



DALHOUSIE UNIVERSITY

CSCI 5409

Advance Topics in Cloud Computing

Jaskaran Singh

MACS

B00948857

js356337@dal.ca

Term Project

Deployment URLs:

<http://testenvironment.eba-9gbjccrm.us-east-1.elasticbeanstalk.com/>

Project Description:

SnapSearch is a photo management web application with its user-friendly interface, streamlining the uploading and organization of cherished memories. Harnessing advanced image recognition technology using Amazon Rekognition, the app enables seamless searching through vast photo collections, ensuring instant accessibility to every precious moment.

This application allows users to login and register in the application and then upload photos in the cloud storage. It also allows users to search photos from the cloud storage based on search criteria.

Below are some of the screenshots of the application:

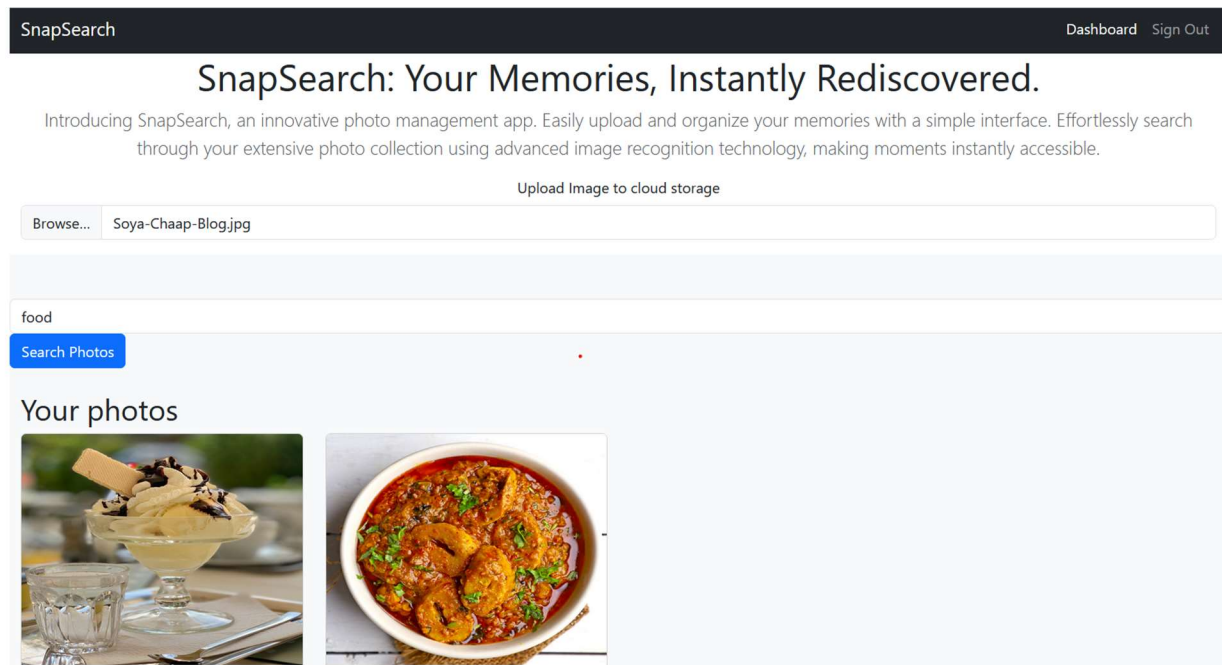
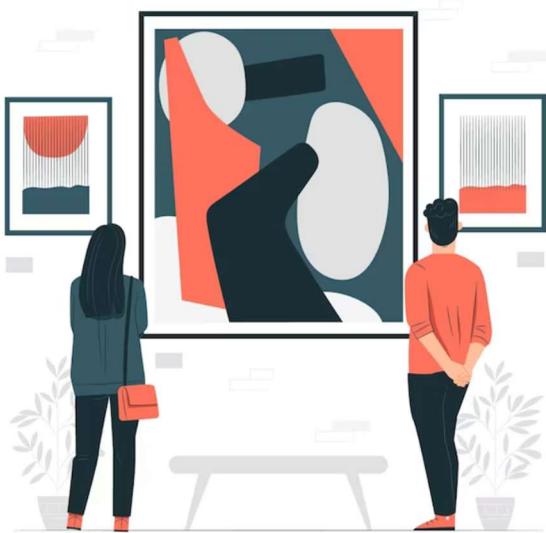


Fig. 1: Dashboard page to upload, search and view photos on SnapSearch App

SnapSearch

Sign inRegister



Login

Email address

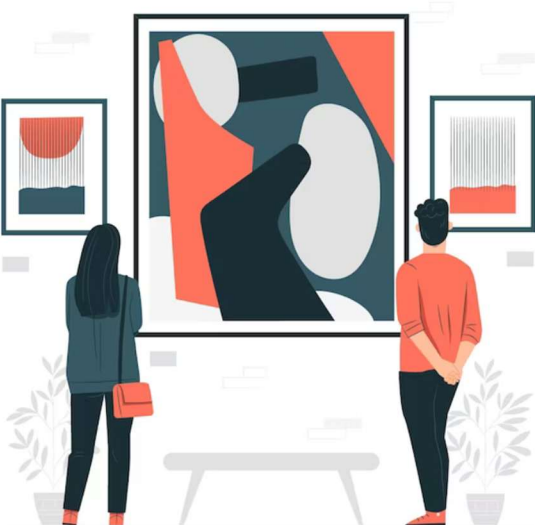
Password

[Login](#) or [Click to Register](#)

Fig. 2: Login page on SnapSearch App

SnapSearch

Sign inRegister



Register

Name

Email address

Password

[Register](#)

Fig. 3: Registration page on SnapSearch App

Requirements:

Below are the services that I have used to build SnapSearch web application:

Compute (2):

1. AWS Lambda:

AWS Lambda is a serverless computing service that lets you run code without provisioning or managing servers. It automatically scales and executes code in response to events, making it ideal for building scalable and cost-effective applications.

Lambda seamlessly integrates with various AWS services, making it easy to create comprehensive serverless architectures. This includes integrations with Amazon S3, DynamoDB, Amazon API Gateway, and more. Developers can write functions in languages like Node.js, Python, Java, C#, and more. They can also use pre-built templates or create custom functions, making it accessible to both beginners and experienced developers.

2. AWS Elastic Beanstalk:

I opted for AWS Lambda to execute my code due to its straightforward and effective approach to function execution without the hassle of server management. With Lambda, I can concentrate solely on coding without the burden of server setup and upkeep. Its automatic scaling, adapting to incoming requests, ensures it handles traffic seamlessly without manual resource adjustments. The pay-as-you-go model means I only pay for actual compute usage, making it economical for my application. Lambda's versatility in triggering functions with various events simplifies the creation of event-driven applications, especially with its smooth integration with AWS services like DynamoDB and S3. Overall, AWS Lambda streamlines code execution in a scalable and cost-effective manner, making it an excellent choice for constructing serverless applications—especially when paired with the AWS API Gateway to meet project requirements.

Storage (2):

1. Amazon S3:

For my project, I've chosen to store project-related images in an S3 bucket. The information is stored in a jpg/jpeg file, eliminating the need for a traditional database. While there's an option to store images in DynamoDB using base64 format, it's not considered a suitable method for image storage due to its inherent limitations.

2. Amazon DynamoDB:

I've opted for Amazon DynamoDB as the storage solution for my project details within AWS. Its user-friendly interface and automated management eliminate concerns about server setup and backups, handled seamlessly by AWS. DynamoDB's automatic scalability suits my application's evolving needs, ensuring efficient data handling. Its rapid performance guarantees quick user access, while the flexible design facilitates easy adjustments as my app grows. Seamless integration with other AWS services enhances reliability. DynamoDB's commitment to data safety and availability further solidifies its position as the ideal choice, offering a perfect blend of simplicity, scalability, speed, and reliability for my application.

Network(1):

1. AWS API Gateway:

I've chosen AWS API Gateway as the go-to service for constructing APIs for my project due to its streamlined and hassle-free API creation and management. With API Gateway, the complexities of infrastructure setup and server management are taken care of by AWS. It simplifies the creation of APIs that interact with my lambda functions, providing a straightforward means for my web app to access application APIs. The service seamlessly integrates with AWS Lambda, facilitating the effortless development of serverless APIs. Beyond that, API Gateway enhances API performance and security through features like caching, rate limiting, and request/response transformations. Its built-in monitoring and logging capabilities offer valuable insights into API usage, aiding in the identification and resolution of potential issues. In essence, AWS API Gateway is an optimal choice, streamlining the API creation and management process and contributing to the scalability and reliability of my application.

General(2):

1. AWS CloudFormation:

AWS CloudFormation provides a common language for describing and provisioning all the infrastructure resources in a cloud environment. It allows developers to use a template to create and provision AWS resources, ensuring consistency and repeatability.

2. AWS Rekognition:

AWS Rekognition is a deep learning-based image and video analysis service. It enables developers to add powerful visual analysis to applications, detecting objects, scenes, and faces in images, as well as analyzing and indexing videos.

I have used AWS Rekognition to generate labels related to images and then storing those labels in the DynamoDB table.

Deployment:

I've opted for the public cloud as the hosting solution for my SnapSearch application due to its myriad advantages that align with my requirements. By leveraging the public cloud, my application becomes globally accessible, enabling people worldwide to effortlessly upload their images. The automatic scalability feature ensures uninterrupted website availability, even during peak visitor times, eliminating concerns about crashes or slowdowns. Security is a paramount benefit, with the cloud provider handling data protection through robust measures like encryption and firewalls. Additionally, the cloud provider takes care of server-related tasks, freeing me from responsibilities such as updates and maintenance. This leaves me with more time to enhance my application functionalities and provide user-friendly cloud storage for photos for clients.

In summary, the public cloud simplifies my web application, ensures website stability, and guarantees the safety and security of user's data.

Delivery Model:

For the frontend, I've adopted a Platform as a Service (**PaaS**) delivery model utilizing AWS Elastic Beanstalk. Think of it as my website's efficient assistant, handling server management and technical complexities effortlessly.

Moving to the backend, I've embraced a Function as a Service (**FaaS**) model, employing Lambda functions. With Lambda, the worry of server setup and maintenance is lifted, allowing me to dedicate my focus solely to code creation. Its automatic scalability ensures seamless handling of any incoming traffic.

Shifting to the database, I've opted for **PaaS** approach. Leveraging S3 for file storage and DynamoDB for project data, I'm spared the intricacies of managing the underlying infrastructure. Interactions are facilitated through the AWS SDK for NodeJS, simplifying the entire process.

Architecture Overview:

The project's architecture, illustrated in **Figure 4**, delineates the cohesive interplay among the chosen services. Serving as the project's entry point is an Elastic Beanstalk instance, hosting the React app. The React app, in turn, initiates API calls directed to the API Gateway. The API Gateway orchestrates the invocation of specific lambda functions based on incoming requests. These lambda functions, adept in diverse operations, interact with DynamoDB tables, an S3 bucket, and AWS Rekognition.

AWS Rekognition is a deep learning-based image and video analysis service. It is invoked by lambda to create labels related to images that are present in the S3 bucket and then these labels are stored in DynamoDB table.

