Saveetha School of Engineering

Saveetha Institute of Medical and Technical Sciences Institute of Computer Science Engineering

**A CAPSTONE PROJECT REPORT**

**Detecting Data leaks using SQL**

**(Cloud Computing and Big Data Analytics using Cloud Federation-CSA1583)**

***Submitted to***

**SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES**

***In partial fulfillment for the award of the degree of***

**BACHELOR OF ENGINEERING IN COMPUTER SCIENCE & ENGINEERING**

***By***

**T Geethika (192110330)**

***Course Faculty***

**DR.J.CHENNI KUMARAN**

**JUNE-2024**

**SAVEETHA SCHOOL OF ENGINEERING**

**ACKNOWLEDGEMENT**

This project work would not have been possible without the contribution of many people. It gives me immense pleasure to express my profound gratitude to our Honorable Chancellor **Dr. N. M. Veeraiyan**, Saveetha Institute of Medical and Technical Sciences, for his blessings and for being a source of inspiration. I sincerely thank our Director of Academics **Dr. Deepak Nallaswamy,** SIMATS, for his visionary thoughts and support. I am indebted to extend my gratitude to our Director **Dr. Ramya Deepak,** Saveetha School of Engineering, for facilitating us all the facilities and extended support to gain valuable education and learning experience.

I register my special thanks to **Dr. B. Ramesh,** Principal, Saveetha School of Engineering for the support given to me in the successful conduct of this project. I wish to express my sincere gratitude to my Course faculty **DR.J.CHENNI KUMARAN**, for his inspiring guidance, personal involvement and constant encouragement during the entire course of this work.

I am grateful to Project Coordinators, Review Panel External and Internal Members and the entire faculty of the Department of Predictive Engineering, for their constructive criticisms and valuable suggestions which have been a rich source to improve the quality of this work.

**STUDENT NAME’s**

**T Geethika(192110330)**

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **Sl. No** | **CONTENTS** | **Page no** |
| **1** | **Problem Statement** |  |
| **2** | **Requirements Gathering** |  |
|  | * Identifying the specific requirements |  |
|  | * Determine the necessary features |  |
| **3** | **Choose Cloud Provider best fits your needs** |  |
| **4** | **Develop Frontend** |  |
|  | * Layout |  |
|  | * User-Friendly |  |
|  | * Color Selection |  |
| **5** | **Develop Backend** |  |
|  | * Database Implementation |  |
|  | * Execution |  |
| **6** | **Implementation and Integrate with Cloud Services** |  |
|  | * Implementation |  |
|  | * Cloud Integration and Deployment |  |
|  | * Testing |  |
| **7** | **Performance Evaluation** |  |
| **8** | **Conclusion** |  |

### Problem Statement

In the modern digital age, data breaches have become increasingly common, posing significant risks to personal privacy and security. These breaches can occur through various means such as password theft, keylogging, or ransomware attacks, leading to severe repercussions for users. Current security measures often fall short in effectively preventing these breaches, especially in dynamic environments like e-commerce sites where users frequently log in and interact.

This project aims to address these security vulnerabilities by developing a robust system using cloud technology and securing it with Advanced Encryption Standard with eXtended (AESX) Encryption. The system will be designed to detect and prevent data breaches through comprehensive content inspection and contextual analysis. By leveraging Data Loss Prevention (DLP) solutions, the system will classify users based on their online behaviors and messages. It will identify potentially harmful actions or messages that could invite security vulnerabilities, whether intentional or unintentional, and take appropriate measures to mitigate these risks.

#### Objectives:

* **Develop a Cloud-Based Security System**: Utilize cloud technology to build a scalable and robust system secured with AESX Encryption to protect user data.
* **Detect Data Breaches**: Implement advanced content inspection and contextual analysis to identify potential data breaches.
* **Classify Users**: Analyze user behaviors and messages to classify users and identify potential assaulters or those unknowingly inviting security risks.
* **Prevent Security Vulnerabilities**: Use DLP solutions to intelligently restrict or take action against users whose actions pose security threats.
* **Protect E-Commerce User Data**: Enhance the privacy and security of personal information held on e-commerce sites, reducing the risk of data loss, financial loss, and mental health impacts.

