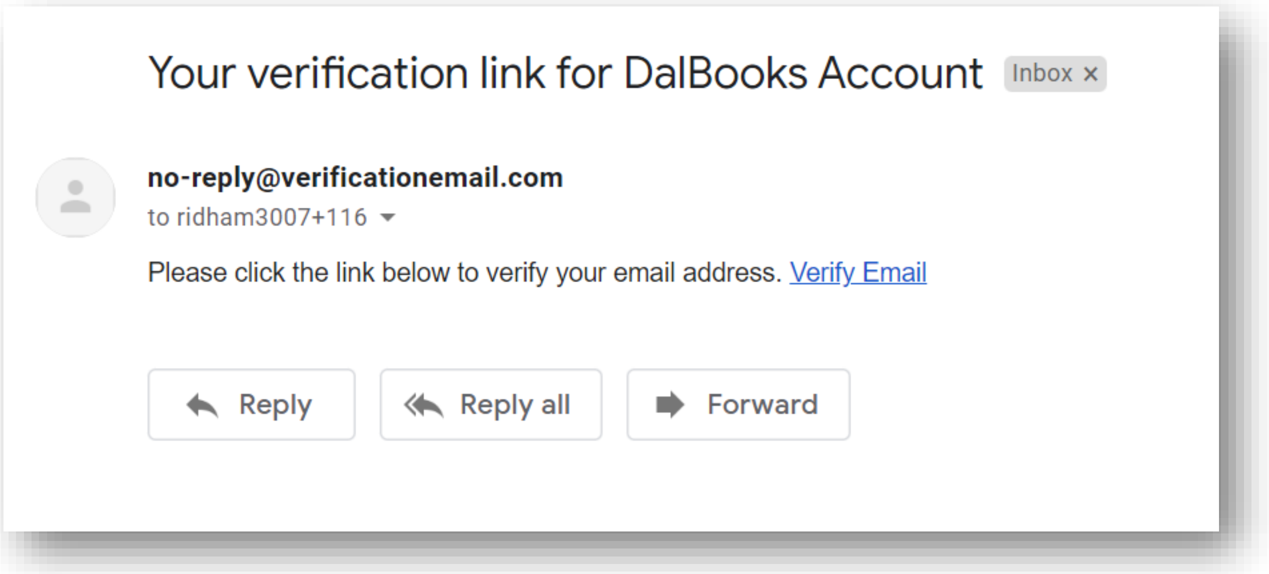
Last Name: K

Email: ridham3007+116@gmail.com

Password: ••••••••

Confirm Password: ••••••••

Figure 4: Registering User[3]

