A Platform for Managing Security Evaluations

Final Report (Draft)

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: February 28th, 2022

Abstract

Information technology is widely applied in modern lives, so information security is valued more. To prove their system's security level, 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 proof with criteria tags and project tags. So it is easier for developers to view and manage the evidence they have. This system also allows evaluators to check the proof and mark them as satisfied with the criteria or not. It can simplify the process of security evaluation.

Acknowledgements

We received help from respected persons who deserve our most incredible gratitude in developing this project. The completion of this project gives us so much confidence and pleasure. We would like 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, thanks to Professor Mostafa Taha, who gave us suggestions for our progress report. We also thank Carleton University for consent to include copyrighted pictures as a part of our final report.

Finally, thanks to all the members of this project group. Your efforts have brought this project from conception to realization.

did you get this consent? ↳ Let's discuss

Contents

List of Figures

↑ more detailed captions will be helpful

List of Tables

List of Abbreviations

VoIP	Voice over Internet protocol
MRI	Magnetic resonance imaging

REVISE

Chapter 1: Introduction

People are becoming increasingly dependent on information technology 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 are made to evaluate those security mechanisms for computer security certification.

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.

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, Information technology is deeply involved in people's daily life. People store their data in the cloud or allow applications to access their private information conveniently. Under this circumstance, the security of these systems is in the spotlight. 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 [1]. 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 should provide proper evidence for security evaluators and regulatory authorities to evaluate their system security.

this section is a bit repetitive from above

the introductory part motivates the problem better than this

1.2 Problem Statement

The evidence provided by 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 of different evidence. It is also hard for evaluators and regulatory authorities to manage certification of different evidence and get the general evaluation result.

clearly state the problem
↳ we need a way to ...

1.3 Proposed Solution

you already did → check tense

We plan to implement a system to manage evidence to solve this problem. This system should provide service 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.

a better summary of your solution is needed

1.4 Accomplishments (INCOMPLETE)

List your accomplishments towards your solution. Remember, this is crucial information that will be used to determine your final grade. Do not make the reader sift through the entire report to determine what you actually completed.

Sometimes, a project does not completely solve the stated problem. This might be due to unexpected technical problems or perhaps an initial under-estimation of the amount of work involved. If this is the case, be sure to point this out.

1.5 Organization of the Report (INCOMPLETE)

Let the reader know what information is included in the report, and where the information is located. Give a brief statement of why the information has been included so the reader will be better prepared to relate the information to the BIG 3 questions:

1. What is the problem being solved?
2. What is your solution to the problem?
3. What did you accomplish?

Chapter 2: The Engineering Project

2.1 Health and Safety

This project is developing a web-based system, which poses few risks to health and safety. Due to the pandemic, our team kept remote corporation with our computers throughout the project on GitHub. The supervisor has a meeting with our team per week to know our progress and problems met.

→ this is not related to H&S

↳ what about H&S risks to you as engineers?

2.2 Engineering Professionalism

Engineering requires a very high degree of professionalism. It is a process that requires methods and logical thinking to solve problems. In the practice of this project, our team strictly followed the software engineering development processes. Those processes include the determination of software requirements and the form of proposals; numerous research and analyses are carried out before the project begins to design; interim progress reports; and systematic development and testing.

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.

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

This should not be related to SE processes but instead relevants aspects from your professional practice course

2.3 Project Management

The development of this 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. One member worked to establish the database, and the others were responsible for implementing web-related functions and frontend. Due to the emergency time, the testing and deployment will proceed parallel. One of our members took the responsibility of figuring out how to deploy the project; the others kept testing 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 design 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 in the backend and designing the corresponding HTML page for the search service. In addition, I customized the project's page template using CSS and made modifications for some frontend pages.

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 corporate 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 developed 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

deploying the database 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

Ruixuan Ni:

I am responsible for abstract, section 1.1, 1.2, 1.3 in introduction and section 2.1, 2.3, 2.4 in chapter 2.

Zijun Hu:

I am mainly responsible for the report's organization section (1.5) and the Flask section (4.3, 4.4) in the technical solution.

Jiawei Ma:

In this final report, I am responsible for discussing the technical solutions and details in Section 4, including the system design and configuration in Section 4.1, the Client-Server part in Section 4.2, and the database section in Section 4.6.

