# Project Introduction:

DalCollab is a web application designed specifically for students at Dalhousie University to upload and share their projects. The platform facilitates collaboration and knowledge sharing among students by allowing them to view projects posted by their peers. The primary users of this application are Dalhousie University students who are engaged in various academic and extracurricular projects. The performance target for this application includes fast response times for uploading and retrieving projects, secure storage and transfer of data, and high availability to ensure students can access the platform at any time.

* **Primary Users: T**he primary users are students at Dalhousie University, engaged in various academic and extracurricular projects.
* **Performance Targets:**
* **Fast Response Times:** Ensuring quick uploads and retrieval of projects to provide a seamless user experience.
* **Secure Data Storage and Transfer:** Protecting user data and project information through secure storage solutions and encrypted data transfer.
* **High Availability:** Ensuring the platform is always accessible to accommodate the varied schedules of students.

# AWS Services:

## Compute:

* **EC2 (Elastic Compute Cloud)**: EC2 instances are chosen to host the frontend and backend due to their flexibility, scalability, and ease of integration with other AWS services. EC2 instances are chosen to host the frontend (Next.js) and backend (Spring Boot) applications. The flexibility of EC2 allows us to configure the environment precisely to meet the application's requirements.
* **Alternative: AWS Elastic BeanStalk**

While AWS Elastic Beanstalk simplifies the deployment and management of applications by automating deployment, scaling, and monitoring, it offers less control over the environment compared to EC2. The flexibility of EC2 allows us to configure the environment precisely to meet the application's requirements. It offers scalability, enabling us to adjust the computing power based on traffic and load, ensuring consistent performance. Thus, EC2 is preferred for its finer control and integration capabilities.

* **Lambda**: Lambda function is used for handling the specific tasks in such as sending the notifications and all. Lambda functions are used to handle specific tasks such as sending notifications when a new project is uploaded, or an account is created. This serverless approach is cost-effective and scalable, as it automatically adjusts to the load and only incurs charges when the function is executed.
* **Alternative: EC2**Running these tasks on EC2 involves managing server instances or containers, which increases operational complexity and cost. While these alternatives offer more control, Lambda's serverless model provides simplicity and cost-effectiveness, making it the preferred choice for handling discrete, event-driven tasks.

## Storage:

* **RDS (Relational Database Service)**: RDS is used to host the MySQL database, which stores project data, user accounts, and other related information. RDS reduces the operational overhead by handling database management tasks such as backups, updates, and scaling, allowing us to focus on application development.
  + **Alternative: DynamoDB**

Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance. However, it may not be ideal for applications requiring complex queries and transactional operations, which are better supported by MySQL in RDS. RDS offers a balance of performance, cost, and functionality, making it the preferred choice for this project**.**

## Network:

* **API Gateway**: Amazon API Gateway is selected to create, publish, maintain, monitor, and secure the API endpoints. API Gateway is used to create and manage the API endpoints for the application. It serves as the entry point for all client requests, routing them to the appropriate backend services, such as EC2 instances and Lambda functions. API Gateway ensures that the API is secure, scalable, and easy to maintain.
  + **Alternative: Lambda Function URLs**

While AWS Lambda Function URLs can be used to create HTTP endpoints for Lambda functions, they lack the full feature set of APIS Gateway, such as throttling, API lifecycle management, and detailed monitoring. API Gateway ensures that the API is secure, scalable, and easy to maintain, making it the better choice for managing API endpoints.

## General:

* **SNS (Simple Notification Service)**: SNS is used to send email notifications to students when they create an account or upload a project on the website. It ensures that users are promptly informed of important actions, enhancing user engagement and experience.
  + **Alternative: Amazon SES**

While Amazon SES is also a viable option for sending emails, SNS's integration capabilities and simplicity in sending notifications made it the better choice. SNS ensures that users are promptly informed of important actions, enhancing user engagement and experience.