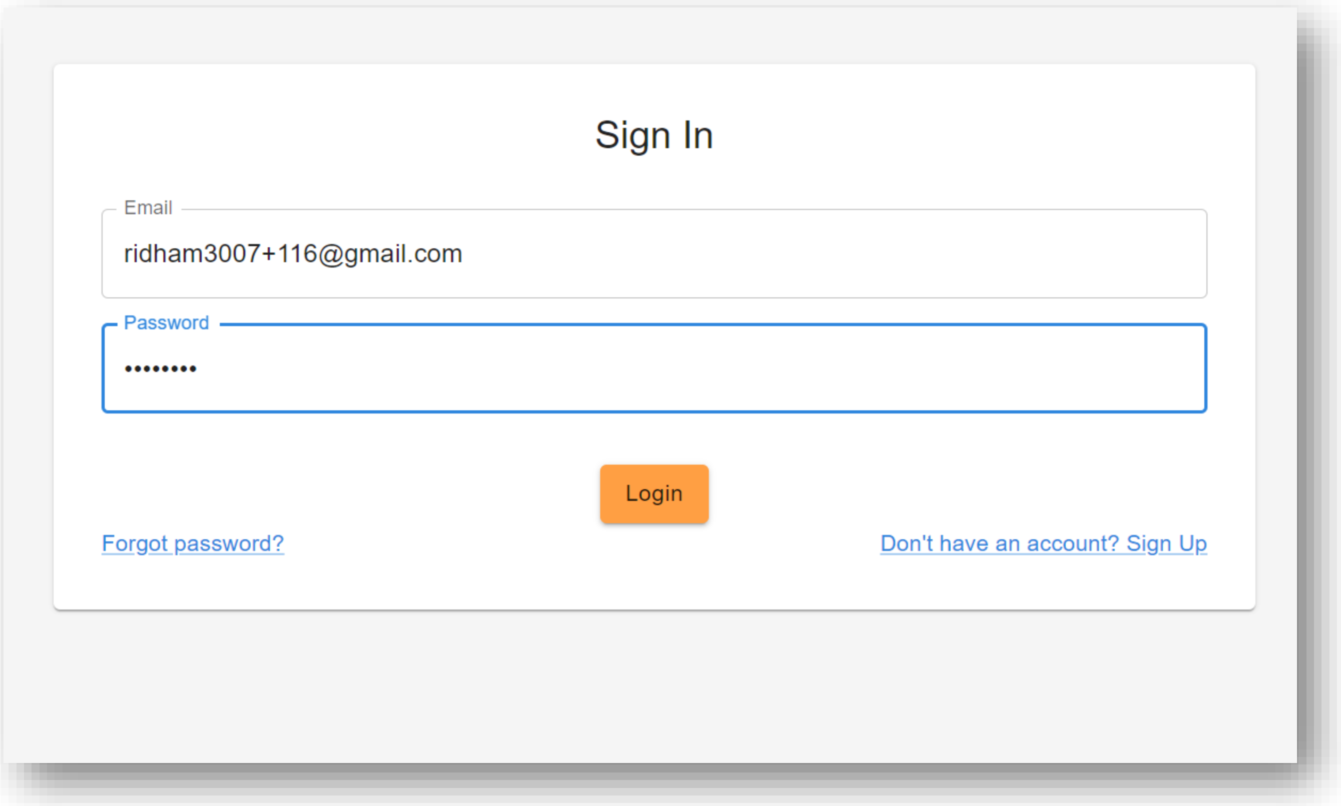

Your verification link for DalBooks Account

no-reply@verificationemail.com

to ridham3007+116

Please click the link below to verify your email address. Verify Email

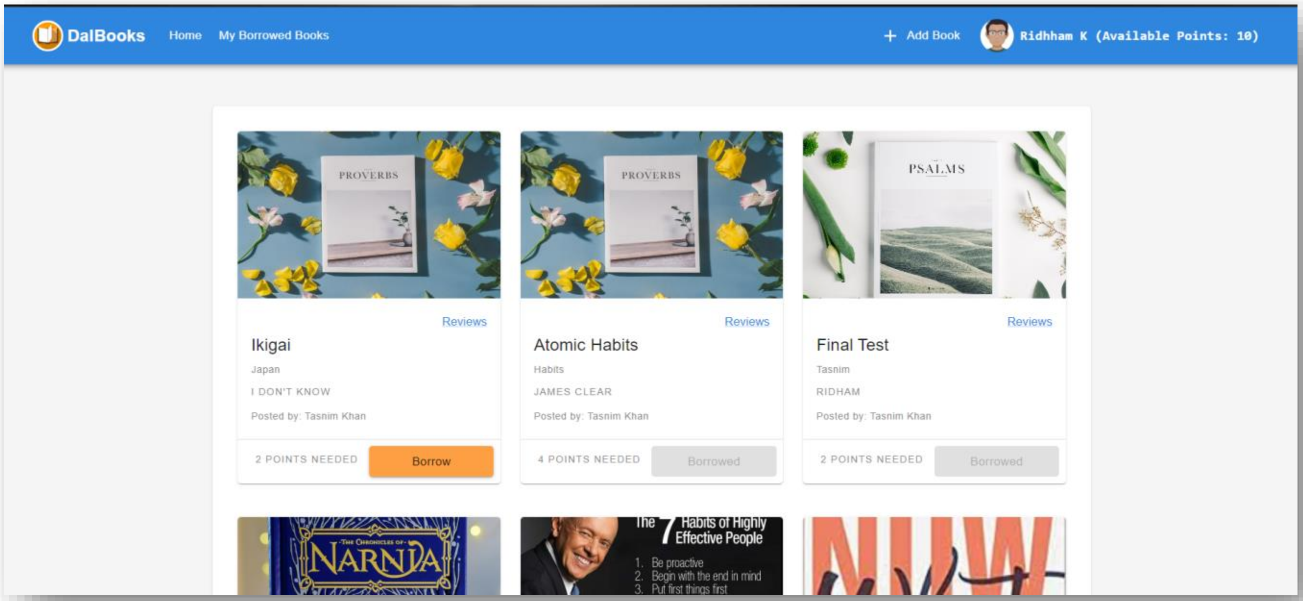
Figure 5: Verification mail


Sign In

Email: ridham3007+116@gmail.com

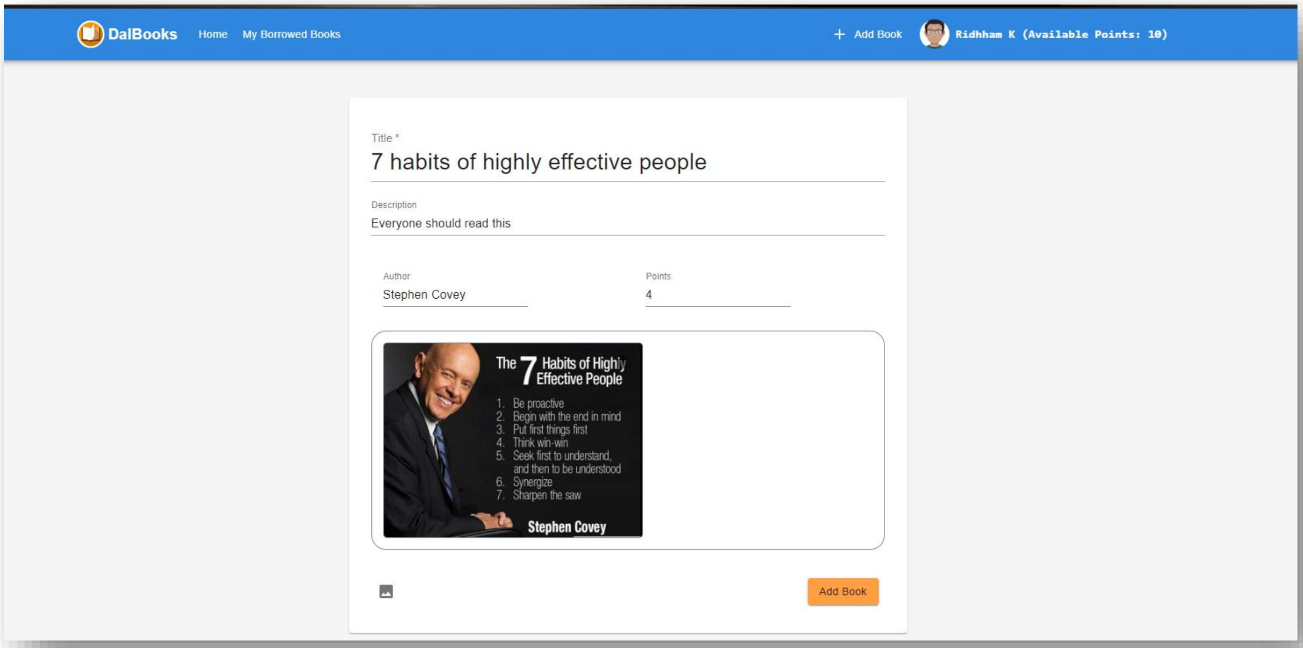
Password: ••••••••

Figure 6: Login page



Reviews	Reviews	Reviews
Ikigai	Atomic Habits	Final Test
Japan	Habits	Tasnim
I DON'T KNOW	JAMES CLEAR	RIDHAM
Posted by: Tasnim Khan	Posted by: Tasnim Khan	Posted by: Tasnim Khan
