A Platform for Managing Security Evaluations

Project Final Report

SYSC 4907 Winter 2022

Team Members
(Last name in alphabetical order)
Zijun Hu 101037102
Tiantian Lin 101095243
Jiawei Ma 101034173
Ruixuan Ni 101092506

Supervisor
Jason Jaskolka, Ph.D., P.Eng.

Date: April 12, 2022

Abstract

Information technology is widely applied in modern lives, and the public pays more attention to information security. To prove the system's security level to the public, developers need to collect evidence during development. However, it is hard to manage this evidence since the amount can be large, and the source of evidence may vary. It is hard to find a system that can currently manage this evidence.

Our team implemented a web-based system used to manage evidence in this project. This system allows developers to upload evidence with criteria tags and project tags. So, it is easier for developers to view and manage their evidence. Criteria describe what should be accomplished to secure the system or information. This system allows evaluators to check the evidence and mark them as satisfied with the criteria or not. It can simplify the process of security evaluation.

Acknowledgments

In developing this project, we received help from respected persons who deserve our most incredible gratitude. The completion of this project gives us so much confidence and pleasure. We want to show our appreciation to Dr. Jason Jaskolka for giving us a well-organized guideline and valuable suggestions for this project throughout these two semesters.

In addition, thank you to Professor Mostafa Taha, who gave us suggestions for our progress report.
Finally, thanks to all the members of this project group. Your efforts have brought this project from conception to realization.

Contents

List of Figures

List of Tables

Chapter 1

1. Introduction

People are already dependent on information technology heavily nowadays. Corporations pay more attention to system security to satisfy people's needs and protect their privacy and assets. There are already multiple well-implemented security mechanisms applied in the working environment. Multiple security evaluation criteria, such as Trusted Computer System Evaluation Criteria (TCSEC), Common Criteria, and European Information Technology Security Evaluation Criteria (ITSEC), are made for computer security certification to evaluate those security mechanisms.[1]

There would be a large amount of security evidence produced during the development lifecycle. This evidence can come from different sources for complex or large projects. An evidence management system is necessary to manage evidence for evaluation corresponding to criteria.

To do the security evaluation, we need to collect evidence from the development process. There are some existing tools used for security testing and vulnerability management. However, those tools usually pay less attention to evidence of security evaluation. An evidence management tool could help developers store and manage evidence and be convenient for security evaluators and regulatory authorities to verify and evaluate the system's security.

This project is to design an evidence management platform. This platform could allow users to upload and manage evidence and criteria. It should also fit the production environment and keep data safe.

1.1 Problem Motivation

Nowadays, people store their data in the Cloud or allow applications to access their private information for convenience. Internet corporations need to pay more effort to protect their data. The customers also want to know if their data is well protected. Thereupon, some evaluation criteria (for example, Common Criteria) are built in response to people's needs.[2] These criteria give introductions to corporations about functionalities related to the implementation of security services. It also provides a common standard for customers to know if a corporation can provide reliable protection to their privacy and assets. Developers need to store and manage the security evidence for further evaluation throughout the development lifecycle. However, there is no such tool existing on public platforms currently. It is also hard for evaluators to evaluate and mark evidence files with criteria without such a tool.

1.2 Problem Statement

The evidence developers can include documentation and codes from multiple sources throughout the development lifecycle. For large-scale project development, it can be hard to manage and trace the information and change other evidence. It is also hard for evaluators and regulatory authorities to address the certification of additional evidence and get the general evaluation result. So, we need a way to allow developers to store and manage the evidence files during their group development.

1.3 Proposed Solution

We plan to implement a system to manage evidence to solve this problem. This system should provide services to developers and security evaluators. It should allow developers to upload and update different evidence. Each evidence-related operation will be recorded with the operator's name, making the change and management traceable. This system can help developers to manage evidence throughout the development lifecycle. It should also allow security evaluators to view them and evaluate if they satisfy the criteria.

1.4 Accomplishments

The team successfully implemented a web-based platform for evidence management and evaluation in this project. The team has built a register and login system to help manage users based on their roles. Users are allowed to create new accounts and log in to the system. The access rights (user, developer, quality Analyst, administrator) would be granted by the administrator account. The account registered first in this platform would be automatically set as an administrator. The administrator account has all permissions in this system. Furthermore, only the administrator account can upload and delete criteria in the system. The criteria uploading allow administrators to upload one by one by providing name and description or upload various by uploading a CSV file.

The administrator can assign a user as a developer. A developer is allowed to upload and manage evidence files. They could upload evidence files (in text form) by entering their name and project name and selecting the criteria that this file corresponds to. The evidence file will then be able to be viewed by other developers or quality analysts. It can be searched via a search page. This page not only supports specific search with evidence id but also supports fuzzy search, which allows users to filter the searching result with multiple tags. The evidence file can also be updated by developers. The system will record the username and edit time for each update.

For quality analyst accounts, they can change the status of evidence files. For new uploaded evidence, the status will default set to pending. After evaluating this evidence file, a quality analyst can change the status to approved or disapproved. These statuses are visible to developers. Developers can update their evidence files that the quality analyst disapproved. After the update, the status of the evidence file will be reset to pending.

We have also designed the HTML pages and CSS files to make the platform user-friendly. We implemented a navigation bar fixed at the top of the web pages to make the switching between functions easily.

1.5 Organization of the Report

This report is organized into five chapters. The introduction chapter summarizes the background, solved problems, and our project's accomplishments. After the introduction, Chapter 2, The Engineering Project, describes the engineering skills we applied during the processing of this project, which includes the management of the project and the contribution of each team member.

Chapter 3, Background Literature Review, summarizes the key concepts related to this project. Our team showed our background research on criteria, security evaluation, and security evidence in this chapter. The research towards products that have something in common with our project is also discussed in chapter 3.

Chapter 4, Background Literature Review, are the technical solutions for this problem. In this chapter, our team provides our solution to the problem in detail. We discussed how we design the project structure, build the database, implement the web frame, and deploy the project to the Cloud. We also explained some decisions we made by comparing the options.

Finally, Chapter 5, Conclusions, is the overall conclusion of what we have done and features that could be added or improved in the future.

Chapter 2

2. The Engineering Project

2.1 Health and Safety

This project is developing a web-based system, which means it poses few risks to health and safety. Due to the pandemic, our team kept remote cooperation with our computers throughout the project on GitHub.

2.2 Engineering Professionalism

Engineering requires a very high degree of professionalism. In the practice of this project, our team strictly followed professional practice principles. This project includes planning, designing the system, composing ideas and code, and evaluating and testing the system.

Our group made a series of decisions about the project during the development process. By carefully analyzing and studying the different options, we ensured that these decisions did not affect or harm the public and team members. Also, during many team meetings, we discussed how to divide the project work equally based on each member's expertise field, making the team members work pretty in the development process.

As a group, each team member demonstrated professionalism. Through the different expertise learned by each team member, this project can be considered more comprehensively.

2.3 Project Management

Considering this project's scale and time management, our project follows a waterfall model. Our team listed this project's functional and non-functional requirements in the proposal. With the analysis of these requirements, our team could design the system structure with diagrams. We drew several diagrams like the class diagram and sequence diagram for functions to make a corresponding system design and start implementation.
After that, we start the implementation separately. We created a GitHub repository and cooperated through Git. One member worked to establish the database, and the others were responsible for implementing web-related

functions and frontend. Since this project is web-based, we decided to do the testing and deployment parallelly. One of our members took the responsibility of figuring out how to deploy the project, and the others kept doing the testing of the functions.

2.4 Individual Contributions

2.4.1 Project Contributions

Zijun Hu

In the project design, I designed the project's general structure, summarized the main requirement needed, and drew the project structure diagram.

In the project development process, I implement the flask structure, database code structure, whole authentication section, and part of the evidence and criteria related sections, including criteria management, evidence view, and evidence delete.

At the same time, I also wrote the readme documentation for the project.

