Initial Report of Group attackflow_01

Project: Building a dataset of real-world cyber-attacks with Attack Flow

Team members

Se Jin Yoon: a1706219

Ting-Wei Chin: a1782423

Faisal Hawsawi: a1822781

Lina Nehme: a1802697

Ran Qi: a1675122

Joseph Toubia: a1753547

Zemin Wong: a1780385

Jixing Ye: a1798631

Yu Zheng: a1739446

Contents

Project Vision

The Attack Flow project provides very useful data for the cyber security professionals to better prepare and protect against cyberattacks. In the past, defenders had to track down each of the attacks individually which was then hard to relate between the attacks. With the help of Attack Flow, the defenders now can focus on the sequences of attacks and analyse them to have a better understanding of the cyberattacks.

This project aims to expand the idea of the Attack Flow project and build a system which is easy-to-use but powerful enough to provide useful attack flow models that describe real-world cyberattacks. The system provides users tools to upload and annotate incident report documents, automatic standard dataset generation using the annotated data, and a UI to effectively visualise attack flows to enable users to study the attack flow in detail.

We have one goal, which is to provide a simple tool to help defenders against the real-world-cyberattacks. We want to keep it simple and easy-to-use but powerful enough to deliver useful data for the defenders.

Customer Q&A

Q: Is the flow of application where we allow a user to upload a document, annotate it, then an attack flow is generated (where the application will give a visualisation of the attack flow), in which we will have a validation point at the end, where the admin will then approve the annotated document and attack flow before being added to the database?

A: That is exactly the flow of application where a user uploads a document, annotate then an attack flow is generated, which will then have a validation point at the end, where the admin will approve the annotated document and attack flow before being added to database.

Q: Will attackflow input and output files be provided?

A: Yes, sample incident report files and their attack flow files will be provided. See MITRE attackflow project website.

Q: Given that the document should be editable, not be visualised. Does that mean when we have to incorporate a tool where you can edit the document and annotate, then ‘publish’, which will then be shown in the attack flow format?

A: A tool has to be incorporated where the document can be annotated but not editing

Reflection

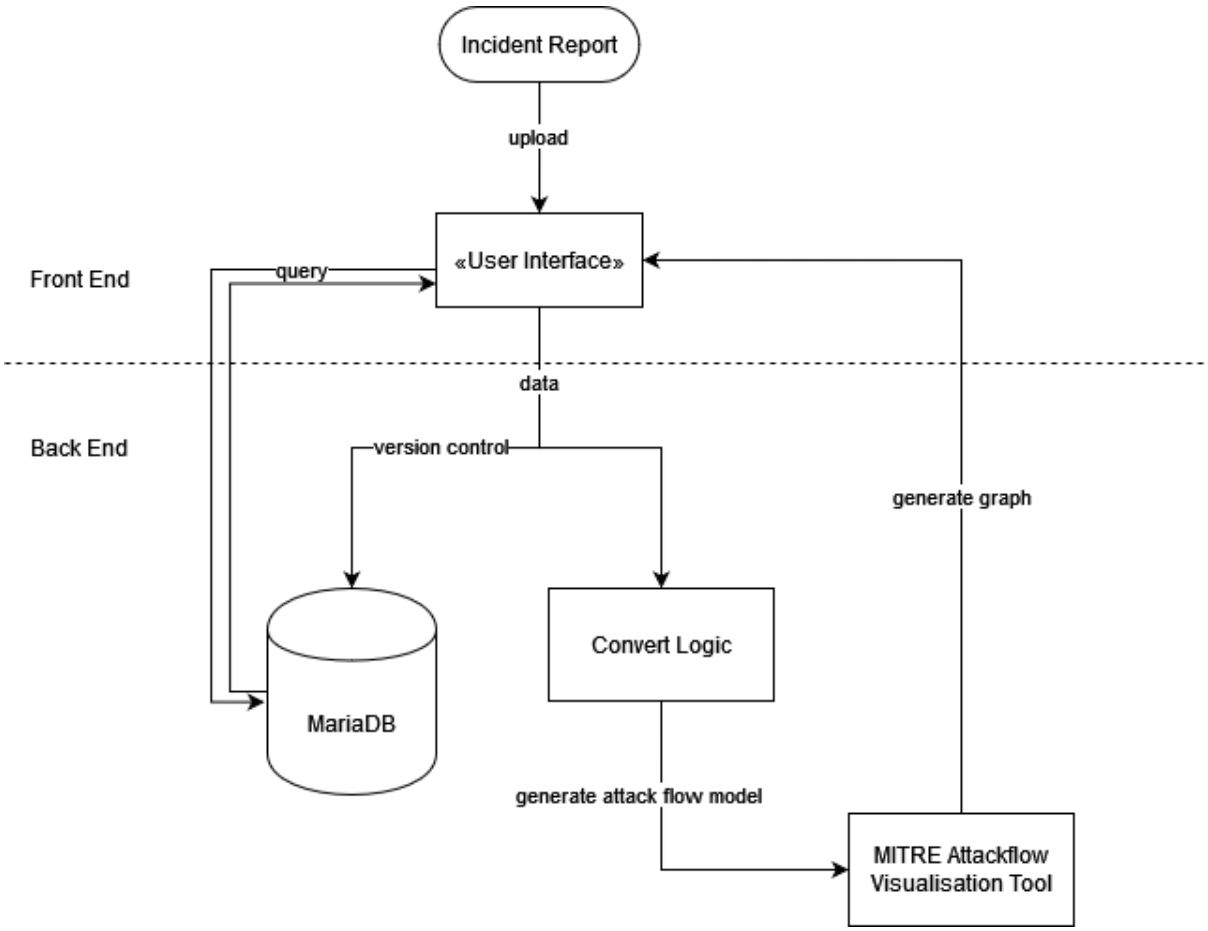
The Kick-off meeting gave lots of insight into the project where everyone can understand what the project is and what it requires. However, before the meeting we could have read through the provided document and conduct in-depth research to formulate questions that can provide assistance further down the project. Because after reading and reviewing the questions asked, it can be concluded that more insightful questions could have been asked, in which we will improve for the next meeting.

Users

The users of this project are **cyber security professionals** who are seeking to enhance their understanding of cyber adversaries, their tactics, techniques, and procedures. These professionals may come from a variety of roles within the field of cyber security, including but not limited to:

- Cybersecurity Specialists/Researchers: Use Attack Flow helps researchers stay ahead of emerging threats and develop effective countermeasures, they can leverage the detailed behaviour-based threat intelligence products created through the system to understand adversary tactics and techniques more comprehensively.
- Programmers: They might contribute to creating algorithms that automate parts of the attack flow generation processor, enhance the visualisation capabilities, and ensure its functionality, usability, and the integration with other tools give the best efficiency.
- Threat Intelligence Analysts: They can provide insights to their organisations or clients about potential threats and vulnerabilities. The machine-readable language of Attack Flow could make justifying defence techniques more understandable and facilitates collaboration and interoperability across organisations.
- Security Operations: Attack Flow assists in identifying gaps in coverage and helps teams make data-driven decisions, this process could happen by identifying weak points in their defences and simulating different attack scenarios, and prioritising security measures to mitigate risks.
- Incident Response Teams: Post-incident analysis, they can map out the attack flows to understand how an attack occurred, where defences failed, and how they can improve their incident response plans. Improving future responses depends a lot on analysis in learning from past incidents.
- Red Team Members: utilising Attack Flow they can model realistic sequences of tactics, techniques, and procedures to plan and execute adversary emulation scenarios, this would result in more red flags on defences against threats and assess an organisation's defensive capabilities.
- Risk Assessors: Evaluate potential risks based on observed sequences of attack steps, this information helps them understand the potential impact of various attack scenarios on an organisation's assets and operations.
- Government Agencies: Sharing their insights with other organisations to release top tier secure national standards and increase awareness of national cybersecurity.

Software Architecture



We chose to build the tool as a web application which allows users to interact with the tool through a web page. The data uploaded by the users will be processed by a converting logic in the back end that converts data to attackflow models using the format specified by the user. The original documents and annotations will also be saved in MariaDB, which can be queried by the user. The MITRE attackflow visualization tool will be integrated into the system to allow users to download attackflow models as graph.

A web application was selected as our chosen software architecture due to the following factors:

- Scalability: increasing number of users can be accommodated by scaling web applications and database.
- Portability: Users can interact with the web application on various devices with an internet connection.
- The user-centric-familiar user interface of a web application ensures that users can easily access and interact with the website. Multiple users can collaborate when uploading and annotating documents and visualising attack flows.
- Easy to maintain-web applications can make changes through the use of centralised updates on the server side which makes maintenance time-effective.
- Security-Secure authorisation and authentication system can be in place on the user interface. These access control mechanisms can ensure only authorised users access certain data which protects sensitive data.

Tech Stack and Standards

Front-end

Use HTML and CSS to build a simple, clean, easy to use user interface.

Back-end

- Use JavaScript, node.js to implement functionalities like annotating, saving documents, generating attackflow, etc.
- Use MariaDB to store incident reports and annotation data. We chose MariaDB because it is a high percentage open-source database software.
- Natural language processing models like ChatGPT for parsing incident report and assisting user annotation.

Frameworks

- Vue3 for seamless user interactions
- Express
- The open-source attack flow framework as guideline
- Axios, GraphQL

Libraries

Font Awesome, Element UI plus

Communication Tools

- Discord for informal communication
- Slack for formal communication
- Zoom for meetings

IDE

vscode

Diagrams

draw.io

Package manager

npm

Database Visualisation tool

Visual Studio Code extension: Database Client

Coding standards

- consistent naming conventions.

```
# Example: Use camelCase for variable names and PascalCase for class names in Java.
int myVariable = 10;
class MyClass {}
```

- Use Intention-Revealing variable names.
    - use underscore in variable_name
    - capitalise global variable names
    - Function name: do_this

- Detailed commenting

```
# Example: Use comments to explain the purpose and behavior of functions.
/**
* Calculates the area of a circle.
*
* @param {number} radius - The radius of the circle.
* @returns {number} The area of the circle.
*/
function calculateArea(radius) {
return Math.PI * radius * radius;
}
```

  - proper grammar and spelling
  - Be clear and concise
  - Avoid obvious comment
  - Comment above each function

```
int addTwoNumbers(int a, int b) {

//the procedure that sums given two numbers

       int result = a + b;

       return result;

}
```

- Break down long code into functions

```
# Example: Break down complex code into smaller, reusable functions.
def calculate_area(radius):

return 3.14 * radius * radius

def calculate_circumference(radius):
       return 2 * 3.14 * radius
def display_circle_properties(radius):
       area = calculate_area(radius)
       circumference = calculate_circumference(radius)
print(f"Area: {area}, Circumference: {circumference}")
```

- Keep indentation consistent: use tab

```
# Example: Use consistent indentation (e.g., 2 or 4 spaces) throughout the code.
def my_function():
       if condition:

       # Do something
       Pass

       Else:

       # Do something else
       pass
```

- Graceful error and exception handling, e.g., use try and catch

```
# Example: Use try-catch blocks to handle exceptions.
try {
	int result = 10 / 0;
} catch (ArithmeticException e) {
	System.out.println("Division by zero is not allowed.");
}
```

- Avoid magic numbers, because using magic numbers makes your code of test hard to understand due to lack of context. Using magic numbers limits the flexibility of code since If requirement changes happen, you have to update all occurrences in your code. Using magic numbers also makes your code hard to maintain, since If you use magic numbers in your tests and need to change the value later, you might have to search and replace the value across your codebase.

```
# Example: Use named constants instead of hardcoding numeric values.
const double Pi = 3.14159265359;
double area = Pi * radius * radius;
```

- Security: Identify potential security issues Including
  - buffer overflows
  - SQL or command injection
  - cross-site scripting
  - vulnerable versions of libraries
  - exposure of sensitive data
  - threading problems,race conditions, and so forth.
- Testing

```
# Example: Write unit tests to verify the behavior of functions.
@Test
public void testAddition() {
int result = Calculator.add(2, 3);
assertEquals(5, result);
}
```

  - Integration and unit tests
  - Black and white box test

Group Meetings and Team Member Roles

We will meet on Zoom twice a week, to talk about assignments, plan tasks, and help each other remove roadblocks.

- Weekly Meeting 1: Wed 6.30 - 7.30 pm
- Weekly Meeting 2: Fri 6.30 - 7.30 pm

We will also have retrospective meeting at the end of each sprint. To prepare for the retrospective meeting, each member need to:

- review tasks allocated in the sprint – Were the tasks reasonably allocated? What tasks have been completed and what haven't? What were the main roadblocks?
- Set the agenda of the meeting.

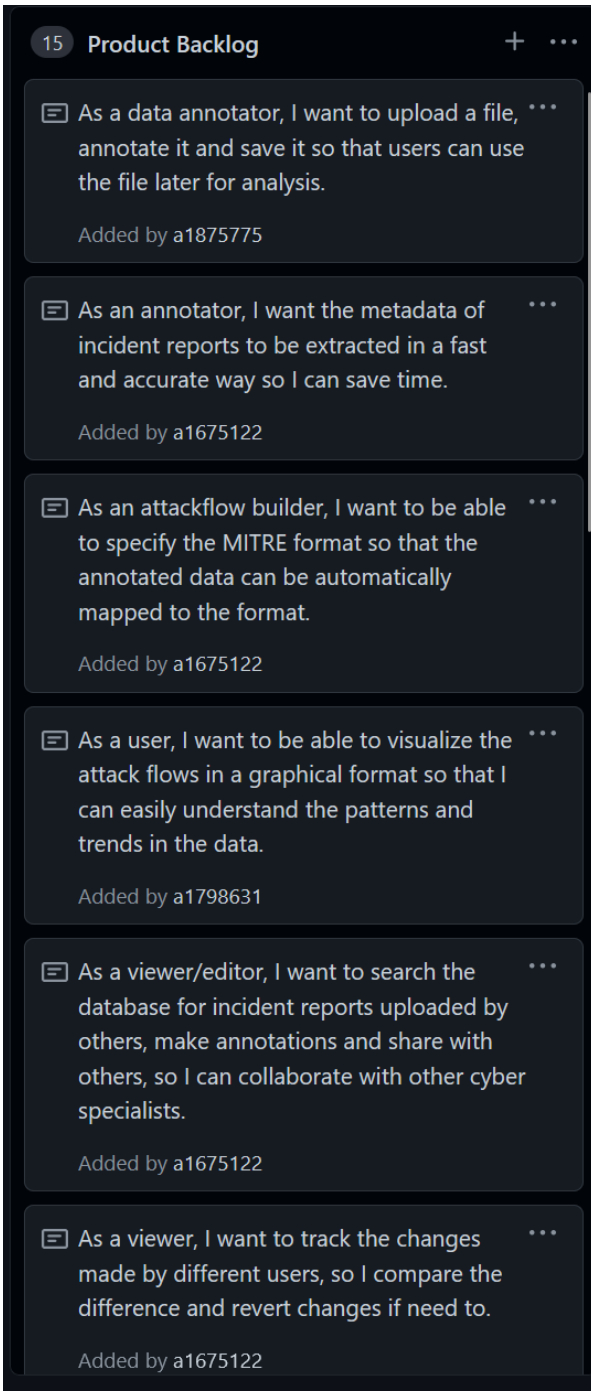
The group mainly communicate with the tutor via Slack. We also meet with tutor every two weeks on Zoom for 25 minutes give summary on work done in the last sprint, receive feedback, and plan for the next sprint.

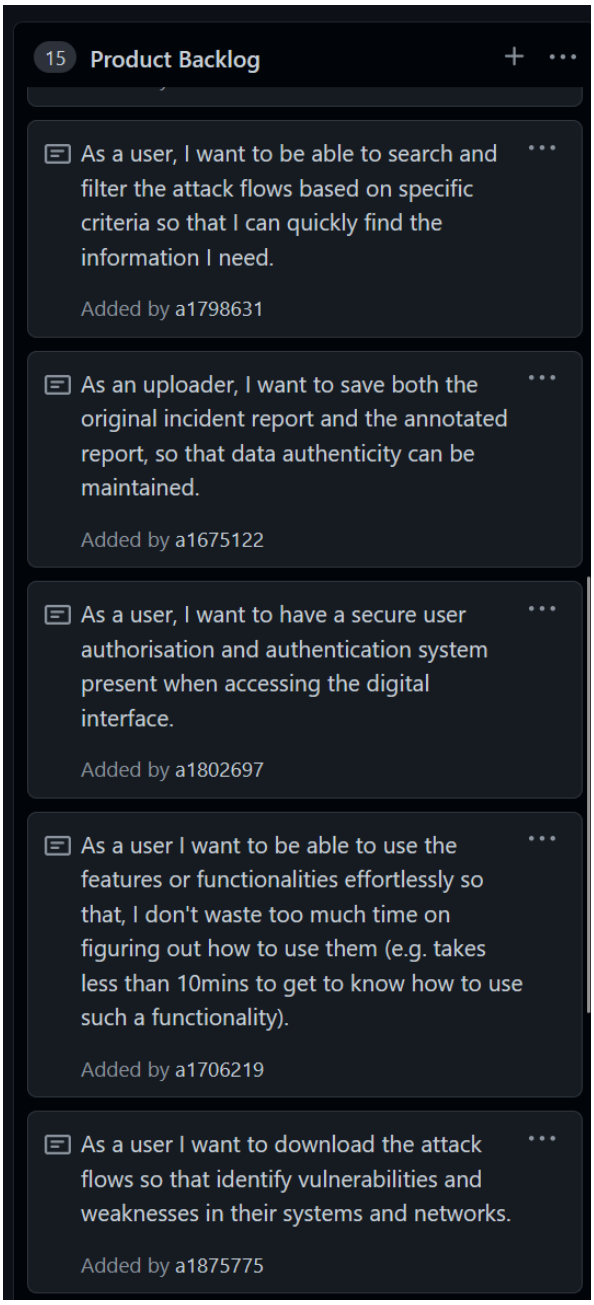
Scrum Master Schedule

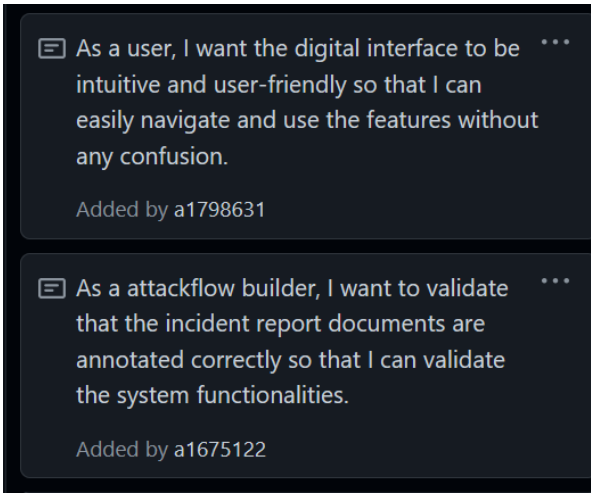
Sprint 1	Ran Qi
Sprint 2	Ting-Wei Chin
Sprint 3	Lina Nehme
Sprint 4	Yu Zheng
Sprint 5	Joseph Toubia

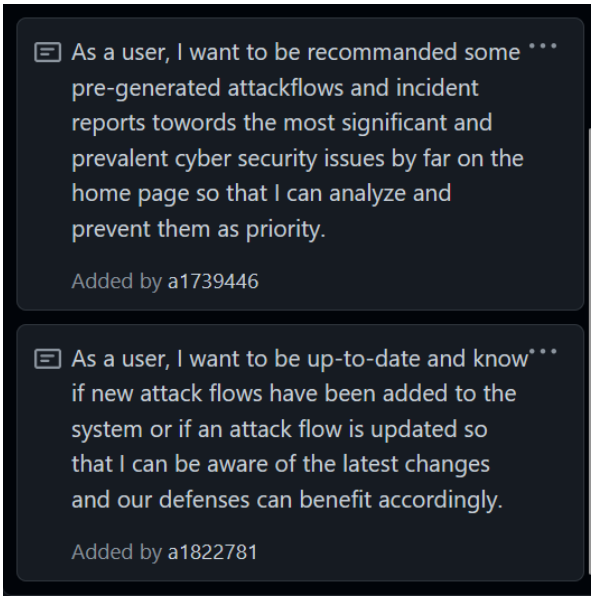
Snapshots

Product Backlog

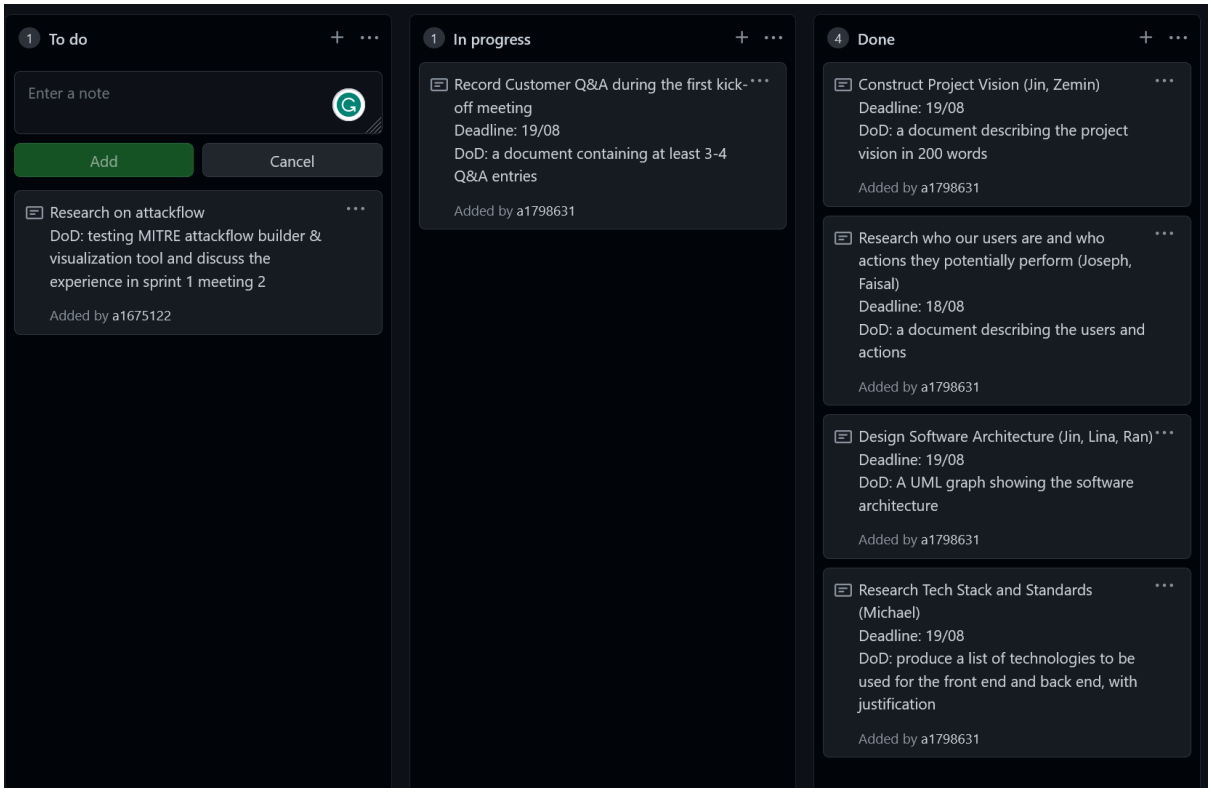




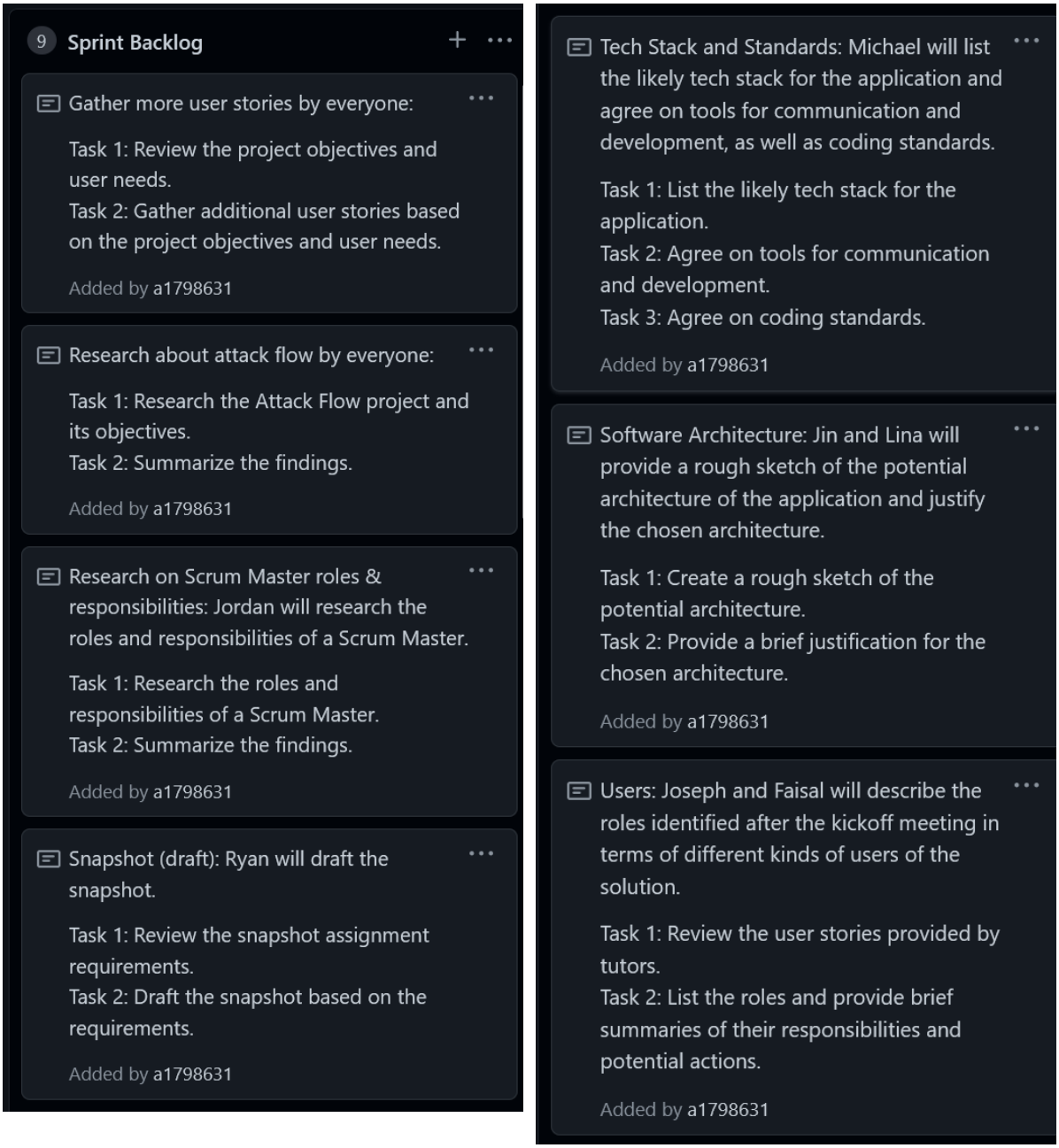

As a user, I want the digital interface to be intuitive and user-friendly so that I can easily navigate and use the features without any confusion.
Added by a1798631
As a attackflow builder, I want to validate that the incident report documents are annotated correctly so that I can validate the system functionalities.
Added by a1675122