Tiantian Lin:

I am responsible for acknowledgement, introduction, section 2.2 in chapter 2 and HTML & CSS templates in section 4.5.

Chapter 3: Background Literature Review (INCOMPLETE)

Provide only the background information needed to clarify the problem. This might include details of the problem, an analysis of related work, and a clear statement of terminology. Any terminology introduced here should relate to the problem, not your solution.

Chapter 4: Technical Solutions and Details

The following sections contain detailed information about the Security Evaluation System functionalities.

4.1 The System Design and Configuration

In our project, there will be three kinds of roles in a company that use 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.

We also chose to deploy the whole platform onto Heroku, a cloud platform as a service. It can provide online services for all users. Besides, 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.

More details will be explained in the following sections.

4.2 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 those evidence are eligible or not. This primary structure suits the Client-Server framework. Client-Server is also a design pattern widely used in web application development.

* all of Section 4.1 is difficult to understand without the system requirements.

Figure 1

The following figure shows the project Structure, which illustrates the relationship between the client, server, and the database:

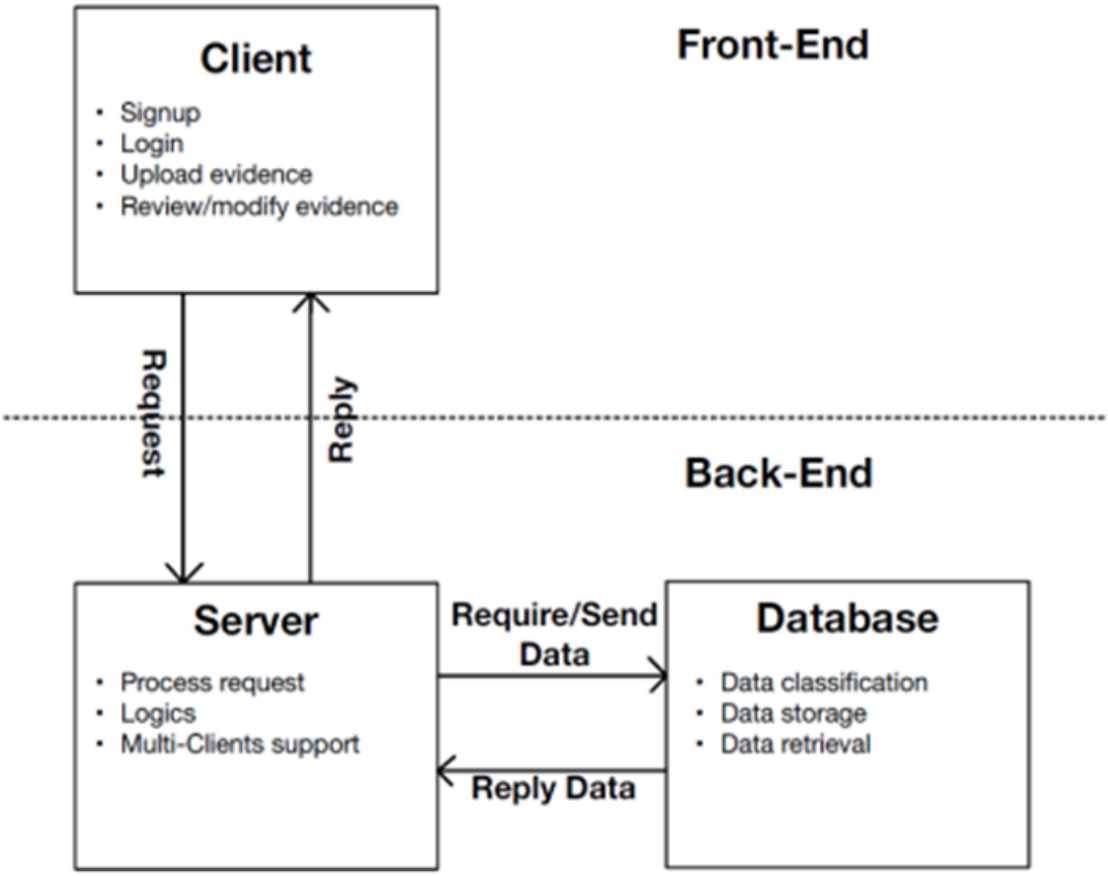

Figure 1 project structure

The server can process requests and operations from different users concurrently. After the server and database are deployed to the cloud, all users can do their operations at any time and anywhere. And it would increase accessibility and usability.