2 POINTS NEEDED Borrow	4 POINTS NEEDED Borrowed	2 POINTS NEEDED Borrowed

Figure 7: Home page after successful signIn


DalBooks
Home
My Borrowed Books
Add Book
Ridhham K (Available Points: 10)

Title *
7 habits of highly effective people

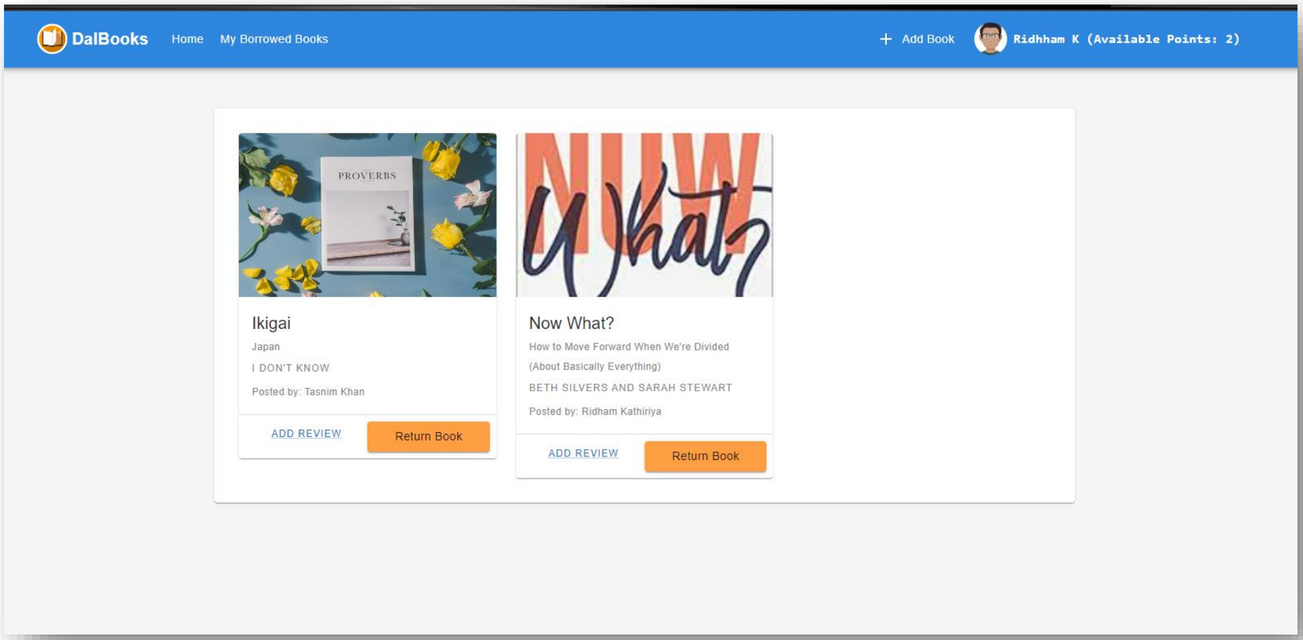
Description
Everyone should read this

Author
Stephen Covey

Points
4

The 7 Habits of Highly Effective People
1. Be proactive
2. Begin with the end in mind
3. Put first things first
4. Think win-win
5. Seek first to understand, and then to be understood
6. Synergize
7. Sharpen the saw
Stephen Covey
Add Book