Tiantian Lin

In the early stage, I designed the user case diagram, access control matrix, and the flow charts for each function, like upload, search, and view. During the development process, I was responsible for implementing the search function and designing the HTML page for the search service. In addition, I customized the project's page template using CSS.

Ruixuan Ni

In the design stage, I'm responsible for drawing some of the graphs used to specify the functions needed in the management system.

In the developing stage, I'm responsible for implementing part of the functionalities in the backend and the function-related pages in the frontend. I have implemented the upload function for evidence and criteria. I also cooperate with Tiantian Lin on decorating these web pages (the implementation of CSS files).

Jiawei Ma

During the design phase, I developed object diagrams to ensure the relationship between each object component within the system. I also created an ER diagram and a database schema diagram for good database design and implementation.

I am responsible for the database-related implementation during the implementation phase, including designing, creating, and building the database. I'm also responsible for implementing the backend logic to store or retrieve data from the database using Flask-SQLAlchemy and SQLite.

During the deployment phase, I am responsible for deploying the system to the service cloud platform Heroku. The deployment process includes configuring the server and the database so that they can be deployed to the Cloud. The deployment process also includes reconfiguring database settings, as Heroku only supports PostgreSQL cloud databases, unlike local database implementations that use SQLite.

2.4.2 Report Contributions

Zijun Hu

I was responsible for Section 1.4 Accomplishments, Section 4.3 Technical and Alternative Solutions, and Section 4.4.1 Router Logic.

Tiantian Lin:

I was responsible for Acknowledgement, Section 1 introduction, Section 3.2 Evidence for Security Evaluation, Section 4.4.2 HTML Templates.

Ruixuan Ni

I was responsible for the Abstract, Sections 1.1-1.3, 1.5 Organization of the Report, Section 2.1, 2.3, 2.4 in chapter 2, and Section 3.1, 3.3 in chapter 3.

Jiawei Ma

I was responsible for discussing Sections 4.1-4.2, Section4.4.3 Database, and Section4.4.4 User Case for Technical Solution.

Chapter 3

3. Background Literature Review

3.1 Criteria and Security Evaluation

Criteria are requirements for developers to secure information security. It lists rules that developers should follow while developing a security system. There are various criteria created and applied in multiple regions. For example, Common Criteria are widely used in western countries [3]. Common Criteria include requirements for developers in project design, implementation, documentation, risk analysis, etc.

Criteria give requirements and define the standard way for systems or products to be evaluated. For example, Common Criteria has seven Evaluation Assurance Levels (EALs) [3]. The higher the level, the more requirements the system or product satisfies in the Common Criteria. The process of evaluation is security evaluation. This certification aims to assure the customers that a third party has evaluated and verified the products they bought or used.

3.2 Evidence for Security Evaluation

The evidence can be collected during the development process. The evaluation combines the evidence that a system meets or does not meet [4]. The security evaluation is to see if the evidence meets the criteria and assures a system's security level. Therefore, evidence plays an essential part in security evaluation.

3.3 Projects Related

Developers have implemented various tools to manage their projects or do security testing automatically. GitHub is a widely used tool for project management during the development lifecycle, allowing developers to cooperate through Git. It has something in common with this project. It allows the uploading, management, and updating of files related to development throughout the lifecycle. It also has a mature log system which will enable developers to check the operations and rollback. However, it is a system that concentrates on cooperation in project development. It would be hard to manage the evidence file on GitHub since it does not support searching a particular file within the project or labeling any file. This makes this platform impossible to handle a large number of evidence files.

Chapter 4

4. Technical Solution

The security level of a system or product is evaluated based on the security evidence provided by developers. Thus, when companies plan to develop a new product, developers will document the evidence related to user security according to the criteria they should follow. Before a new product is put on the market, the Quality Analysts would inspect the evidence to verify the safety of the product.

Therefore, the main objective of our project is to allow developers to upload the evidence and allow Quality Analysts to inspect the evidence. And the following sections will state detailed information about the Security Evaluation System functionalities.

4.1 The System Requirements

To achieve the main objective that mentioned above, the system requirements need to be specified. The platform should have three roles for users: the administrator, the inspector, and the developer. The platform should store all evidence that is uploaded by developers according to different criteria. And after the evidence is uploaded to the platform, the platform should allow administrators and inspectors to inspect the evidence and approve or disapprove the evidence. Besides, authorized users will be able to search for related evidence by typing some searching keywords to help them find related evidence easily.

To ensure the security of stored text files, all existing evidence should only be accessed by authorized developers. Therefore, the platform should use the developer's username and password as identity verification to identify the developer's authority. And administrators should be able to have full management rights to all users' information, such as roles and credentials. The system should identify the users' roles and access rights once they log in. Moreover, unauthorized users will not be able to access and upload any evidence.

To increase the performance and usability of the platform, authorized developers can use this platform to manage text evidence anytime and anywhere. To reach this goal, we designed the platform to interact with developers through a web page. Developers can access the platform using any commonly used web browsers such as Chrome, Safari, Firefox, Microsoft Edge, etc.

4.2 The System Design and Configuration

In our project, there will be three kinds of roles in a company that uses our system: the Administrator, the Developer, and the Quality Analyst. All those users may use the system concurrently, and the server will handle multiple requests simultaneously. Therefore, our group used the client-server pattern to implement our platform.

We use web pages as the client-side of our platform because users can check, maintain, and add new criteria evidence through web pages at any time and anywhere. And the web pages are also accessible for our group to implement and add new features.

To satisfy the requirement of allowing users to use this platform anytime and anywhere, we chose to deploy the whole platform onto Heroku, a free cloud platform-as-a-service. The alternative plan is to deploy the database and server separately to the Amazon Web Services (AWS). Details of this alternative plan and a comparison between Heroku and AWS are discussed in Section 4.6.5, “Deploying the Platform to the Cloud”. Additionally, Heroku has a Heroku Postgres service that supports the online PostgreSQL database. Therefore, users can access the cloud server and corresponding data in the cloud database at any time.

4.3 Technical and Alternative Solutions

4.3.1 Client-Server Pattern

The pattern of the project is based on the Client-Server pattern. The Client-Server design pattern is a network architecture consisting of a server and several clients. It allows the server to provide service to multiple client components. In this pattern, users could interact with the client, the web page in our project. The client would pass these requests from users to the server. The server would be responsible for processing these requests and passing the result.

Our group built the platform with web pages as the user interface and databases to store evidence uploaded by developers and administrators. And the Quality Assurant can inspect those evidence and decide whether that evidence is eligible. This primary structure suits the Client-Server framework. Client-Server is also a design pattern widely used in web application development. As well as that, the target customer of this project will be companied. Due to the large number of users, the client-server pattern will also be the best fit.

Figure 1 Project Structure shows the project Structure, which illustrates the client, server, and database relationship.

