CSCI 5409 - CLOUD COMPUTING

FINAL REPORT

GROUP 40- CLOUD TECHIES

Members and Contributors

Guryash Singh Dhall	**Soham Kansodaria**	**Samarth Jariwala**
B00910690	B00865680	B00899380
Guryash.dhall@dal.ca	sh788512@dal.ca	sm228153@dal.ca

Faculty of Computer Science
Dalhousie University

July 26, 2022

Table Of Content

Video URL:
https://dal.brightspace.com/d2l/common/viewFile.d2lfile/Database/MTI4MjY3MzY/CSCI5409_
Group40.mp4?ou=222417

Alternative URL:
https://youtu.be/-kJc_5X_wd0

GitLab URL:
https://git.cs.dal.ca/courses/2022-summer/csci4145-5409/group-40/-/tree/main/

1) Project Background

The Trip Management System will give you access to buy a trip from various companies that connected to from a main organization. Customers can see the whole itinerary and same trip options from some various companies with some changes in trip details.

Here in this system customer can see whole details of the trip with like where the journey starts and what connection use in whole itinerary like start from the airplane ticket or train ticket for journey and where bus pick up you from place to travel other places. In addition, it includes where you are staying like stay type hotel or any other accommodation with breakfast, lunch and dinner information and other information. In the trip if there are some tourist places where there are some attractions then other information provided about how much extra you must pay if you want to go there.

In our system, we included features like user management where user can register and after that they can login to their account. Other feature is the trip management where user search the trip, buy trip, and cancel the trip and in this admin will add trip for users. For checkout admin will observe the payment of customer.

This will help customers have a hassle-free experience of making bookings for their trips. It prevents customers from getting hooked by various schemes by the travel agents ultimately spoiling the travel experience.

2) Project Details

2.1 User Management

Features included:
a. Log In
b. Sign Up

We are implementing user registration and login using AWS Cognito. Amazon Cognito provides a built-in and customizable UI for user sign-up and sign-in. After registering with the basic information, the user can add much more information such as an address and a profile photograph. We are using the Amazon S3 storage service because photos are involved.
We are the AWS Lambda function to upload photos to the S3 service, which also serves as the application's backend. The AWS API Gateway is used to communicate between the frontend and backend of the application to access the Lambda function as an application programming interface (API) call.
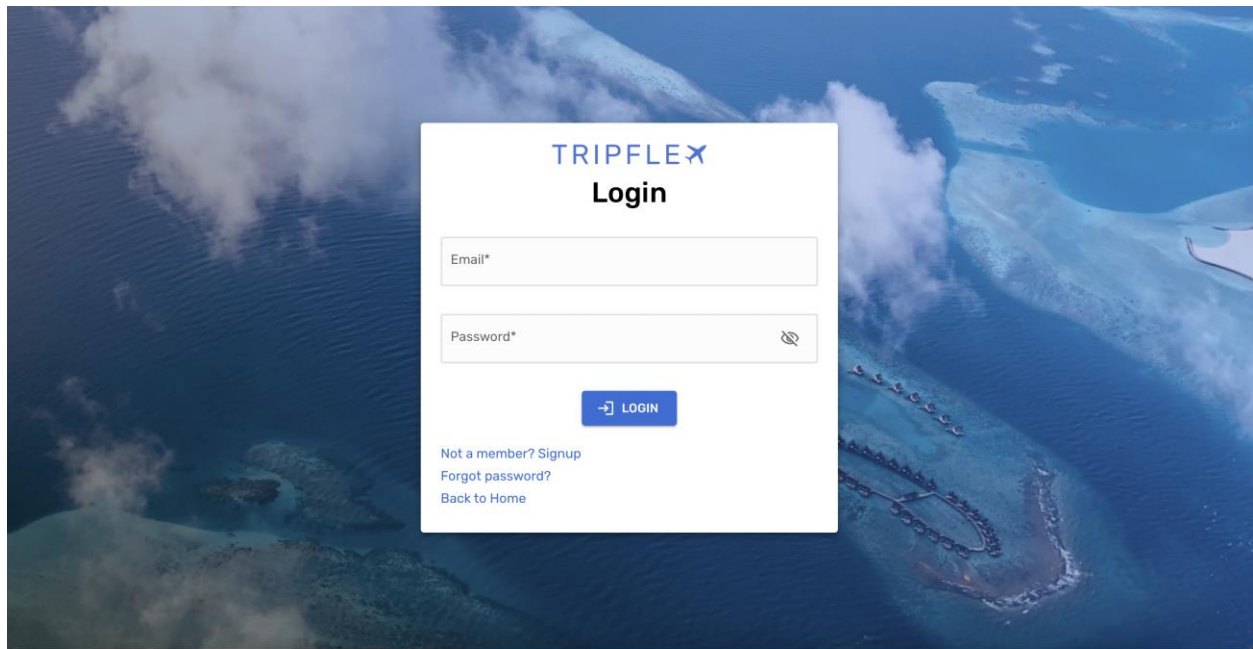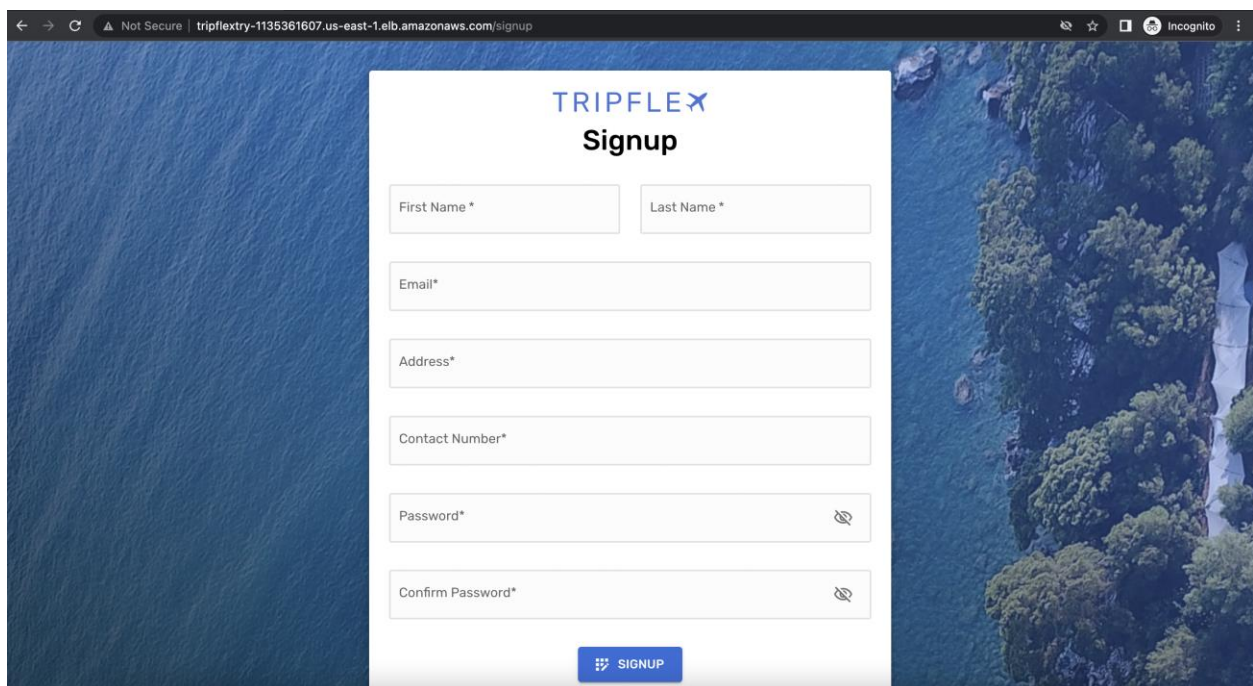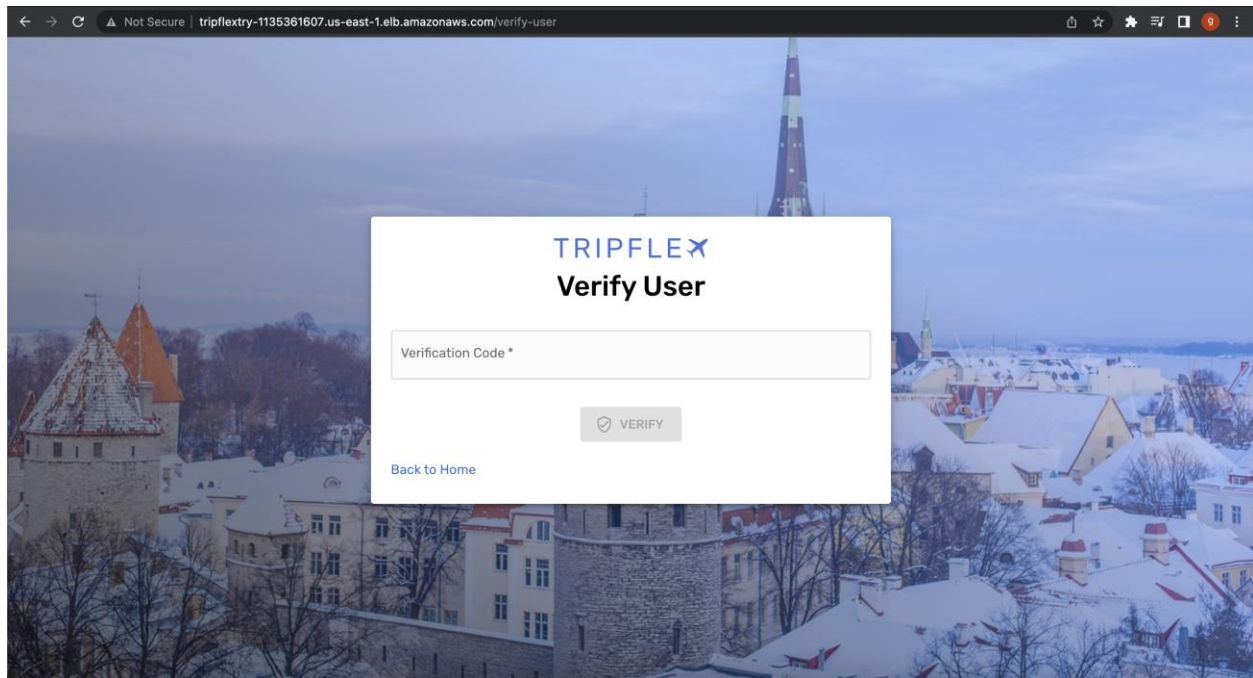
Figure 1 Login Page

