

# AWS Frontier

Student: Saul Castillo  
Product Owner: *Masoud Sadjadi*  
Instructor/Faculty: *Masoud Sadjadi*, Florida International University

## PROBLEM

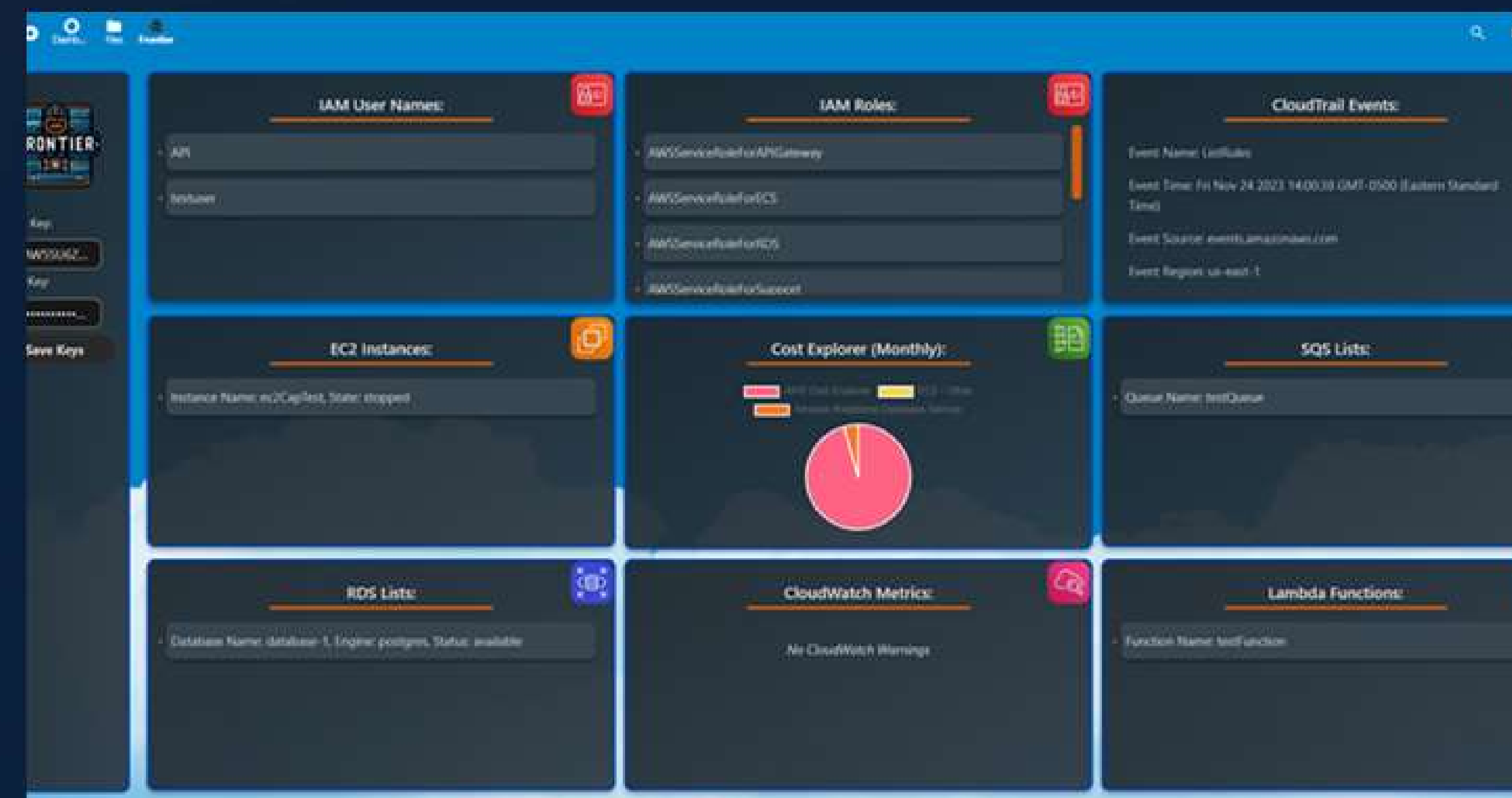
When using a service such as AWS, there are many different resources that an account can create. However, there is not a centralized location to view every resource on an account. Our main goal was to assess this issue while also maintaining security of the application. The security of an application is one of the most important factors when developing. While working with AWS accounts, we wanted to make sure that our users would be able to interact with the application without doubting if their AWS information was in jeopardy. For this reason, we decided to create AWS Frontier from the ground up.

## CURRENT SYSTEM

Our current version of AWS Frontier has the following features:

- Encryption/Decryption of AWS user Access and Secret keys
- Display a list of IAM Users on an account
- Display a list of IAM Roles on an account
- Display a list of parsed CloudTrail API Logs
- Display a list of EC2 Instances
- Display a list of SQS Queues on an account
- Display a list of RDS Databases on an account
- Display a list of Lamba functions on an account
- Cost Explorer Chart (monthly)

## RESULTS



```
function encrypt(text, encryptionKey) {
    const key = CryptoJS.enc.Utf8.parse(encryptionKey);
    const iv = CryptoJS.lib.WordArray.random(16);
    const cipherText = CryptoJS.AES.encrypt(text, key, {
        iv: iv,
        mode: CryptoJS.mode.CBC,
        padding: CryptoJS.pad.Zero,
    });
    return iv.toString() + cipherText.toString();
}

function decrypt(ciphertext, encryptionKey) {
    const key = CryptoJS.enc.Utf8.parse(encryptionKey);
    const iv = CryptoJS.AES.decrypt(ciphertext, key, {
        mode: CryptoJS.mode.CBC,
        padding: CryptoJS.pad.Zero,
    });
    return iv.toString() + CryptoJS.AES.decrypt(ciphertext, key, {
        mode: CryptoJS.mode.CBC,
        padding: CryptoJS.pad.Zero,
    }).toString();
}

// generate a 32 byte (256 bits) key for AES encryption
const generateEncryptionKey = () => {
    const bytesLength = 32;
    const key = CryptoJS.lib.WordArray.random(bytesLength);
    return key.toHex().toString();
}

function saveKeysToLocalStorage() {
    const accessKey = encrypt(document.getElementById("accessKey").value, generateEncryptionKey());
    const secretKey = encrypt(document.getElementById("secretKey").value, generateEncryptionKey());
    localStorage.setItem("accessKey", accessKey);
    localStorage.setItem("secretKey", secretKey);
    alert("All keys saved!");
}

// load encryption key or create one if none
const loadEncryptionKey = () => {
    const storedEncryptionKey = localStorage.getItem("encryptionKey");
    if (!storedEncryptionKey) {
        localStorage.setItem("encryptionKey", generateEncryptionKey());
        storedEncryptionKey = localStorage.getItem("encryptionKey");
    }
}
```

## IMPLIMENTATION



## SOLUTION

Our team successfully developed a monitoring panel hosted on Nextcloud for AWS. This panel enables users to securely enter their AWS access and secret keys, which are essential for connecting to their AWS accounts. These keys are encrypted and decrypted as needed, using AES, IV, and CBC. The panel currently features the most commonly utilized AWS resources, and a backend designed for future additions. Also, we focused on delivering a user-friendly interface, characterized by its ease of use and clarity. Each resource is neatly organized within custom-designed containers, facilitating a seamless and efficient user experience.

## Summary

The development of AWS Frontier involved thorough research into the functionalities and potential of Nextcloud. Our team dedicated significant effort to ensure seamless integration and compatibility of our code with Nextcloud's framework. This project has not only enriched our professional skill set but also provided a solid foundation for future capstone students to expand upon. The groundwork we have established with AWS Frontier opens up limitless opportunities for further innovation and development.

## Contributions

- In this project, my primary focus was on the backend development of the application. I took the lead in designing and implementing the template for API integration, and managed the backend construction of complex API calls. These were crucial for ensuring efficient and seamless interaction with the frontend CSS/HTML interface.
- A key aspect of my role also involved bolstering the security of our system. Recognizing the sensitivity of API access keys, particularly those from AWS used in monitoring panel elements
- AES, IV, CBC

## Verification

- Verified that the application was pulling the correct information for an account by testing multiple accounts
- Verified the information that was retrieved for every resource by check manually through the AWS Console
- Verified the Access and Secret key inputs were being encrypted and decrypted to ensure security

## ACKNOWLEDGEMENT

The material presented in this poster is based upon the work supported by XXXXX (name of the project sponsor: federal, state or local government, or corporate partners, where applicable). We/I thank XXXX for his/her/their assistance and mentorship that I/we received throughout the senior design project.