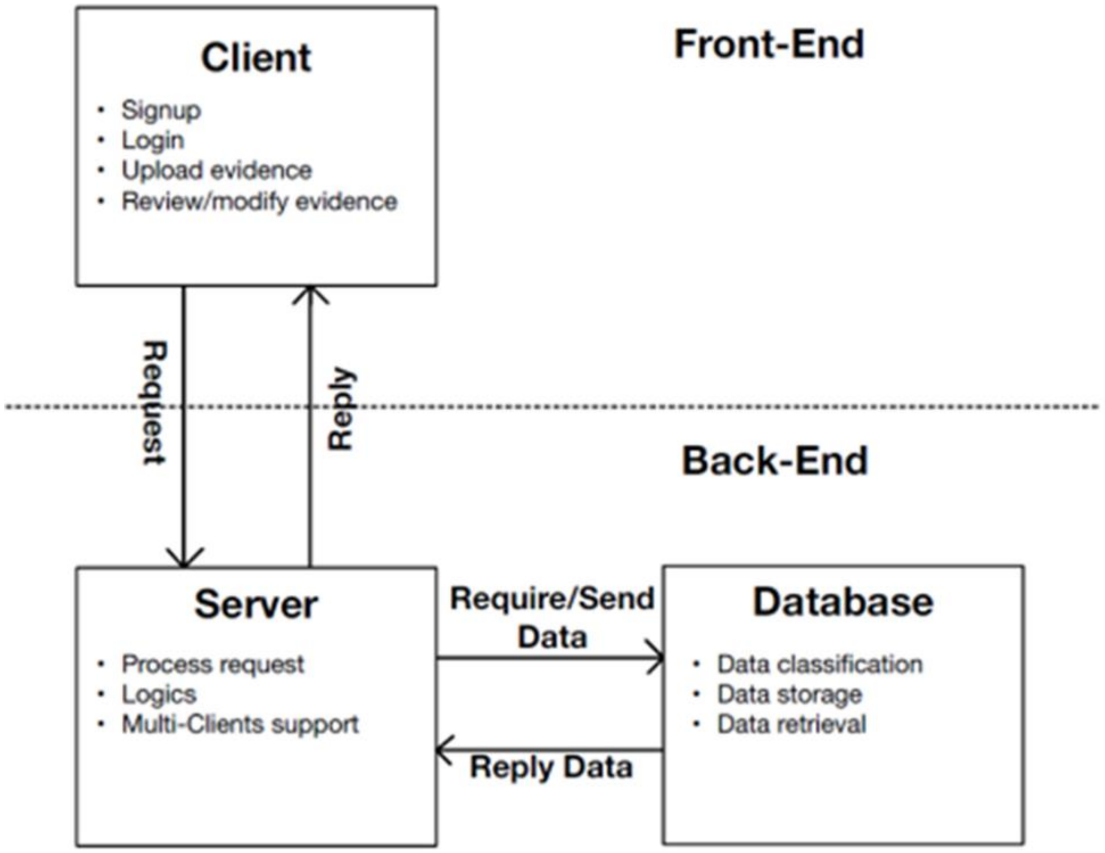


Figure 1 Project Structure

The project structure could be divided into two main sections: Front-end and Back-end. The primary purpose of the frontend is to interact with the user, allowing users to upload/view the evidence and display needed information. The backend includes a server that can simultaneously process requests and operations from different users and a database to categorize and store all involved evidence and information.

4.3.2 Web Application vs. Desktop Application

Graphical User Interface (GUI) is one form of user interface (UI) that allows users to interact with the program to perform needed commands [5]. The designer could easily control each pixel on the users' screen and the access direction using GUI. As well as that, the GUI can efficiently guide the user to view the information/pages the designer wants. Also, it can force the user to enter information before entering the next step by using the pop-up windows [6]. However, it makes users' operation flexibility lower. At the same time, it also takes longer for users to familiarize the system and operating methods.

A web interface is a kind of interface that requires the users to access via the web browser on their phone, computer, or tablet. Compared with traditional UI, it provides users more flexibility and is accessible to update. Users can access it through any smart device with a browser. At the same time, because the webpage itself is stored on the service provider's server, developers got chances to update the web page easily without asking users to download any updates packages.

After balancing the pros and cons, the web page format was selected for the project's frontend development due to the ease of updating and the flexibility in accessing.

4.3.3 Why Flask & Why Python?

Python is a high-level scripting language in programming with an extensive library. It has modules and packages which support functions like data management or network transmission [7]. Python also supports accessing databases like MySQL with a wrapper, simplifying data storage and transmission functions [8]. These functions could help our group build web pages as interfaces, establish databases to store evidence, and combine these different modules.

The team has also considered Java. Compared to Java, Python is a more readable programming language and easier to learn. Python is also more efficient than Java in development. Besides, all of our team members have essential experience in Python programming. Our group believes that Python is a programming language that is more suitable for our present project architecture and development plan.

After choosing Python as the programing language, the team decides to use Flask as the web application framework for this project. Flask is a microframework that provides basic features for a web application. By using Flask with Python, it helps developers to easily get started on and expand using Python's libraries and extensions.

4.4 Technical Detail

The structure for this project can be divided into three components,

- Router Logic, which includes the server logic in response to users' requests and router between different HTML pages;
- HTML Templates, which display needed information to users;
- Database, which stores all related data and provides it to the server when needed.

4.4.1 Router Logic

The router logic can be split into three main sections:

- Authentication, which maintains the users' session, supports login/signup, and provides users' profile management functionality;
- Evidence, which includes the server logic on evidence upload, search and view;
- Criteria, which provides criteria management-related features.

4.4.1.1 Authentication

The authentication section covers multiple features, including signup/login, user management, and profile management.

Login & Signup:

The login and signup feature imports the flask login library, which provides user session management for the Flask app. Users can sign up by providing their name, email, and any password. The email is required as the unique. The password is encrypted using SHA-256 with a 32-bit salt value. The SHA-256 is used because it is the secure and trusted industry standard. And the salt value is used to avoid the encrypting process generating the same hash due to the same password. There are also password rules to make sure the password is at least eight characters, including at least one lowercase, capital letter, and number.

The login mechanism will ask the current user to provide their email address and password. If the email exists and the password matches, the system will log the user in and maintain the login session for the current user.

Different User Roles and the Access Control Matrix:
Besides the login mechanism, the system also integrates the access control method. The access control method used is RBAC which is known as role-based access control. As mentioned before, the target customer of this project is companies, and for most companies, the employees are devised into different groups/roles. So, the team decided to use role-based access control to manage users. For now, there is a total of four different roles; they are,

- Developer, who mainly responds to upload and modify the evidence
- Administrator, who has full access to the system and can manage other users and manage criteria.
- Quality Analyst, who mainly responds to make approval to the evidence.
- User, who is regular users have no access to features.

The first registered user in the system will automatically be assigned to administer. The following registered user will be assigned as normal users. All router logic checks the user's role before redirecting to ensure the page is only displayed to users with the right access privileges. The same check also happens before interacting with the database.

Role/Source	**Criteria**	**Evidence**	**Users**
Developer	View	Create, Modify, Delete	N/A
Administers	View, Create, Delete	Create, Modify, Delete, Approval	Modify, Delete
Quality Analyst	View	Read, Approval	N/A
User	N/A	N/A	N/A

Table 1 Access Control Matrix

User Management Mechanism:
The user management mechanism provides user management services for the Admin users. Admin users can change other users' names and roles through the admin page. They can also delete users' accounts or reset other users' passwords via the page. Please notice that the admin users can only manage users other than themselves. This restriction ensures at least one admin user is in the system.

Profile Management
The profile management feature allows users to change their name and password. Using the profile section on the Welcome Page, users can interact with this feature.

4.4.1.2 Evidence
The following sections are related to evidence management functions, including evidence uploading and searching.

Upload
In terms of evidence management, we have implemented an upload form that allows developers and administrators to input the evidence name, description of evidence (optional), and the project name related to this evidence. Uploaders can choose criteria from the criteria list as a tag of this evidence and upload the evidence file. Currently, only text files are allowed to be uploaded as evidence.

Every time an uploader tries to upload a file, the system requests a list of criteria from the database to ensure that the version of the criteria list is up to date. Since we have not deployed the web to the Cloud yet, the upload function only saves uploaded files to the local directory and passes the local directory to the database. If a file is not successfully uploaded, the submit button will redirect the uploader to the upload page with a flash of the file not successfully uploaded. If evidence is successfully uploaded, the uploader would be redirected to the view page, which displays the detailed information of this evidence. The server will record the user id of the uploader and pass this information to the database. The database then will record the time of evidence upload and the last edit time of the evidence. Each evidence uploaded to the database would have a specific evidence id which is automatically generated.

Search
In terms of searching for security evidence, we have added several filters to make it easier for users to search for the evidence they need. In the current version of the management platform, under the search page, we provide users the following search conditions: ID, criteria, project name, uploader, creation time, and last modification time. After users enter those searching conditions, the server will record those conditions and pass those filters to the database to search. Then, the database will return the search result to the server. The HTML page then reads the search results for the server page and displays the results as a list with the evidence name and the corresponding URL link. When we click the link, we will be redirected to a view page to see the details of the chosen evidence.

