FRIC

Findings and Reporting Information Console - FRIC

codefactor A-

Scope of Product

The Cyber Experimentation & Analysis Division (CEAD) recognizes the complexity and the time it takes to manage task assignments, progress, vulnerability discovery during a cyber engagement and generate custom reports that presents the discovered vulnerabilities and potential issues to CEAD's target audience. They want a system that would aid the management of task, collection of evidence, and report generation during a cyber engagement.

The University of Texas at El Paso (UTEP) and CEAD are collaborating to develop Findings and Reporting Information Console (FRIC) system that will provide the ability to manage task assignment and progress, and facilitate the collection of evidence on existing vulnerabilities, and generation of custom reports.

This is the FRIC system, as developed by Team 8 (SBSG) of the Fall 2020 CS 4311 cohort at the University of Texas at El Paso. FRIC uses the following technologies for its development.

- PWA/SPA — Single page app
- Built with Vue.js, Buefy & Bulma
- SCSS sources with variables
- Mobile Responsive

Table of Contents

Deploy with Docker □ (Work in Progress...)

⚠ We recommend you go through the manual installation first to ensure proper installation of the FRIC tool. Deploying through Docker is still a work in progress.

In an effort to simplify the process of deploying our application, we use Docker container technology to eliminate the need to manually install any dependencies or services on your host machine. We setup a Docker system with containers for each of our services needed to run FRIC.

> To learn about Docker and how it works read the official documentation here https://www.docker.com/why-docker
>
> Follow these steps to install docker on your machine https://docs.docker.com/get-docker/

□To run FRIC using **Docker** follow these steps

- Download the source code.
- Build the frontend ui by `cd frontend` and then running the following

```
docker build -t kevinapodaca/team8:frontend -f Dockerfile .
```

- Now you can run the frontend container by issuing the following command

```
docker run --name fric -e VUE_APP_API_HOST=127.0.0.1 -e VUE_APP_API_PORT=300
```

> Note that the ports being exported could be changed if needed. See https://docs.docker.com/engine/reference/commandline/run/

- Run the command `docker ps` to list the current processes, and copy the *Network ID* of the process named **kevinapodaca/team8:frontend**.
- You can now run `docker start id` and replace the word *id* with the network id you copied in the previous step

> This has now started the frontend services container for FRIC.

- Build the backend by cd backend and then run the following

```
docker build -t kevinapodaca/team8:backend -f Dockerfile .
```

- Now you can run the backend container by issuing the following command

```
docker run --name fric-backend -e DATABSE_HOSTNAME=IP-ADDR -d kevinapodaca/t
```

Kubernetes Cluster Deployment

To run the app using a Kubernetes cluster follow the commands from the root directory:

1. Deploy the mongoDB instance with `kubectl apply -f k8s-mongodb.yml`
2. Deploy the backend instance with `kubectl apply -f k8s-backend.yml`
3. Deploy the frontend instance with `kubectl apply -f k8s-frontend.yml`

Deploying Manually Quick Start

> Use this method of installation until the Docker setup is completed. This notification will be removed when that is complete.

FRIC uses the `MEVN` stack. This is a full javascript-based technology stack. Before attempting to run the project you should check that you have the following things installed.

Mongo □

This is needed to be able to create a database instance of FRIC and be able to perform CRUD operations on it. To check if you have mongo installed type the command

> mongo -version

To install mongoDB please use the following guides:

macOS Install

Using brew `brew tap mongodb/brew` and then `brew install mongodb-community@4.4`

Next make sure you start the services using `brew services start mongodb-community@4.4`

Finally, to begin using mongo run the following command `mongo`

> For more details on installing mongoDB on mac please reference https://docs.mongodb.com/manual/tutorial/install-mongodb-on-os-x/

Windows Install

Download the installer from https://www.mongodb.com/try/download/community?tck=docs_server

Run through the installation process and make sure to:

1. Use the `Complete` setup type when prompted for one.

2. Select `Install mongoDB as a Service` when configuring the service

Now connect the shell to the mongoDB instance by navigating (in file explorer to) `C:\Program Files\MongoDB\Server\4.4\bin\` and double click on the `mongo.exe` file

> For more details on installing mongoDB on windows please reference https://docs.mongodb.com/manual/tutorial/install-mongodb-on-windows/

Linux Install

We tested the installation on the *CentOS*, a Red-Hat-based distribution, to install mongoDB use the following steps:

1. `gedit /etc/yum.repos.d/mongodb.repo`
2. Now add the following lines to this file

```
[mongodb-org-4.0]
name=MongoDB Repository
baseurl=https://repo.mongodb.org/yum/redhat/$releasever/mongodb-org/4.0x86
gpgcheck=1
enabled=1
gpgkey=https://www.mongodb.org/static/pgp/server-4.0.asc
```

Now you can just install mongo with `yum install -y mongodb-org server`

To start the services run the command `systemctl enable mongod && systemctl start mongod`

NodeJS □

FRIC relies on the javascript runtime environment *NodeJS* to compile, transpile, and serve content. As such, it is necessary that you have this installed to be able to run any `npm` command.

To install nodeJS on your machine navigate to https://nodejs.org/en/download/ and select the appropriate installer for your operating system.

> ⓘ As of 12/05/2020 we recommend you use the LTS installer for nodeJS, we cannot guarantee our tool is supported on the latest release of nodeJS

Finally, ensure that you have both nodeJS installed and the node package manager by running the command `node -v` and `npm -v` . FRIC was developed and tested with node version **v15.0.1** and npm with version **7.0.3**

□ Congrats! You should now have everything ready to use our tool! Follow the next steps □

Get the repo

Download the source code with
`git clone https://github.com/KevinApodaca/FRIC-team8-SBSG.git`

Install

To install all dependencies for the **backend** services: `cd backend` followed by `npm install`

> When running the install in the backend folder, it is also necessary to also run this command
>
> `npm install --save-dev @babel/cli`

To install all dependencies for the **frontend** : `cd frontend` followed by `npm install`

Serve

To pre-compile & hot-reload for development, launch two terminal windows/tabs (one in which you are in the frontend directory, the other in which you are in the backend directory) and use the following commands.

1. `npm run serve` in the frontend directory
2. `npm start` in the backend directory

You should now be given a localhost url in which you can visit the project through the browser.

Build

Production-ready with minified bundle `npm run build`

Browser Support

Currently supports

- Firefox
- Chrome
- Safari
- Brave
- Opera
- Edge
- Min

Reporting Issues

We are accepting issues and pull requests on Github, if you notice an issue be sure to raise an issue up.

If you notice something that could be improved, feel free to send a pull request up and explain what's happening and how your solution fixes it.

Useful Links

- Vue.js Guide
- Vue CLI Guide
- Buefy
- Bulma

FRIC Backend

Introduction

This section will document the FRIC API and the ways in which our tool interacts with the database and web server

---

Database Notes for Developers

Developers interested in the structure of our mongoDB database and wishing to directly manipulate information and documents would need a GUI tool for mongo.

We have used MongoDB compass which you can get from https://www.mongodb.com/products/compass

Alternatively, you can use tools like TablePlus, Studio3T, or noSQLBooster

---

API Notes for Developers

Developers interested in testing our API without having to use the frontend client would need a tool for making HTTP requests directly to our endpoints.

We have used Postman for this purpose which allows for `GET` , `POST` , `DELETE` , and `PATCH` requests in an easy-to-use application. You can get Postman from https://www.postman.com/downloads/ or `brew cask install postman` for brew users

FRIC API

The development of FRIC required a custom RESTful API to be created in order to fulfill all the necessary requirements. To test our API endpoints you can use a tool such as Postman.

GET Get Artifacts

```
http://localhost:3000/<artifact>/all
```

This endpoint allows the system to get data from artifacts.
<artifact> in the URL indicates the artifact collection such as Transactions, Analysts, Events, Systems, Tasks, Subtasks or a Findings.

Request

Path Parameters

transactions
REQUIRED
`string`

This will display all Transactions in the database.

analysts
REQUIRED
`string`

This will display all Analysts in the database.

events
REQUIRED
`string`

This will display all Events in the database.

systems
REQUIRED
`string`

This will display all Systems in the database.

tasks
REQUIRED
`string`

This will display all Tasks in the database.

subtasks
REQUIRED
`string`

This will display all Subtasks in the database.

findings

REQUIRED

`string`

This will display all Findings in the database.

Response

● 200: OK

Returns if the connection was successfully made with the backend and the request was accepted. Sample POST request that would generate a 200

GET all tasks

```
{
        "subtasks": "1",
        "system": "",
        "analyst": "",
        "title": "SampleTaskKevin",
        "systems_for_task": "AlexAlex",
        "findings": "3",
        "task_priority": "Medium",
        "task_progress": "Transferred",
        "created": "2020-12-02",
        "description": "asdfasdf",
        "task_association": [],
        "subtask_association": [],
        "finding_association": [],
        "systems": "",
        "progress": 0,
        "priority": "",
        "task_progresses": "Not Started",
        "id": "5fab4352d5032a469de0d2a0"
    },
```

● 404: Not Found

Returns if there was an issue with the request on that artifact or endpoint. For example, if you try to reach an endpoint that is misspelled you might see something like this.

```
<!DOCTYPE html>
<html lang="en">

<head>
```

```
  <meta charset="utf-8">
  <title>Error</title>
</head>

<body>
  <pre>Cannot GET /tasksk/all</pre>
</body>

</html>
```

POST Post Artifacts

```
http://localhost:3000/<artifact>/
```

This endpoint allows the system to add data to the artifacts.
<artifact> in the URL indicates the artifact collection such as Transactions, Analysts, Events, Systems, Tasks, Subtasks or a Findings.

Request

Path Parameters

transactions
REQUIRED
`string`

This will add a new artifact to Artifacts in the database.

analysts
REQUIRED
`string`

This will add a new artifact to Analysts in the database.

events
REQUIRED
`string`

This will add a new artifact to Events in the database.

systems
REQUIRED
`string`

This will add a new artifact to Systems in the database.

tasks
REQUIRED
`string`

This will add a new artifact to Tasks in the database.

subtasks
REQUIRED
`string`

This will add a new artifact to Subtaks in the database.

findings

REQUIRED

`string`

This will add a new artifact to Findings in the database.

Response

● 200: OK

This appears when you are able to submit a request to our API and it is accepted. For example if you create a new system you might see something like this be returned to you.

```
{
        "name": "SampleSystemName",
        "system": "",
        "description": "asdf",
        "location": "123",
        "router": "asdf",
        "switches": "fds",
        "room": "fdsa",
        "plan": "afsd",
        "analyst": "",
        "title": "",
        "findings": "0",
        "progress": 0,
        "tasks": "0",
        "status": "0",
        "type": "",
        "classification": "a simple security guide",
        "risk": "",
        "system_confidentiality": "Medium",
        "system_integrity": "High",
        "system_availability": "Low",
        "task_association": [],
        "subtask_association": [],
        "finding_association": [],
        "task": "",
        "id": "5f87e1567e88ea1048cd348d"
    },
```

PATCH Patch Artifacts

```
http://localhost:3000/<artifact>/:id
```

This endpoint allows the system to modify data from artifacts.
<artifact> in the URL indicates the artifact collection such as Transactions, Analysts, Events, Systems, Tasks, Subtasks or a Findings.
:id indicates the unique identifier for the artifact.

Request

Path Parameters

:id
REQUIRED
`string`

The unique identifier for the artifact that is to be displayed and modified in the database.

Response

● 200: OK

A sample response you would get if you changed the title of a finding.

```
{
    "message": "Finding was updated successfully."
}
```

DELETE Delete Artifacts

```
http://localhost:3000/<artifact>/:id
```

This endpoint allows the system to remove an artifact.
<artifact> in the URL indicates the artifact collection such as Transactions, Analysts, Events, Systems, Tasks, Subtasks or a Findings.
:id indicates the unique identifier for the artifact.

Request

Path Parameters

:id
REQUIRED
`string`

The unique identifier for the artifact that is to be removed from the database.

Response

● 200: OK

After a succesfull deletion of a system, you might see this response.

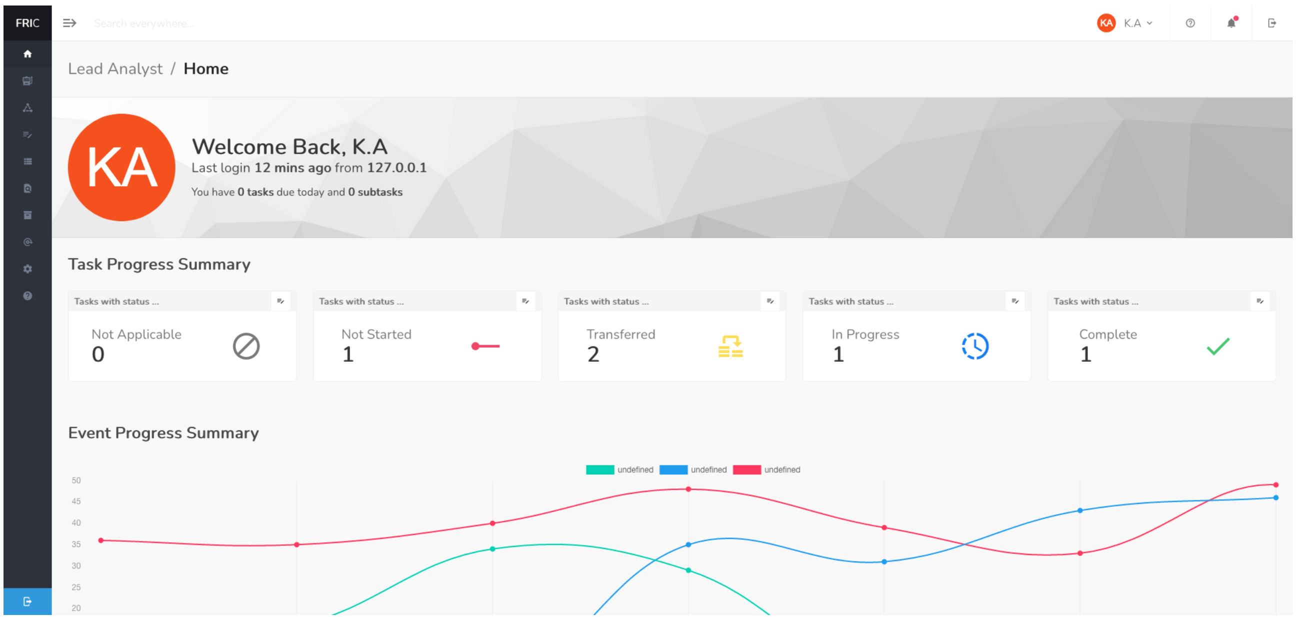
```
{
    "message": "System was deleted successfully!"
}
```

FRIC Frontend

Introduction

This section will document the FRIC frontend codebase as well as contain some information for developers

---



Welcome Back, K.A
Last login **12 mins ago** from **127.0.0.1**
You have **0 tasks** due today and **0 subtasks**

Task Progress Summary

Tasks with status ...	Tasks with status ...	Tasks with status ...	Tasks with status ...	Tasks with status ...
Not Applicable 0	Not Started 1	Transferred 2	In Progress 1	Complete 1

Event Progress Summary

FRIC Homepage

Home

This page is the initial home page for an analyst that will display the analyst progress summary and the log history.

> http://localhost:8080/#

Analyst Progress Summary (Home.vue)

Name	Description
Task Progress Summary	This is used to display the number of tasks with certain status
Event Progress Summary	This is used to visually display on a Chart how many systems, findings, and tasks have been created.
Subtask Progress Summary	This is used to display the number of Subtasks with certain status'
Recent Changes	This is used to display the logs of all changes throughout the FRIC tool

Computed

Name	Description
titleStack()	This is used to put the title of the Subtask at the top of the page
mounted()	This is used to get the data for the status of Tasks, Subtasks, and Chart at runtime

Methods

Name	Description
getTaskStatus()	This is used to gather the status' of all the Tasks in the tool
getSubtaskStatus()	This is used to gather the status' of all the Subtasks in the tool
randomChartData()	This is used to fill the Chart with the data of Systems, Findings, and Tasks in the tool

Event □

Events are a period time in which CEAD will test a set of systems. Events have a set of systems to test, along with findings related to them.

> ⓘ http://localhost:8080/#/events

Event Overview Table (Events.vue)

This view provides the general information about the Events, such as the Event Name, Number of Systems, Number of Findings within each Event, Progress of the Event and Created date when the Event was created. The Event Overview Table page is composed of two files, `Events.vue` and `EventOverviewTable.vue`.

Event Name	No. of Systems	No. of Findings	Progress	Created (DD-MM-YYYY hh:mm:ss)
foo	2	3	100%	11-11-2020: 15:13:38
bar	1	5	60%	12-01-2020 14:12:14

Methods:

Name	Description
getEventData()	description goes here
trashModal()	description goes here
displayError()	description goes here
removeRow()	description goes here

trashConfirm()	description goes here
trashCancel()	description goes here
logAction()	description goes here

Event.vue

Methods:

eventHelp	description goes here
onSearchToggle	description goes here

Create Event

http://localhost:8080/#/create-event

Frontend Fields

Event Name	Name of the Event that is being written text box
Event Description	Decription of the Event text box
Event Lead Analyst	Initials of the Lead Analyst for the Event drop down menu
Event Type	Type of Event (CVPA, CVI, VOF) drop down menu
Event Version	The version of the Event text box
Assessment Date	Assessment Datepicker menu
Organization Name	Organization Name text box
Security Classification Title Guide	Title Guide text box
Event Classification	Event Classification drop down menu
Declassification Date	Declassification Datepicker menu
Customer Name	Customer name text box

CreateEvent.vue

Computed:

titleStack()	Defines the path of the page

heroTitle()	Determines the title of the page
heroRouterLinkTo()	Defines where to go when the header button is clicked
heroRouterLinkLabel()	Defines the name of the header button
formCardTitle()	Defines the name of the card

Methods:

Submit()	Method is used to log and create an Event
logAction()	Method is used to create a log for the creation of an Event
displayError()	Method is to display an error

Event Detailed View

http://localhost:8080/#/events/:id

:id is to indicate the unique identifier for the Event artifact

Frontend Fields

Event Name	Name of the Event that is being written text box
Event Description	Description of the Event text box
Event Lead Analyst	Initials of the Lead Analyst for the Event drop down menu
Event Type	Event type (CVPA, CVI, VOF) drop down menu
Event Version	The version of the Event text box
Assessment Date	Assessment Datepicker menu
Organization Name	Organization Name text box
Security Classification Title Guide	Title Guide text box
Event Classification	Event Classification drop down menu
Declassification Date	Declassification Datepicker menu
Customer Name	Customer name text box
Derived From	Derived From text box

EventForm.vue

Computed:

titleStack()	Defines the path of the page
heroTitle()	Determines the title of the page, in this case it will show the name of the Event
heroRouterLinkTo	Defines where to do when clicking the header button
heroRouterLinkLabel	Defines the name of the header button
formCardTitle	Defines the name of the card
created	

Methods:

add()	Method is to add Lead Analysts and Analysts to the Event
getOldData()	Method is used to get current data on the Event
getData()	Method is get data from user when editing an Event
submit()	Method is log and save changes to the Event
logAction()	Method is to create a log of the change in Event
displayError()	Method is to display an error

Systems □

This view is made up using two files `Systems.vue` and `SystemOverviewTable.vue`

ⓘ http://localhost:8080/#/systems

System Overview Table (Systems.vue)

This view provides the general information about the Systems, such as the System Name, Number of Tasks, Number of Findings within each System, Progress of the System and Created date when the System was created. The System Overview Table page is composed of two files, `Systems.vue` and `SystemOverviewTable.vue`

System Name	No. of Tasks	No. of Findings	Progress
Foo	2	3	100%
Other	1	5	60%

Computed Values

Name	Description
trashObjectName	Returns the name of the name of the system which will be deleted

Methods

Name	Parameter	Description
data()		Is the private memory of each component where you can store any variables you need.

created()		Called synchronously after the instance is created.
getSystemData()		Load all systems and renders it to the table
getEventData()		Gets the current event from the database.
deleteSystem()		Removes the selected system from the database
trashModal()		Activates the trash model, which then displays a confirmation.
displayError()	e	Handles displaying the error with its proper message. Where 'e' is an object that contains the message (e.message)
removeRow()		Handles removing the instance of the item from the table.
removeFromEvent()		Removes the selected system from the current event.
trashConfirm()		Gets called when user confirms action from snackbar, which then removes system from database and from table.
trashCancel()		Gets called when user cancel action from snackbar and table information stays the same.
logAction()		Logs user's action after archiving a system by storing it into the database.
removeItemAction()		Checks if user is on the System page or Archive page and returns 'System Archive' or 'Delete System' as a string.
iconType()		This is used to display an icon depending if user is on the Systems page or Archive page
setRemoveItemColor()		This is used to display the color of either delete system or archive system button depending if user is on the Systems page or Archive page

System.vue

Methods:

Name	Description
eventHelp()	Displays an alert with helpful information about the systems artifact
onSearchToggle()	Loads results after a search term is entered

Create System

http://localhost:8080/#/createsystem-form

Editable Fields (CreateSystem.vue)

Name	Description
ID	A unique identifier for the System
System Name	Name of a system
System Description	Description of a system
System Location	The location where a system is physically located
System Router	The router that are accessed
System Switch	Switches that are accessed
System Room	Rooms that are accessed
Test Plan	Name of the test plan
Confidentiality	Confidentiality level of a system
Integrity	Integrity level of a system
Availability	Availability level of a system

Computed Values

Name	Description

titleStack()	Determines the header label of the page. e.g Analyst/ Task
heroTitle()	Determines the title of the page (task, subtask,etc)
heroRouterLinkTo()	Defines where to go when the header button is clicked
heroRouterLinkLabel()	Defines the name of the header button
formCardTitle()	Defines the labels above the overview table

Methods

Name	Parameters	Description
submit()		Submits the changes of the system if any of the fields have been changed.
logAction()		Shows the changes from the old system form to the new one, to display on the home page
displayError()	e	Used to displaying the error with its proper message. Where 'e' is an object that contains the message (e.message)
createSystem()		Its responsible for submitting the form to the database
addSystem()		Its responsible for associating itself with the current event.
getEvent()		Loads the current event and stores the current id of the event

System Detailed View

http://localhost:8080/#/systems/:id

Where `:id` is to indicate the unique identifier for the System artifact

Editable Fields (SystemForm.vue)

Field	Description
ID	A unique identifier for the System
System Name	Name of a system
System Description	Description of a system
System Location	The location where a system is physically located
System Router	The router that are accessed
System Switch	Switches that are accessed
System Room	Rooms that are accessed
Test Plan	Name of the test plan
Confidentiality	Confidentiality level of a system
Integrity	Integrity level of a system
Availability	Availability level of a system

Computed Values

Name	Description

titleStack()	Determines the header label of the page. e.g Analyst/ Task
heroTitle()	Determines the title of the page (task, subtask,etc)
heroRouterLinkTo()	Defines where to go when the header button is clicked
heroRouterLinkLabel()	Defines the name of the header button
formCardTitle()	Defines the labels above the overview table
created()	Triggers a load of the same artifact (system) when the page renders. Used to keep one with the old data and the other for updating the changes.

Methods

Name	Parameters	Description
getData()		Stores the current information of the System after it is edited
getOldData()		Stores the information of the System before it was edited
submit()		Submits the changes of the system if any of the fields have been changed.
logAction()		Shows the changes from the old system form to the new one, to display on the home page
displayError()	e	Used to displaying the error with its proper message. Where 'e' is an object that contains the message (e.message)

Tasks ✔

ⓘ http://localhost:8080/#/tasks

Task Overview Table (Tasks.vue)

This view provides the general information about the Task, such as the title of the task, the system the task is associated to, the analyst that is working on the task, the priority of a task, the overall progress of a task, the no. of subtask's that is related to the task, the no. of findings that is related to the task, tand he Due Date (DD-MM-YYYY) of the task.

Title	System	Analyst	Priority	Progress	No. of Subtasks	No. of Findings	Due Date (DD-MM-YYYY)
Title01	System01	k.a	Medium	Complete	5	5	12-01-2020
Titile02	System01	j.b	In Progress	Transferred	3	4	12-20-2020

Methods

Name	Description:
taskHelp()	Displays the helper text when clicking the help icon
onSearchToggle()	Allows the user to search for a task name
getTaskData()	Load all taks information and render it to the table

Create Task

ⓘ http://localhost:8080/#/createtask-form

Frontend Fields (CreateTask.vue)

Field	Description
Title	The title/name of a particular task
Description	A short description of the task to be completed
Analysts for Task	The analyst(s) that are working on the task
Analyst	The analyst(s) that are playing the role of a collaborator
Related Task	A task that has a relation with the current task
System For Task	A system that has a relation with the current task
Task Priority	Describes the priority/urgency of the task
Task Progress	Describes the task overall progress
Subtasks	The number of subtasks that are related to that task
Findings	The number of findings that are related to that task
Due Date	The timestamp of when a task should be complete by

Computed values

Value	Description
titleStack()	Determines the header label of the page. e.g Analyst/ Task
heroTitle()	Determines the title of the page (task, subtask,etc)

heroRouterLinkTo()	Defines where to go when the header button is clicked
heroRouterLinkLabel()	Defines the name of the header button
formCardTitle()	Defines the labels above the overview table

Methods

Name	Description:
createTask()	Submits the request to create a task
uploadFiles()	Used to upload evidence to the database
getSystems()	Retrieves the list of systems from the backend
getRelatedTasks()	Returns the name of the tasks related to this particular task
getAnalysts()	Retrives the list of analysts from the backend
logAction()	Logs all changes that was made within Task and stores in database
displayError()	Displays an error message on the frontend

Task Detailed View

http://localhost:8080/#/tasks/:id

Where `:id` is to indicate the unique identifier for the Tasks artifact

Frontend Fields (TaskForm.vue)

Field	Description
Title	The title/name of a particular task
Description	A short description of the task to be completed
Analysts for Task	The analyst(s) that are working on the task
Analyst	The analyst(s) that are playing the role of a collaborator
Related Task	A task that has a relation with the current task
System For Task	A system that has a relation with the current task
Task Priority	Describes the priority/urgency of the task
Task Progress	Describes the task overall progress
Subtasks	The number of subtasks that are related to that task
Findings	The number of findings that are related to that task
Due Date	The timestamp of when a task should be complete by

Computed values

Name	Description

titleStack()	Determines the header label of the page. e.g Analyst/ Task
heroTitle()	Determines the title of the page (task, subtask,etc)
heroRouterLinkTo()	Defines where to go when the header button is clicked
heroRouterLinkLabel()	Defines the name of the header button
formCardTitle()	Defines the labels above the overview table

Methods

Name	Description
getOldData()	Stores the information of the task before it was edited
getData()	Sores the current information after the task has been edited
input()	Triggers actions when button to submit is clicked
getSystems()	Retrieves the list of systems from the backend
getRelatedTasks()	Grabs the list of tasks that were related to this current task
getAnalysts()	Retrieves the list of analysts from backend
logAction()	Logs any changes that have been made toTask and stores it in backend and shows a change history
displayError()	Displays an error message on the frontend

Subtasks □

ⓘ http://localhost:8080/#/subtasks

Subtask Overview Table

This view provides the general information about the Subtasks, such as the Subtask Name, Number of Tasks, Assigned Analyst, Progress of the subtask, number of Findings, Due Date of each Subtask. The Subtask Overview Table page is composed of two files, `Subtasks.vue` and `SubtaskOverviewTable.vue.`

Title	Task	Analyst	Progress	No. of Findings	Due Date (DD-MM-YYYY)
foo	2	A.O	Not Started	2	11-11-2020
bar	1	A.O	In Progress	1	12-01-2020

Methods:

Name	Description
getSubtaskData()	This is used to grab substask information from the Backend
trashModal()	This is used for the archive button for each Subtask
removeRow()	This is used to clear the row of an archived Subtask from the table
trashConfirm()	This is used to confirm if the Subtask has been removed
trashCancel()	This is used to cancel a Subtask from being removed
logAction()	This is used to log actions that happen with a Subtask

displayError(e)	This is used to display any error message that may occur
removeItemAction()	This is used to check if we are on the Subtasks page or Archive page
iconType()	This is used to display an icon depending if we are on the Subtasks page or Archive page
setRemoveItemColor()	This is used to display a color depending if we are on the Subtasks page or Archive page

Subtask.vue

Subtask Methods:

Name	Description
subtaskHelp()	This displays summery about Subtasks in the info pop up
onSearchToggle()	This is used to search for the subtasked typed in the search bar

Create Subtask

ⓘ http://localhost:8080/#/createsubtasks-form

Frontend Fields

Field	Description
Title	Text field for Title of the Subtask
Description	Text field for Description of the Subtask
Progress	Dropdown with progress options
Due Date	Date Picker with full working calander
Attachments	Way to attach artifacts to a Subtask
Analyst(s)	Dropdown with selection of analysts assigned to event
Collaborator(s)	Dropdow with a selection of analysts available to collaborate
Task(s)	Dropdown with a selection of Tasks to relate the Subtask to
Subtasks(s)	Dropdown with a selection of Subtasks

CreateSubtask.vue

Computed:

Name	Description
titleStack()	This is used to put the title of the Subtask at the top of the page
heroTitle()	This is used to put the title of the Subtask on the hero bar
heroRouterLinkTo()	This is used to return link to Subtasks

heroRouterLinkLabel()	This is used to return to previous page
formCardTitile()	This is used to return the Form Title
created()	This is used to get Tasks

Methods:

Name	Description
input()	This is used to make a readable date
submit()	This is used to log and create Subtask
getTasks()	This is used to return Task Titles
logAction()	this is used to create a log of the creation of the Subtask
displayError()	This is used to display any errorr message that may occur

Subtask Detailed View

> ⓘ http://localhost:8080/subtaskform/:id

Where `:id` the unique identifier of the subtask

Frontend Fields

Field	Description
Title	Text field for Title of the Subtask
Description	Text field for Description of the Subtask
Progress	Dropdown with progress options
Due Date	Date Picker with full working calander
Attachments	Way to attach artifacts to a Subtask
Analyst(s)	Dropdown with selection of analysts assigned to event
Collaborator(s)	Dropdow with a selection of analysts available to collaborate
Task(s)	Dropdown with a selection of Tasks to relate the Subtask to
Subtasks(s)	Dropdown with a selection of Subtasks

SubtaskForm.vue

Name	Description
titleStack()	This is used to put the title of the Subtask at the top of the page
heroTitle()	This is used to put the title of the Subtask on the hero bar
heroRouterLinkTo()	This is used to return link to Subtasks

heroRouterLinkLabel()	This is used to return to previous page
formCardTitile()	This is used to return the Form Title
created()	This is used to get Tasks, Subtask old and new data, and Analysts

Methods:

Name	Description
input()	This is used to make a readable date
submit()	This is used to log and create Subtask
getTasks()	This is used to return Task Titles
logAction()	this is used to create a log of the creation of the Subtask
displayError()	This is used to display any errorr message that may occur
getOldData()	This is used to get current data on the Subtask
getData()	This is used to get the data being generated when the user edits a Subtask
getSubtasks()	This is used to grab the Subtask being editied
id()	This is used to grab the individual ID of a Subtask from the database

Findings □

Findings are vulnerabilities. A finding will either lead to a true vulnerability or just a data point that the client should be made aware of.

This view is made up using two files `Findings.vue` and `FindingOverviewTable.vue`

http://localhost:8080/#/findings

Finding Overview Table

This view provides the general information about the Finding, such as the title of the finding, its ID, the system/task/subtask that it links to, the analyst that created it, the current status of the finding, its classification, the type of finding, and the risk associated to it.

Title	ID	System	Task	Subtask	Analyst	Status	Classification	Type	R
foo	78x7	bar	t1	sub2	k.a	open	information	Encryption	V

Reports

This view provides the user the ability to download and generate reports with the data stored in the database. By clicking on any one of the three buttons `ERB Report` `Risk Matrix` or `Final Report` there is an associated report that is downloaded to the machine.

To test generating a report try out the following command

```
node FRIC-team8-SBSG/frontend/src/reports/<report>/<report_file.js> && open
```

Where

- `report` is the folder of the type of report you want to create, such as erb-report, risk-matrix, final-report.
- `report_file.js` is the code itself that will generate the report, such as erb_report.js, final*report.js, riskmatrix*.js.
- `report_name `is the report file name with its appropriate extension, docx, xlsx, pptx

Methods

Name	Parameter	Type	Description
onGenerateReport()	report	Object	Generates the appropriate report based on the required type triggered by user.
findingHelp()			Triggers an alert to be displayed, showing the user what the page is about.
onSearchToggle()			Loads search results when a user types a value in the search box.

Create Finding

http://localhost:8080/#/create-finding

Frontend Fields (CreateFinding.vue)

Field	Description
ID	A unique identifier for the finding
Title	A title/name for the finding
Host Name	The host where the finding was documented
IP Port	The IP and the Port in which the finding was discovered
Description	A short description of the finding
Long Description	A longer description of the finding with more details
Status	The current status of the finding
Type	The type of vulnerability that was found
Classification	The classification of the finding
Related Findings	A potential relationship to another finding
Evidence	A file system for analysts to upload evidence media for the finding
System	A potential relationship to a system in which the finding was discovered
Task	A potential relationship to a task in which the finding was discovered

Subtask	A potential relationship to a subtask in which the finding was discovered
Confidentiality	The value of confidentiality for the finding
Integrity	The value of integrity for the finding
Availability	The value of availability for the finding
Analyst	The analyst which will be used for logging the finding
Collaborator	An analyst that worked on the finding with the original analyst
Posture	The location of the analyst under which the finding was discovered
Mitigation Description	A brief sentence that describes a potential mitigation of the issue
Mitigation Long Description	A longer, more detailed text aread to explain the mitigation
Relevance	How relevant the finding is to the system under test
Effectiveness Rating	A value of the countermeasure for the finding
Impact Description	Describes how impactful the finding is to the system under test
Impact Level	How important and need of attention the finding is
Risk	A value for the risk the finding presents
Likelihood	A value that describes how likely an issue is to arise