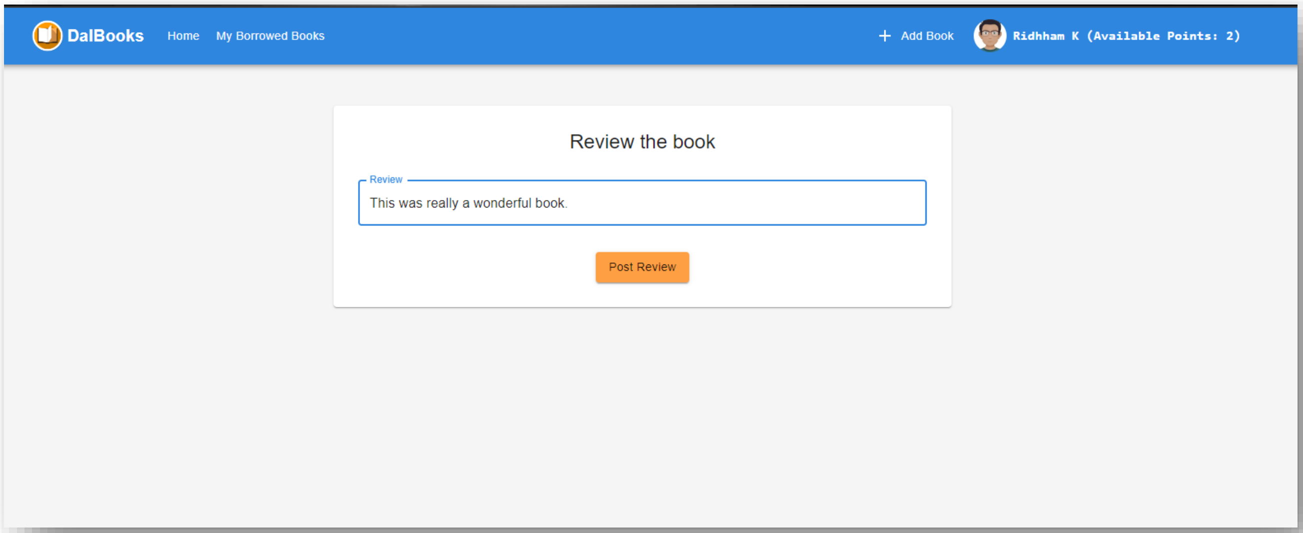
Figure 8: Add book page



Ikigai
Japan
I DON'T KNOW
Posted by: Tasnim Khan

Now What?
How to Move Forward When We're Divided (About Basically Everything)
BETH SILVERS AND SARAH STEWART
Posted by: Ridham Kathiriya

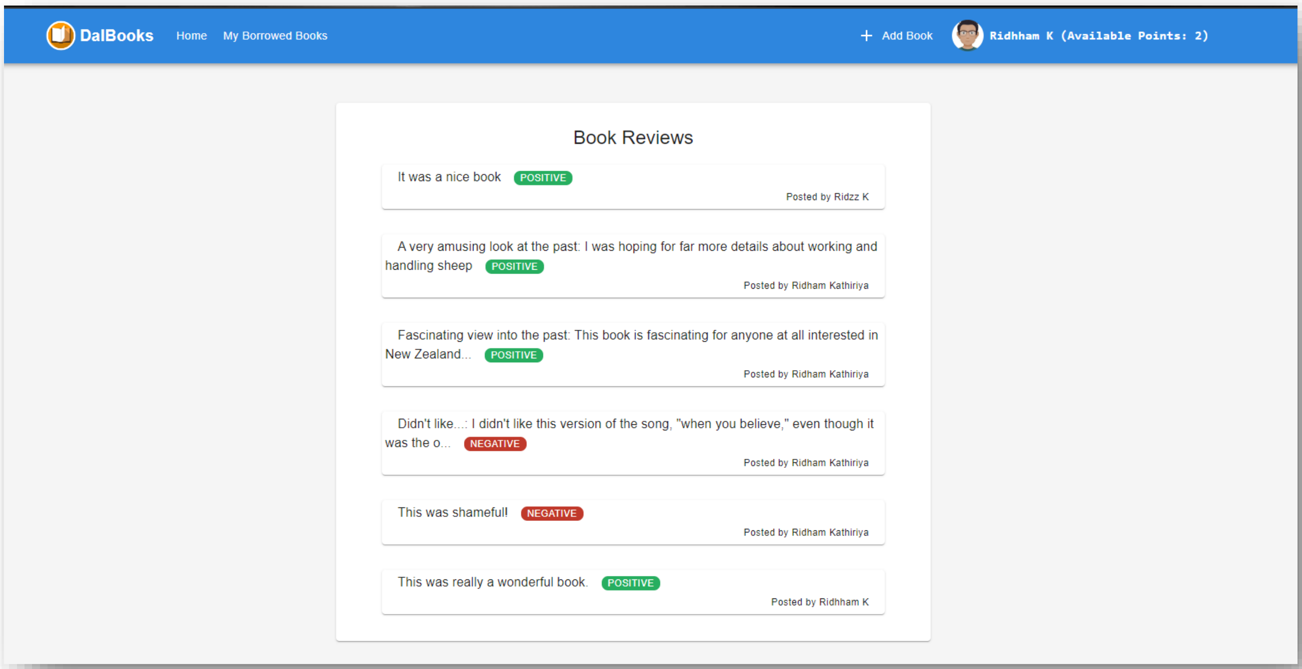
Figure 9: My Borrowed book page



Review the book

Review
This was really a wonderful book.

Figure 10: Providing Book review



Book Reviews

It was a nice book POSITIVE
Posted by Ridzz K

A very amusing look at the past: I was hoping for far more details about working and handling sheep POSITIVE
Posted by Ridham Kathiriya

Fascinating view into the past: This book is fascinating for anyone at all interested in New Zealand... POSITIVE
Posted by Ridham Kathiriya