4.4.1.3 Criteria
Criteria are necessary for evidence management. Only Admin Users can upload criteria in the current system, including the name and description. The system would record the uploader’s user id to ensure the activities are traceable. The database would also generate a criteria id for each criteria uploaded. Criteria uploaded would be saved within the database and used as tags as evidence.

4.4.2 HTML & CSS Templates
In Flask, templates are files containing both static and dynamic data. The template is rendered with specific data and presents the result [9]. In this project, the template is the base.html. All the other pages are developed based

on the base.html. In the current base.html, we added our own designed logo, copyright, and menu bar, which could direct users to other pages.

In addition, we used CSS to customize our web page to improve the user experience. We chose the same color as our designed logo to make the entire program look consistent.

There are four kinds of pages in the current management security platform: initial pages, evidence-related pages, authentication-related pages, and criteria manage pages.

4.4.2.1 Initial Pages

Initial pages include base.html and welcomepage.html.

base.html

The base.html is the initial page that every other page extends from it. In base.html, we check users' roles and show different available options based on their roles' permission. As shown in Figure 2 base.html - Menu bar, there are six options, Admin, Upload, Search, Criteria, and Logout.

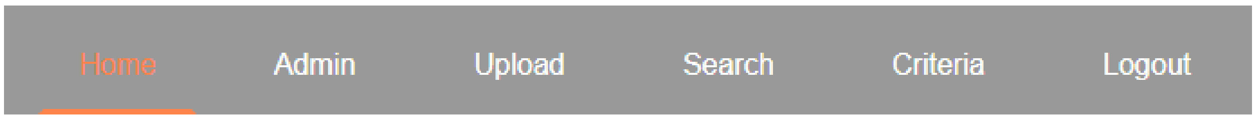


Figure 2 base.html - Menu bar

welcomepage.html

The welcomepage.html welcomes and directs users to the login page. After the user login, this page will also display the user's profile. Figure 3 welcomePage.html - Before Login and Figure 4 welcomepage.html - After Login shows the welcome page before and after login.

Before login, users could click either the orange button below the page title or the login option on the menu bar to log in.

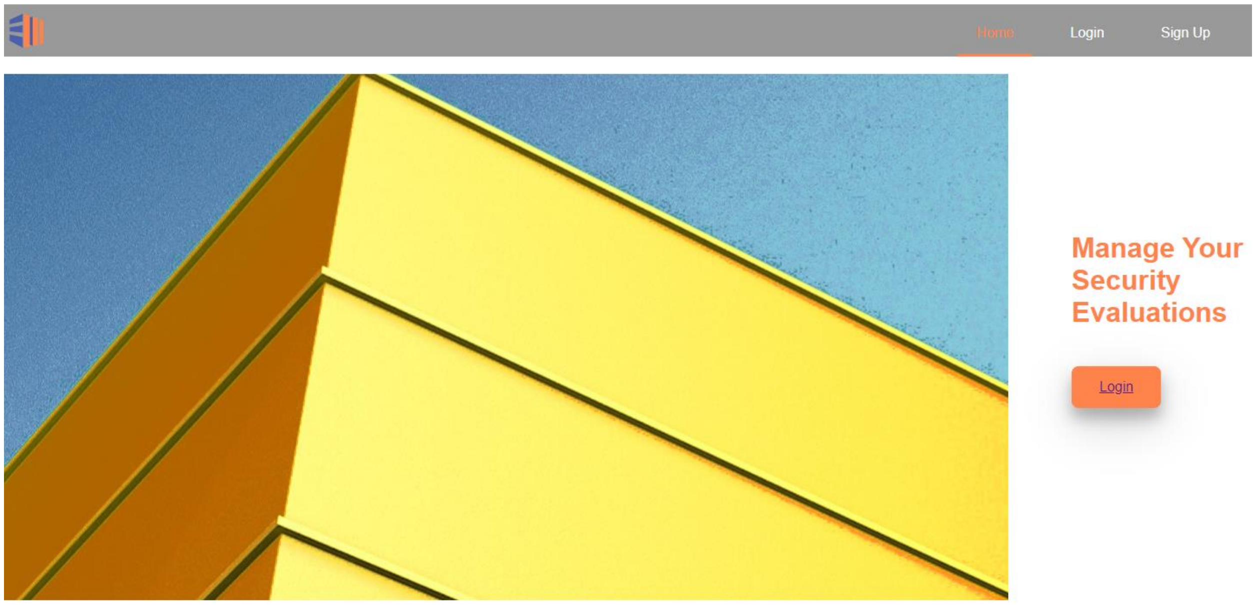
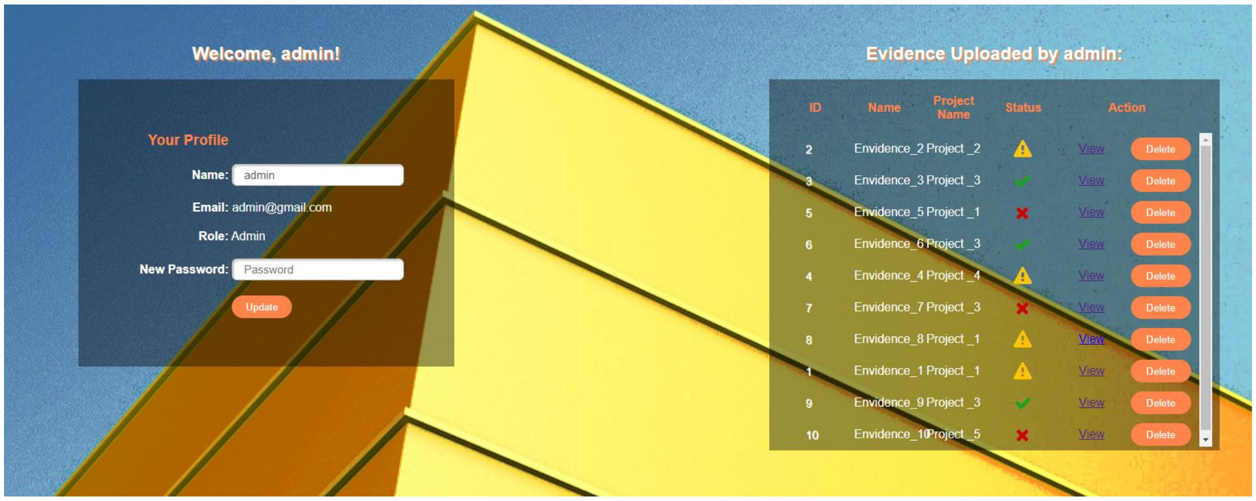


Figure 3 welcomePage.html - Before Login

After Login, the welcome page will show the user's profile with their name, email address, and the corresponding role. This page also provides a service for users to change their passwords. In addition, on the right side of the profile section, there is a form that shows the uploaded evidence by this user.


Welcome, admin!