Figure 2 Signup Page

Figure 3 Verify code for account verification

Homepage: The homepage is the default page which shows the trips created by admin. User can see this page even without getting logged in the application.
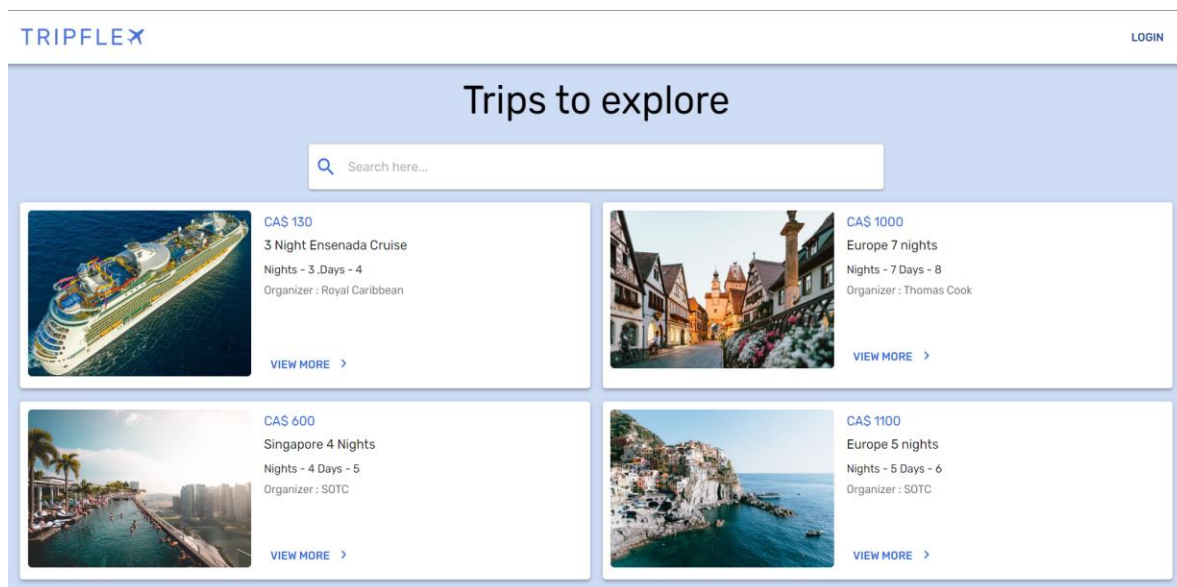

Figure 4 Homepage

2.2 Trip Management

Features included:
a. Search trips
b. Create new trip - Admin
c. Book Trip - Customer

The user is presented with a dashboard showcasing some of the trip options after logging into the application. A user can select their preferred mode of transport. A list of products that match your criteria will be presented. Because trip details and photos must be kept, we are using S3.
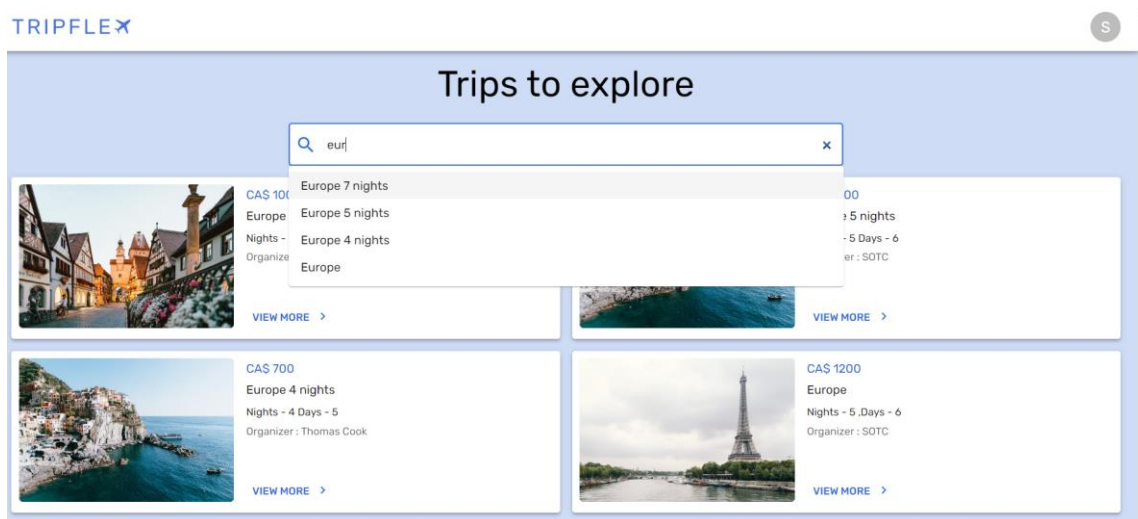

Figure 5 Search Trip

Figure 6 Book Trip

2.3 Checkout

The customer after finalizing the trip package and details has the option of paying via Credit/Debit Card.

Figure 7 Payment

Figure 8 Payment Confirmation

2.4 Admin

Features included:
  a. Create Trip
  b. View Trip, sub-trip, payment details

The create trip option is only available to admin. Admin can view the trip, sub-trip and payment details table

Figure 9 Admin Login

Figure 10 Admin Homepage

ID	Trip Name	Organizer	Accomodation	Duration	Extra Charges	Itinerary	Trip Price	Trip Type
M6EY	3 Night Ensenada Cruise	Royal Caribbean	03 nights at Standard Roo...	Nights - 3 ,Days - 4	If you want to acc...	Day 1 - LOS ANGELES - EX...	130	Cruise
6EUX	Europe 7 nights	Thomas Cook	03 nights at Hotel Holiday...	Nights - 7 Days - 8	Any increase in th...	Day 1 - Welcome to Zurich....	1000	Trip
P4L5	Singapore 4 Nights	SOTC	Hotel Ibis Novena / Similar	Nights - 4 Days - 5	Singapore Visa ch...	Day 1 - Arrive Singapore: ...	600	Trip
4GD8	Europe 5 nights	SOTC	03 nights at Hotel Holiday...	Nights - 5 Days - 6	Any increase in th...	Day 1 - Welcome to Zurich....	1100	Trip
5YAF	Europe 4 nights	Thomas Cook	03 nights at Hotel Holiday...	Nights - 4 Days - 5	Any increase in th...	Day 1 - Welcome to Zurich....	700	Trip

1 row selected   1–5 of 8

Day 1 - Welcome to Zurich. The Tour Manager / Station Manager will meet you outside Zurich airport. Proceed for dinner and then to the hotel and check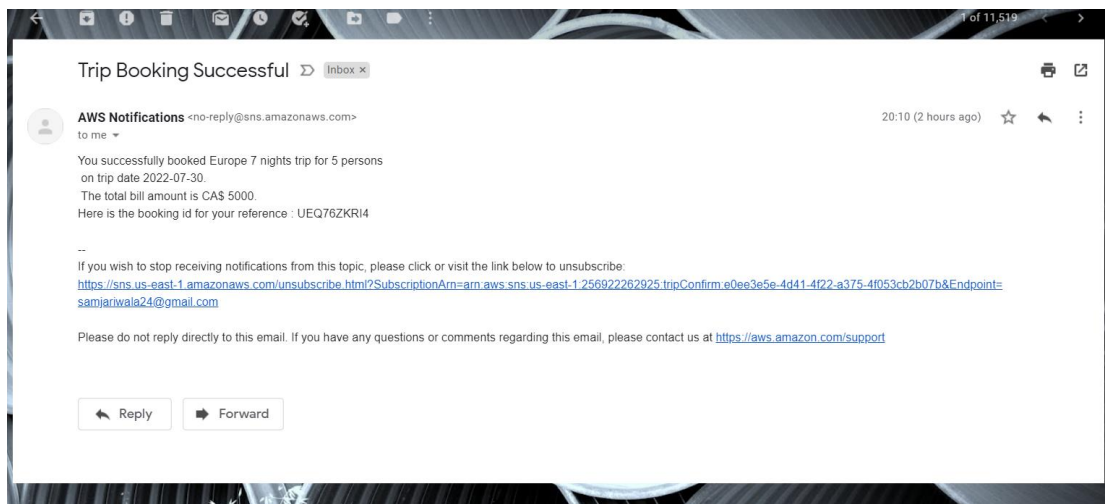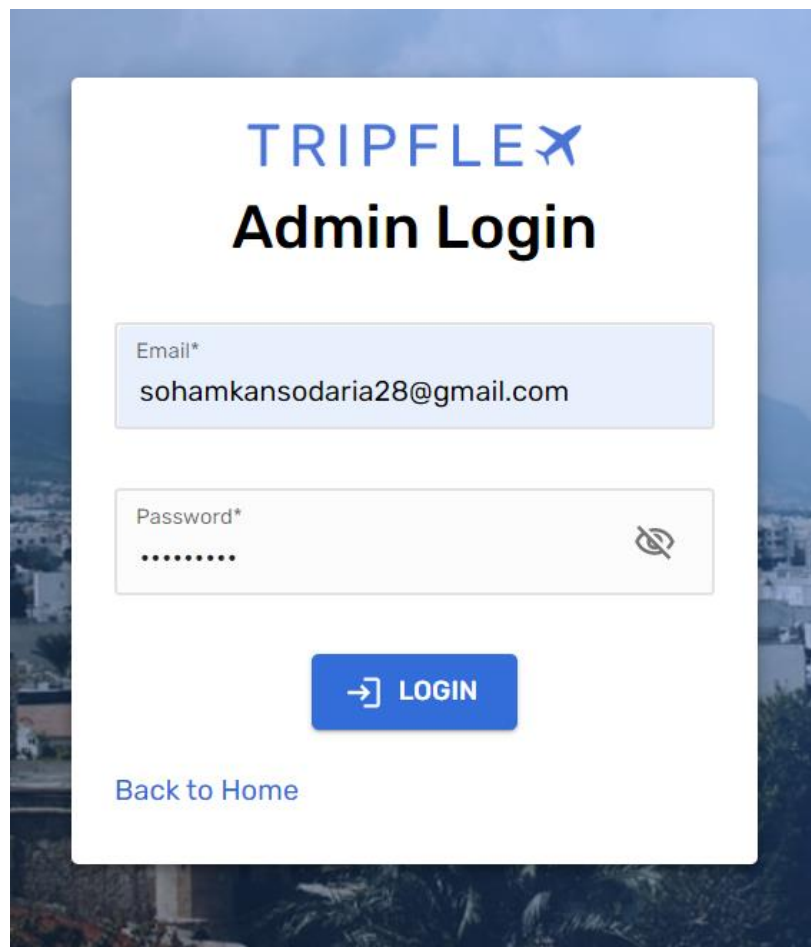-in. Overnight in Central Switzerland (D). Day 2 - Visit Mt. Titlis – Orientation tour of Lucerne. This morning, proceed to Mt Titlis where you will table cable car rides including the world's first rotating cable car, the Rotair – to the top of Mt. Titlis at 3,020 meters. Do not forget to visit the "Cliff Walk" a suspension bridge along the cliff of Mt. Titlis. Day 3 - Day at leisure OR Excursion to Jungfraujoch and Interlaken (Optional). Day is at leisure OR you can join us on an adventurous ride to visit Jungfraujoch - The Top of Europe. Take a cogwheel train from Lauterbrunnen / Grindelwald Grund to a height of 3,454 meters to Europe's highest railway station. Visit the Ice Palace and the Sphinx observatory deck to view the longest glacier in the Alps. Later, take a photo stop at Interlaken. Day 4 - See Rhine falls – Black Forest – Orientation tour of Heidelberg. This morning, check out & drive to Schaffhausen to see the Rhine Falls. Next visit the heart of Black Forest, for a demonstration on cuckoo clocks. Later drive to

Figure 11 Trip Table

Figure 12 Create Trip

3) Final Architecture Diagram
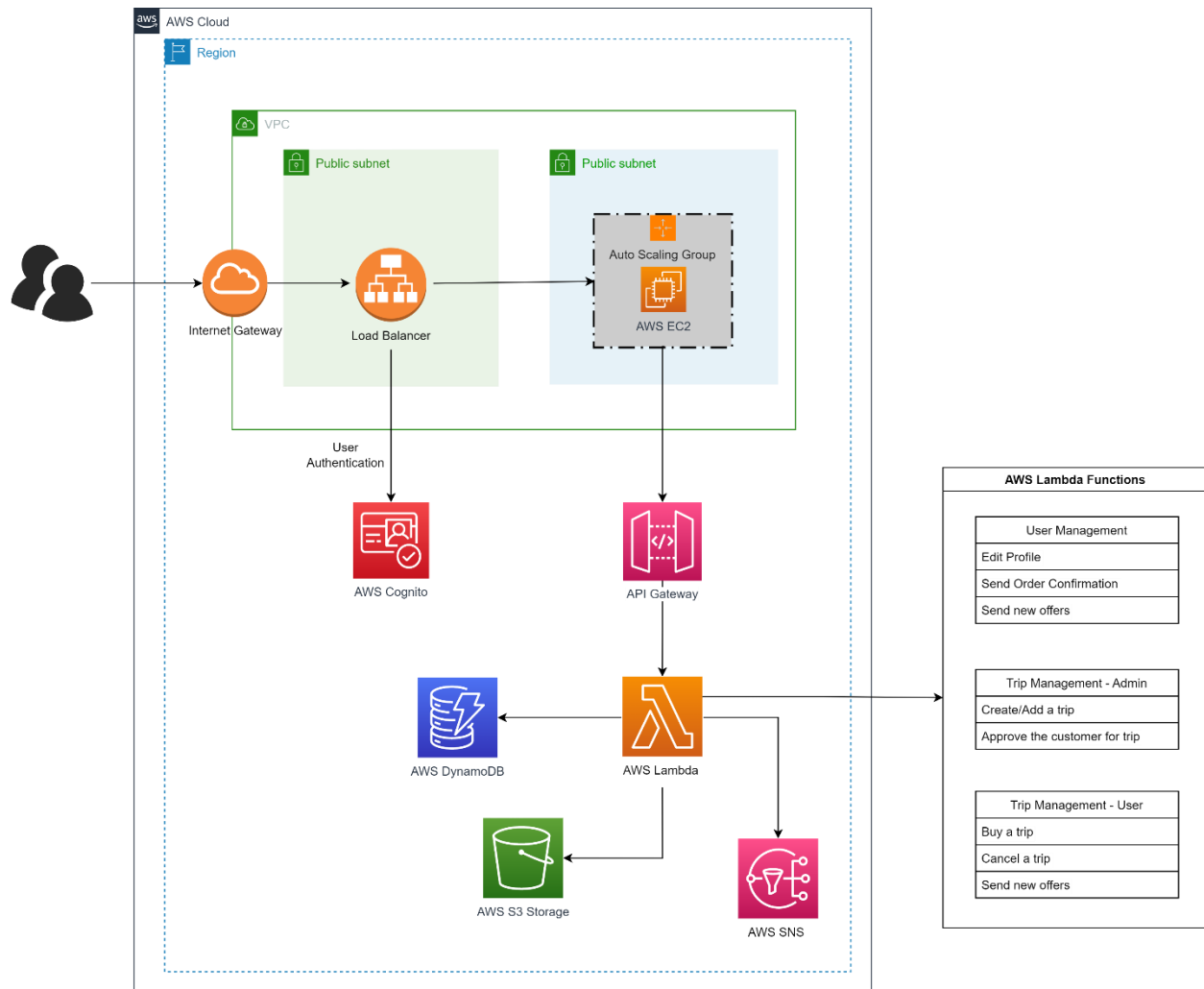
Figure 13. Architecture of Tripflex

3.1 Cloud Mechanisms and flow of data through the system

Our cloud application saves user information in AWS Cognito. Moreover, user have option to edit their information as you can see in Figure 13, there is an edit profile function to edit the profile. Other user information related to trip management is added into AWS Dynamo DB like Trip Id, Trip Name, Trip Duration, Trip Accommodation, Trip Price, Trip Itinerary, Trip Extra Charges, Trip Travel Information and, etc. Also, we are storing payment details connected to trip details with customers' status like user id, username, trip id, number of persons in trip, payment date, order status, etc.

The flow of data of our system is, first user is registering into our system and then logging to explore the trip, book the trip, edit the profile, etc. To execute all these

functions, we will make lambda functions like an API connected to the API gateway. The process is that the front-end will send invoke notification to API gateway with Lambda function, which then will reply with the respected response after doing the task mentioned in the request. Here lambda function will communicate with S3 and Dynamo DB for the requested operation. In addition, SNS will also send the message, which will trigger by the lambda function to send details of the trip to users and send some offers after adding trips from the admin side.