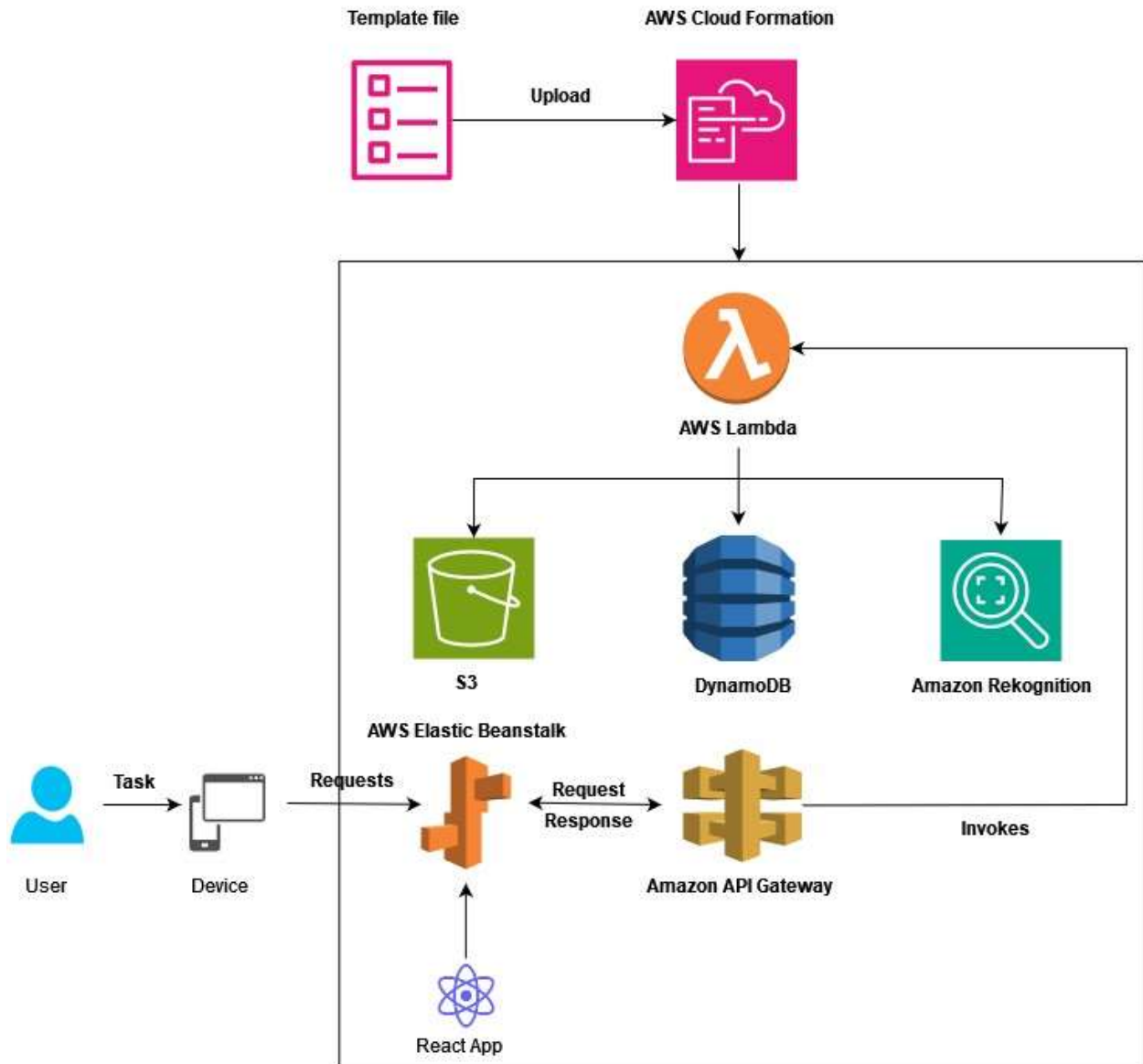


Fig. 4: Service Architecture of SnapSearch

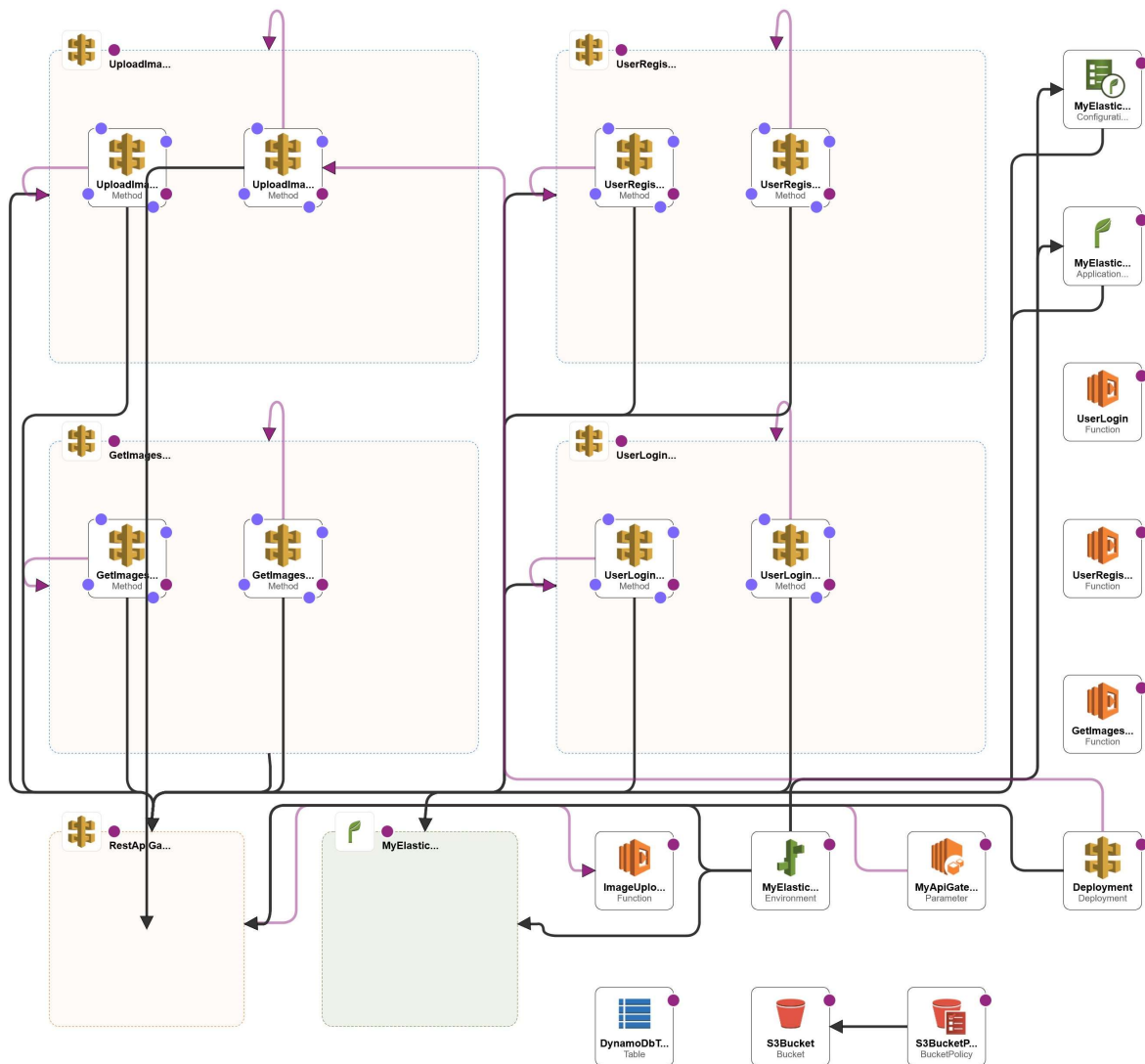


Fig. 5: Designer View in Cloudformation

Security:

For security, my application is mostly dependent on AWS.

I am storing user passwords in encrypted format on DynamoDB. All user data stored in Amazon DynamoDB is fully encrypted at rest. For encryption key management on DynamoDB table, I have chosen AWS KMS key which is owned and managed by DynamoDB.