#### Key Challenges:

* **Implementation of AESX Encryption**: Ensuring the encryption mechanism is robust and effectively secures data both at rest and in transit.
* **Content Inspection and Contextual Analysis**: Developing algorithms and tools for accurate and real-time analysis of user content and behavior.
* **User Classification**: Creating a reliable classification system to distinguish between malicious users and those inadvertently causing security risks.
* **Integration of DLP Solutions**: Seamlessly integrating Data Loss Prevention measures to proactively prevent data breaches and protect sensitive information.
* **User Privacy and Compliance**: Ensuring the system adheres to privacy laws and regulations while effectively monitoring and restricting harmful activities.

By addressing these challenges, this project aims to create a comprehensive security solution that not only protects user data but also fosters a safer online environment on e-commerce platforms.

### Requirements Gathering

#### 1. Identifying the Specific Requirements

**A. Cloud Technology Platform**

* **Selection of Cloud Provider**: Choose a cloud service provider (AWS, Azure, GCP) that supports the necessary services for building, deploying, and scaling the system.
* **Compliance**: Ensure the chosen cloud provider meets compliance standards such as GDPR, HIPAA, etc.

**B. Encryption and Security**

* **AESX Encryption**: Implement AESX encryption for securing data both at rest and in transit.
* **Key Management**: Utilize cloud provider’s key management service (KMS) for handling encryption keys securely.
* **Multi-Factor Authentication (MFA)**: Enforce MFA for user authentication to enhance security.

**C. Data Breach Detection**

* **Content Inspection**: Implement tools and algorithms to inspect content for sensitive data.
* **Contextual Analysis**: Develop mechanisms to analyze the context of data usage to detect anomalies and potential data breaches.
* **Behavioral Analysis**: Monitor and analyze user behavior to identify suspicious activities.

**D. User Classification and Monitoring**

* **User Profiling**: Develop a system to profile users based on their activities and behaviors.
* **Machine Learning Models**: Use machine learning to classify users as potential threats or safe based on their activities and messages.
* **Real-Time Monitoring**: Implement real-time monitoring to detect and respond to suspicious activities instantly.

**E. Data Loss Prevention (DLP)**

* **DLP Policies**: Define and implement DLP policies to prevent unauthorized data access and leaks.
* **Automated Responses**: Develop automated responses to restrict users or take action when suspicious activities are detected.
* **Incident Management**: Create a framework for managing and responding to security incidents.

**F. User Interface and Experience**

* **Dashboard**: Create a user-friendly dashboard for administrators to monitor activities, view alerts, and manage incidents.
* **Notifications**: Implement notification systems to alert users and administrators of potential security breaches.

**G. Compliance and Legal**

* **Regulatory Requirements**: Ensure the system complies with relevant regulations such as GDPR, HIPAA, PCI DSS, etc.
* **Data Privacy**: Implement measures to protect user data privacy according to legal standards.
* **Audit Logs**: Maintain detailed audit logs for all activities and accesses for compliance and forensic purposes.

#### 2. Determine the Necessary Features

**A. Cloud Technology Platform Features**

* **Scalability**: Ensure the platform can scale to handle increased data and user loads.
* **Reliability**: Implement redundancy and failover mechanisms to ensure high availability.
* **Integration**: Ensure compatibility with other security tools and services.

**B. Encryption and Security Features**

* **AESX Encryption**: Implement robust AESX encryption algorithms.
* **Key Management Service (KMS)**: Integrate with the cloud provider’s KMS.
* **Multi-Factor Authentication (MFA)**: Integrate MFA into the user authentication process.

**C. Data Breach Detection Features**

* **Content Inspection Tools**: Develop or integrate tools for inspecting content for sensitive information.
* **Contextual Analysis Algorithms**: Implement algorithms for contextual analysis to detect anomalies.
* **Behavioral Analysis Tools**: Utilize machine learning for behavioral analysis.

**D. User Classification and Monitoring Features**

* **User Profiling System**: Create a system to profile and classify users based on behavior.
* **Machine Learning Models**: Develop and train machine learning models for user classification.
* **Real-Time Monitoring**: Implement tools for real-time monitoring of user activities.

**E. Data Loss Prevention (DLP) Features**

* **DLP Policy Engine**: Develop an engine to define and enforce DLP policies.
* **Automated Response System**: Implement a system for automated responses to suspicious activities.
* **Incident Management Framework**: Create a framework for handling security incidents.

**F. User Interface and Experience Features**

* **Admin Dashboard**: Develop a comprehensive dashboard for administrators.
* **User Alerts and Notifications**: Implement systems for sending alerts and notifications.
* **User Management Interface**: Create an interface for managing user profiles and permissions.