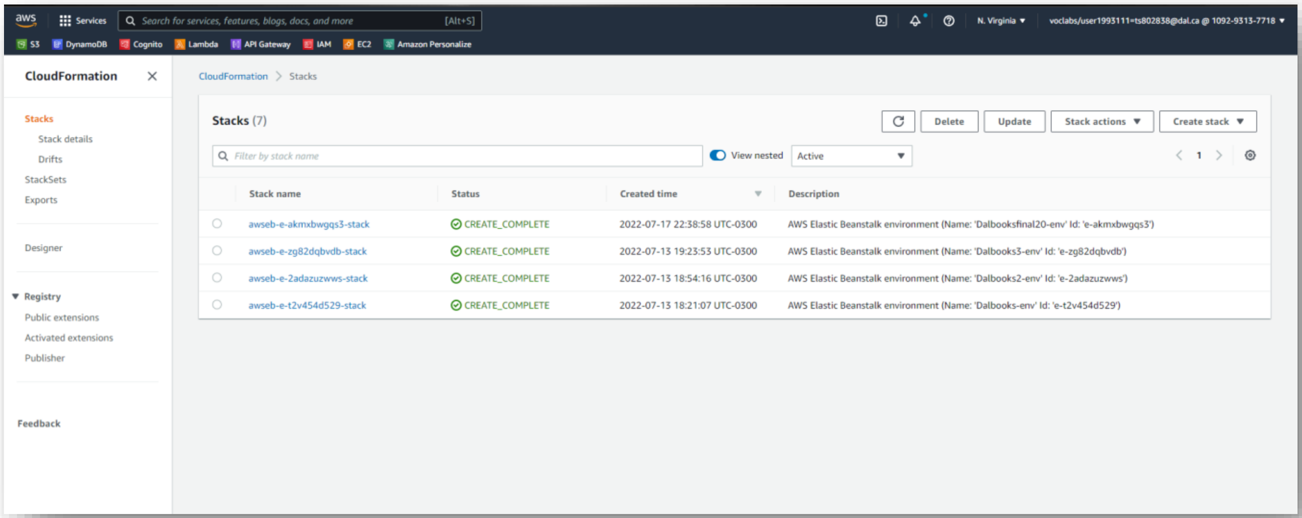
Didn't like...: I didn't like this version of the song, "when you believe," even though it was the o... NEGATIVE
Posted by Ridham Kathiriya

This was shamefull NEGATIVE
Posted by Ridham Kathiriya

This was really a wonderful book. POSITIVE
Posted by Ridhham K

Figure 11: Book reviews with its sentiment

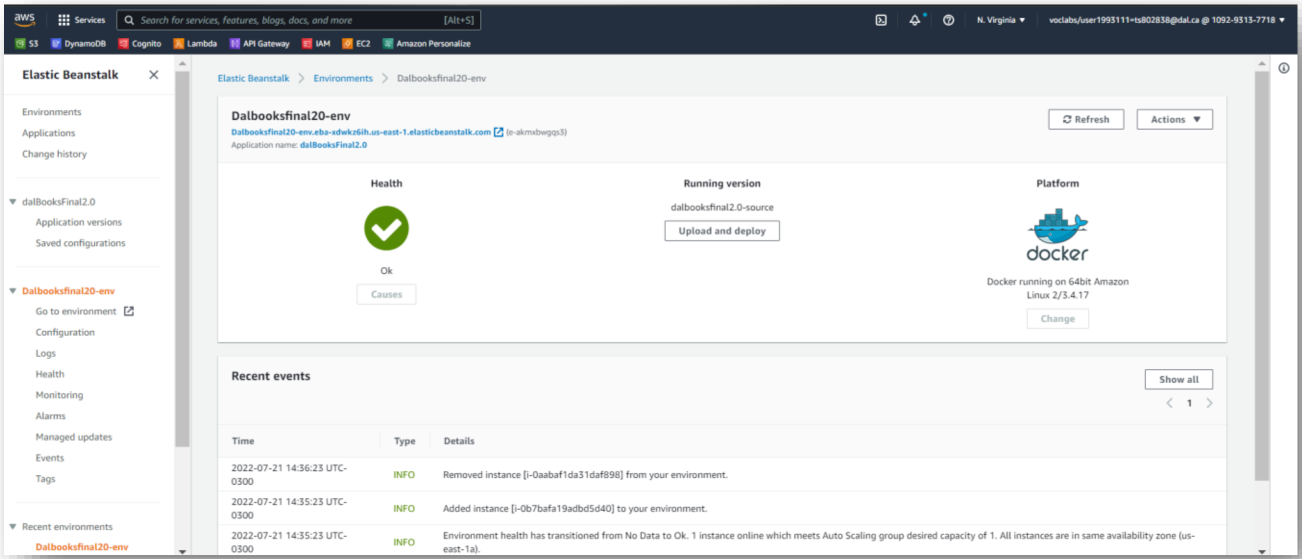
Screenshots of the AWS Service configuration



Stacks (7)

Stack name	Status	Created time	Description
awseb-e-akmxbwgqs3-stack	CREATE_COMPLETE	2022-07-17 22:38:58 UTC-0300	AWS Elastic Beanstalk environment (Name: 'Dalbooksfinal20-env' Id: 'e-akmxbwgqs3')
awseb-e-zg82dqbvdb-stack	CREATE_COMPLETE	2022-07-13 19:23:53 UTC-0300	AWS Elastic Beanstalk environment (Name: 'Dalbooks3-env' Id: 'e-zg82dqbvdb')
awseb-e-2adazuzwws-stack	CREATE_COMPLETE	2022-07-13 18:54:16 UTC-0300	AWS Elastic Beanstalk environment (Name: 'Dalbooks2-env' Id: 'e-2adazuzwws')
awseb-e-t2v454d529-stack	CREATE_COMPLETE	2022-07-13 18:21:07 UTC-0300	AWS Elastic Beanstalk environment (Name: 'Dalbooks-env' Id: 'e-t2v454d529')