For data in transit, all AWS services support encryption using TLS protocol.

I have defined security groups which is acting as virtual firewall for my ec2 instance which hosts web application's frontend.

My application's backend uses AWS Lambdas which are protected by AWS API Gateway.

Due to time constraints, I have not implemented my REST APIs authentication and authorization on API gateway. This work I am planning to do during the semester break.

Also, I could have used VPC in my project to create my application more secure.

Cost metrics:

I have used AWS Pricing Calculator to calculate estimate costs.

Here are the cost metrics for my project:

Current Estimates:

URL: <https://calculator.aws/#/estimate?id=860c0543bfebbcad529cf4f19b5e1756899a656b>

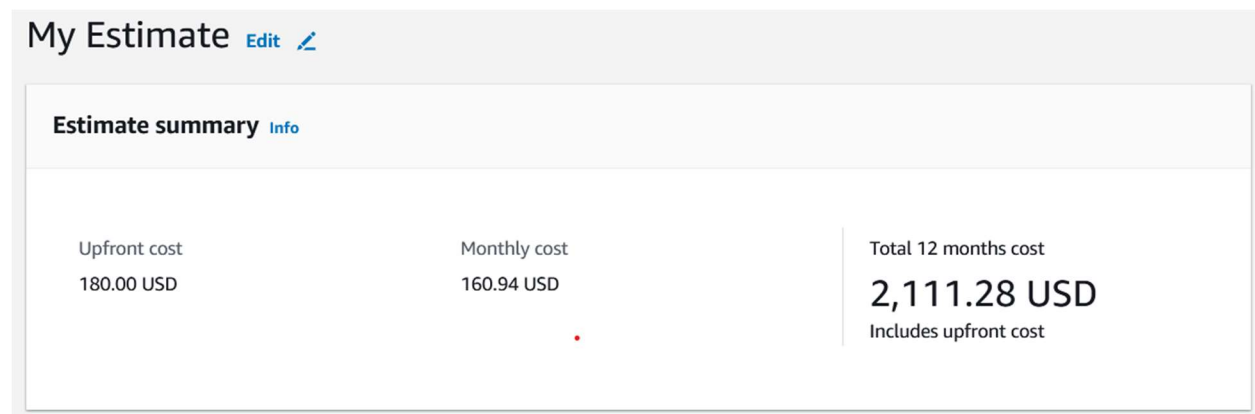


Figure 6: Upfront, monthly and annual cost (approx.)

My Estimate									
<input type="text" value="Find resources"/>				<button>Duplicate</button>	<button>Delete</button>	<button>Move to</button>	<button>Create group</button>	<button>Add support</button>	<button>Add service</button>
<div>< 1 > ⌂</div>									
<input type="checkbox"/>	Service Name	Status	Upfront cost	Monthly cost	Description	Region	Config Summary		
<input type="checkbox"/>	Amazon Simple Stora...	-	0.00 USD	0.00 USD	-	US East (Ohio)			
<input type="checkbox"/>	Amazon API Gateway	-	0.00 USD	7.00 USD	-	US East (Ohio)	REST API request unit...		
<input type="checkbox"/>	AWS Lambda	-	0.00 USD	0.00 USD	-	US East (Ohio)	Architecture (x86), Arc...		
<input type="checkbox"/>	Amazon EC2	-	0.00 USD	27.30 USD	-	US East (Ohio)	Tenancy (Shared Insta...		
<input type="checkbox"/>	Amazon Rekognition	-	0.00 USD	100.00 USD	-	US East (Ohio)	Number of images pr...		
<input type="checkbox"/>	Amazon DynamoDB	-	180.00 USD	26.64 USD	-	US East (Ohio)	Table class (Standard)...		
<input type="checkbox"/>	AWS CloudFormation	-	0.00 USD	0.00 USD	-	US East (Ohio)	Number of third-part...		