**G. Compliance and Legal Features**

* **Compliance Checks**: Implement tools to ensure compliance with relevant regulations.
* **Data Privacy Mechanisms**: Develop mechanisms to protect user data privacy.
* **Audit Logging**: Implement detailed logging for all activities and accesses.

### Integration with Existing Systems

**A. User Management and Authentication Systems**

* **LDAP/Active Directory Integration**: Integrate with existing user management systems for authentication.
* **SSO Solutions**: Implement Single Sign-On (SSO) for unified access control.

**B. Security Systems**

* **Existing IDS/IPS Systems**: Integrate with existing Intrusion Detection and Prevention Systems.
* **SIEM Solutions**: Ensure compatibility with current Security Information and Event Management systems for centralized logging and monitoring.

**C. Compliance and Audit Systems**

* **Audit Logs**: Integrate with existing audit logging systems to maintain a unified log management system.
* **Compliance Tools**: Ensure the new system works seamlessly with existing compliance management tools.

By gathering these requirements and determining the necessary features, the project aims to build a robust and secure system that protects user data, prevents data breaches, and ensures compliance with industry standards and regulations.

**Choosing a Cloud Provider**

#### Amazon Web Services (AWS)

**Key Features:**

* **Wide Range of Services**: AWS offers a comprehensive suite of services including computing power (EC2), storage (S3), databases (RDS, DynamoDB), machine learning (SageMaker), and more.
* **Security and Compliance**: AWS provides robust security features such as AWS Key Management Service (KMS) for managing cryptographic keys, AWS CloudHSM for hardware-based key storage, and AWS Shield for DDoS protection. It is compliant with numerous regulations, including GDPR, HIPAA, and PCI DSS.
* **Encryption Services**: AWS supports encryption at rest and in transit. AWS KMS and AWS CloudHSM enable secure key management and AES encryption.
* **DLP Solutions**: AWS Macie is a fully managed data security and data privacy service that uses machine learning and pattern matching to discover and protect sensitive data.

**Why AWS?**

* AWS offers a mature, reliable, and comprehensive set of security tools and services that align well with the needs of this project.
* The extensive ecosystem of AWS services can support the development, deployment, and scaling of the application.
* AWS’s global infrastructure ensures high availability and disaster recovery capabilities.

### Developing the Frontend with AWS

To develop the frontend for the project, we will utilize AWS services such as AWS Amplify for hosting, managing authentication, and connecting to backend services. The following sections outline the design considerations, layout, user-friendliness, and color selection.

#### 1. Layout

**A. Home Page:**

* **Header**: Contains the logo, navigation menu (Home, Features, Pricing, Contact), and user login/sign-up buttons.
* **Banner**: A large, eye-catching banner with a brief introduction to the service and a call-to-action button.
* **Features Section**: Highlights key features with icons and brief descriptions.
* **How It Works**: A step-by-step guide on how to use the service.
* **Testimonials**: User feedback and success stories.
* **Footer**: Contact information, social media links, and legal disclaimers.

**B. Dashboard:**

* **Sidebar**: Contains navigation links to different sections (Overview, Messages, Security Alerts, Settings).
* **Main Content Area**: Displays the user's recent activities, alerts, and data breach analysis.
* **Top Bar**: Quick access to profile settings and logout options.

**C. User Profile:**

* **Profile Picture**: Upload and update profile picture.
* **Personal Information**: Display and edit personal details such as name, email, and contact information.
* **Security Settings**: Manage password, two-factor authentication, and other security options.

#### 2. User-Friendly Design

**A. Intuitive Navigation:**

* Clear and logical navigation structure with easily accessible menus and links.
* Breadcrumbs for better user orientation and navigation history.

**B. Responsive Design:**

* Mobile-first approach ensuring the site is fully responsive and looks great on all devices.
* Use of media queries and flexible grid layouts (CSS Grid/Flexbox).

**C. Accessibility:**

* Adherence to WCAG (Web Content Accessibility Guidelines) to make the site accessible to users with disabilities.
* Proper use of ARIA (Accessible Rich Internet Applications) roles and attributes.

**D. Performance Optimization:**

* Optimized images and assets for fast loading times.
* Use of lazy loading for images and content.
* Implementing caching strategies using AWS CloudFront.

**E. User Feedback:**

* Interactive elements like hover effects, tooltips, and confirmation dialogs.
* Feedback mechanisms like loading spinners and progress bars during data processing.