* **Secret Manager**: Secrets Manager is used for storing credentials for the RDS database. It securely manages sensitive information, such as database passwords, ensuring that the credentials are not hardcoded in the application code and are protected from unauthorized access.
  + **Alternative: AWS Parameter Store**

While AWS Parameter Store can also store configuration data and secrets, Secrets Manager provides additional features such as automatic rotation of credentials and tighter integration with other AWS services. This makes Secrets Manager the preferred choice for managing sensitive information securely.

# Deployment Model:

Hence, for DalCollab, I chose AWS’s public cloud since it offers a strong service market and elastic platform. Since the frontend and the backend reside on the EC2 instances, it is possible to create conditions depending on the work conditions to provide the best optimization. Amazon RDS handles MySQL database and offers features such as automated database backup, user-pooling and automated database deletion, or resizing as required that help to save time on management and administration. These are functions to take backend responsibilities such as notifications thus being economical and elastic. Also, API Gateway caters to API operations, facilitating safe and elastic access to a business’s backends. AWS’s multiple data centre location around the globe strengthens the application accessibility and performance and on the other side, AWS has an open bottomless pricing model that makes it easier to make application more cost-effective. In general, all the AWS integration features and a wide range of services provided will complement DalCollab requirements to address the efficient implementation and further administration of the application.

# Delivery Model:

**EC2:** Infrastructure as a service

**AWS Lambda:** Function as a service

**RDS:** Platform as a service

**API Gateway:** Platform as a service

**SNS:** Software as a service

**Secret Manager:** Software as a service

For DalCollab, we utilized a range of AWS service models to optimize deployment and management of the application.

* **EC2 (Elastic Compute Cloud):** Used as Infrastructure as a Service (IaaS), EC2 has adequate computing capabilities to host a frontend app (Next.js) and a backend app (Spring Boot). This choice allows one to freely set up the environment and modulate the amount of traffic which is processed to maintain constant quality.
* **AWS Lambda:** Acting as Function as a Service (FaaS), Lambda is used to perform such functions as sending notifications. This serverless model supports cost-optimisation and is elastic, only charging the execution time thus cutting down on operational costs.
* **RDS (Relational Database Service):** When offered as Platform as a Service (PaaS), RDS takes the responsibility for MySQL database, its backups, updates and scaling. The stated way significantly minimizes the operational pressure that accompanies database management while drastically enabling the enhancement of development and performance.
* **API Gateway:** Also, a Platform as a Service (PaaS), the API Gateway, enables the formation of API endpoints, its administration and protection. It provides a clean and efficient interface between the client requests and the backend services when dealing with the authentication, routing of the requests and scaling.
* **SNS (Simple Notification Service):** Employed as Software as a Service (SaaS), SNS is used for sending email notifications to users. It simplifies the process of managing notifications and ensures users are informed of important updates without requiring infrastructure management.
* **Secrets Manager:** Deployed as Software as a Service (SaaS), Secrets Manager securely manages sensitive information such as database credentials. It protects against unauthorized access and reduces the complexity of handling secrets in application code.

Thus, the choice of these service models as IaaS, FaaS, PaaS, and SaaS allows DalCollab to use AWS’s opportunities optimally and minimize time and resource consumption, as well as proposing a secure, efficient, and cost-effective solution to our users.

# Architecture:

A screenshot of a video game

Description automatically generated

## Architecture Overview:

* EC2: Dalcollab’s frontend (react.js) and backend (springboot) is hosted on the ec2 with the help of the docker.
* RDS: RDS is used for storing all the data in the MySQL database.
* API Gateway: Api gateway routes the API request to the lambda.
* Lambda: Handle the logic of sending the email to the users.
* Secret Manager: It created for storing the credentials for the RDS.
* SNS: Its main purpose is to send the notifications to the users.