Figure 7: Detailed view for the current estimate

From above figures 6 and 7, there will be 180 USD upfront cost and the monthly cost will be 160.94 USD.

Cheaper Alternative Estimates:

I could save money by downgrading ec2 instance from t4 to t2 family, also I can limit how many images each user can upload on my application for no charge to lower cost from AWS Rekognition Additionally, I can shift from DynamoDB to S3 for storage.

These measures can help to reduce the cost by approx. 70 percent from the original estimate, bringing down the monthly cost to 73.79 USD and upfront cost to 0 USD.

URL: <https://calculator.aws/#/estimate?id=13eced22374cd89e5b80545c9b6346329e88ed92>

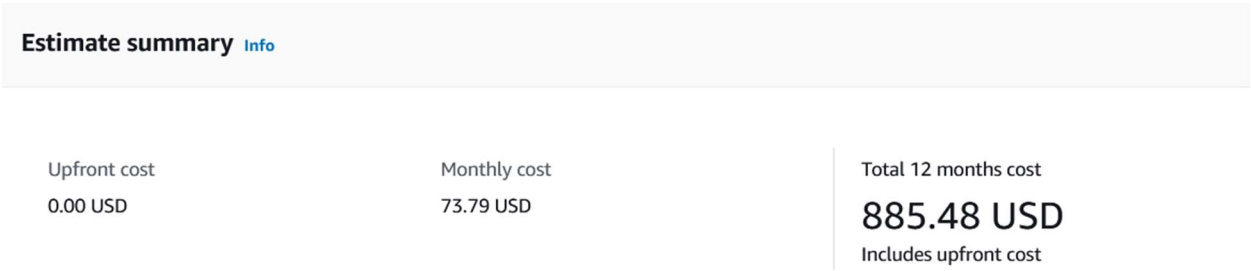


Figure 8: Upfront, monthly and annual cost (approx.) for cheaper approach

<input type="checkbox"/>	Service Name	Status	Upfront cost	Monthly cost	Description	Region	Config Summary
<input type="checkbox"/>	Amazon Simple Stora...	-	0.00 USD	0.00 USD	-	US East (Ohio)	
<input type="checkbox"/>	Amazon API Gateway	-	0.00 USD	7.00 USD	-	US East (Ohio)	REST API request unit...
<input type="checkbox"/>	AWS Lambda	-	0.00 USD	0.00 USD	-	US East (Ohio)	Architecture (x86), Arc...
<input type="checkbox"/>	Amazon EC2	-	0.00 USD	16.79 USD	-	US East (Ohio)	Tenancy (Shared Insta...
<input type="checkbox"/>	Amazon Rekognition	-	0.00 USD	50.00 USD	-	US East (Ohio)	Number of images pr...
<input type="checkbox"/>	AWS CloudFormation	-	0.00 USD	0.00 USD	-	US East (Ohio)	Number of third-part...

Figure 9: Detailed view for the cheaper estimate

Expensive Alternative Estimates:

URL: <https://calculator.aws/#/estimate?id=3af1dfed5b8254dac8833df3eb5058214c47e2a0>

<input type="checkbox"/>	Service Name	Status	Upfront cost	Monthly cost	Description	Region	Config Summary
<input type="checkbox"/>	Amazon Simple Storage Service	-	0.00 USD	0.00 USD	-	US East (Ohio)	
<input type="checkbox"/>	Amazon API Gateway	-	0.00 USD	7.00 USD	-	US East (Ohio)	REST API request unit...
<input type="checkbox"/>	AWS Lambda	-	0.00 USD	0.00 USD	-	US East (Ohio)	Architecture (x86), Arc...
<input type="checkbox"/>	Amazon EC2	-	0.00 USD	16.79 USD	-	US East (Ohio)	Tenancy (Shared Insta...
<input type="checkbox"/>	Amazon Rekognition	-	0.00 USD	50.00 USD	-	US East (Ohio)	Number of images pr...
<input type="checkbox"/>	AWS CloudFormation	-	0.00 USD	0.00 USD	-	US East (Ohio)	Number of third-part...