Your Profile

Name: admin

Email: admin@gmail.com

Role: Admin

New Password: Password

Evidence Uploaded by admin:

ID	Name	Project Name	Status	Action	
2	Envidence_2	Project _2	⚠	View	Delete
3	Envidence_3	Project _3	✓	View	Delete
5	Envidence_5	Project _1	✗	View	Delete
6	Envidence_6	Project _3	✓	View	Delete
4	Envidence_4	Project _4	⚠	View	Delete
7	Envidence_7	Project _3	✗	View	Delete
8	Envidence_8	Project _1	⚠	View	Delete
1	Envidence_1	Project _1	⚠	View	Delete
9	Envidence_9	Project _3	✓	View	Delete
10	Envidence_10	Project _5	✗	View	Delete

Figure 4 welcomepage.html - After Login

4.4.2.2 Evidence Related Pages

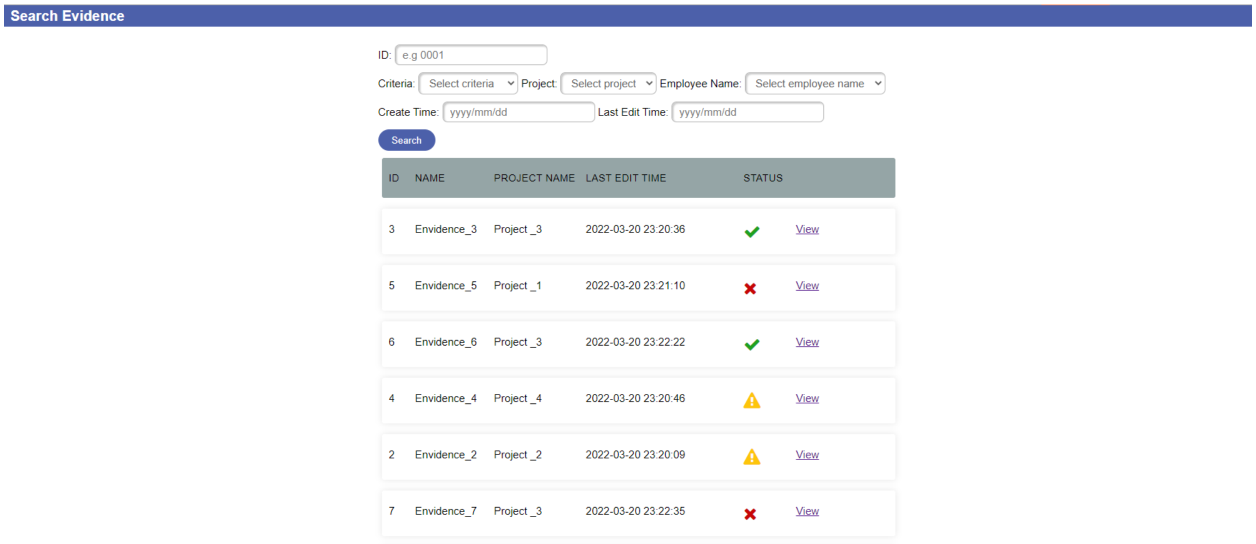
Evidence-related pages include search.html, upload.html, and view.html.

search.html

The search.html is used for users to search the security evidence. This page contains the dropdown boxes and the input boxes for users to apply filters to evidence tags when searching.

After they click the search button, the search result will display its name, id, project name, last edit time, and the corresponding URL link to the view evidence page, view.html.

We chose a table to list the searched items in the page custom design. This form of expression allows users to have a more intuitive understanding of evidence, including information about each piece of evidence.


Search Evidence

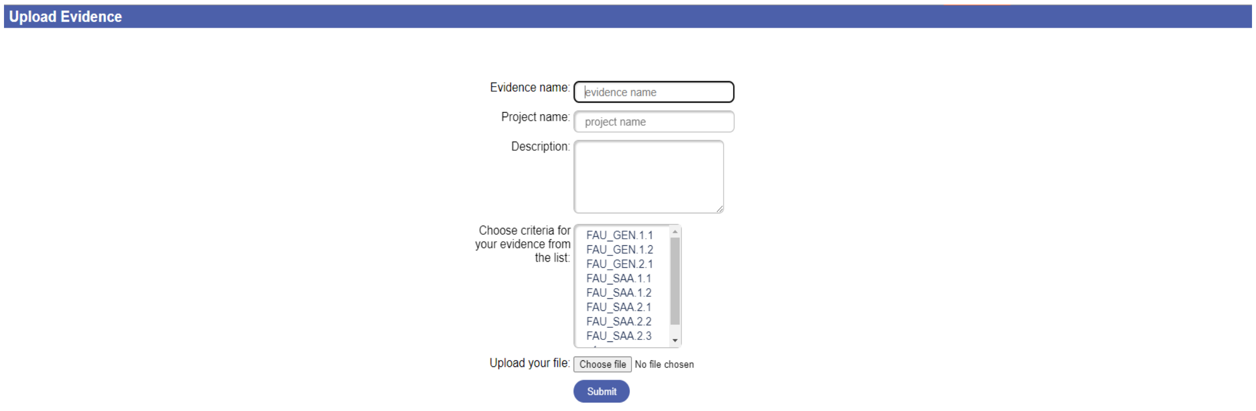
ID: e.g 0001
Criteria: Select criteria Project: Select project Employee Name: Select employee name
Create Time: yyyy/mm/dd Last Edit Time: yyyy/mm/dd
Search

ID	NAME	PROJECT NAME	LAST EDIT TIME	STATUS	
3	Envidence_3	Project _3	2022-03-20 23:20:36	✓	View
5	Envidence_5	Project _1	2022-03-20 23:21:10	✗	View
6	Envidence_6	Project _3	2022-03-20 23:22:22	✓	View
4	Envidence_4	Project _4	2022-03-20 23:20:46	⚠	View
2	Envidence_2	Project _2	2022-03-20 23:20:09	⚠	View
7	Envidence_7	Project _3	2022-03-20 23:22:35	✗	View

Figure 5 search.html

upload.html

The upload.html is for users to upload the security evidence. By entering the related information such as evidence name, project name, description, and criteria, users could create new evidence and save the new evidence to the database. After the evidence has been uploaded successfully, the page will show the message.

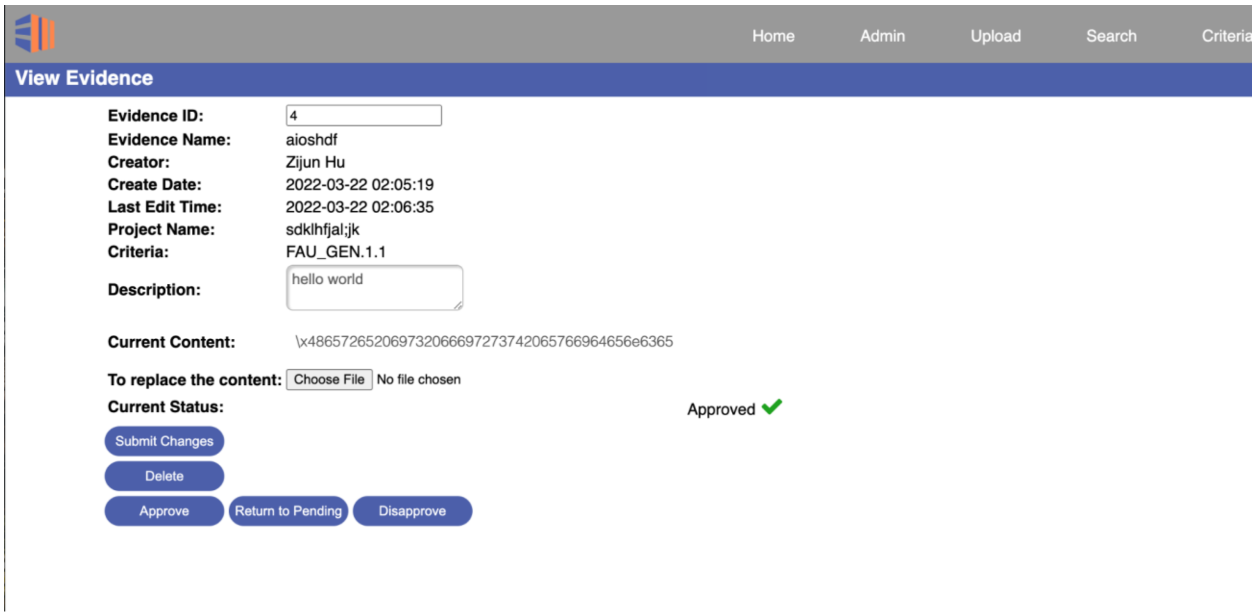

Upload Evidence

Evidence name:	evidence name
Project name:	project name
Description:	
Choose criteria for your evidence from the list:	FAU_GEN.1.1 FAU_GEN.1.2 FAU_GEN.2.1 FAU_SAA.1.1 FAU_SAA.1.2 FAU_SAA.2.1 FAU_SAA.2.2 FAU_SAA.2.3
Upload your file:	Choose file No file chosen

Figure 6 upload.html

view.html

The view.html is for the user to view the detail of the specific evidence. Currently, the view page only supports the text content. If a user wants to update the content of this evidence, they could upload a new file to cover the original file.