4.3 Flask

what are the alternatives?

We choose Flask to implement our platform because Flask is a lightweight micro web framework in Python and supports a strong third-party library. Flask uses Flask-SQLAlchemy for database operation, Flask-WTF for authentication forms, Flask-Login for login management, Flask-Mail for sending and receiving emails, etc.

The flask structure can be divided into three components,

i) Router Logic, which includes the server logic on response to users' requests and router between different HTML pages, which will be explained in detail in Section 4.4 Router Logic

ii) HTML & CSS Templates, which display needed information to users, which will be explained in detail in Section 4.5 HTML & CSS Templates

iii) Database stores all related data and provides it to the server when needed, which will be explained in detail in Section 4.6 Database.

4.4 Router Logic

The router logic can be split into three main sections:

i) Evidence, which includes the server logic on evidence upload, search, and view

ii) Authentication, which maintains the users' session, supports login/signup and provides users' profile management functionality

iii) Criteria, which provides criteria management-related features

4.4.1 Evidence

The following sections are functions directly related to evidence management, 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 with 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.

View（Incomplete）

4.4.2 Authentication

you need to explain why all of this is needed!

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 and provides user session management for the Flask app. Users can sign up by providing their name, email, and password. The email is required as the unique. The first registered user will be assigned as the Admin user to ensure the system has at least one Admin user. The password check mechanism now allows any type of password which is not ideal, and some password rules will be developed and added soon. More detail will be provided later in the coming section: *Secure Password*.

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:

Each user is assigned a role in the system to differentiate user privileges and support an access control matrix (ACM). 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. *Modify: View + Change

Table 1 ~~Access Control Matrix~~ listed the detailed privileges for each type of role.

Roles\Source	Criteria	Evidence	User Management
Developer	View	Modify*/Create/Delete	N/A
Admin	View/Create/Delete	Modify*/Create/Delete	Modify/Delete
Quality Analyst	View	View	N/A
User	N/A	N/A	N/A

Table 1 Access Control Matrix

Modify: View + Change

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 in the Welcome Page, users can interact with this feature.

4.4.3 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 unique id for each criteria uploaded. Criteria uploaded would be saved within the database and used as tags to evidence.

4.5 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 [3]. 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 colour 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.

4.5.1 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. In

general, there are four options, as Figure 2 base.html - Menu bar shows: Home, 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.

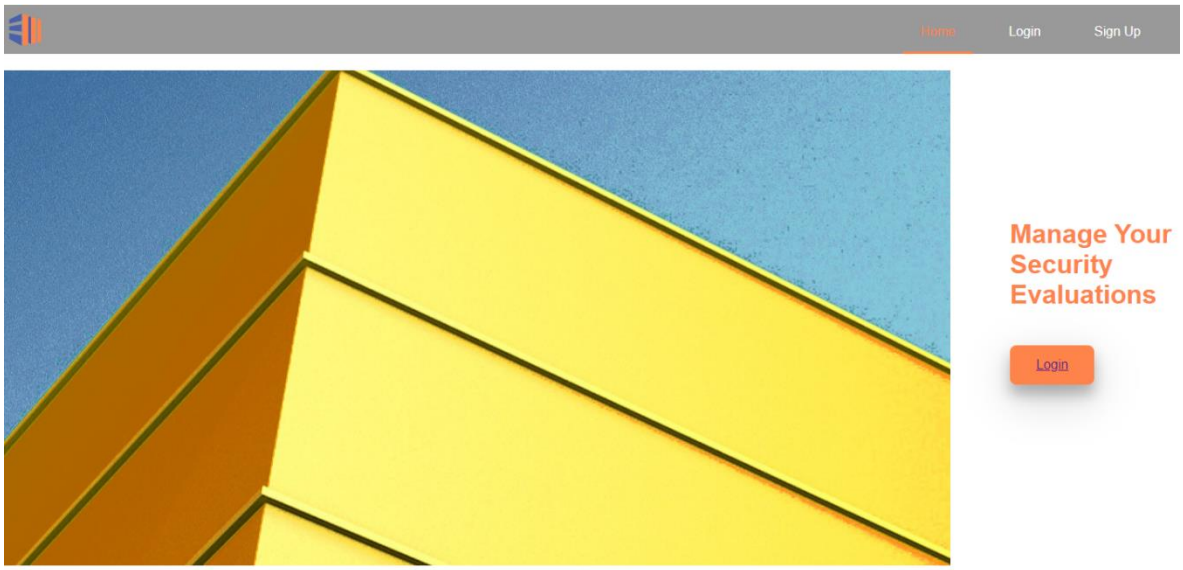

Figure 3 welcomePage.html - Before Login

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.

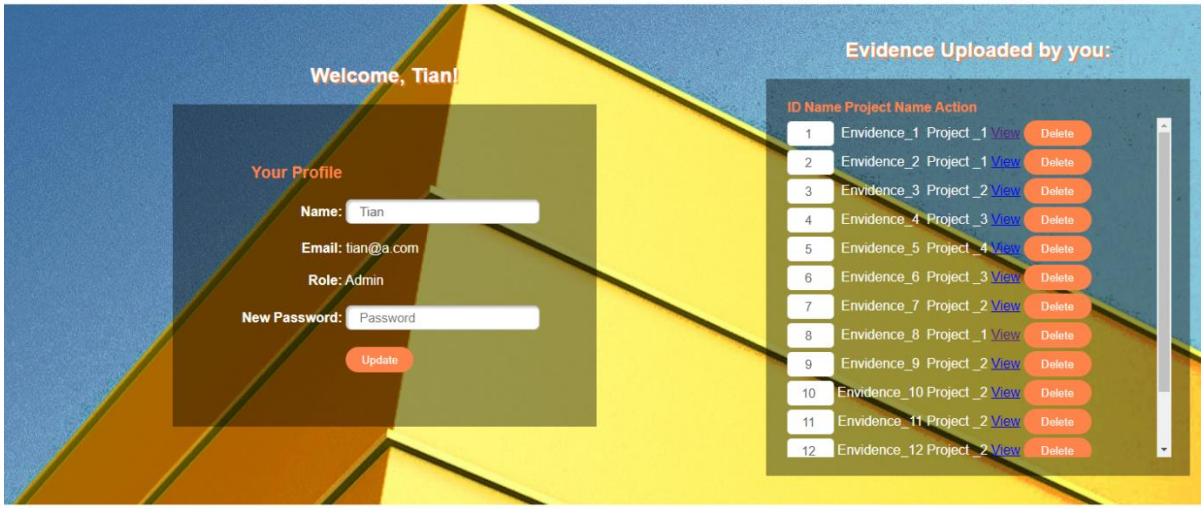

Figure 4 welcomepage.html - After Login

4.5.2 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 page custom design. This form of expression allows users to have a more intuitive understanding of evidence, including information about each piece of evidence.

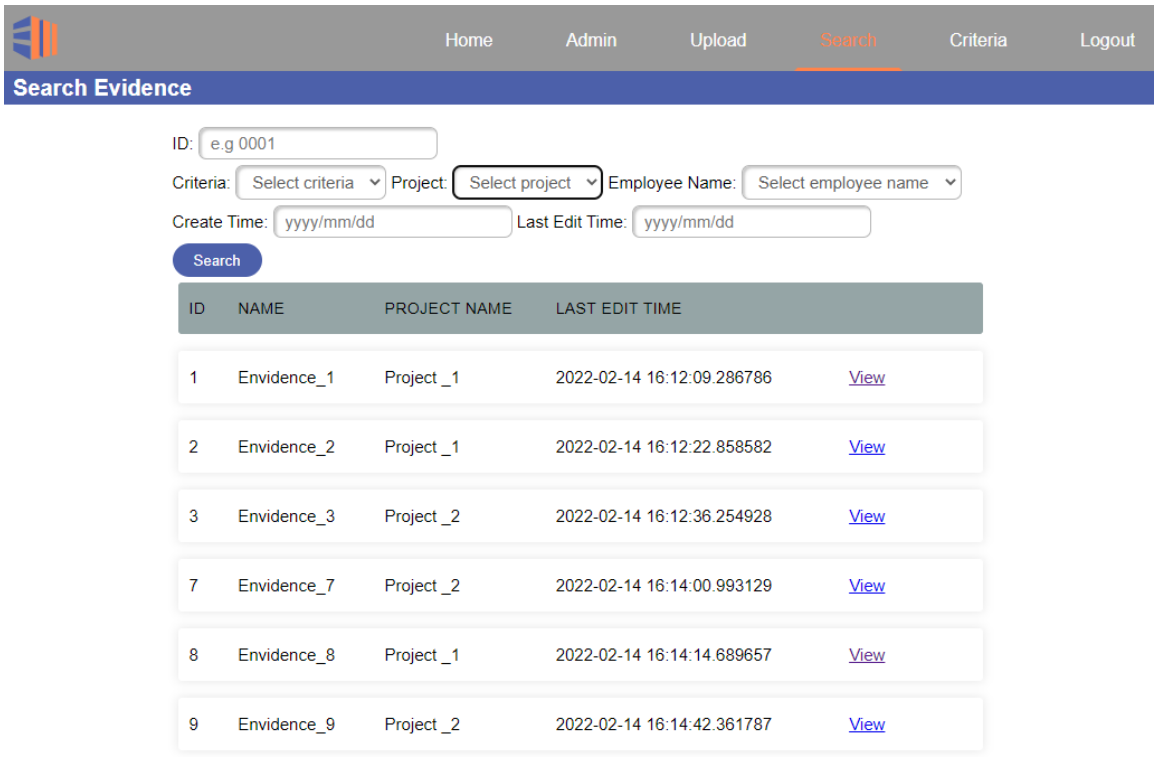

Figure 5 search.html

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.

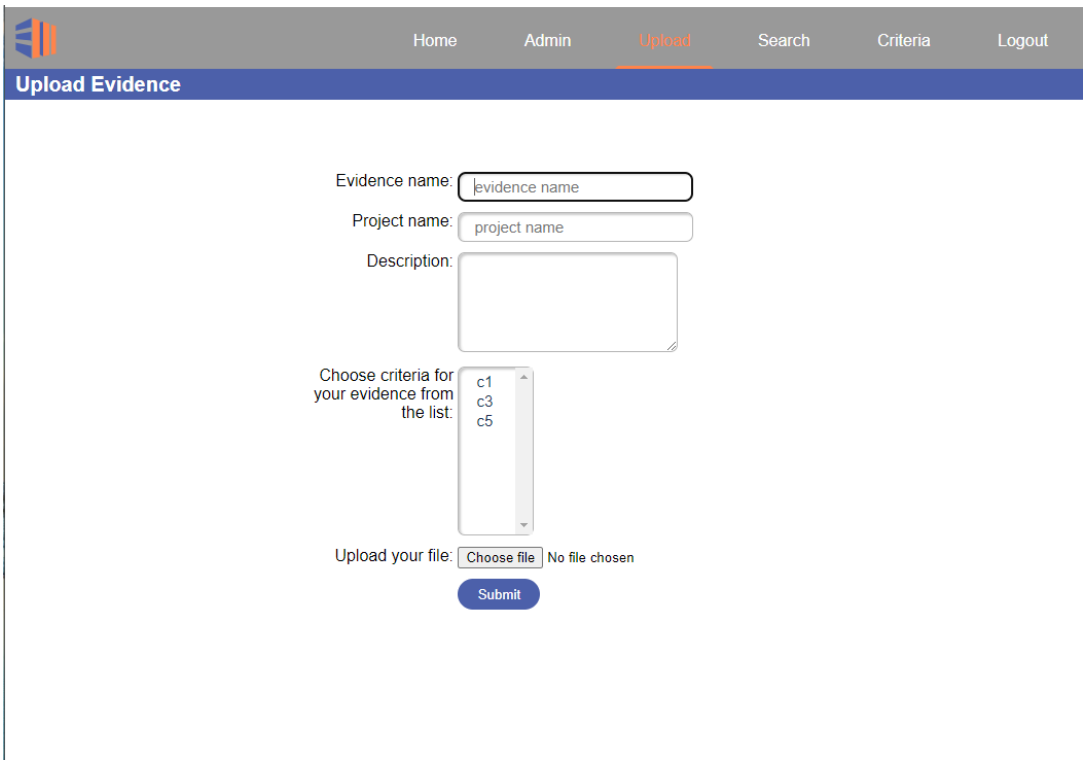

Figure 6 upload.html

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.

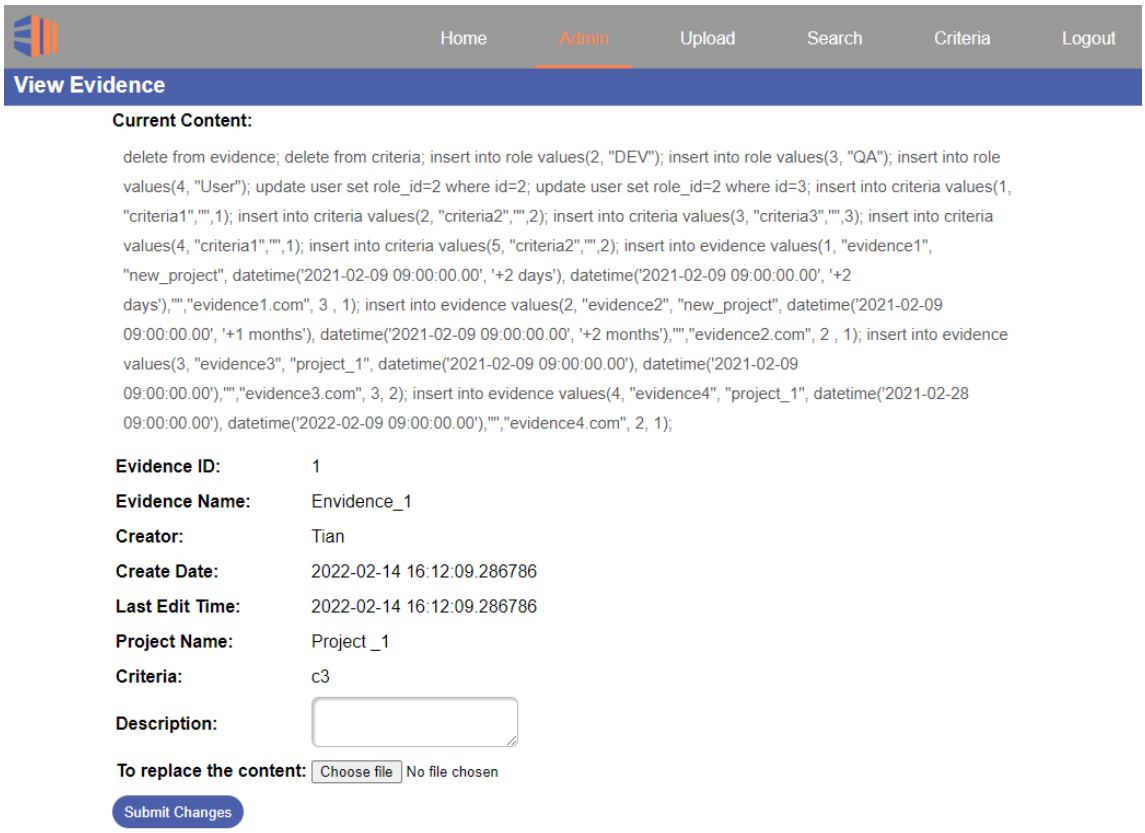

Figure 7 view.html

4.5.3 Authentication Related Pages

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

Figure 8 admin.html

4.5.4 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.

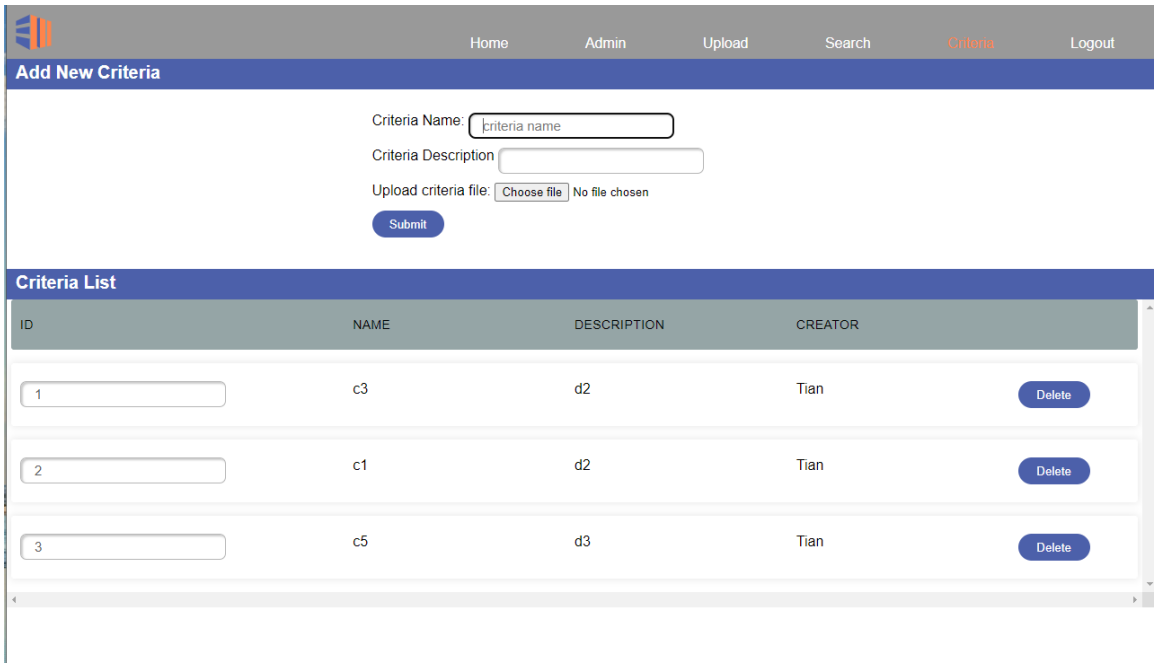

Figure 9 criteria.html

all of the UI implementation sections seem fine
↳ just remember to justify any decisions you made

4.6 Database

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

4.6.1 Design of the database

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

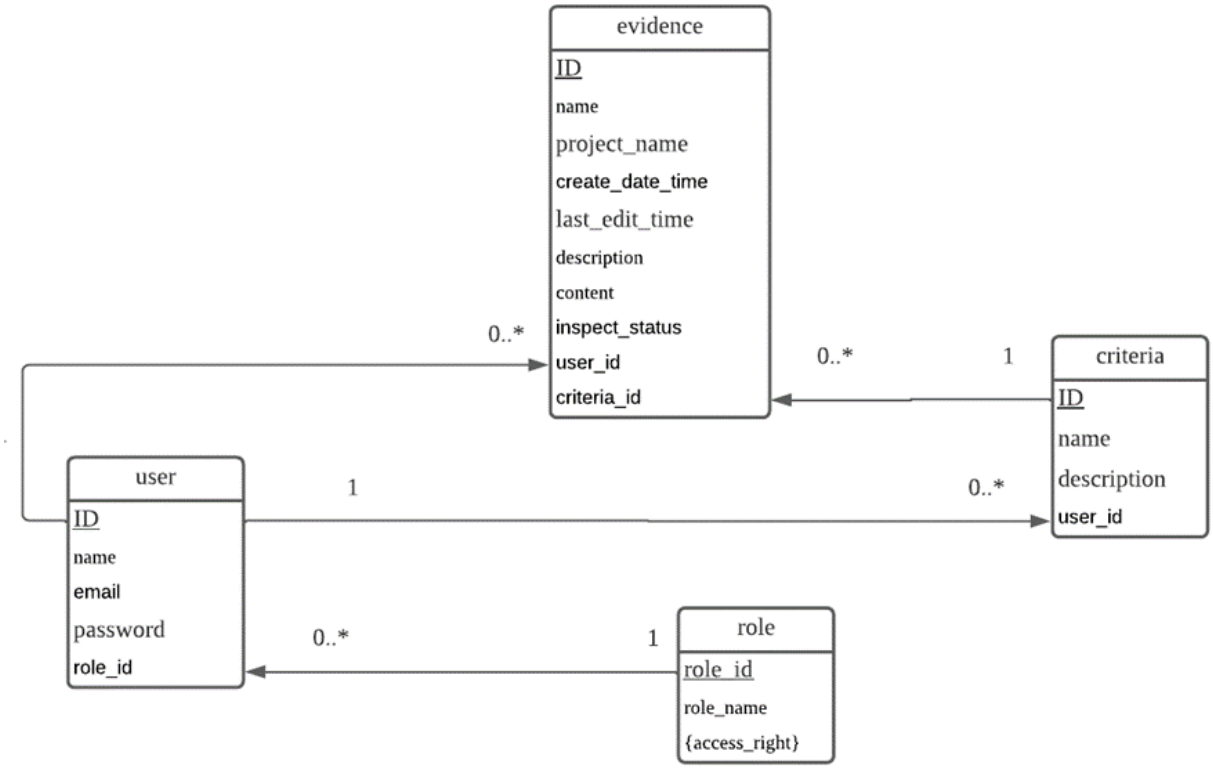

Figure 10 Database schema diagram

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

i) The user table stores the user-related data such as the user's id, name, password, email, and role type.

ii) 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.

iii) The evidence table stores the evidence's id, name, the creation date and time, the latest modified date and time, description, status, URL, and project name.

It also stores the developer's user id who uploads the evidence and the criteria's id which the evidence belongs to

iv) The criteria table stores the criteria's id, name, description, and the admin's user id who create the criteria

what are the alternatives?

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. And 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. We applied an interface SQLAlchemy to query the desired data using raw SQL when querying the related data.

4.6.2 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 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 [4]. Using SQLite, the implementation process does not need to install any other software to check the stored data for feature testing. And the data can be easily invoked through the PyCharm IDE.

4.6.3 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 [5]. This situation 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.

4.6.4 Heroku Cloud Server and Database

"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." [6]. 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 and deploy the new updated

version to the cloud. Heroku CI lets us efficiently manage the CI/CD automation across our security evaluation platform.

* A use case to demo the whole solution would be nice

Chapter 5: Conclusions (INCOMPLETE)

Briefly summarize the problem, the solution, and your accomplishments towards the solution. (Remember, you want to be very clear about the answers to the BIG 3 questions!) Be sure to point out any features that might make your solution more desirable than other existing solutions to similar problems.

Make suggestions about how the project might be extended or modified to solve bigger or related problems. You do not have to give many details here, but a brief description will help the reader follow your suggestion.

During a project (and particularly near the end), it is not unusual for new alternative solutions to be discovered. Often, the project is committed to a particular solution, and too much work would be involved to incorporate the new solution. If this happens, you might want to describe the new solution and recommend that the new solution might be preferable to the one you implemented.

References

[1] C. Criteria, "Common Criteria for Information Technology Security Evaluation - Part 1: Introduction and general model," April 2017. [Online]. Available: https://www.commoncriteriaportal.org/files/ccfiles/CCPART1V3.1R5.pdf. [Accessed 27 September 2021].

[2] T. Me and R. You, "A great result," Wonderful Journal, vol. 5, no. 9, pp. 1–11, 1998.

[3] J. Him and K. Her, "An even better result that you won't believe," Best Journal Ever, vol. 4, no. 8, pp. 55–66, 2002.

[4] “SQLITE3 - DB-API 2.0 interface for SQLite databases” *sqlite3 - DB-API 2.0 interface for SQLite databases - Python 3.10.2 documentation* [Online]. Available: https://docs.python.org/3/library/sqlite3.html. [Accessed: 25-Feb-2022].

[5] “SQLite on Heroku: Heroku dev center,” *SQLite on Heroku | Heroku Dev Center*, 15-Nov-2017. [Online]. Available: https://devcenter.heroku.com/articles/sqlite3. [Accessed: 25-Feb-2022].

[6] S. Rostad, “What is Heroku? A simple explanation for Non-Techies," *TriFin Labs*, 22-Jun-2018. [Online]. Available: https://trifinlabs.com/what-is-heroku/. [Accessed: 26-Feb-2022].

Appendix A: Supporting Appendices

Appendices should be labelled with upper case letters rather than numbers (i.e. A, B, . . .), and each appendix should have a title. Be sure that all appendices are listed in the table of contents and that each appendix is referenced at least once in the report body.