3.2 Programming Language

Programming languages used to make this application are:
- **Node JS:** We have used JavaScript to build our application. Apart from that a lambda function is written in Node JS.
- **React:** We have used React JS to build the frontend of our application. The reason to go with React JS and Node JS is because all the group members were familiar with these languages to develop web application.
- **Python:** Almost all lambda functions are written in python. The Boto3 library of the python makes it easier to create lambda function of the services used in the application.

The user interaction with the application has code which is written in React and NodeJS. The lambda function to use the AWS services requires code. The cloud formation is created using YAML file.

Libraries used in the project:
- **Boto3:** It is the AWS SDK for Python, which provides a Python API to build applications in AWS infrastructure services.[2]
- **MUI:** It is a library for React which helps to design the front-end of the application.[1]

3.3 System Deployment

To deploy our frontend system, we used an EC2 instance which will require pm2 to run continuously. For deployment of the frontend react app we first need to install node, git, pm2, and serve. After deploying the app, we have done load balancing, which automatically diverts to another region if viewership increases. For overall system deployment, we created a cloud formation file that will provision all required resources for the project if needed in future times.
In the backend functionality, we are using the API gateway with lambda function which requires a JWT token for some requests in the frontend. To manage all users we are using the Cognito user pool with built-in attributes and profile types to differentiate the user and admin. Here we have three dynamo DB tables that will manage all the data of trips and payments. Here the trip will store main trip detail information whereas, the sub-trip table will store the date wise trip capacity and other details. In addition, the payment table will store the payment card, trip name, the number of people booked, user mail, and other

necessary information. To store the image of the trip we are using the S3 bucket which will give publicly access to object to fetch in frontend to show the users.

Figure 14, depicts the target group I created with ec2 instance.
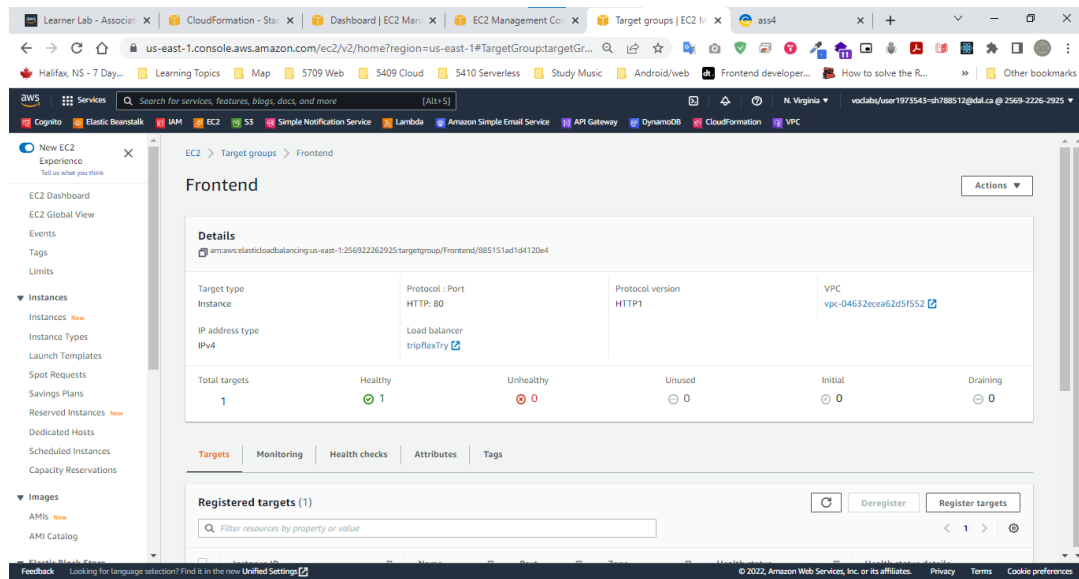
Figure 14. Target group connected with load balancer

In Figure 15, as you can see the load balancer with respective target group which have ec2.
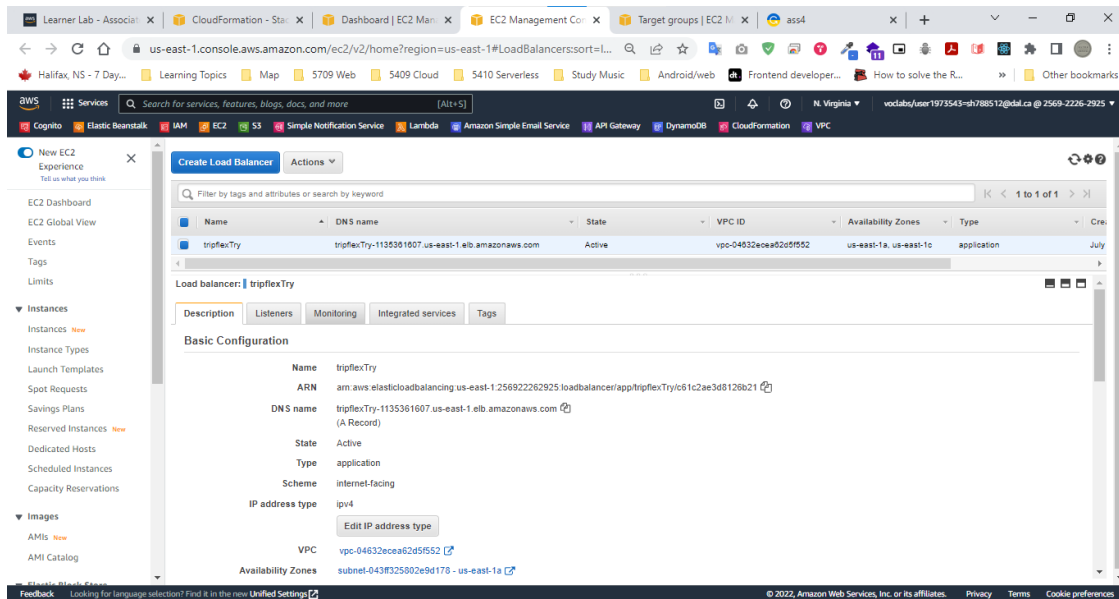
Figure 15. Load balancer connected with respective ec2 instance