View Evidence

Evidence ID:	4
Evidence Name:	aioshdf
Creator:	Zijun Hu
Create Date:	2022-03-22 02:05:19
Last Edit Time:	2022-03-22 02:06:35
Project Name:	sdklhfjal;jk
Criteria:	FAU_GEN.1.1
Description:	hello world
Current Content:	\x48657265206973206669727374206576696465656e6365
To replace the content:	Choose File No file chosen
Current Status:	Approved ✓

Figure 7 view.html

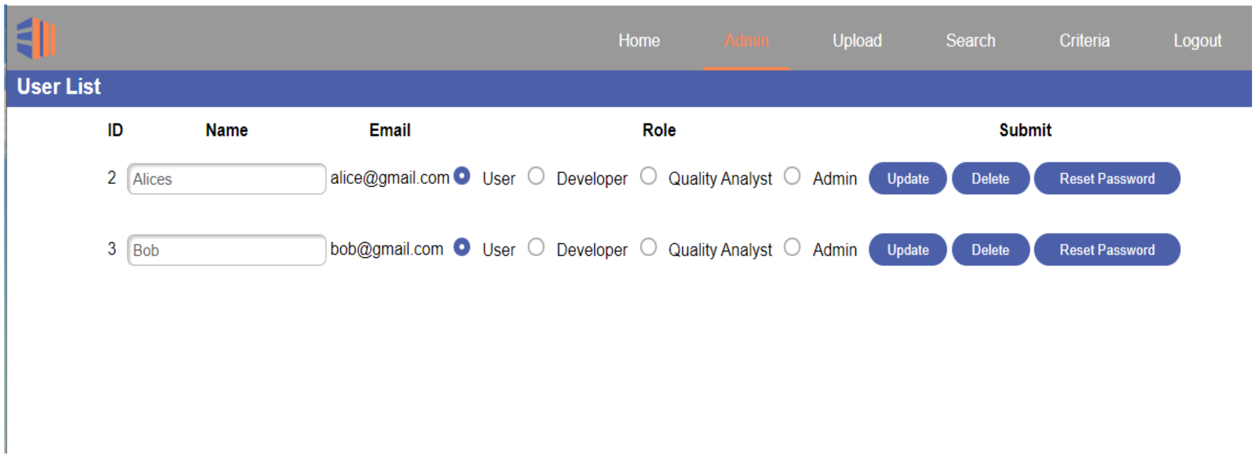
4.4.2.3 Authentication Related Pages

Authentication-related pages include login.html, signup.html, and admin.html.

login.html
The login.html allows users to log in to the system by entering the correct username and password. Each email address is unique. If a user enters an email address that does not exist in the database, the user will be redirected to the signup page.

signup.html
The signup.html is for users to register to the system. By entering the username, email and password, a user will be registered.

admin.html
The admin.html is for Admin users to manage other users. In the current version, the first user has the administrator role. The Admin user could assign different roles to later registered users.

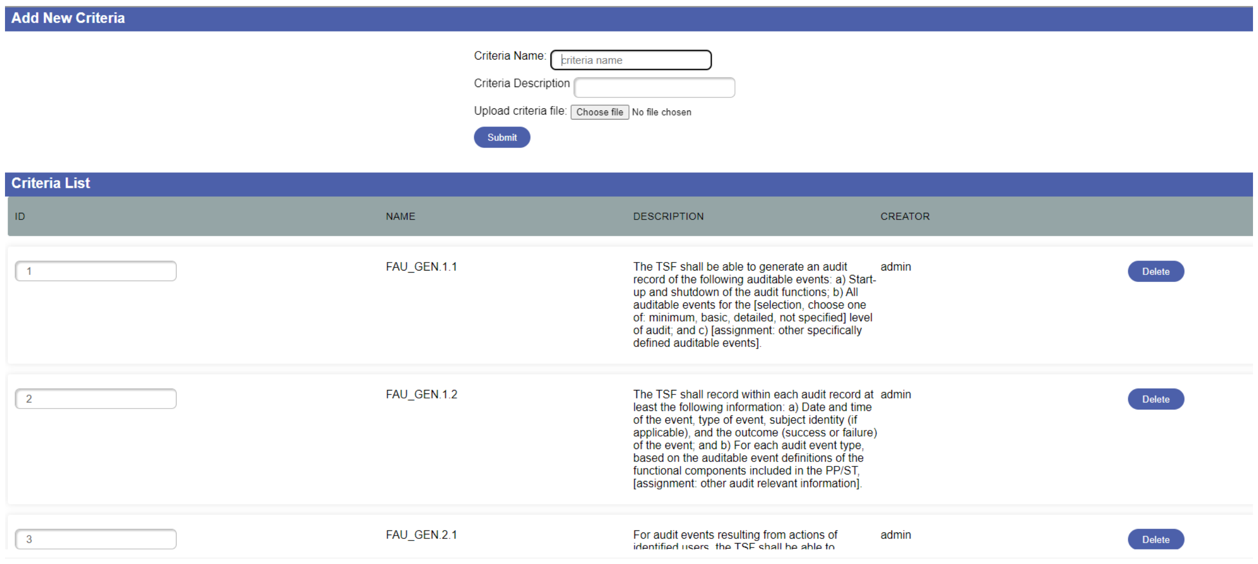


User List

ID	Name	Email	Role	Submit
2	Alices	alice@gmail.com	[✓ User] Developer Quality Analyst Admin	Update Delete Reset Password
3	Bob	bob@gmail.com	[✓ User] Developer Quality Analyst Admin	Update Delete Reset Password

Figure 8 admin.html

4.4.2.4 Criteria Management Page - criteria.html
The criteria.html is for users to create new criteria. The new criteria could be created and stored in the database by entering the name and description. The created criteria will then be shown on the page with its name, creator, unique id, and description. Admin users could also delete the criteria by clicking the delete button.



Add New Criteria

Criteria Name: criteria name

Criteria Description

Upload criteria file: Choose file No file chosen

Submit

Criteria List

ID	NAME	DESCRIPTION	CREATOR	
1	FAU_GEN.1.1	The TSF shall be able to generate an audit record of the following auditable events: a) Start-up and shutdown of the audit functions; b) All auditable events for the [selection, choose one of: minimum, basic, detailed, not specified] level of audit; and c) [assignment: other specifically defined auditable events].	admin	Delete
2	FAU_GEN.1.2	The TSF shall record within each audit record at least the following information: a) Date and time of the event, type of event, subject identity (if applicable), and the outcome (success or failure) of the event; and b) For each audit event type, based on the auditable event definitions of the functional components included in the PP/ST, [assignment: other audit relevant information].	admin	Delete
3	FAU_GEN.2.1	For audit events resulting from actions of identified users, the TSF shall be able to	admin	Delete

Figure 9 criteria.html

4.4.3 Database

There were several technical solutions for the database in different periods. And details will be explained in the following subsections.

4.4.3.1 Design of the database

Figure 10 Database Schema is the database schema diagram to illustrate the relationship between the four tables.

