**Student Information Management System**

A serverless web application built using AWS services that allows users to manage student information. The project demonstrates the integration of various AWS technologies to create a fully functional application.

**Table of Contents**

* [Project Overview](#project-overview)
* [Features](#features)
* [Technologies Used](#technologies-used)
* [Project Structure](#project-structure)
* [Screenshots](#screenshots)
* [AWS Architecture](#aws-architecture)
* [How to Use](#how-to-use)
* [Setup Instructions](#setup-instructions)
* [Known Issues](#known-issues)
* [Future Enhancements](#future-enhancements)
* [License](#license)
* [Acknowledgments](#acknowledgments)

**Project Overview**

The Student Information Management System is a simple yet effective project designed to demonstrate the integration of AWS services. It provides functionality to save student details (ID, name, class, and age) and view all student records through a web interface.

**Features**

* **Frontend**: Built using HTML, CSS, and JavaScript, hosted on AWS S3.
* **Backend**: AWS Lambda functions (written in Python) handle data storage and retrieval.
* **Database**: Student data is stored in a DynamoDB table.
* **API Gateway**: Exposes the Lambda functions as RESTful APIs.
* **CloudFront**: Provides HTTPS support for secure content delivery.

**Technologies Used**

* **AWS Services**: S3, DynamoDB, Lambda, API Gateway, CloudFront
* **Languages**: HTML, CSS, JavaScript (for frontend), Python (for backend)
* **Tools**: Git, GitHub, AWS Management Console

**Project Structure**

plaintext

Copy code

/ (Root Directory)

│

├── index.html # HTML file for the web interface

├── styles.css # CSS file for styling the web interface

├── scripts.js # JavaScript file for handling frontend logic

├── lambda\_function.py # Python code for AWS Lambda functions

├── README.md # Project documentation

└── .gitignore # Files to exclude from version control

**Screenshots**

**Home Page**

<!-- Replace with your actual screenshot file name -->

**Save Student Data**

<!-- Replace with your actual screenshot file name -->

**View All Students**

<!-- Replace with your actual screenshot file name -->

**AWS Architecture**

The project uses various AWS services in the following way:

1. **S3**: Hosts the static website (HTML, CSS, JS).
2. **API Gateway**: Routes HTTP requests to the appropriate Lambda functions.
3. **Lambda**: Handles two main operations:
   * POST to save student information in DynamoDB.
   * GET to retrieve all student records from DynamoDB.
4. **DynamoDB**: Stores student information in a table.
5. **CloudFront**: Distributes the frontend content securely over HTTPS.

**How to Use**

1. **Save Student Data**: Enter student details (ID, name, class, age) in the input fields and click the "Save" button to store the data.
2. **View All Students**: Click the "View All Students" button to fetch and display all student records.

**Setup Instructions**

Follow these steps to set up the project on AWS:

**1. Frontend Deployment**

* Upload index.html, styles.css, and scripts.js to an S3 bucket.
* Configure the bucket to host a static website.
* Enable public access for the bucket and note the website URL.

**2. DynamoDB Setup**

* Create a DynamoDB table named Students with studentId as the primary key.

**3. Lambda Functions**

* Create two AWS Lambda functions:
  + **POST Function**: Saves student data to DynamoDB.
  + **GET Function**: Retrieves all student records from DynamoDB.
* Use the provided lambda\_function.py code for both functions.

**4. API Gateway**

* Set up an API Gateway to create RESTful APIs for the Lambda functions.
* Map the HTTP methods (POST and GET) to the respective Lambda functions.

**5. CloudFront Distribution**

* Create a CloudFront distribution to serve the content from your S3 bucket securely over HTTPS.

**Known Issues**

* **Decimal Serialization**: If you encounter serialization errors, ensure that numeric data from DynamoDB is converted to JSON serializable types.

**Future Enhancements**

* Add authentication to secure data access.
* Implement data validation on the frontend.
* Enhance the user interface for a better user experience.

**License**

This project is open-source and available under the MIT License.

**Acknowledgments**

* Special thanks to the AWS documentation and community forums for providing helpful resources.