Figure 12: Screenshot of successfully executed stacks of the cloud formation



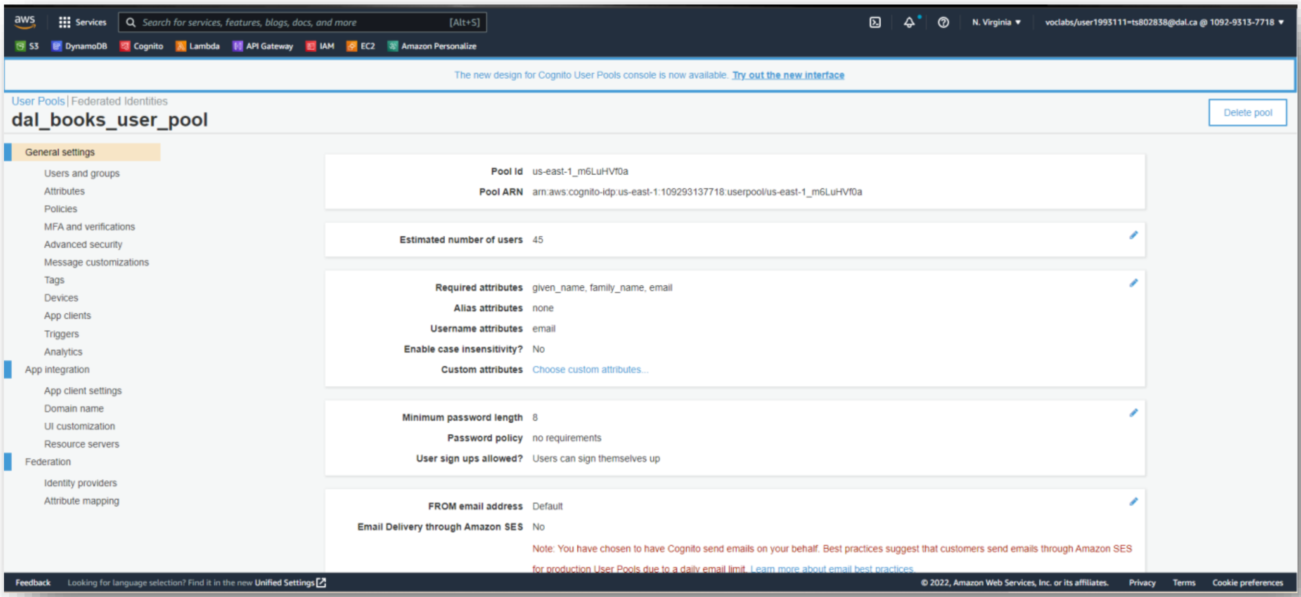
Dalbooksfinal20-env
Dalbooksfinal20-env.eba-xdwkz6ih.us-east-1.elasticbeanstalk.com (e-akmxbwgqs3)
Application name: dalBooksFinal2.0

Health: Ok
Running version: dalbooksfinal2.0-source
Platform: Docker running on 64bit Amazon Linux 2/3.4.17

Recent events

Time	Type	Details
2022-07-21 14:36:23 UTC-0300	INFO	Removed instance [i-0aabaf1da31daf898] from your environment.
2022-07-21 14:35:23 UTC-0300	INFO	Added instance [i-0b7bafa19adbd5d40] to your environment.
2022-07-21 14:35:23 UTC-0300	INFO	Environment health has transitioned from No Data to Ok. 1 instance online which meets Auto Scaling group desired capacity of 1. All instances are in same availability zone (us-east-1a).

Figure 13: Frontend Application deployed on AWS Beanstalk[1]



The new design for Cognito User Pools console is now available. Try out the new interface