#### 3. Color Selection

**A. Color Palette:**

* **Primary Color**: A calming blue (#007ACC) to convey trust and professionalism.
* **Secondary Color**: A complementary green (#28A745) for call-to-action buttons and highlights.
* **Neutral Colors**: Shades of grey (#F8F9FA for background, #6C757D for text) for a clean and modern look.

**B. Color Accessibility:**

* Ensuring sufficient color contrast for readability.
* Avoiding color combinations that are problematic for color-blind users (e.g., red-green color blindness).

**C. Branding Consistency:**

* Using colors consistently across all UI components to maintain a cohesive brand identity.
* Aligning color choices with the organization’s existing branding guidelines if applicable.

### Implementation with AWS Amplify

1. **Setup AWS Amplify:**
   * Initialize an AWS Amplify project.
   * Configure Amplify for hosting, authentication, and connecting to backend services.

bash

Copy code

amplify init

amplify add hosting

amplify add auth

amplify add api

amplify push

1. **Developing the Frontend:**
   * Use a modern frontend framework like React for building the UI.
   * Structure the project with components for Home Page, Dashboard, and User Profile.

bash

Copy code

npx create-react-app frontend

cd frontend

npm install aws-amplify @aws-amplify/ui-react

1. **Integrating AWS Amplify:**
   * Configure AWS Amplify in your React project to manage authentication and API calls.

javascript

Copy code

// src/index.js

import React from 'react';

import ReactDOM from 'react-dom';

import './index.css';

import App from './App';

import Amplify from 'aws-amplify';

import config from './aws-exports';

Amplify.configure(config);

ReactDOM.render(<App />, document.getElementById('root'));

1. **Designing the Layout:**
   * Create React components for the layout sections (Header, Banner, Features, etc.).
   * Apply CSS for responsive design and styling.

javascript

Copy code

// src/components/Header.js

import React from 'react';

const Header = () => (

<header>

<nav>

<ul>

<li><a href="/">Home</a></li>

<li><a href="/features">Features</a></li>

<li><a href="/pricing">Pricing</a></li>

<li><a href="/contact">Contact</a></li>

</ul>

</nav>

<div>

<button>Login</button>

<button>Sign Up</button>

</div>

</header>

);

export default Header;

1. **Styling with CSS:**
   * Define CSS styles for the components, ensuring responsiveness and accessibility.

css

Copy code

/\* src/index.css \*/

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

}

header {

display: flex;

justify-content: space-between;

background-color: #007ACC;

color: white;

padding: 1rem;

}

nav ul {

list-style: none;

display: flex;

}

nav ul li {

margin-right: 1rem;

}

nav ul li a {

color: white;

text-decoration: none;

}

button {

background-color: #28A745;

color: white;

border: none;

padding: 0.5rem 1rem;

cursor: pointer;

}

button:hover {

background-color: #218838;

}

By following these steps, you can create a well-designed, user-friendly, and visually appealing frontend for the project using AWS Amplify and modern web development practices.

### Performance Evaluation Process

1. **Loading Speed:**
   * Run a Google Lighthouse audit and obtain a score for performance.
   * Use WebPageTest to analyze loading times, TTFB, FCP, LCP, and TTI.
   * Address identified performance bottlenecks (e.g., large images, unoptimized scripts).
2. **Responsiveness:**
   * Test the application in responsive design mode for various screen sizes (e.g., mobile, tablet, desktop).
   * Ensure fast touch/click response times by optimizing event handlers.
3. **Accessibility:**
   * Perform a Google Lighthouse accessibility audit and fix any issues.
   * Use Axe DevTools to identify and address additional accessibility concerns.
   * Verify with the WAVE tool to ensure compliance with WCAG.
4. **Security:**
   * Test SSL/TLS configuration with Qualys SSL Labs and ensure a high rating.
   * Run OWASP ZAP to find and fix security vulnerabilities.
   * Implement security headers and verify with the Security Headers tool.
5. **Scalability:**
   * Perform load testing with Apache JMeter to evaluate the application under heavy usage.
   * Conduct stress testing using BlazeMeter to understand limits and performance under extreme conditions.
   * Monitor and analyze AWS CloudWatch metrics to ensure the application can scale as needed.

### Conclusion

By systematically evaluating these performance aspects, you can ensure that the developed frontend is fast, responsive, accessible, secure, and scalable. This comprehensive performance evaluation will help in identifying potential areas of improvement and implementing necessary optimizations to provide an excellent user experience.