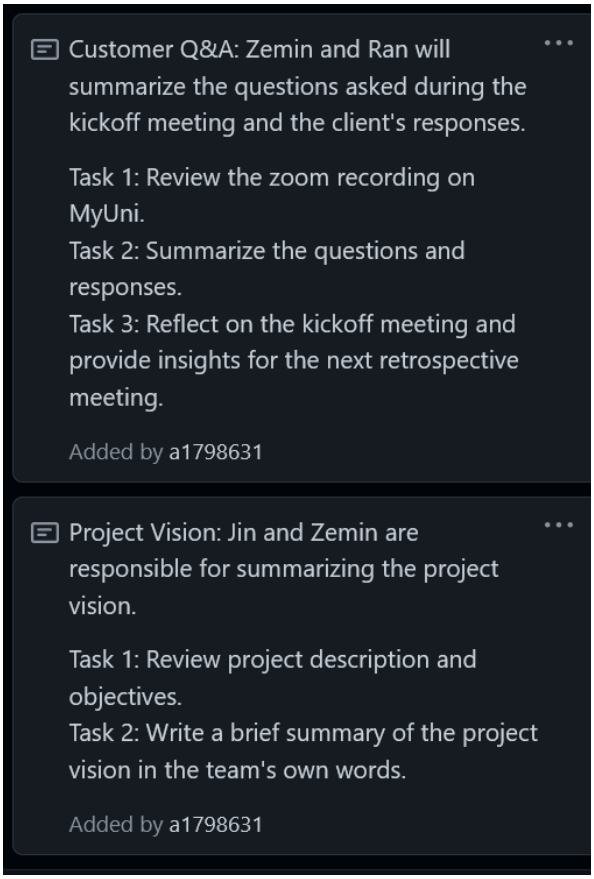

As a user, I want to be recommanded some pre-generated attackflows and incident reports towords the most significant and prevalent cyber security issues by far on the home page so that I can analyze and prevent them as priority.
Added by a1739446
As a user, I want to be up-to-date and know if new attack flows have been added to the system or if an attack flow is updated so that I can be aware of the latest changes and our defenses can benefit accordingly.
Added by a1822781

Task Board


1 To do
Enter a note
Add
Cancel
Research on attackflow
DoD: testing MITRE attackflow builder & visualization tool and discuss the experience in sprint 1 meeting 2
Added by a1675122
1 In progress
Record Customer Q&A during the first kick-off meeting
Deadline: 19/08
DoD: a document containing at least 3-4 Q&A entries
Added by a1798631
4 Done
Construct Project Vision (Jin, Zemin)
Deadline: 19/08
DoD: a document describing the project vision in 200 words
Added by a1798631
Research who our users are and who actions they potentially perform (Joseph, Faisal)
Deadline: 18/08
DoD: a document describing the users and actions
Added by a1798631
Design Software Architecture (Jin, Lina, Ran)
Deadline: 19/08
DoD: A UML graph showing the software architecture
Added by a1798631
Research Tech Stack and Standards (Michael)
Deadline: 19/08
DoD: produce a list of technologies to be used for the front end and back end, with justification
Added by a1798631

Sprint Backlog





User Stories & DoD

During the first sprint, most of the work is to understand client's requirements and deduce user stories. After meeting with the clients, we selected the most important user stories, which are the key feature of our product:

> *"As a data annotator, I want to upload an incident report, annotate it and save it so that users can use the file later for analysis."*

> *"As a user, I want to be able to visualize the attack flows in a graphical format so that I can easily understand the patterns and trends in the data."*

Based on the user stories, we derived below tasks:

- Research on attack flow (sample files, current projects, tools available, etc)
    - DoD: testing MITRE attackflow builder & visualization tool and discuss the experience in sprint 1 meeting 2
- Propose software architecture
    - DoD: A UML graph showing the software architecture
- Research tech stack and code standards
    - produce a list of technologies to be used for the front end and back end, with justification

Summary of Changes

In this initial snapshot, we have laid the foundation for our project by defining the user stories based on the project's objectives. The team has been actively collaborating to set up the product backlog , task board, meeting schedules, communication platforms, etc. Preliminary discussions with Associate Professor Hung Nguyen have provided valuable insights, guiding our approach. The team is motivated and committed to ensuring the project's success.

Changes in this week's snapshot:

- Defined a list of user stories, functional & non-functional requirements.
- Decided on the platform (web app) and proposed software architecture accordingly.
- Decided on the tech stack (database, framework, coding language, IDE)