Figure 10: Upfront, monthly and annual cost (approx.) for expensive approach

Find resources				< 1 > ⚙			
<input type="checkbox"/>	Service Name	Status	Upfront cost	Monthly cost	Description	Region	Config Summary
<input type="checkbox"/>	Amazon Simple Storage Service	-	0.00 USD	0.00 USD	-	US East (Ohio)	
<input type="checkbox"/>	Amazon API Gateway	-	0.00 USD	7.00 USD	-	US East (Ohio)	REST API request unit...
<input type="checkbox"/>	AWS Lambda	-	0.00 USD	0.00 USD	-	US East (Ohio)	Architecture (x86), Arc...
<input type="checkbox"/>	Amazon EC2	-	0.00 USD	30.08 USD	-	US East (Ohio)	Tenancy (Shared Insta...
<input type="checkbox"/>	Amazon Rekognition	-	0.00 USD	210.00 USD	-	US East (Ohio)	Number of images pr...
<input type="checkbox"/>	AWS CloudFormation	-	0.00 USD	0.00 USD	-	US East (Ohio)	Number of third-part...
<input type="checkbox"/>	Amazon DynamoDB	-	180.00 USD	26.39 USD	-	US East (Ohio)	Table class (Standard)...

Figure 11: Detailed view for the expensive estimate

For deploying my SnapSearch app in a private cloud, I've planned a cost-effective infrastructure. To ensure basic server functionality without the need for extensive compute power, I've selected affordable servers at \$600 each. Considering the importance of availability, I'm opting for three servers, totaling \$1800. The tech stack involves Linux and Nginx for frontend serving and MySQL as the DBMS, incurring no additional costs. Open-source software will synchronize server activities seamlessly. Domain acquisition is estimated at \$20 to \$100. Annual maintenance, covering both hardware and software, is projected at \$100. Electricity costs are estimated at \$50 per month, excluding external cooling for this

small-scale setup. The upfront investment is \$1900, with an annual commitment of \$720 to \$800 for electricity, domain, and maintenance.

For ongoing management, DynamoDB monitoring will be crucial in the current cloud formation to control costs effectively. Future enhancements to the project may involve leveraging Amazon SES for personalized email communication, SQS for efficient chat support, and Amazon Lex for advanced customer service capabilities. These improvements aim to enhance user engagement and support features.

References:

[1] "AWS Lambda", AWS, (Available).

<https://aws.amazon.com/lambda/> [Last accessed on 30 Nov, 2023].

[2] "AWS Beanstalk", AWS, (Available).

<https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/Welcome.html> [Last accessed on 30 Nov, 2023].

[3] "Amazon S3", AWS, (Available).

<https://aws.amazon.com/s3/> [Last accessed on 30 Nov, 2023].

[4] "Amazon DynamoDB", AWS, (Available).

<https://aws.amazon.com/dynamodb/> [Last accessed on 30 Nov, 2023].

[5] "Amazon API Gateway", AWS, (Available).

<https://aws.amazon.com/api-gateway/> [Last accessed on 30 Nov, 2023].

[6] "AWS CloudFormation", AWS, (Available).

<https://aws.amazon.com/cloudformation/> [Last accessed on 30 Nov, 2023].

[7] "Amazon Rekognition", AWS, (Available).

<https://aws.amazon.com/rekognition> [Last accessed on 30 Nov, 2023].

[8] "Draw.io", Draw.io, (Available).

<https://app.diagrams.net/> [Last accessed on 30 Nov, 2023].

[9] "AWS Pricing Calculator", AWS, (Available).

<https://calculator.aws/#/> [Last accessed on 30 Nov, 2023].

[10] "Types of Cloud Computing", AWS, (Available)

<https://aws.amazon.com/types-of-cloud-computing/> [Last accessed on 30 Nov, 2023].