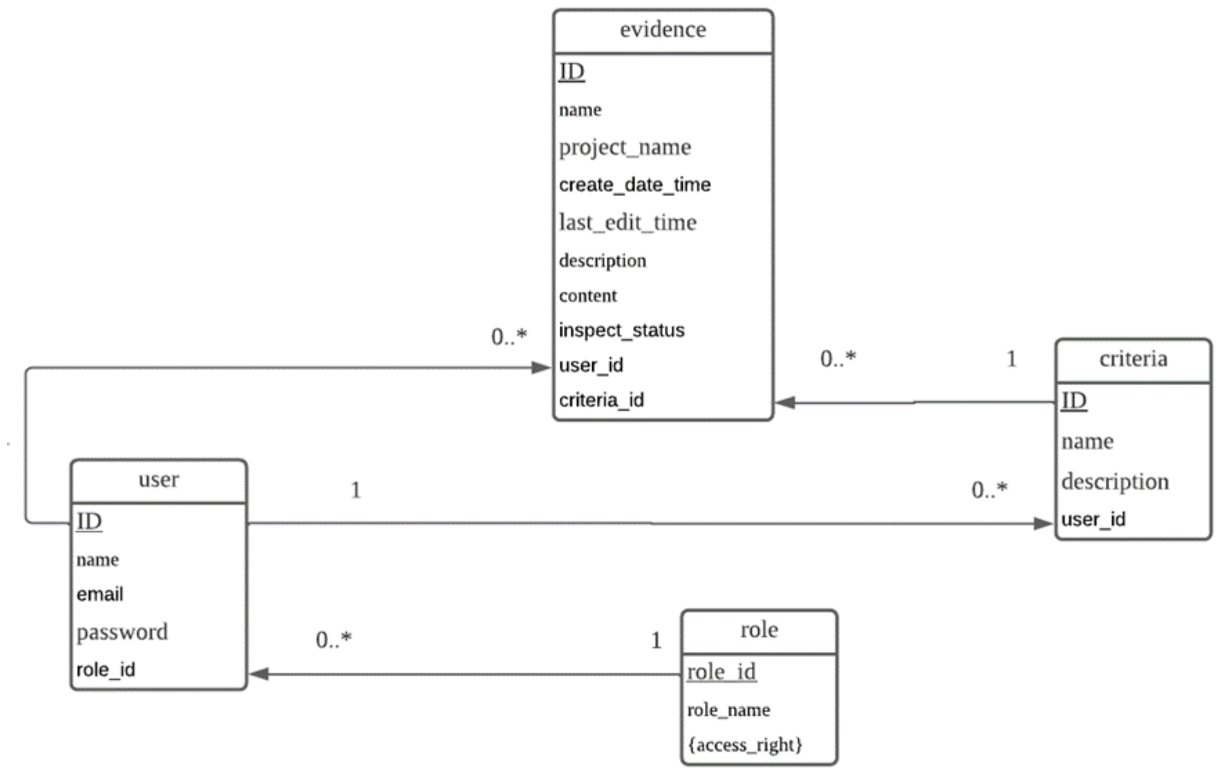


Figure 10 Database Schema

There are four object-oriented classes: user, role, evidence, and criteria.

- The user table stores the user-related data such as the user's id, name, password, email, and role type.
- The role table stores the role's id and name. The roles' access rights are coded in a python file rather than in the database. When a user is redirected to another web page, the program will repeatedly retrieve the access control matrix to find the correct data. Thus, this design would increase the speed when searching a user's access right.
- The evidence table stores the evidence's id, name, the creation date and time, the latest modified date and time, description, URL, and project name. It also stores the developer's user id who uploads the evidence and the criteria's id to which the evidence belongs to.
- The criteria table stores the criteria's id, name, description, and the admin's user id who create the criteria.

The Flask-SQLAlchemy uses object-relational Mapping (ORM) to create tables and manage the data in the database. The relationship between each object/table should be declared. For example, each user can only have one role, and each role can contain many users. Therefore, the relationships between role and user, user and evidence, user and criteria, and criteria and evidence are all one-to-many relationships. The inverse relationships are many-to-one.

Several methods support the backend logic to store and retrieve data in the database, such as storing, updating, querying, and deleting the information in four tables. When querying the related data, we applied an interface SQLAlchemy to query the desired data using raw SQL.

4.4.3.2 Object Relational Mapping vs. Plain SQL Database

There are two types of databases: Object-Relational Mapping (ORM) and the Plain SQL Database.
The ORM supports webpage development. It can store lightweight data, and these light data can be stored and retrieved quickly. It does not use plain SQL to interact with the database. Instead, ORM provides a way to interact with the database using object-oriented language to facilitate database development. However, the ORM syntax would become complicated when dealing with multi-table join queries and queries with multiple and complex conditions.

A plain SQL database is used to manage the heavyweight database. Plain SQL databases such as SQLite and PostgreSQL use raw SQL statements to create tables and store and retrieve data in the tables. The developers can fully control the plain SQL database by using raw SQL statements.

Although the plain SQL database is well-used in a heavyweight database, and we can have full control of all data in the database, we ended up choosing ORM. We use ORM to structure and implement our database because Flask-SQLAlchemy has APIs to support the ORM directly. It is also because there is no solution to adapt the plain SQL database with the Flask-SQLAlchemy.

To overcome the disadvantage of ORM, which can be complicated to handle multi-table join queries and the queries with multiple and complex conditions, we found a way in SQLAlchemy to execute the query in raw SQL statements. At present, we applied the way into most of the queries such as query all criteria id and name, query the evidence info by the given filters from backend with the information in the tables of criteria, evidence, and user. By overcoming the disadvantage of ORM, we can easily implement the query methods and support the backend logic better.

4.4.3.3 SQLite

As mentioned above, the Flask-SQLAlchemy components can be used for database operations. For the local implementation (before deploying to the Cloud), SQLAlchemy and Python support using the SQLite database. Therefore, the local implementation of each feature used SQLite database to store and operate data.

SQLite provides a lightweight disk-based database, and it doesn't require a separate server process. SQLAlchemy also allows accessing the database using both raw SQL language and a nonstandard variant of the SQL query language [10]. The implementation process does not need to install any other software to check the stored data for feature testing by using SQLite. And the data can be easily invoked through the PyCharm IDE.

4.4.3.4 PostgreSQL

Although SQLite is ideally for implementing the new features locally, Heroku, which is the cloud platform for deploying the whole system, does not support the SQLite database. Otherwise, the entire database will be lost at least once every 24 hours [11]. This is because the SQLite database will be stored as a separate file with the extension name of Sqlite3. However, Heroku does not support updating individual files every second. Therefore, we connect the ORM database to the Heroku Postgre database to allow the database to run in the Cloud along with the server.

4.4.3.5 Deploying the Platform to the Cloud

When developing the platform, we use a local server and a local database for ease of implementation. After we have developed the platform's main features, we need to deploy the server and database to the Cloud. So that users can access the platform anytime, anywhere, using their own devices.

There are two alternative plans for deploying the platform to the Cloud: Heroku and Amazon RDS.

Amazon Web Services (ASW) has Amazon RDS and Amazon Elastic Compute Cloud (Amazon EC2) as cloud web services. RDS provides the cloud relational database service and is compatible with MySQL, PostgreSQL, MariaDB, and so on. Amazon EC2 provides cloud server services that allow local servers to be deployed to the Cloud. In conclusion, AWS provides a powerful cloud platform-as-a-service.

However, AWS charges are expensive beyond our budget. Besides, the setup processes are really complicated due to their powerful features. Therefore, we found another alternative which is Heroku. "Heroku runs on Amazon Web Services (AWS). AWS is an Infrastructure as a Service (IaaS) provider, meaning they are responsible for managing large, shared data centers." [12]. It does not have complicated configurations on both cloud servers and cloud databases. When the Heroku project connects to the GitHub project, the program is automatically deployed to Heroku and can be run on the web page.

4.4.3.6 Heroku Cloud Server and Database

We used Heroku to run the implemented server and database in the Cloud. Because Heroku can provide the online service for our implemented security platform anytime and anywhere for the users, we choose to deploy our project onto Heroku and reconfigure the database to make it connect to the Heroku Postgres component.

Heroku also has continuous integration and delivery pipelines (CI/CD). It can connect to the GitHub repository. When the new features are implemented, tested, and deployed to the main branch, the Heroku CI will automatically build the new updated version and deploy them to the Cloud. Heroku CI lets us easily manage the CI/CD automation across our security evaluation platform.

4.4.4 User Case for Technical Solutions

Figure 11 User Case Diagram illustrates the user case of the platform technical solutions for four roles of users: Developers, Administrators, Quality Analysts, and Normal Users.

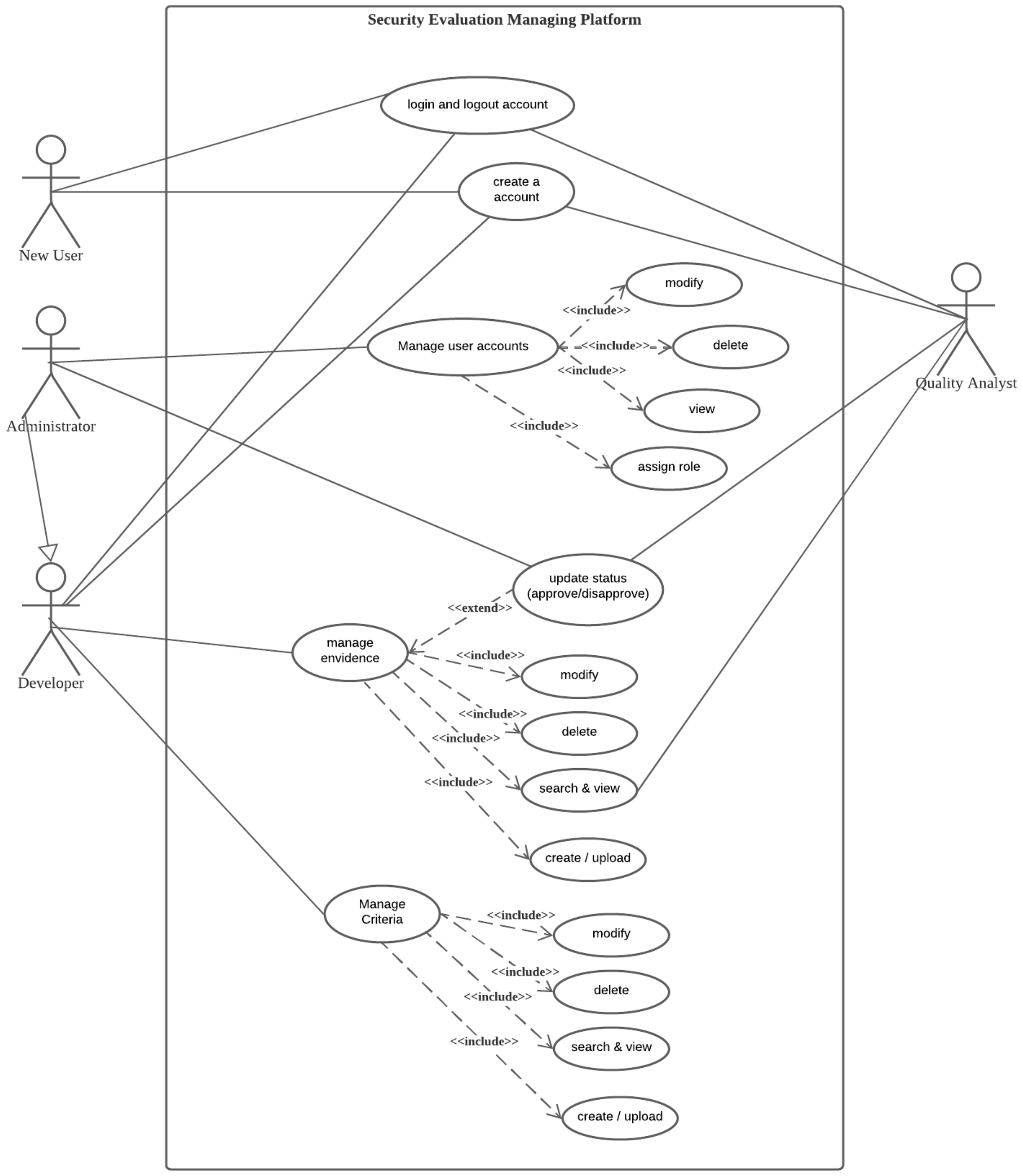


Figure 11 User Case Diagram

Chapter 5

5. Conclusions

Due to people’s reliance on the information, the system’s security has become a top priority for development companies. To perform a security evaluation on a system, we need to gather evidence to see if a system meets security criteria. However, during the development of a system, much evidence will be generated. Therefore, we need an evidence management system to store and upload these evidence files so that the evaluator can evaluate the system’s security and review the evidence later.

Based on this problem, our team designed a security evidence management system to help developers upload, search, and manage evidence. At the same time, the evaluator can also approve the evidence which meets the criteria.

At the end of the semester, we successfully designed a web-based application. This program realizes the storage and management of evidence. It can provide users with registration and login functions. At the same time, the platform gives different roles to different users, such as developer, QA, and administrator. In addition to evidence management, the platform can also manage criteria and change the evidence status. Currently, the platform has been deployed to the Cloud.

However, there are still several extensions that future developers could make. The system could be extended to support all format files on evidence and criteria upload. As well as that, evidence of online viewing, evidence modification history, and rollback will also be pretty useful features to have.

References

[1]"Evaluation Criteria of Systems Security Controls - dummies", Dummies.com, 2022. [Online]. Available: https://www.dummies.com/article/academics-the-arts/study-skills-test-prep/cissp/evaluation-criteria-systems-security-controls-254878/. [Accessed: 12- Apr- 2022].

[2]Commoncriteriaportal.org, 2022. [Online]. Available: https://www.commoncriteriaportal.org/files/ccfiles/CCPART1V3.1R5.pdf. [Accessed: 12- Apr- 2022].

[3]"The Common Criteria | CISA", Cisa.gov, 2022. [Online]. Available: https://www.cisa.gov/uscert/bsi/articles/best-practices/requirements-engineering/the-common-criteria. [Accessed: 12- Apr- 2022].

[4]Cl.cam.ac.uk, 2022. [Online]. Available: https://www.cl.cam.ac.uk/~rja14/Papers/SEv2-c26.pdf. [Accessed: 12- Apr- 2022].

[5]"What is user interface (UI)? Definition from SearchAppArchitecture", SearchAppArchitecture, 2022. [Online]. Available: https://www.techtarget.com/searchapparchitecture/definition/user-interface-UI. [Accessed: 13- Apr- 2022].

[6]"The Difference Between Web Design and GUI Design", Nielsen Norman Group, 2022. [Online]. Available: https://www.nngroup.com/articles/the-difference-between-web-design-and-gui-design/. [Accessed: 13- Apr- 2022].

[7]"The Python Standard Library — Python 3.10.4 documentation", Docs.python.org, 2022. [Online]. Available: https://docs.python.org/3/library/. [Accessed: 13- Apr- 2022].

[8]A. Lukaszewski, MySQL for Python. Olton Birmingham England: Packt Pub. Ltd., 2010.

[9]"Templates — Flask Documentation (2.0.x)", Flask.palletsprojects.com, 2022. [Online]. Available: https://flask.palletsprojects.com/en/2.0.x/tutorial/templates/#:~:text=Templates%20are%20files. [Accessed: 13- Apr- 2022].

[10]"sqlite3 — DB-API 2.0 interface for SQLite databases — Python 3.10.4 documentation", Docs.python.org, 2022. [Online]. Available: https://docs.python.org/3/library/sqlite3.html. [Accessed: 13- Apr- 2022].

[11]"SQLite on Heroku | Heroku Dev Center", Devcenter.heroku.com, 2022. [Online]. Available: https://devcenter.heroku.com/articles/sqlite3. [Accessed: 13- Apr- 2022].

[12]S. Rostad, "What is Heroku? A Simple Explanation for Non-Techies", TriFin Labs, 2022. [Online]. Available: https://trifinlabs.com/what-is-heroku/. [Accessed: 13- Apr- 2022].