## Flow of the cloud architecture:

* User can access the web application from the EC2 in which our both frontend and backend is hosted.
* After that, user will see the dashboard of the dalcollab in which user can do register and login.
* When user do register, at that time the lambda function for sending the subscription request is sent to user regarding subscribe the channel.
* Once user do register and login, user will see the dashboard of the dalcollab in which user can see all the projects that are listed by other students.
* User can also add the project of themselves by giving all the details for that along with the facility to edit and delete the project.
* User can filter the services as per the technology type of the project.
* In this whenever the new project is added by any user, it will trigger another lambda function and send the notifications to all other users regarding that by giving all the descrition.

## Data Storage:

Most of the structure data for DalCollab is kept in the Amazon RDS which has MySQL database database for user details projects and its associations metadata. Moreover, Lambda functions might use temporary data in memory at run-time and any storage requirements of any kind are met through RDS. For the ones that require notification and sensitive data for example credntials, then Amazon SNS and Secrets Manager are used respectively for notification and secure management. Such a setup helps to minimize loss off data as it organizes data storage and management within the application space effectively and with high security.

## Programming Languages:

* **Frontend (Client): Next.js**

For DalCollab, Java was used for backend development with Spring Boot, chosen for its robustness and scalability in building enterprise-level applications.

* **Backend (Server): Spring Boot**

JavaScript (with TypeScript) was used for the frontend with Next.js, providing an interactive and dynamic user interface.

* **Lambda Functions: Python**

Python was utilized for Lambda functions, chosen for its simplicity and efficiency in handling serverless tasks such as notifications and other backend processes. This combination ensures a well-rounded, scalable, and efficient application.

## How is the system deployed to the cloud?

In order to deploy the system to the cloud for the first time, it can be achieved by using AWS cloud formation which automates the provisioning of the resources by running he cloud formation file and create all the resources which is defined in the cloud formation.

Other than that, in order to deploy any changes that are made in the frontend and backend and want to get that updated version then just need to build the image of the backend and frontend with the newly updated code. After that just need to reconnect the EC2 ssh and update the newly image so it will automatically make changes.

For updating the lambda function, just make sure that to deploy the new lambda function after making changes.

## Data Security:

* **JWT Authentication:**

Dalcollab Uses a JWT authentication to secure user interactions with the application. It’s basically using a token-based authentication as Upon successful login, users receive a JWT that contains encoded user information and claims. This token is used for subsequent API requests, ensuring that each request is authenticated without the need for re-entering credentials. It is stateless as server do not worry about storing it and it will manage It in the front-end side.

* **Secret Manager:**

This application stores the credentials of the database in the secret manager for the security purpose. This will ensure the security of the user’s data. It securely manages sensitive information, such as database passwords, ensuring that the credentials are not hardcoded in the application code and are protected from unauthorized access.

## Vulnerability:

Despite having the strong security, it has some vulnerabilities as in this application it is storing the jwt token in the local storage so there are some chances for exposing the token to the user and will affect by the unauthorised user and attack.

In this right now, this application is not encrypting the password to store in the database, so it has increased the risk to get attacked by unauthorised user.

This application is not fully serverless as it has only two lambda function and all other api are running on the spring boot. So now its tightly coupled with each other and its better to do all the whole application server less on the lambda function.

# Cost Estimation:

## Private Cloud Cost Estimation:

* **Compute Resources:**
  + **Server**: Purchase mid-range servers which is equivalent to the t2.m Gb micro ec2 instance or servers with the same specifications. Servers with the 4 vCPUs and 2 GB RAM would be sufficient.
  + **Estimated Cost**:
* **Serverless Functions:**
* **Database Storage:**
* **API Management**
* **Notifications**
* **Secret Manager**

## Future Improvements:

In the dalcollab, many enhancements can be made to make it very good and capable website. First, by adding the functionality of participating in each other’s project, it will not only increase the relation between the students but also will get the information of the different projects which will increase the knowledge of them.

In the future, it will be great if it has a chat feature in which students can talk with each other and discuss some topics which enhance the knowledge and information. Having this on the website, they can share the problems while developing the projects to each other and they can get the help from another students.

It would be very beneficial if the whole application is serverless so it will be able to handle the loads that are coming on the website. By doing this this application will become the scalable, secure and fully serverless.

# References: