OSTBAYERISCHE TECHNISCHE HOCHSCHULE REGENSBURG

Informatik und Mathematik

REGENSBURG

INTRODUCTION TO CLOUD COMPUTING - KICC

Travel Guide using AWS

Lecturer: **Mr. Maximilian Schön**
Student: **Nghia Le Minh**
Student ID: **3397327**

05/2023

Table of content

Abstract

Amazon Web Service(AWS) is a globally recognized cloud computing platform that changes the way organizations build, deploy, and manage their applications. It offers a wide variety of services, hosted in the cloud, such as computing power, storage, database, artificial intelligence, networking, and so much more. These services are always offered with high availability and scalability to satisfy the ever-evolving needs of businesses in the modern world. Businesses can also gain access to these services through cloud computing, which eliminates the need to invest in building and maintaining highly expensive data centers around the world. This helps to lower the bar for entrepreneurs to start their businesses much more easily. Since its birth, AWS has redefined the world of cloud computing, offering organizations a diverse toolset that can change seamlessly according to their need and with cost-efficiency.

In order to demonstrate the benefits that AWS brings to the table, I have utilized many different services to create a travel-focused website which helps users to search and display travel-relevant information gathered from many sources over the Internet through their API.

Overview

In this project, I have created a website, called “The Travel Plan”, focused on gathering and displaying information regarding places of interest that can be found using a query provided by users. There are three main pages offered by the website, consisting of a landing page, a search page, and a page that ranks location search results based on their views. The website provides users with the ability to search for places using free-from text query and also the ability to view how many times certain locations have been searched and viewed by other users before.

The tech stack of the website consist of:

- Front-end: HTML, CSS, JavaScript using the React library.
- Back-end: Node.js in AWS Lambda functions.
- Database: AWS DynamoDB

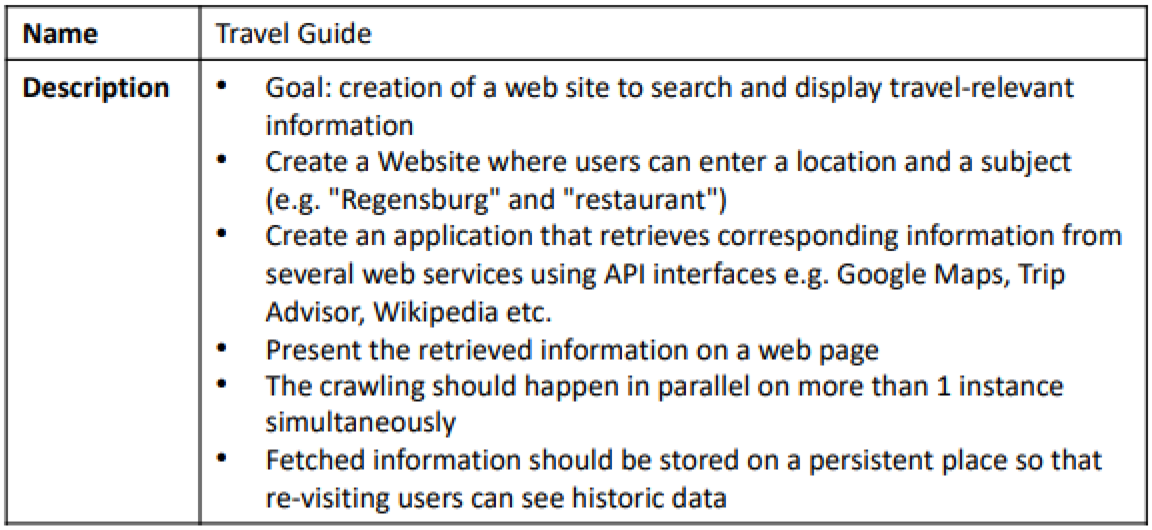
The website also used other online resources for styling, including:

- Fonts offered by Google Fonts.
- Icons offered by Font Awesome.
- Styling offered by the library, React - Bootstrap
- The ability to use AJAX from the jQuery library
- Routing capability from the react-router package

Requirements Analysis

Through the requirements of this project on the topic of a travel guide website implemented with AWS services. Several critical features can be extracted from the specification, including:

- Free-form text query.
- API Information Retrieval.
- Data Display.
- Parallel Crawling.
- Information Storage.



Name	Travel Guide
Description	• Goal: creation of a web site to search and display travel-relevant information • Create a Website where users can enter a location and a subject (e.g. "Regensburg" and "restaurant") • Create an application that retrieves corresponding information from several web services using API interfaces e.g. Google Maps, Trip Advisor, Wikipedia etc. • Present the retrieved information on a web page • The crawling should happen in parallel on more than 1 instance simultaneously • Fetched information should be stored on a persistent place so that re-visiting users can see historic data

Image 1: *Requirements*

1) Free-form text query:

The first requirement is that users should be able to enter a location and a subject into the search bar, submit them and receive back a list of results from the website containing the corresponding locations and their basic information: name, address, type of location, contact information and additional details.

For example:

i. User submitted a query: “Regensburg cafe”.
ii. Website responded with a list of coffee shops located in the city of Regensburg, for example “Cafe-Bar Regensburg”, “Cafe Lila”,...
iii. For each location, there is also their address, type of location and contact information included with the result.

2) **API Information Retrieval:**

The second requirement is that the information about the corresponding locations should be retrieved from web services using the API interface. The data should then be reformatted and displayed onto the web page.

For example:

i. A query of “Regensburg cafe” is submitted to the website.
ii. The website sends that data to HereAPI through the API interface.
iii. HereAPI sends back the list of related locations such as “Cafe-Bar Regensburg”, “Cafe Lila”,...
iv. The website reformats the data to be suitable to be displayed.

3) **Data Display:**

The third requirement is that users should be able to view specific information such as the location website, a brief summary about the place, its user ratings and reviews, … by clicking on one of the location results displayed by the website after a search was carried out.

For example:

i. User clicks on “Cafe-Bar Regensburg”.
ii. More specific information about the place such as its description, its contact information, its rating among users etc, is revealed to the customer.

4) **Parallel Crawling:**

The fourth requirement is that when a query is submitted to the website, it should be sent to multiple instances of APIs in parallel. Whenever an API returns the result, it should be immediately displayed to the user without waiting for the others to finish.

For example:

i. A query is submitted to the website.
ii. The query is then processed and sent to multiple APIs. In this case, they are HereAPI and SerpAPI.
iii. The data is then returned from the APIs.
iv. Whichever data arrives first to the website is then immediately displayed onto the web page to the user.

5) Information Storage:

Information collected from the API calls should be stored in a persistent database so that returning users can see the statistics about the searches conducted by the website.

For example:

i. Whenever a search result is returned from a query submitted by a user, that result is then logged into the database and then the view count is increased accordingly.
ii. The user can then access the ranking page.
iii. The logged data is then pulled from the database and displayed to the user.

Functionality

From the requirements specified previously, several functionality has been included into the website, which are:

- Location search: A function that takes in a query provided by the user, calls the APIs and then displays the result onto the webpage in the form of a list of locations of interest.
- View detail: The user can view additional information about a location returned by the "Location search" function.
- Article search: A function that also takes in the query and calls the corresponding API at the same time as the "Location search" function, which then will return a list of relevant articles gathered from a Google search.
- View result ranking: Users can view the ranking of locations based on the number of "views" - the time that this location has been returned from the "Location search" function.

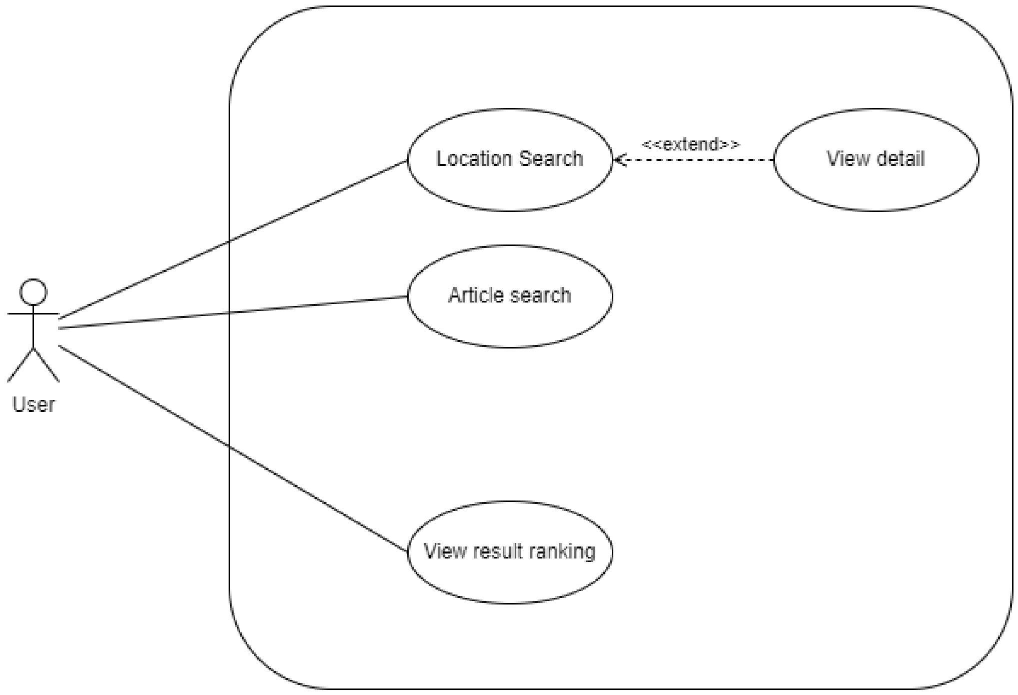


Image 2: *Use-case diagram*

1) Location search:

- Name: Location search
- Description: User can submit query to search for locations
- Normal flow:
    a) User enters the query into the search bar and presses the "Search" button.
    b) The website first cleared the screen of any result from the previous search.
    c) The query is passed as a parameter to an AJAX call to a Lambda function called "apiCallForLoc", along with the current location in the form of longitude and latitude of the user.
    d) The Lambda function receives the AJAX call, then extracts the data from the parameter of the call. The function then sends a request to the HereAPI to get a list of related locations.
    e) For each location in the result returned by HereAPI that contains a reference ID from TripAdvisors, another three requests are made in order to retrieve information on the location's images, its reviews, and its general information(containing its description, ratings, and total user reviews that will be used).
    f) The result list is then returned back to the front-end of the website in the form of JSON.
    g) After receiving a response from the Lambda function, the website proceeds to extract the location results and their details and display them as search results for the user.
    h) The website then creates a POST request to a Lambda function called "writeTable", to log the received results into a DynamoDB database called "locationResult".

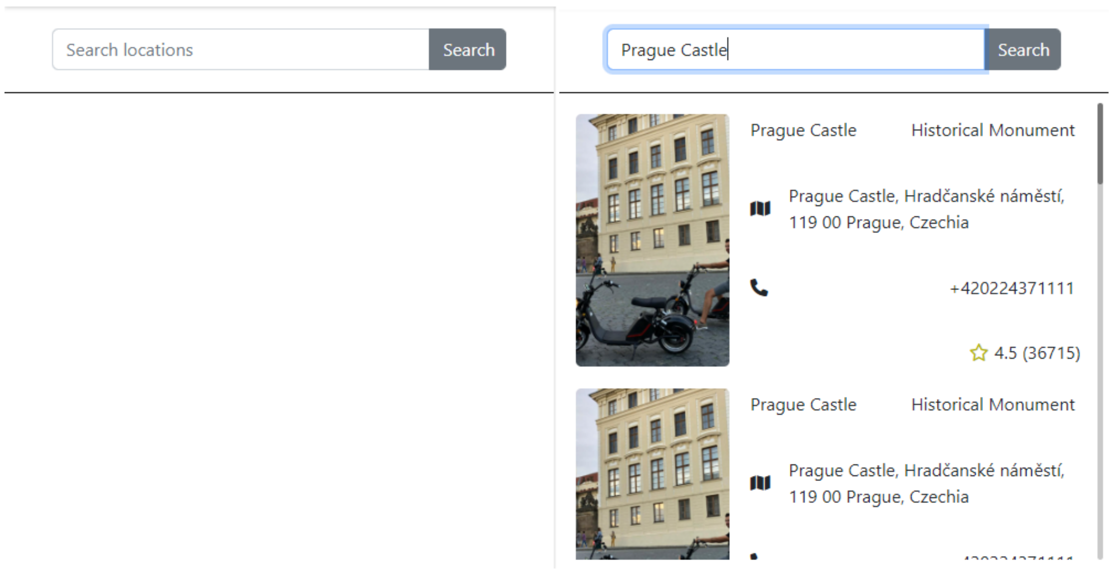


Image 3: *Search area before and after a search on "Prague Castle"*

2) View detail:

- Name: View detail
- Description: User can view additional detail about a location returned in the "Location search" function.
- Normal flow:
    a) User clicks on a card in the <u>search result area</u> to signal to the website that they want to view the location in greater detail.

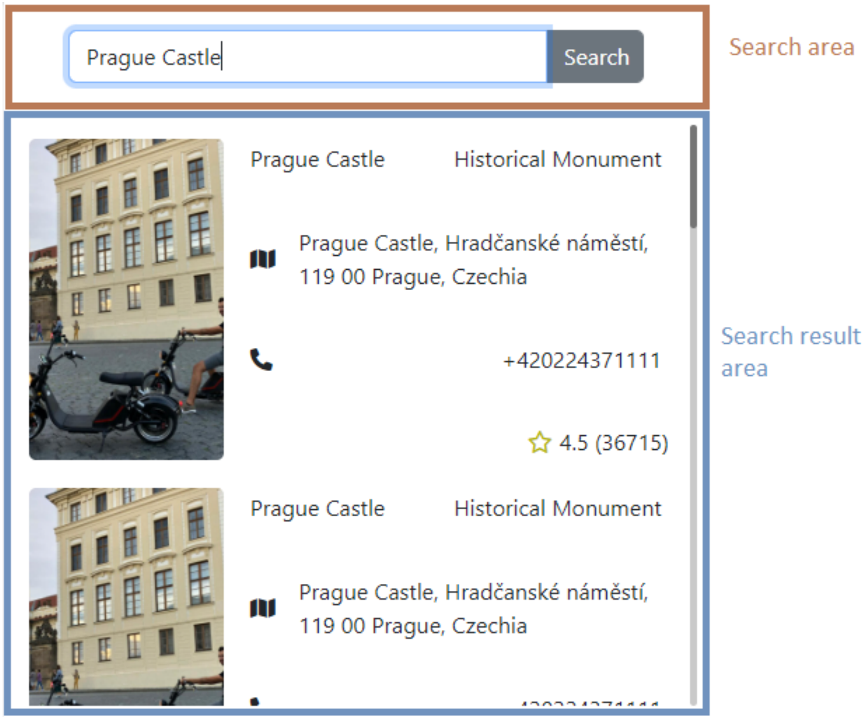
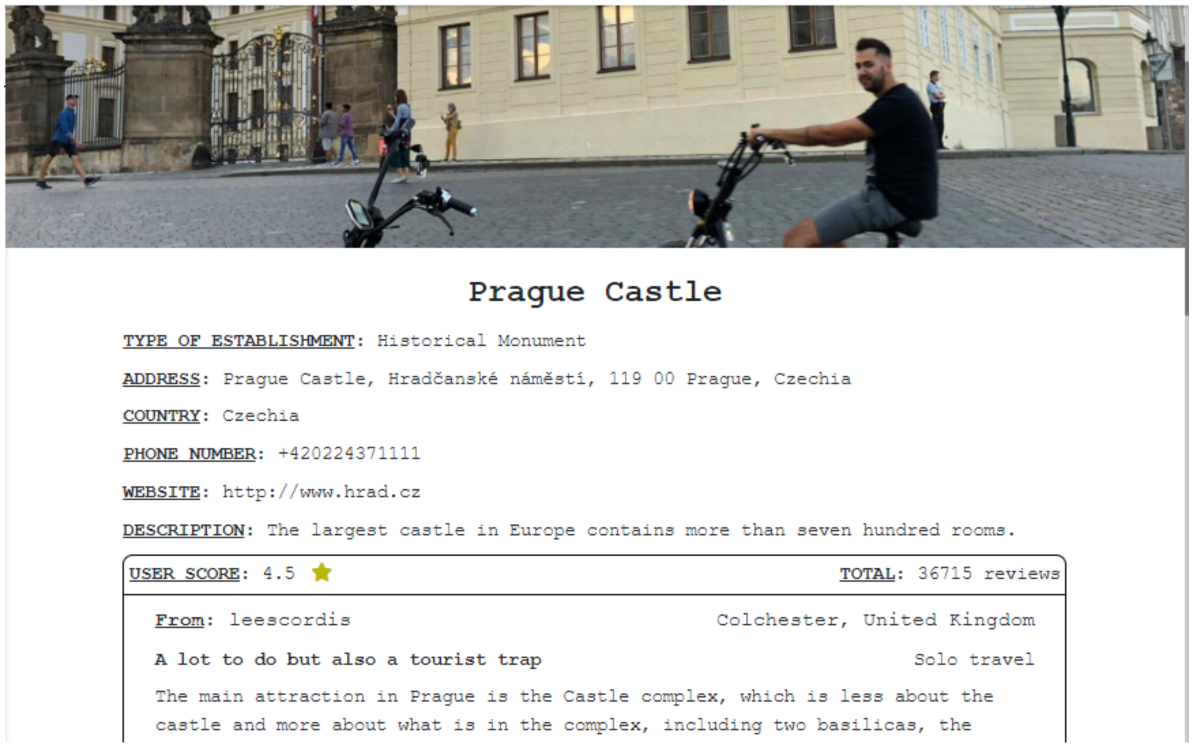


Image 4: *Areas of the front-end of the Location search function*

b) The website receives the signal and uses the information from the JSON object corresponding to the clicked location to display a detailed view of the chosen location. The detailed view contains information such as:
   + Image of the place
   + Name
   + Type of establishment
   + Address
   + Country
   + Phone number
   + Website
   + Brief description
   + User ratings
   + User reviews and scores



Prague Castle

TYPE OF ESTABLISHMENT: Historical Monument

ADDRESS: Prague Castle, Hradčanské náměstí, 119 00 Prague, Czechia

COUNTRY: Czechia

PHONE NUMBER: +420224371111

WEBSITE: http://www.hrad.cz

DESCRIPTION: The largest castle in Europe contains more than seven hundred rooms.

USER SCORE: 4.5 ★ TOTAL: 36715 reviews

From: leescordis Colchester, United Kingdom

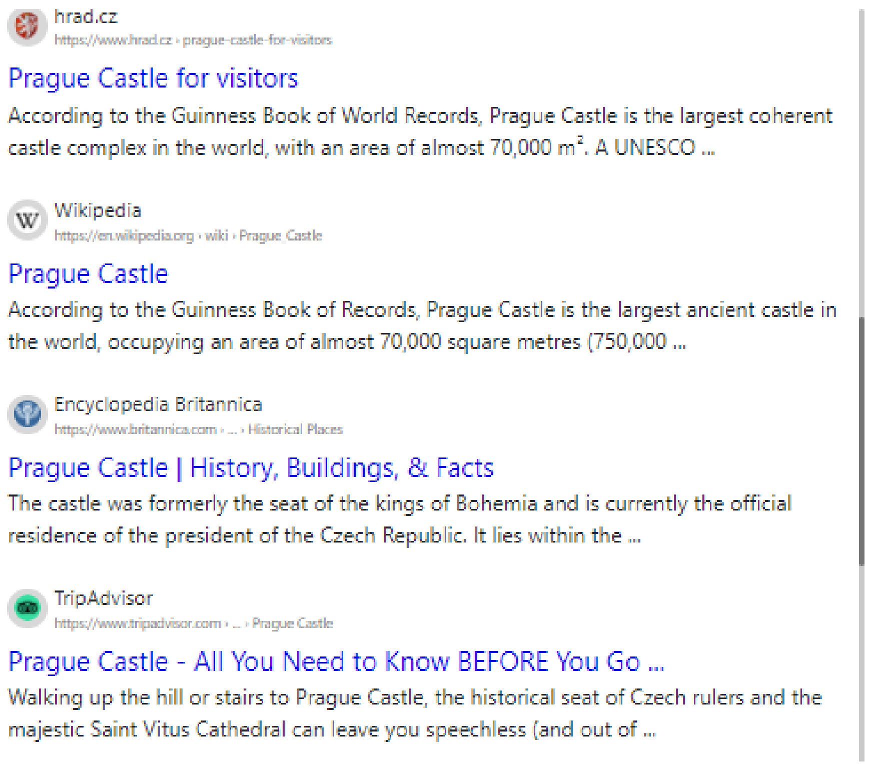
A lot to do but also a tourist trap Solo travel

The main attraction in Prague is the Castle complex, which is less about the castle and more about what is in the complex, including two basilicas, the

Image 5: *Detailed view of Prague Castle location*

3) Article search:

- Name: Article search
- Description: Using the query provided by the user, the website searches for related articles from Google.
- Normal flow:
    a) User enters the query into the search bar and presses the "Search" button.
    b) The website first cleared the screen of any result from the previous search.
    c) The query is passed as a parameter to an AJAX call to a <u>Lambda function</u> called "apiCallForPost".
    d) The Lambda function makes a request to SerpAPI with the query as one of the parameters.
    e) SerpAPI returns the results in the form of a search using Google Search Engine.
    f) The Lambda function proceeds to extract the organic result from the result received from SerpAPI and reformats them.
    g) The list of articles is then returned to the website to be presented to the user.



hrad.cz
https://www.hrad.cz › prague-castle-for-visitors
Prague Castle for visitors
According to the Guinness Book of World Records, Prague Castle is the largest coherent castle complex in the world, with an area of almost 70,000 m². A UNESCO ...

Wikipedia
https://en.wikipedia.org › wiki › Prague Castle
Prague Castle
According to the Guinness Book of Records, Prague Castle is the largest ancient castle in the world, occupying an area of almost 70,000 square metres (750,000 ...

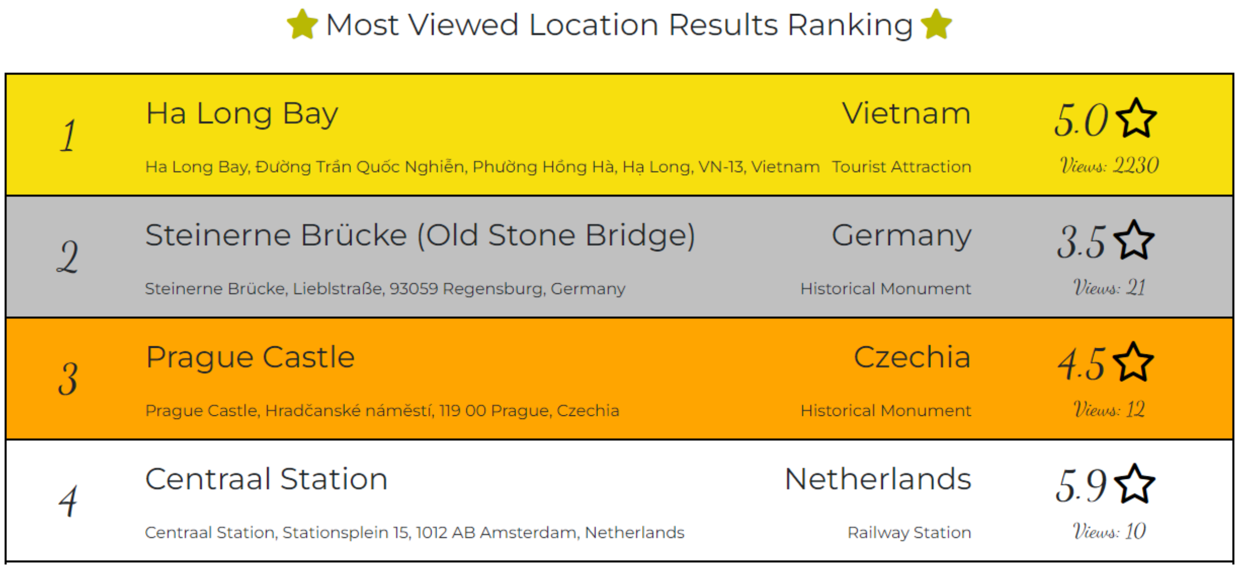
Encyclopedia Britannica
https://www.britannica.com › ... › Historical Places
Prague Castle | History, Buildings, & Facts
The castle was formerly the seat of the kings of Bohemia and is currently the official residence of the president of the Czech Republic. It lies within the ...

TripAdvisor
https://www.tripadvisor.com › ... › Prague Castle
Prague Castle - All You Need to Know BEFORE You Go ...
Walking up the hill or stairs to Prague Castle, the historical seat of Czech rulers and the majestic Saint Vitus Cathedral can leave you speechless (and out of ...

Image 6: *Articles list displayed for query "Prague Castle"*

4) View result ranking:

- Name: View result ranking
- Description: User can view the number of times a location comes up as a result from a user's search carried out by the website.
- Normal flow:
    a) User clicks on the Ranking button on the navigation bar to access the result ranking board.
    b) The website loads the page, then makes a GET request to a Lambda function called "getRanking".
    c) The Lambda function gets information about the locations from the DynamoDB table, "locationResult".
    d) The Lambda function returns the data in the form of JSON back to the website.
    e) The website proceeds to sort the data based on the number of views.
    f) The reordered data is then displayed onto the ranking board for the user to see.



Most Viewed Location Results Ranking

1	Ha Long Bay Ha Long Bay, Đường Trần Quốc Nghiễn, Phường Hồng Hà, Hạ Long, VN-13, Vietnam	Vietnam Tourist Attraction	5.0 ☆ Views: 2230
2	Steinerne Brücke (Old Stone Bridge) Steinerne Brücke, Lieblstraße, 93059 Regensburg, Germany	Germany Historical Monument	3.5 ☆ Views: 21
3	Prague Castle Prague Castle, Hradčanské náměstí, 119 00 Prague, Czechia	Czechia Historical Monument	4.5 ☆ Views: 12
4	Centraal Station Centraal Station, Stationsplein 15, 1012 AB Amsterdam, Netherlands	Netherlands Railway Station	5.9 ☆ Views: 10

Image 7: *Ranking Board with data from DynamoDB*

AWS Architecture

The website was developed using a wide variety of services that AWS offered. The services can be categorized into five groups based on their usage:

- Hosting: AWS Amplify
- Back-end hosting: AWS API Gateway, AWS Lambda, AWS DynamoDB, AWS S3.
- Networking: AWS Virtual Private Cloud(VPC)
- Log access: AWS Cloudwatch
- Security: AWS Identity & Access Management(IAM)

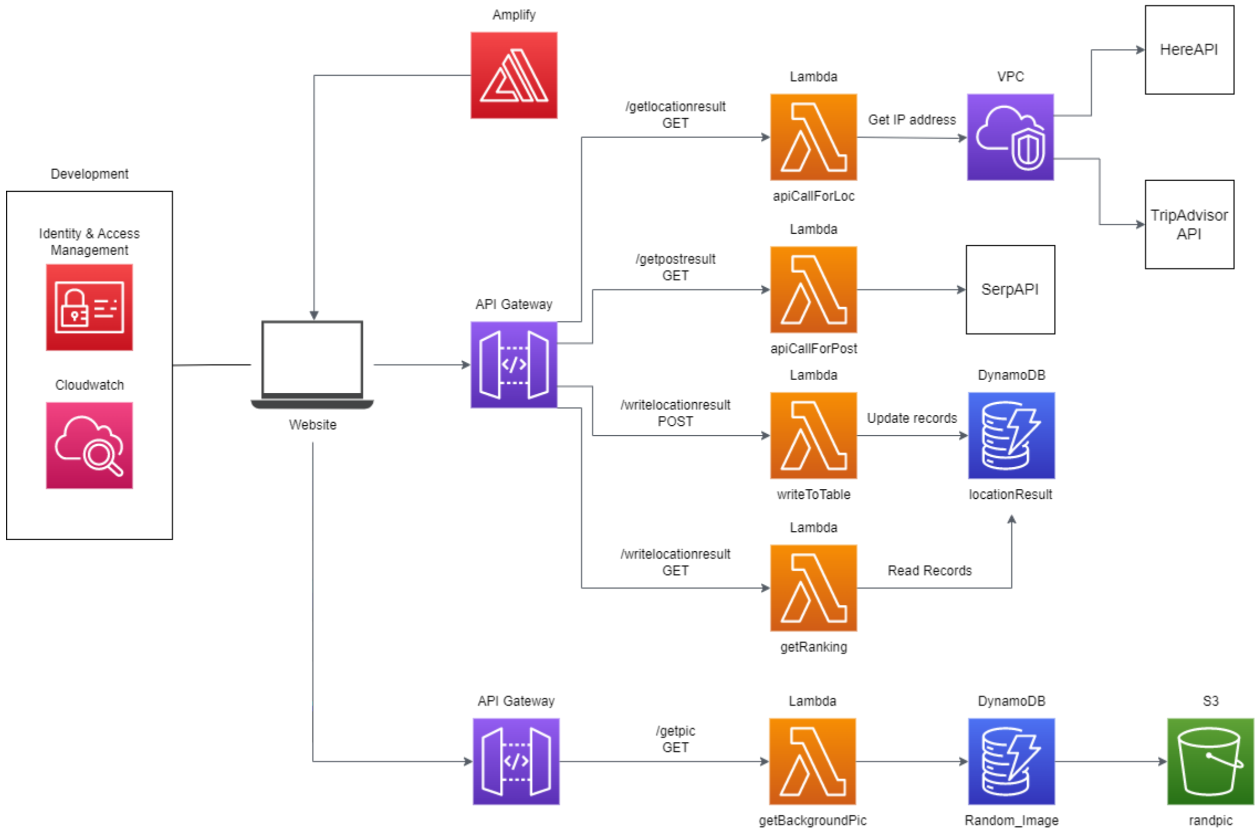


Image 8: *AWS Architecture of the website*

1) AWS Amplify:

"AWS Amplify is a set of purpose-built tools and features that enables frontend web and mobile developers to quickly and easily build full-stack applications on AWS."[1] It provides developers with services and tools that allows for a streamlined development workflow and continuous deployment. Amplify provides two main services: Amplify Hosting and Amplify Studio.

Amplify Hosting provides developers the ability to host their full-stack serverless web apps based on Git, and allows for continuous deployment whenever a git push is recorded. A wide variety of frameworks is also supported, including React, Angular, and Vue.js.

Amplify Studio is a visual development environment for building web and mobile apps. Its services include an UI library that offers a variety of pre-built components such as forms, e-commerce cards, buttons etc. It also allows for easy integration with other AWS services such as Lambda, Cognito, … in order to create the backend side of the app.

In this project, AWS Amplify is connected to a Github repository containing the code for the website, with continuous deployment enabled, in order to host the website on AWS. Amplify provides an auto-generated link for the hosted website, which can be used to view the frontend built from React by the Amplify Hosting service.

[1] AWS, AWS Amplify Documentation - "Welcome to AWS Amplify Hosting" - https://docs.aws.amazon.com/amplify/latest/userguide/welcome.html

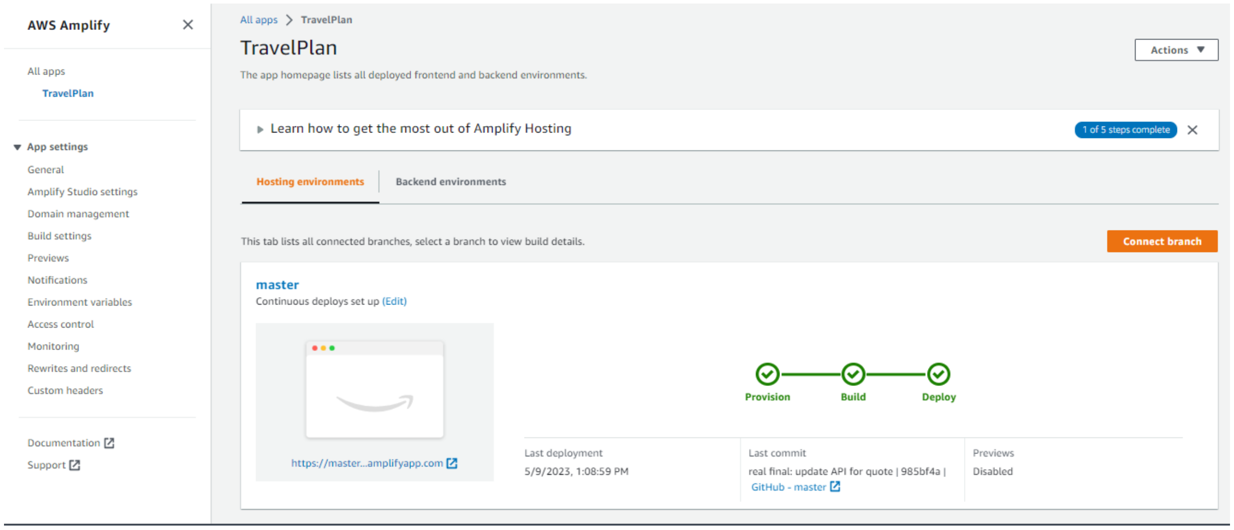


Image 9: *AWS Amplify Hosting console interface*

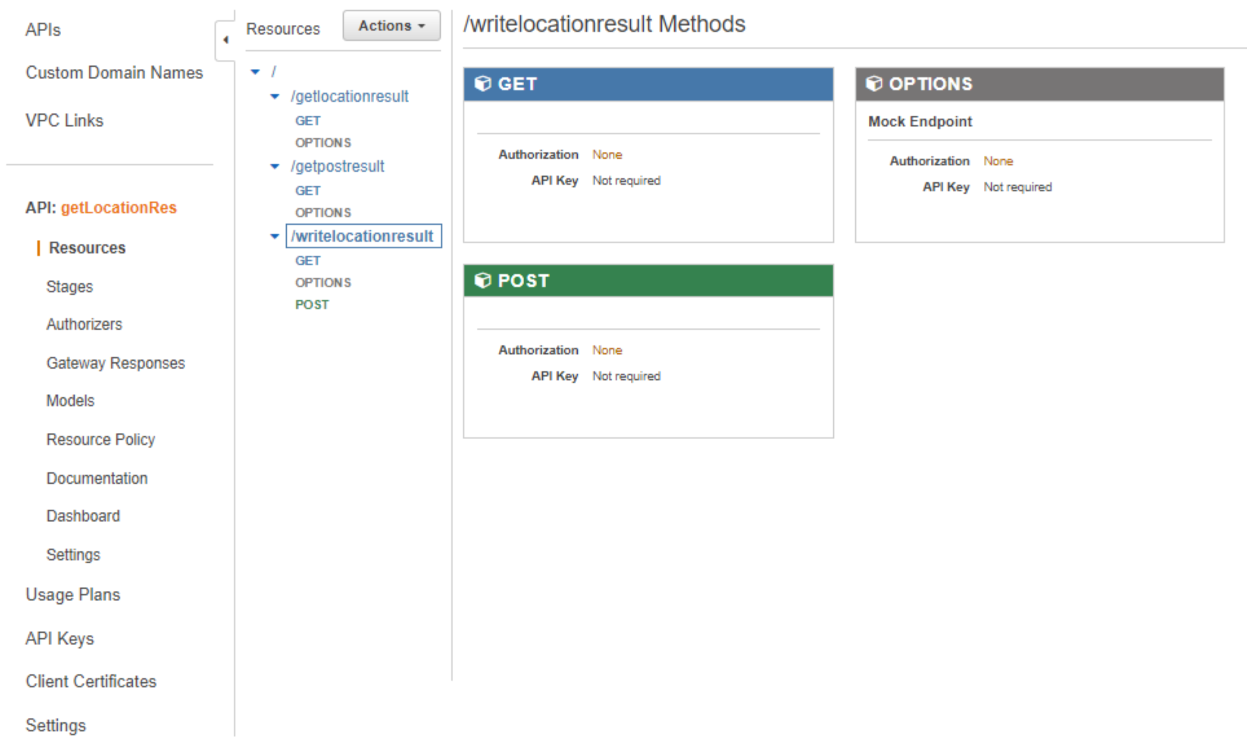
2) AWS API Gateway:

"Amazon API Gateway is an AWS service for creating, publishing, maintaining, monitoring, and securing REST, HTTP, and WebSocket APIs at any scale."[2] API Gateway enables developers to create API more easily with a visual interface for creating resources and methods of the API. Furthermore, API Gateway also offers integration with other AWS services, most importantly, AWS Lambda to handle backend activities, and also services such as AWS DynamoDB, AWS S3, etc.

It also supports various security measures to disallow unauthorized traffic from accessing the API through the use of AWS Identity and Access Management, Cognito user pools and custom authorizers. The service also allows developers to manage and monitor the traffic to the API by giving them the ability to set up throttling and caching rules, and to access a dashboard of API usage, latency, errors.

The website uses two API Gateways for its backend. One is for giving the website the resources they need to render out the frontend. In this case, it is a list of images that can be used in various pages of the website. The other one is for implementing the search and ranking function of the website. This API connects to several Lambda functions to help carry out several requests to the corresponding APIs behind the scene. It is also used for logging and accessing the data of the DynamoDB database.

[2] AWS, AWS API Gateway Documentation - "What is Amazon API Gateway?" - https://docs.aws.amazon.com/apigateway/latest/developerguide/welcome.html



Authorization	None
API Key	Not required

Mock Endpoint

Image 10: *AWS API Gateway interface (/writelocationresult method)*

3) AWS Lambda:

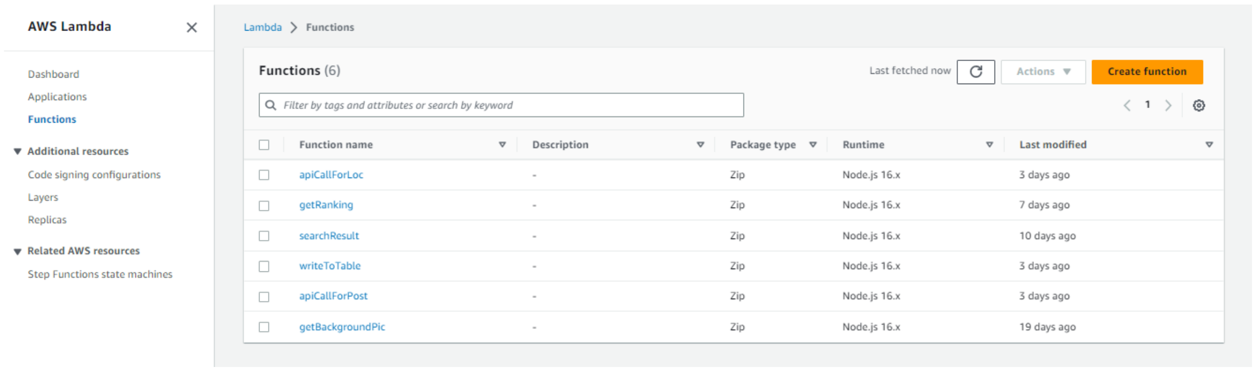
"AWS Lambda is a compute service that lets you run code without provisioning or managing servers."[3] AWS Lambda allows code to be executed in a self-provisioning and cost-effective manner, based on the number of incoming requests or events.

AWS Lambda allows developers to focus on writing code for their application functions without the need to worry about server management and scaling problems. It supports a diverse set of languages that developers can choose from, including Go, Java, Nodejs, Python, Ruby and .NET.

AWS Lambda automatically allocates the suitable amount of resources needed to execute the function, then releases it after the execution is complete. This helps to abstract away the need for server management and manual scaling resource provision. Furthermore, this event-driven execution style of Lambda functions also eliminates the need for continuously running servers and still being able to consistently respond to requests. This combined with the pay-per-use model of AWS Lambda helps to establish itself as a cost-effective method for running backend.

In this project, a total of five Lambda functions are used to implement the functionality of the website. They can be categorized into three main groups: API utilization, Database management, Resource acquisition. For the **API utilization** group, two Lambda functions, "apiCallForLoc" and "apiCallForPost" is used to access the public APIs and get back the data from a query provided by the user. The **Database management** group contains two functions, whose purposes are to write records of the returned search results and get the records from the database. The **Resource acquisition** group contains one function, which is used to get the link to images that are needed to render the website.

[3] AWS, AWS Lambda documentation - "What is AWS Lambda?" - https://docs.aws.amazon.com/lambda/latest/dg/welcome.html



Function name	Description	Package type	Runtime	Last modified
apiCallForLoc	-	Zip	Node.js 16.x	3 days ago
getRanking	-	Zip	Node.js 16.x	7 days ago
searchResult	-	Zip	Node.js 16.x	10 days ago
writeToTable	-	Zip	Node.js 16.x	3 days ago
apiCallForPost	-	Zip	Node.js 16.x	3 days ago
getBackgroundPic	-	Zip	Node.js 16.x	19 days ago

Image 11: *AWS Lambda interface (list of all Lambda functions)*

4) AWS DynamoDB:

"Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability."[4] AWS DynamoDB provides developers with a database without the responsibility of managing a distributed database.

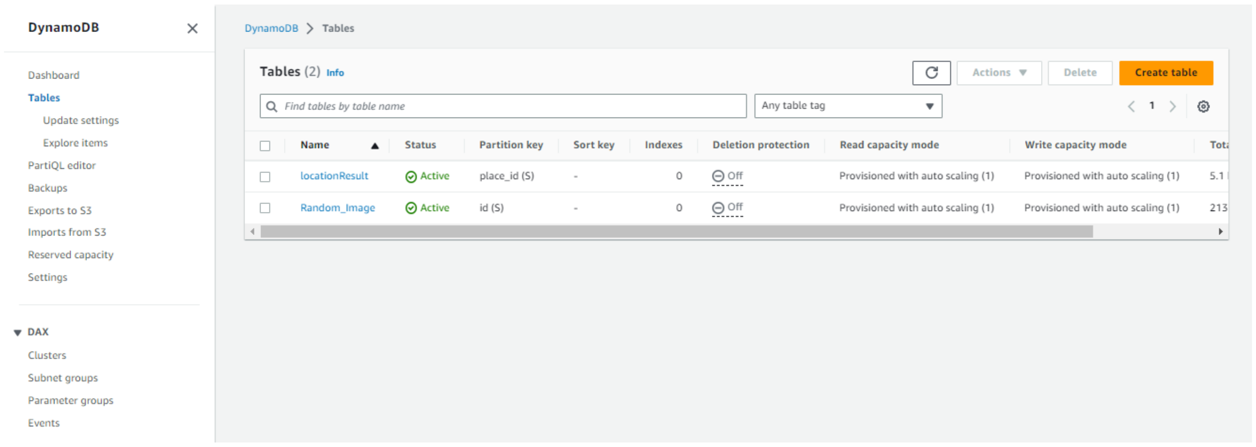
One of the characteristics of DynamoDB is a lack of requirement for a predefined schema, meaning that the database does not need to adhere to a previously established set of rules on fields in the table. This allows for flexibility in data modeling and seamless adaptation to evolving application design.

At its core, DynamoDB is a distributed database, which allows for automatic scalability and data protection. As data volume grows, DynamoDB can automatically segment data across multiple servers in order to be able to handle requests efficiently and effectively. Furthermore, DynamoDB can ensure the reliability of data at all times by replicating it among many availability zones. This helps to prevent serious impact in case of failures, making it perfect for large businesses.

Another benefit of DynamoDB is that it supports PartiQL. PartiQL is a query language that combines the familiar syntax of SQL databases with the flexibility and scalability of NoSQL databases. PartiSQL helps to lower the learning curve of the DynamoDB service.

The website uses two main tables in DynamoDB. "locationResult" is used to store records on the results that were returned from user's queries, which can later be used to create rankings for the locations. "Random_Image" is used to provide the website with links to images that are needed to be rendered onto the web page. PartiQL is also used in some functions in order to add or get records from the database.

[4] AWS, AWS DynamoDB documentation - What is Amazon DynamoDB? - https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html



Name	Status	Partition key	Sort key	Indexes	Deletion protection	Read capacity mode	Write capacity mode	Tot
locationResult	Active	place_id (S)	-	0	Off	Provisioned with auto scaling (1)	Provisioned with auto scaling (1)	5.1
Random_Image	Active	id (S)	-	0	Off	Provisioned with auto scaling (1)	Provisioned with auto scaling (1)	213

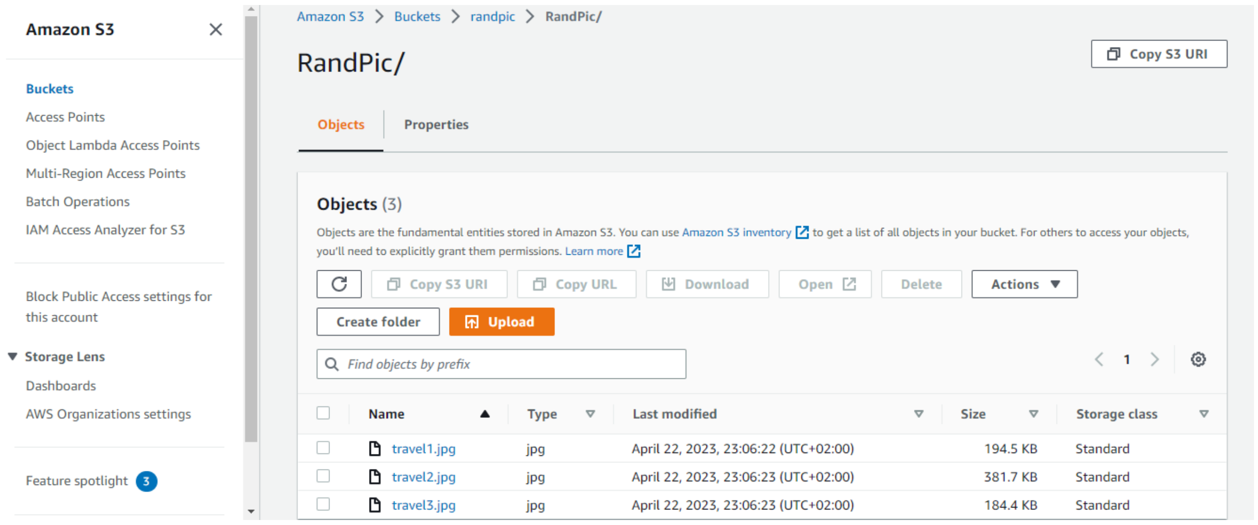
Image 12: *AWS DynamoDB interface (list of all tables)*

5) AWS Simple Storage Service(S3):

"Amazon Simple Storage Service (Amazon S3) is an object storage service that offers industry-leading scalability, data availability, security, and performance."[5] It provides organizations the ability to store and retrieve any amount and any types of data, ranging from a few bytes of documents to terabytes of images, videos and more. The basic logical container of S3 is buckets. They acted as a top-level folder or directory, which can be used to store data in. Each bucket in the S3 system has a globally unique name, which can be used for identification.

Data stored inside S3 buckets are replicated across many availability zones, which ensures high availability and resistance against data loss. It is "designed to provide 99.999999999% durability and 99.99% availability of objects over a given year" [6]. S3 also offers many security features to protect data from unauthorized access such as integration with AWS IAM to regulate access and also encryption using AWS Key Management Service (KMS) or user-provided keys.

In the project, a S3 bucket is created to store images which then can be randomly chosen to render into the banner of the website and many other parts.



Objects (3)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
travel1.jpg	jpg	April 22, 2023, 23:06:22 (UTC+02:00)	194.5 KB	Standard
travel2.jpg	jpg	April 22, 2023, 23:06:23 (UTC+02:00)	381.7 KB	Standard
travel3.jpg	jpg	April 22, 2023, 23:06:23 (UTC+02:00)	184.4 KB	Standard

Image 13: *AWS S3 interface (list of items inside a bucket)*

[5] AWS, AWS S3 documentation - What is Amazon S3? - https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html

[6] AWS, AWS S3 documentation - Data protection in Amazon S3 - https://docs.aws.amazon.com/AmazonS3/latest/userguide/DataDurability.html

6) AWS Virtual Private Cloud(VPC):

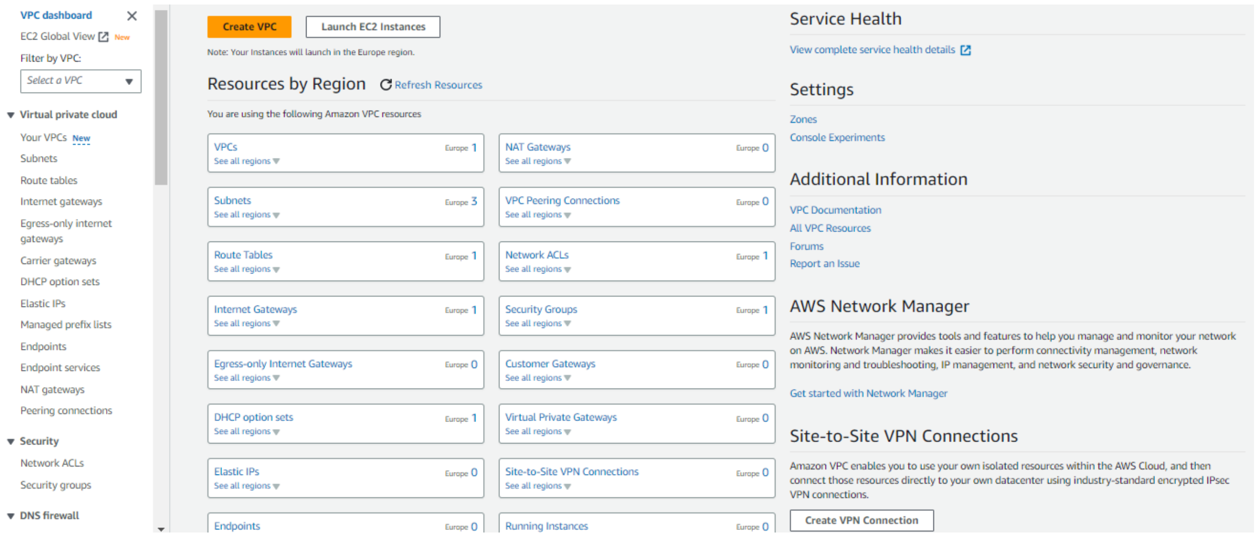
"With Amazon Virtual Private Cloud (Amazon VPC), you can launch AWS resources in a logically isolated virtual network that you've defined. This virtual network closely resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS."[7] VPC provides networking services including: creating subnets, routing tables, network gateways, allocating IP addresses etc, which allows developers to create and customize the network environment according to their needs.

Security is also one of the most important features of VPC. This is implemented through the usage of security groups and network access control list(ACL). Security groups act as virtual firewalls, controlling inbound and outbound traffic using predefined rules. Network ACL helps filter traffic at the subnet level.

In this project, AWS VPC is used to establish a unique, static IP address for the Lambda functions in order to access the APIs, which only allow certain white-listed user-defined IP addresses. Normally, when an API request is made with a Lambda function, AWS will provide the function with one temporary IP address from its pool. Therefore, the address of the Lambda function will change with each execution.

When assigned an Elastic IP address, the request from a Lambda function residing in the private subnet will be routed to NAT gateway in the public subnet, assigning the Elastic IP address. Then, it will be forwarded to the Internet gateway and sent to the specified API. After a response is sent back to the Internet gateway, it is delivered to the NAT gateway and dispatched to the original Lambda function.

[7] AWS, AWS VPC documentation - What is Amazon VPC? - https://docs.aws.amazon.com/vpc/latest/userguide/what-is-amazon-vpc.html



Note: Your Instances will launch in the Europe region.

Resources by Region

You are using the following Amazon VPC resources

VPCs	Europe	1
NAT Gateways	Europe	0
Subnets	Europe	3
VPC Peering Connections	Europe	0
Route Tables	Europe	1
Network ACLs	Europe	1
Internet Gateways	Europe	1
Security Groups	Europe	1
Egress-only Internet Gateways	Europe	0
Customer Gateways	Europe	0
DHCP option sets	Europe	1
Virtual Private Gateways	Europe	0
Elastic IPs	Europe	0
Site-to-Site VPN Connections	Europe	0
Endpoints	Europe	0
Running Instances	Europe	0

See all regions

Service Health

View complete service health details

Settings

Zones
Console Experiments

Additional Information

VPC Documentation
All VPC Resources
Forums
Report an Issue

AWS Network Manager

AWS Network Manager provides tools and features to help you manage and monitor your network on AWS. Network Manager makes it easier to perform connectivity management, network monitoring and troubleshooting, IP management, and network security and governance.

Get started with Network Manager

Site-to-Site VPN Connections

Amazon VPC enables you to use your own isolated resources within the AWS Cloud, and then connect those resources directly to your own datacenter using industry-standard encrypted IPsec VPN connections.

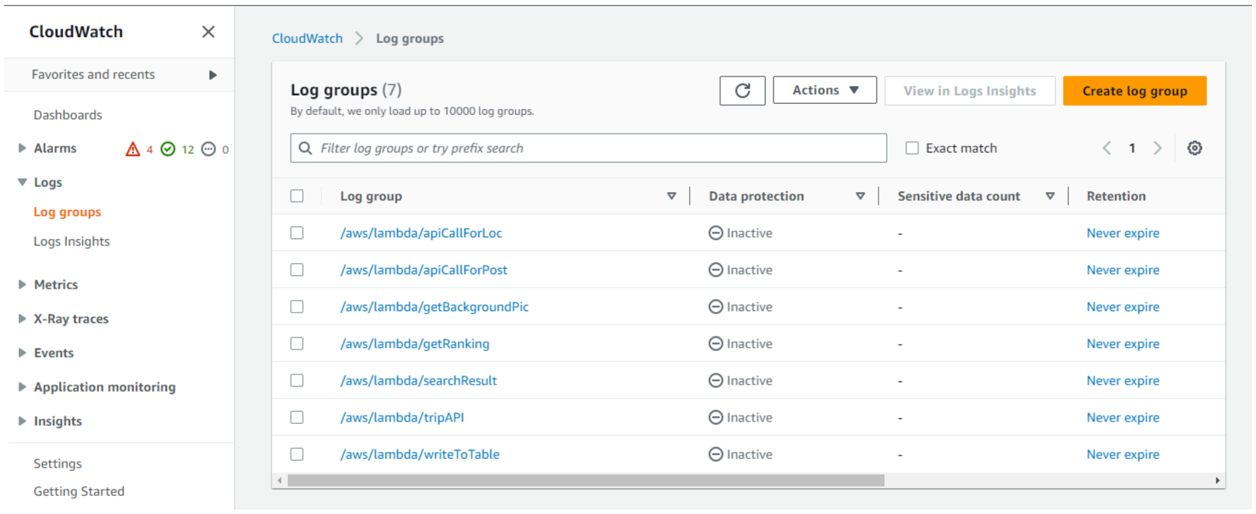
Image 14: *AWS VPC interface (dashboard)*

7) AWS Cloudwatch:

"Amazon CloudWatch monitors your Amazon Web Services (AWS) resources and the applications you run on AWS in real time. You can use CloudWatch to collect and track metrics, which are variables you can measure for your resources and applications."[8] Using AWS CloudWatch, developers can gain access to customizable dashboards and visualization, which grant insights into the operation performance, health and efficiency of the AWS services and infrastructure.

AWS CloudWatch offers the monitoring of logs, which are organized into log groups based on the service that generates it. Using these logs, users can see the system behavior and can be used to debug the application in case of errors.

AWS CloudWatch was used in this project in order to gain insights into the behavior of AWS Lambda functions, which helps highlight undefined behaviors of the functions in the case of errors.



Log groups (7)
By default, we only load up to 10000 log groups.

Log group	Data protection	Sensitive data count	Retention
/aws/lambda/apiCallForLoc	Inactive	-	Never expire
/aws/lambda/apiCallForPost	Inactive	-	Never expire
/aws/lambda/getBackgroundPic	Inactive	-	Never expire
/aws/lambda/getRanking	Inactive	-	Never expire
/aws/lambda/searchResult	Inactive	-	Never expire
/aws/lambda/tripAPI	Inactive	-	Never expire
/aws/lambda/writeToTable	Inactive	-	Never expire

Image 15: *AWS CloudWatch interface (list of all log groups)*

[8] AWS, AWS Cloudwatch documentation - What is Amazon CloudWatch? - https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/WhatIsCloudWatch.html

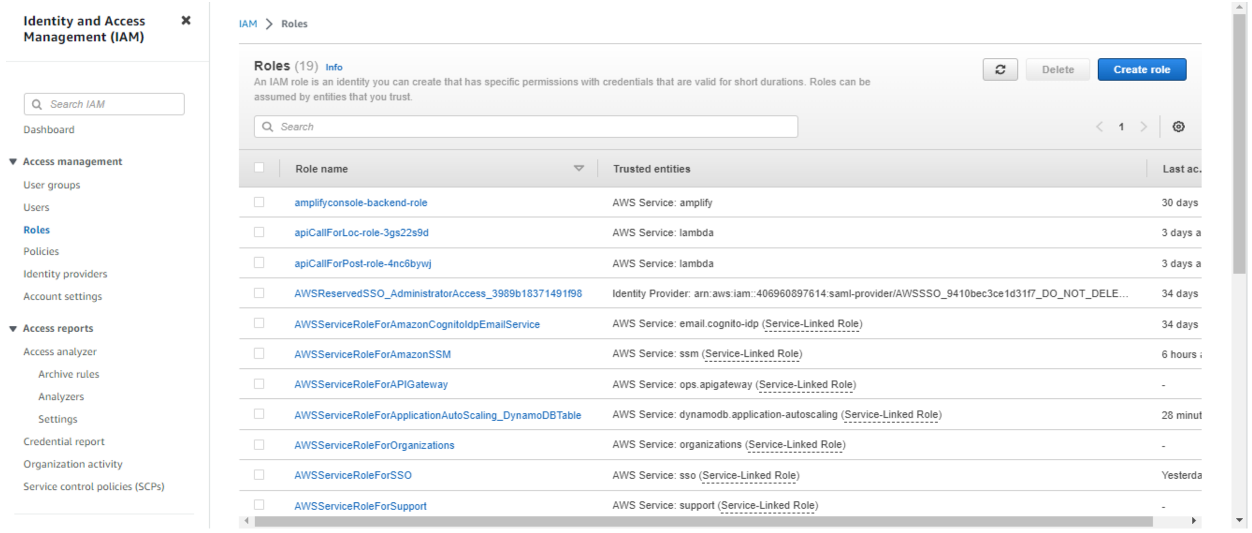
8) AWS Identity & Access Management(IAM):

"AWS Identity and Access Management (IAM) is a web service that helps you securely control access to AWS resources. With IAM, you can centrally manage permissions that control which AWS resources users can access. You use IAM to control who is authenticated (signed in) and authorized (has permissions) to use resources."[9] IAM provides an interface for creating and managing users, roles, and groups. Users can be individuals, such as employees or administrators of the system. Meanwhile, groups are a collection of users which are granted the same access to services. Roles are assigned to users and administrators can give certain access to users who have a specific combination of roles.

IAM supports least privileges policies, promoting best security practices of only providing users with the bare minimum permissions to perform their tasks. It also allows conditional policies, which enables users to gain access to certain services when certain conditions are met such as time, IP address and multi-factor authentication.

In this project, IAM is used to enable resource to resource access and also website to resource access, including:

- Lambda functions to DynamoDB databases.
- Website to S3 buckets
- Lambda functions to be executed in subnets provided by VPC.



Roles (19) Info

An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.

Role name	Trusted entities	Last ac.
amplifyconsole-backend-role	AWS Service: amplify	30 days
apiCallForLoc-role-3gs22s9d	AWS Service: lambda	3 days a
apiCallForPost-role-4nc6bywj	AWS Service: lambda	3 days a
AWSReservedSSO_AdministratorAccess_3989b18371491f98	Identity Provider: arn:aws:iam::406960897614:saml-provider/AWSSSO_9410bec3ce1d31f7_DO_NOT_DELE...	34 days
AWSServiceRoleForAmazonCognitoIdpEmailService	AWS Service: email.cognito-idp (Service-Linked Role)	34 days
AWSServiceRoleForAmazonSSM	AWS Service: ssm (Service-Linked Role)	6 hours
AWSServiceRoleForAPIGateway	AWS Service: ops.apigateway (Service-Linked Role)	-
AWSServiceRoleForApplicationAutoScaling_DynamoDBTable	AWS Service: dynamodb.application-autoscaling (Service-Linked Role)	28 minut
AWSServiceRoleForOrganizations	AWS Service: organizations (Service-Linked Role)	-
AWSServiceRoleForSSO	AWS Service: sso (Service-Linked Role)	Yesterda
AWSServiceRoleForSupport	AWS Service: support (Service-Linked Role)	-

Image 16: *AWS IAM interface (list of all roles)*

[9] AWS, AWS IAM documentation - What is IAM? - https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html

Implementation

1) Page structure:

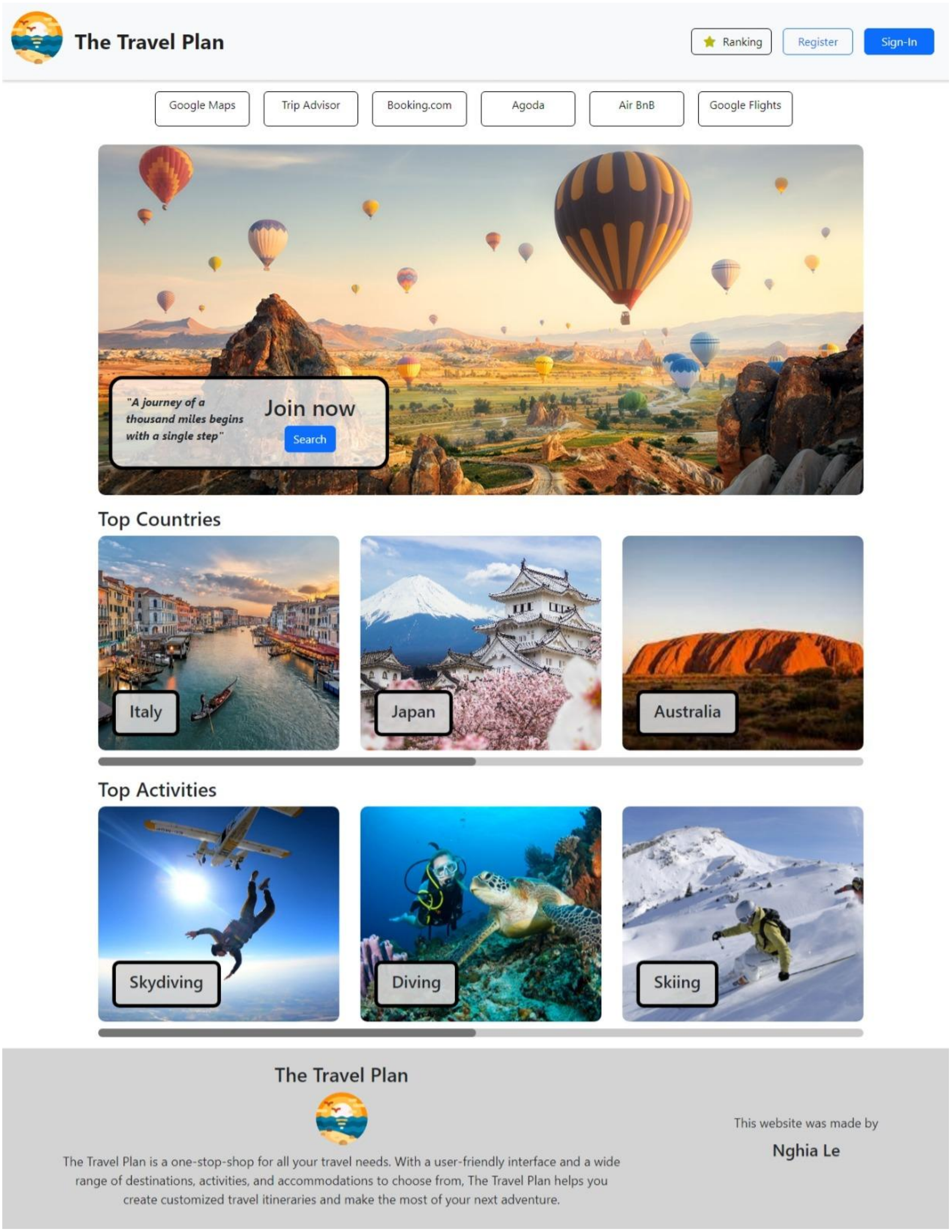
The website structure contains three main pages: a landing page which is the index page for the website, a search page and a ranking page.

a) Landing page:

Path: index

Description: The landing page is the first point of contact from users to the website. The page features many parts:

- Navigation bar
- List of utility websites
- Banner with a link to the search page
- Lists of countries and activities
- Footer



The Travel Plan is a one-stop-shop for all your travel needs. With a user-friendly interface and a wide range of destinations, activities, and accommodations to choose from, The Travel Plan helps you create customized travel itineraries and make the most of your next adventure.

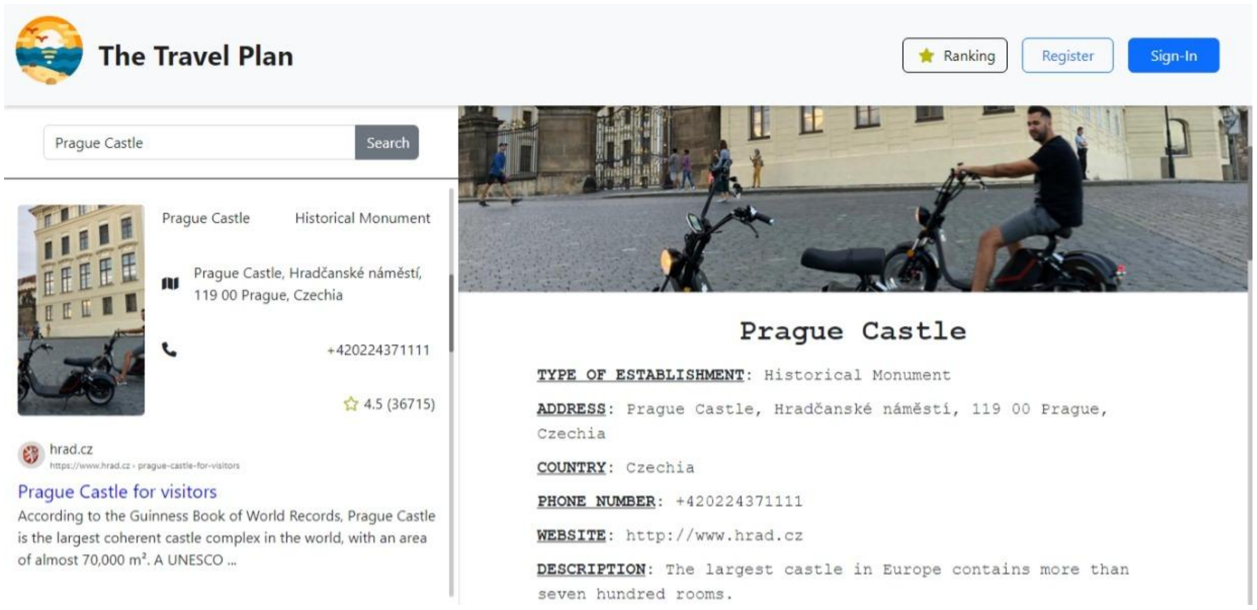
This website was made by

Nghia Le

Image 17: *Landing page*

b) Search page:
Path: /search
Description: The search page offers to the user the "Location search", "View detail", and "Article search" functionality. The page has several parts, which are:

- Navigation bar
- Search bar for entering and submitting query
- A result page which contains location cards and article cards containing search result
- The detail area, in which the selected result will be displayed with greater details.



Prague Castle, Hradčanské náměstí, 119 00 Prague, Czechia

+420224371111

Prague Castle for visitors
According to the Guinness Book of World Records, Prague Castle is the largest coherent castle complex in the world, with an area of almost 70,000 m². A UNESCO ...

Prague Castle

TYPE OF ESTABLISHMENT: Historical Monument

ADDRESS: Prague Castle, Hradčanské náměstí, 119 00 Prague, Czechia

COUNTRY: Czechia

PHONE NUMBER: +420224371111

WEBSITE: http://www.hrad.cz

DESCRIPTION: The largest castle in Europe contains more than seven hundred rooms.

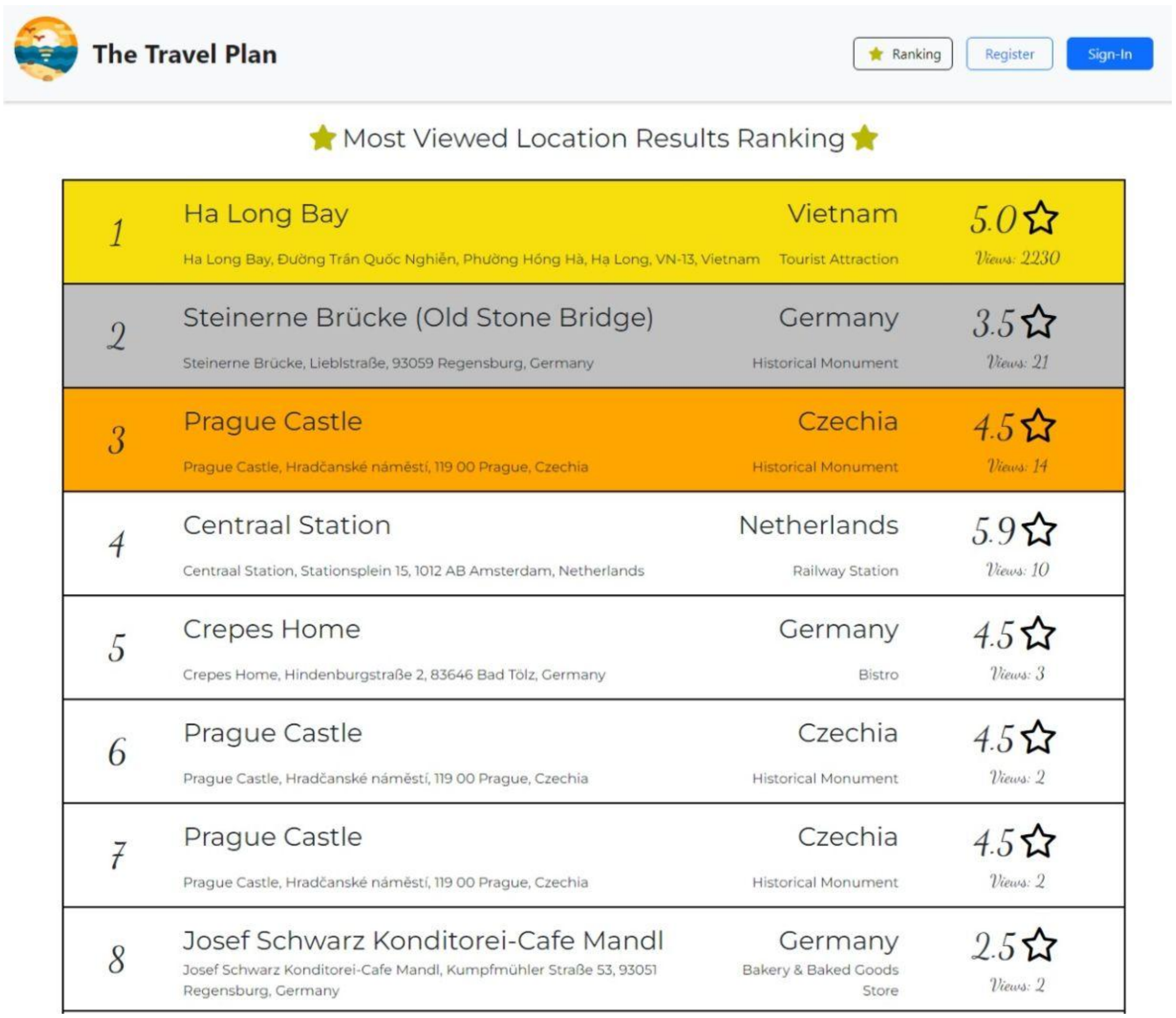
Image 18: *Search page (Prague Castle details)*

c) <u>Ranking page:</u>

Path: /board

Description: The ranking page allows the user to see the ranking on how many times certain locations show up in the search result from the user's query, which is the "View result ranking" function. The components on the webpage are:

- Navigation bar
- Ranking board



Most Viewed Location Results Ranking

1	Ha Long Bay Ha Long Bay, Đường Trần Quốc Nghiễn, Phường Hồng Hà, Hạ Long, VN-13, Vietnam	Vietnam Tourist Attraction	5.0 ☆ Views: 2230
2	Steinerne Brücke (Old Stone Bridge) Steinerne Brücke, Lieblstraße, 93059 Regensburg, Germany	Germany Historical Monument	3.5 ☆ Views: 21
3	Prague Castle Prague Castle, Hradčanské náměstí, 119 00 Prague, Czechia	Czechia Historical Monument	4.5 ☆ Views: 14
4	Centraal Station Centraal Station, Stationsplein 15, 1012 AB Amsterdam, Netherlands	Netherlands Railway Station	5.9 ☆ Views: 10
5	Crepes Home Crepes Home, Hindenburgstraße 2, 83646 Bad Tölz, Germany	Germany Bistro	4.5 ☆ Views: 3
6	Prague Castle Prague Castle, Hradčanské náměstí, 119 00 Prague, Czechia	Czechia Historical Monument	4.5 ☆ Views: 2
7	Prague Castle Prague Castle, Hradčanské náměstí, 119 00 Prague, Czechia	Czechia Historical Monument	4.5 ☆ Views: 2
8	Josef Schwarz Konditorei-Cafe Mandl Josef Schwarz Konditorei-Cafe Mandl, Kumpfmühler Straße 53, 93051 Regensburg, Germany	Germany Bakery & Baked Goods Store	2.5 ☆ Views: 2

Image 19: *Ranking page (top 8 results)*

2) APIs usage:

APIs play an important role in the functionality of the website as they are the main source of information that the website needs to return the data according to the user's query. There are three main APIs that are in used in this project, which can be split into two categories based on the purpose of their usage:

- Location API: HereAPI and TripAdvisor API.
- Article API: SerpAPI.

a) HereAPI:

HERE is a prominent company specialized in the field of mapping, navigation and other location-based services. The main focus of the company is to provide individuals and businesses with accurate and latest location data. Its offers include: Maps & Data, Location Services, Platform Tools, and Applications developed by the company.

In this project, the website uses the Location Services offer of the HERE company, or more specifically the HERE Geocoding & Search API. The website uses the *discover* use case of the API to search for suitable locations based on the query provided by the user and their current location.

The service provides the website with a list of locations with their basic information such as their names, their addresses, their contact information etc. However, HERE Geocoding & Search API also provides the location ID from other travel-theme websites(if available), which will be used later to get more information on the location from the TripAdvisor API.

b) TripAdvisor API:

TripAdvisor is a well-known online platform focused on providing travelers with information, reviews and booking services to various travel-related fields, such as hotels, restaurants, tours, flights and more. It is among one of the biggest travel websites globally, with a vast amount of user-generated data and resources.

TripAdvisor provides such data to individuals and organizations through their API, which provides access to various services, including:

+ Location Details
+ Location Photos
+ Location Reviews
+ Location Search
+ Nearby Location Search

In this project, three out of the five provided services are used simultaneously to get the additional information, photos and reviews about the location returned by the HereAPI through its reference ID to the TripAdvisor website.

c) SerpAPI:

SerpAPI focuses on the extraction of search engine results in a structured format, which allows developers to get the information and integrate them into their application easily. SerpAPI gathers the result on many search engines such as: Google, Bing, Yahoo etc from a search query and returns the up-to-date, location-based data containing many types of results, including organic results, recipes, locations, images, questions and many more.

SerpAPI was used in this project as a way to gather articles from the organic search results that the API returns from a search query provided by the user and forwarded to SerpAPI.

3) Code structure:

The code uses *React* for the front-end side and uses the *Vite* to build the website and the *react-router* package to handle the routing inside the website. The entry point of the website is the *index.html* and *index.jsx* files at the top-level folder of the repository. These 2 files host and enable the routing of the website.

The majority of the rest of the code used to build the front-end of the website is placed in the *src* folder:

- <u>*API* folder:</u> This folder contains the necessary JavaScript code to make AJAX calls to the API Gateway provided by AWS to run the Lambda functions to both search location results and write such results to the DynamoDB database.
- <u>*components* folder:</u> The folder contains the many small components that make up the website, containing one or more JSX files and their corresponding, personal CSS files. The list of the components are the following:
  + Footer: The Footer at the end of the Landing page.
  + Leaderboard: The ranking board that is used inside the Ranking page.
  + Menu: The list of buttons to other websites that is displayed inside the Landing page.
  + NavBar: The navigation bar that is rendered in every page.
  + Picture: The banner inside the Landing page.
  + Quote: The container for decorative areas that use quotes from an API, is used inside the Register and Signin page.
  + Ranking: The container for the lists of suggested countries and activities in the Landing page.
  + Register: The register form
  + Result: The detailed view area of the chosen location result from a location search.
  + Searcher: The areas in which a user can enter a query and receive the result from such query.
  + Signin: The sign in form
- <u>*css* folder:</u> The folder contains the file *global.css,* which is used mainly to define the global variables to be used in the css file of the components and to set specific styles that should be in every page.

- *data* folder: The folder contains data for rendering the website and also to test the website before making real API calls to the back-end.
- *start* folder: The folder contains the JSX files, containing components that should be rendered out when switching between routes inside the website.

Meanwhile, the back-end of the website is handled by the AWS Lambda functions and communicates/transfers data between the two ends by using AJAX calls.

The full codebase will be submitted together with this report.

Result

In this project, I have successfully created a travel-orientated website called the "*The Travel Plan*" and implemented some of its necessary functionalities. The features in the website and their results are:

- "Location search": The website was able to take the query from the user and return to them the list of location results, while taking into consideration only their location, disregarding other factors such as search history, preferences,...
- "View detail": The website can display to the user further information about a chosen location such as: a brief overview, pictures taken there, and its reviews and scores. However, only locations that have a reference ID to the TripAdvisor website can have its data crawled from the website. Furthermore, only five images and five reviews of the place can be crawled from the TripAdvisor API.
- "Article Search": The website can also use the query provided by the user to search for articles from the Google search engine and display them in the form of cards in the search result area.
- "View result ranking": Records of the previous search results can also be queried from the database to be displayed on the ranking page. However, a full scan of the table is required in order to get all the data, which is inefficient considering the distributed and scalable nature of the DynamoDB database.

Furthermore, the process of development and deployment of the website was also able to utilize many AWS services and the features that they have to offer, including:

- Hosting the website based on code in the GitHub repository using AWS Amplify Hosting
- Creating the back-end of the website with AWS Lambda functions.
- Communication between the two ends by making API requests through AWS API Gateway.
- Assigning an static IP address to outbound traffic from Lambda functions with AWS VPC.
- Recording and querying data using PartiQL to AWS DynamoDB databases.
- Object storage with AWS S3.
- Managing security roles with AWS IAM.
- Viewing logs for debugging purposes with AWS CloudWatch.

Conclusion

In conclusion, this travel guide website is somewhat successfully implemented, leveraging a diverse set of AWS services and the React framework. The project was able to showcase the versatility of the tools that AWS can offer to organizations and their effectiveness.

Some of the notable accomplishments of this project are:

- All of the requirements that are set out by the project description are successfully implemented.
- A diverse set of AWS services was used during the development process of the website and also in the implementation of its functionalities.
- The website not only uses HTML and CSS but also uses the React framework, which helps to showcase the support of AWS Amplify Hosting for similar frameworks.
- The usages of APIs in the website was used reasonably and was organized to run concurrently when possible to help optimize the user's experience.

However, there are still some areas that could use some room for improvements for this project:

- <u>Database querying</u>: Some of the querying of the DynamoDB database is a scan for all the items in the database, which is considered to be quite efficient when taking into account the distributed nature of the database based on the hash key.
- <u>Location search engine</u>: The search engine for the "Location search" functions only takes in the query and the user's current location as parameters for the search, which can make the results not suitable for the user's needs. Other factors that should be taken into consideration are user's search history, location's popularity, user's preference etc
- <u>Advanced features</u>: In this project, I have only been able to utilize some of the basic features that the AWS services have to offer. For example, with AWS Amplify, only the AWS Amplify Hosting feature was used in the project.
- <u>Website feature expansion</u>: Even though all of the requirements of the project were met, there is still room for further exploration of other functionalities that logically should be available in the website, such as user management, autosuggestion, and better ranking algorithms.

Future plans

Even though the project has met all of the requirements set out by the project description, there are still some possible functionalities that should be implemented to help further improve the user's experience:

- User management: An user system should be implemented using AWS Cognito to provide a secure registration and login process. Further personalization of the user, including custom avatars and more can be enabled using AWS S3 buckets.
- Admins: An administrative role should be created for the website, along with dashboards and customizations that the admins can perform such as changing the list of suggested countries, activities, or adding more images to the banner.
- Autosuggestion: While a user is entering a query, location results should be auto-suggested based on the user's previous searches to help create a better user experience.
- User's personal log: With the creation of user management, users should be able to see their previous searches. Furthermore, such data could be used for other purposes which can help enhance the quality of the website.
- User-defined location lists: Users should be able to create lists of locations which can serve a variety of functions: saving favorite locations, trip-planning etc.
- More personalized search results: With the availability of users' personalized log, the data should be analyzed and fed into machine learning algorithms in order to give users better search results.
- Map feature: A map should be added to the website to help visualize displayed search results and help the user with navigation and trip-planning.

References

AWS documentation

[1] AWS, ***AWS Amplify documentation***

[2] AWS, ***AWS API Gateway documentation***

[3] AWS, ***AWS Lambda documentation***

[4] AWS, ***AWS DynamoDB documentation***

[5] AWS, ***AWS S3 documentation***

[6] AWS, ***AWS VPC documentation***

[7] AWS, ***AWS IAM documentation***

[8] AWS, ***AWS CloudWatch documentation***

APIs documentation

[9] HERE, ***HERE documentation,*** https://developer.here.com/documentation

[10] TripAdvisor, ***TripAdvisor API documentation,*** https://tripadvisor-content-api.readme.io/reference/localization

[11] SerpAPI, ***SerpAPI documentation,*** https://serpapi.com/search-api

Others

[12] AWS, ***PartiQL documentation,*** https://partiql.org/dql/overview.html

[13] AWS, ***Generate a static outbound IP address using a Lambda function, Amazon VPC, and a serverless architecture,*** https://docs.aws.amazon.com/prescriptive-guidance/latest/patterns/generate-a-static-outbound-ip-address-using-a-lambda-function-amazon-vpc-and-a-serverless-architecture.html

[14] AWS, ***Build a Serverless Web Application,*** https://aws.amazon.com/getting-started/hands-on/build-serverless-web-app-lambda-apigateway-s3-dynamodb-cognito/