dal_books_user_pool

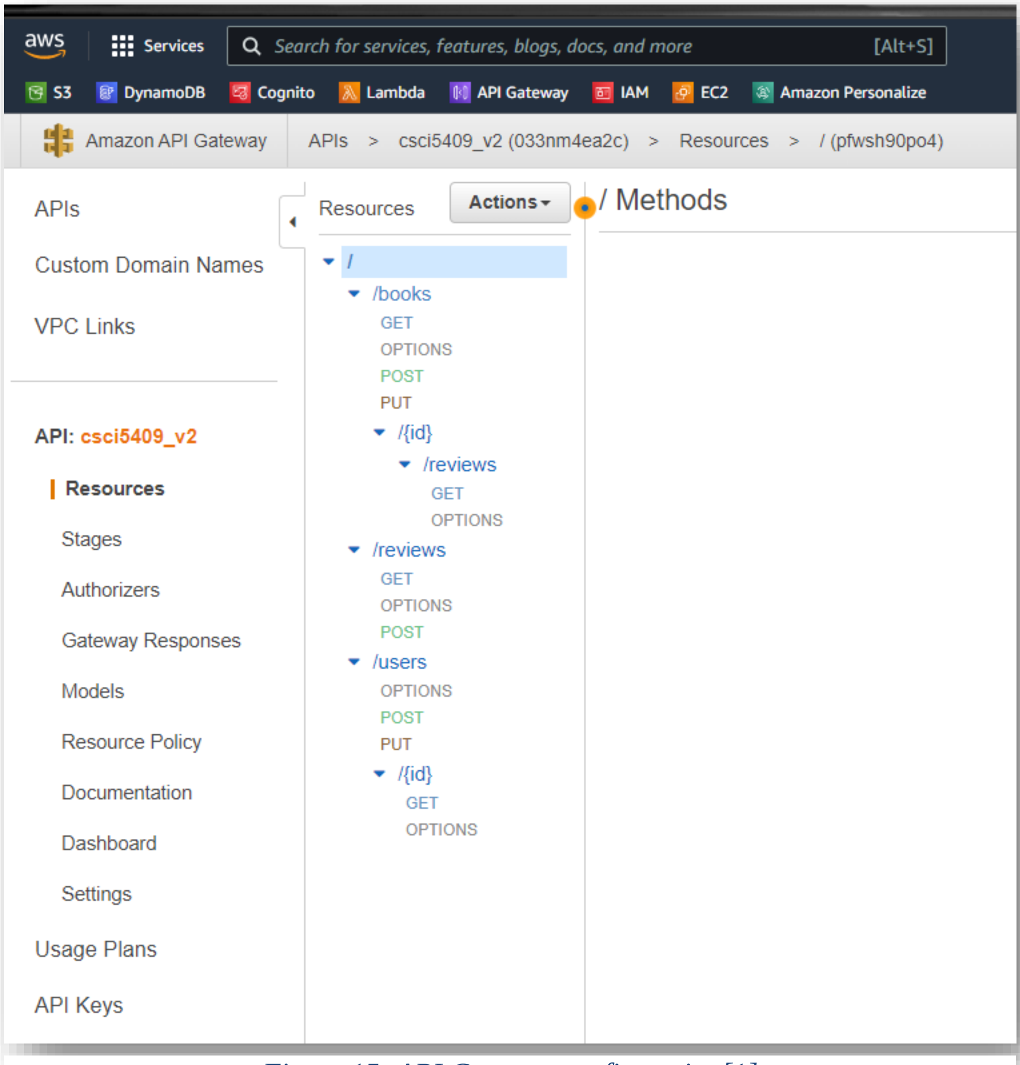
Pool Id	us-east-1_m6LuHVf0a
Pool ARN	arn:aws:cognito-idp:us-east-1:109293137718:userpool/us-east-1_m6LuHVf0a
Estimated number of users	45
Required attributes	given_name, family_name, email
Alias attributes	none
Username attributes	email
Enable case insensitivity?	No
Custom attributes	Choose custom attributes...
Minimum password length	8
Password policy	no requirements
User sign ups allowed?	Users can sign themselves up
FROM email address	Default
Email Delivery through Amazon SES	No

Note: You have chosen to have Cognito send emails on your behalf. Best practices suggest that customers send emails through Amazon SES for production User Pools due to a daily email limit. Learn more about email best practices.

Looking for language selection? Find it in the new Unified Settings

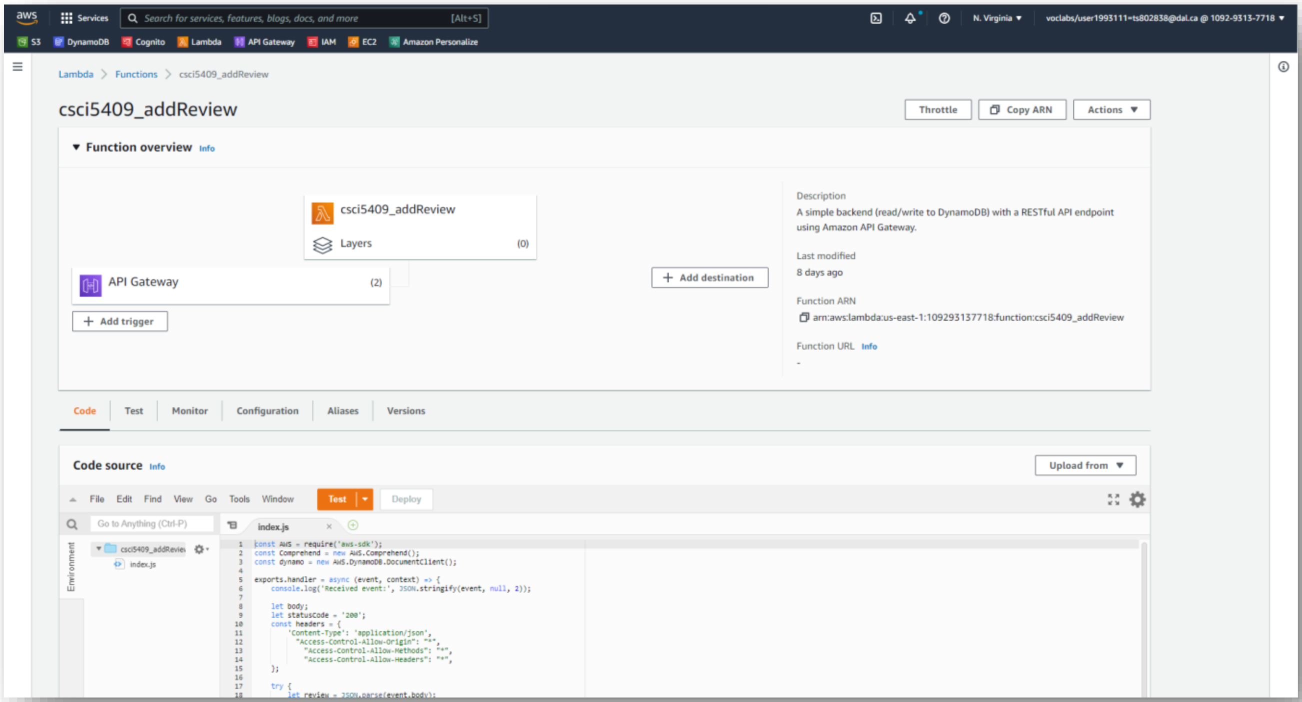
© 2022, Amazon Web Services, Inc. or its affiliates.