In below Figure , you can see the security and region division configuration for load balancer we used.
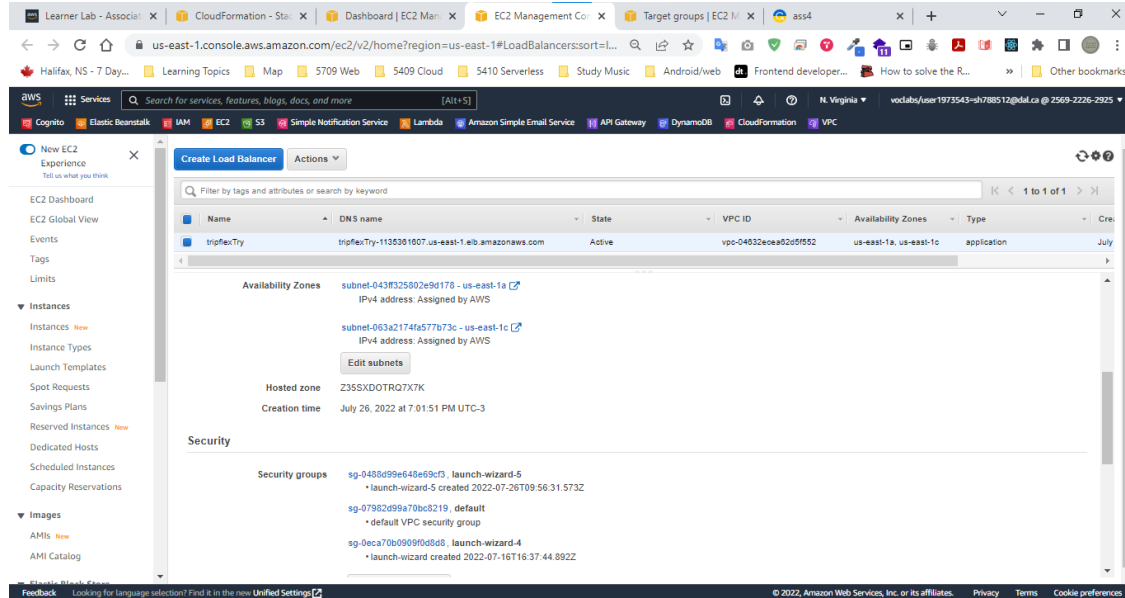
Figure 16. Security and region division in load balancer

4) Determine Architecture

The architecture of our application remains same as the original project proposal. The services used by our application that are cost based are:

AWS EC2, AWS VPC, AWS API Gateway, AWS S3, AWS Dynamo DB, AWS Cognito, AWS Lambda, AWS SNS.

The change which would have resulted in more cost-efficient architecture is removing API gateway. API gateways are usually for developers to easily publish, monitor and operate APIs at any scale. The pricing of API gateways is:

- HTTP APIs:
  - First 300 million: $1.00 per million
  - 300+ million: $0.90 per million
- Caching:
  - Price increases from 0.5 GB: $0.02 per hour
  - To 237 GB: $3.80 per hour
- Message Transfers:
  - First 1 billion: $1.00 per million
  - Over 1 billion: $0.80 per million
- Connection Minutes:
  - Per million connection minutes: $0.25

5) Future Scope

The application can be updated with features like My Trips for the users, which shows the booked trip of the user and Delete Trip for the admin, which deletes the created trip by admin. The AWS services which will be used to fulfill these features functionality are lambda function and Dynamo DB table. Although, these two services are already used by the application. Another feature which can be implemented is real time payment.

The other feature for the application is sending the user an email if they are booking the trip which is full. This would require a new topic in SNS which sends user an email mentioning the trip they are trying to book is full. The Newsletter feature sends subscribed customers an email about new offers on trip.

We would use AWS Deep Compose as a machine learning tool which help us to build a trip recommendation feature for our application.

6) Data Security

- **AWS EC2:** EC2 encryption provides EBS volumes with the help of KMS keys. It does encryption at physical layer. It also does encryption between instances.[7]

- **AWS API Gateway:** Data encryption in API Gateway is at rest and in transit. It supports cache encryption. It does not support unencrypted (HTTP) endpoints.[8]

- **AWS Cognito:** Data is encrypted in two categories in Cognito: encryption at rest and encryption in transit. Data within Amazon Cognito is encrypted at rest in accordance with industry standards. The request made in Cognito are over the Transport Layer Security protocol (TLS).[9]

- **AWS DynamoDB:** Users' data kept at rest, in transit between on-premises clients and DynamoDB, and between DynamoDB and other AWS services located in the same AWS Region are all protected by DynamoDB.[10]

- **AWS Lambda:** Environment variables can be used to safely store secrets for usage with Lambda functions. By default, Lambda encrypts your environment variables using an AWS KMS key. Lambda encrypts the uploaded files, deployment packages and layer archives. Amazon cloud watch logs are also encrypted by default.[11]

- **AWS SNS:** The AWS shared responsibility model provides data protection to all AWS services. But in SNS it's the developer's responsibility to check if the user's sensitive information is not getting shared in email.[12]

7) Security Mechanisms for data security

The security mechanisms that we have used for this application are Hashing and SSO. Hashing is done for the password protection and SSO for generating JWT Token for a user to enhance user security.