Figure 14: Amazon Cognito configured user pool[1]



```
/
  /books
    GET
    OPTIONS
    POST
    PUT
    /{id}
      /reviews
        GET
        OPTIONS
  /reviews
    GET
    OPTIONS
    POST
  /users
    OPTIONS
    POST
    PUT
    /{id}
      GET
      OPTIONS
```

Figure 15: API Gateway configuration[1]



Description
A simple backend (read/write to DynamoDB) with a RESTful API endpoint using Amazon API Gateway.

Last modified
8 days ago

Function ARN
arn:aws:lambda:us-east-1:109293137718:function:csci5409_addReview

Function URL
-

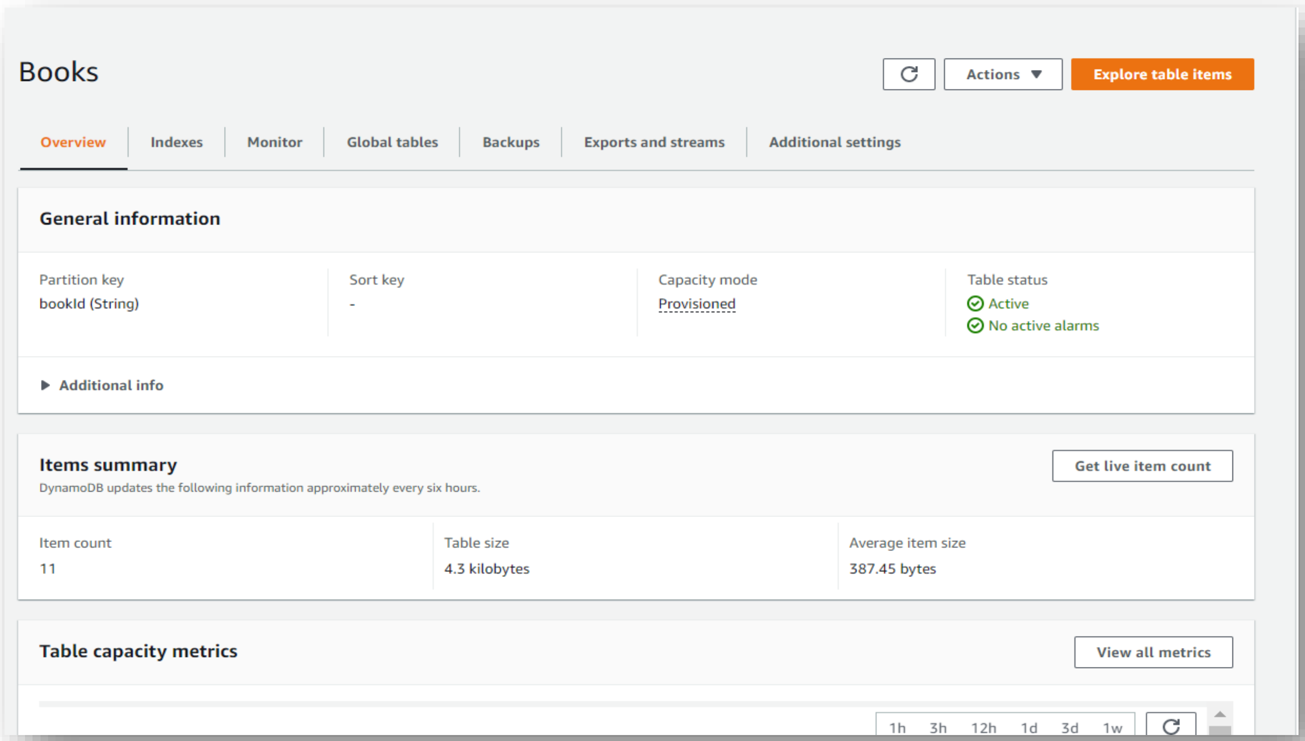
```
const AWS = require('aws-sdk');
const Comprehend = new AWS.Comprehend();
const dynamo = new AWS.DynamoDB.DocumentClient();

exports.handler = async (event, context) => {
    console.log('Received event:', JSON.stringify(event, null, 2));

    let body;
    let statusCode = '200';
    const headers = {
        'Content-Type': 'application/json',
         "Access-Control-Allow-Origin": "*",
          "Access-Control-Allow-Methods": "*",
          "Access-Control-Allow-Headers": "*",
    };

    try {
        let review = JSON.parse(event.body);
```

Figure 16: Amazon Lambda function configuration with associated API gateway[1]



General information

Partition key	Sort key	Capacity mode	Table status
bookId (String)	-	Provisioned	Active No active alarms

Items summary

DynamoDB updates the following information approximately every six hours.

Item count	Table size	Average item size
11	4.3 kilobytes	387.45 bytes

Table capacity metrics

Figure 17: Amazon DynamoDB table configuration[1]

References:

[1] "Cloud Services - Amazon Web Services," [Online]. Available: https://aws.amazon.com/. [Accessed 22 July 2022].

[2] "React” [Online]. Available: https://react.dev/. [Accessed 22 July 2022].

[3] "Express - Node.js web application framework," [Online]. Available: https://expressjs.com/. [Accessed 22 July 2022].

[4] "Elastic Beanstalk CloudFormation Template," [Online]. Available: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/quickref-elasticbeanstalk.html. [Accessed 22 July 2022]

[5] "AWS Cognito CloudFormation Template," [Online]. Available: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-cognito- userpool.html. [Accessed 22 July 2022].