- **SSO (Single Sign On)**

  This approach is used when AWS Cognito generates the JWT Access Token and when we need to grant the authenticated user access to the offered services and the ability to switch between them. We have used AWS Cognito's OAuth as it follows the industry standards. The user can access all the services offered by our application using this token.

- **Hashing**

  In our application hashing of password is done using Cognito, where the customer's credentials are encrypted and stored in Dynamo DB. [13]
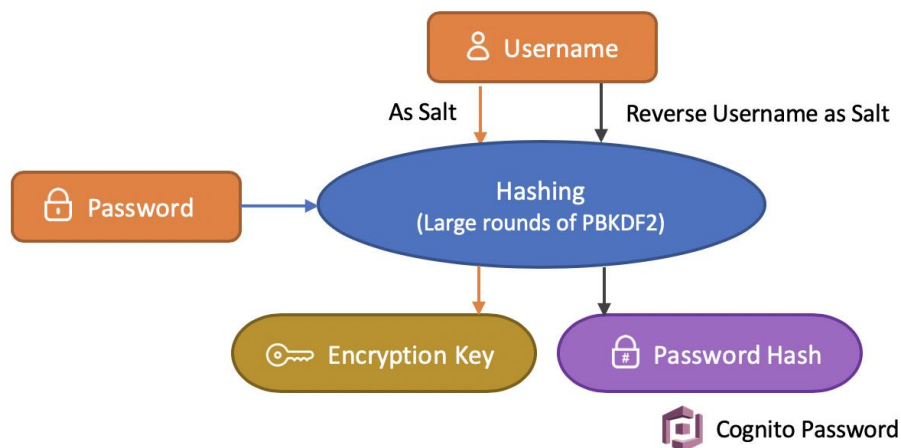
Figure 17 Hashing [13]

8) Cost for Reproduction

For setting up a private cloud we need to take into consideration the following factors:-

- Location: Geographic location to set up the cloud infrastructure taking into consideration the data storage regulations and norms
- Technology Reliability:
- Integration and Scalability: The resources can be integrated with new equipment for future use and is scalable as per the demand

The cost of reproducing the system would require installation of servers, CPUs, hardware, as

well as software resources. We would also need to take into consideration the installation of all this set up and maintenance cost of the set up.

For our application, considering all these factors it will cost around ~3200 $ for setting up and running private cloud for a year.

9) Cost Monitoring for cloud mechanisms

The cloud mechanism which has the most potential for costing money in our Tripflex application is EC2. The cost of EC2 instance that we are using t2 small -8 GB, is around 0.023 $ per hour. It has the major cost contributor in our application.

Another possibility of cost increase is through scalability of the Dynamo DB tables. The price increases when the usage of dynamo DB table increases.

The monitoring mechanism that we have used is CloudWatch. EC2 instance in Cloud Watch falls under free tier. It gives a detail insight in the form of logs, metrics, and events. To monitor the resources, we can set up alarms on the Cloud Watch and visualize the logs which can help track in case of any unusual activities.

References

[1]   "MUI: The React component library you always wanted," *Mui.com*. [Online]. Available: https://mui.com/ [Accessed: 26 July 2022].

[2]   Boto3.amazonaws.com. 2022. *Quickstart — Boto3 Docs 1.24.38 documentation*. [online] Available at: https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html [Accessed 26 July 2022].

[3]   Brightspace[online] Available at: https://dal.brightspace.com/d2l/common/viewFile.d2lfile/Database/MTI0OTM4OTc/Group40-CloudTechies.pdf?ou=222417 [Accessed 26 July 2022].

[4]   Brightspace[online] Available at: https://dal.brightspace.com/d2l/common/viewFile.d2lfile/Database/MTI2MDUzMjA/Group_40_Mechanisms_Critical_Analysis_And_Response.pdf?ou=222417 [Accessed 26 July 2022].

[5]   Brightspace[online] Available at: https://dal.brightspace.com/d2l/common/viewFile.d2lfile/Database/MTI2ODY5Mjg/Group_40_Architecture_Critical_Analysis_And_Response.pdf?ou=222417 [Accessed 26 July 2022].

[6]   Amazon Web Services, Inc. 2022. *Amazon API Gateway Pricing | API Management | Amazon Web Services*. [online] Available at: https://aws.amazon.com/api-gateway/pricing/ [Accessed 26 July 2022].

[7]   AWS [online] Available at: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/data-protection.html [Accessed 26 July 2022].

[8]   *AWS*. [online] Available at: https://docs.aws.amazon.com/apigateway/latest/developerguide/data-protection-encryption.html [Accessed 26 July 2022].

[9]   AWS. [online] Available at: https://docs.aws.amazon.com/cognito/latest/developerguide/data-protection.htm [Accessed 26 July 2022].

[10]   *AWS*. [online] Available at: https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/data-protection.html [Accessed 26 July 2022].

[11]   AWS. [online] Available at: https://docs.aws.amazon.com/lambda/latest/dg/security-dataprotection.html [Accessed 26 July 2022].

[12]   *AWS*. [online] Available at: https://docs.aws.amazon.com/sns/latest/dg/sns-data-protection.html [Accessed 26 July 2022].

[13]   "End to end encryption on Cloud Platforms", *Medium*, 2022. [Online]. Available: https://medium.com/@sterling.shrey/end-to-end-encryption-on-cloud-platforms-411d4a0774ca [Accessed: 26- Jul- 2022].