Computed values

Value	Description
titleStack()	Determines the header label of the page. e.g Analyst/ New Finding

heroTitle()	Determines the title of the page (task, subtask,etc)
heroRouterLinkTo()	Defines where to go when the header button is clicked
heroRouterLinkLabel()	Defines the name of the header button
formCardTitle()	Defines the labels above the overview form

Methods

Name	Description:
submit()	Submits the request to create a finding
displayError()	Used to display an error log, should anything go wrong when submitting the form
getSystems()	Returns the list of systems to be related to the finding
getTasks()	Returns the list of tasks to be related to the finding
getSubtasks()	Returns the list of subtasks to be related to the finding
getFindings()	Returns the list of findings to be related to the finding
getAnalysts()	Loads the analysts that could be linked to the finding, as a collaborator
logAction()	Creates transactions of the creation of a finding

Finding Detailed View

> http://localhost:8080/#/findings/:id

Where `:id` is to indicate the unique identifier for the Finding artifact

Editable Fields (FindingForm.vue)

Field	Description
ID	A unique identifier for the finding
Title	A title/name for the finding
Host Name	The host where the finding was documented
IP Port	The IP and the Port in which the finding was discovered
Description	A short description of the finding
Long Description	A longer description of the finding with more details
Status	The current status of the finding
Type	The type of vulnerability that was found
Classification	The classification of the finding
Related Findings	A potential relationship to another finding
Evidence	A file system for analysts to upload evidence media for the finding
System	A potential relationship to a system in which the finding was discovered

Task	A potential relationship to a task in which the finding was discovered
Subtask	A potential relationship to a subtask in which the finding was discovered
Confidentiality	The value of confidentiality for the finding
Integrity	The value of integrity for the finding
Availability	The value of availability for the finding
Analyst	The analyst which will be used for logging the finding
Collaborator	An analyst that worked on the finding with the original analyst
Posture	The location of the analyst under which the finding was discovered
Mitigation Description	A brief sentence that describes a potential mitigation of the issue
Mitigation Long Description	A longer, more detailed text aread to explain the mitigation
Relevance	How relevant the finding is to the system under test
Effectiveness Rating	A value of the countermeasure for the finding
Impact Description	Describes how impactful the finding is to the system under test
Impact Level	How important and need of attention the finding is
Risk	A value for the risk the finding presents
Likelihood	A value that describes how likely an issue is to arise

Computed values

Name	Description

titleStack()	Determines the header label of the page. e.g Analyst/ Task
heroTitle()	Determines the title of the page (task, subtask,etc)
heroRouterLinkTo()	Defines where to go when the header button is clicked
heroRouterLinkLabel()	Defines the name of the header button
formCardTitle()	Defines the labels above the overview table
created()	Triggers a load of different other artifacts (task, subtask, systems) when the page renders. Used to relate artifacts to the finding

Methods

Name	Description
getOldData()	Stores the information of the finding before it was edited
getData()	Stores the current information of the finding after it is edited
logAction()	Shows the changes from the old finding form to the new one, to display on the home page

Archive □

Archives are a sort of snapshot of artifacts of FRIC. This feature is used to store away artifacts such as findings, tasks, systems, etc for potential use at a later time.

ⓘ http://localhost:8080/#/archive

Archive Page (Archive.vue)

This page displays the overview table of the artifacts of FRIC in order to archive any of them, or restore them. This page displays

- Task overview table
- Subtask overview table
- Findings overview table
- Systems overview table

In addition, when the user clicks on the *New Archive* button in the top right, individual artifacts can be exported to the user's computer, documented below.

Export Data (ExportData.vue)

Users can select any of the following artifacts to directly download them from the database to their computer

Button Label	Description	File Downloaded
Systems	Exports all of the current systems in the database	systems*dateTime.json*
Tasks	Exports all of the current tasks in the database	*tasksdateTime.json*

Subtasks	Exports all of the current subtasks in the database	*subtasksdateTime.json*
Findings	Exports all of the current findings in the database	*findingsdateTime.json*
Transactions	Exports all of the current transaction logs in the database	*transactionsdateTime.json*
All	Exports all of the current data in the database	

To test the exporting of all data you can run the following command

```
mongodump -d db -o Archive
```

> This will dump everything from the db called `db` and then output it to a folder called Archive

Methods

Name	Parameter	Description
onExportFinish()	{ systems | tasks | subtasks | findings | transactions}	Exports the parameter data and generates the file to the user, alerting them on the frontend that it has been exported
archiveAll()		Exports all data, reference the code snippet above

Configuration Content □

http://localhost:8080/#/configuration-content

Methods:

Name()	Description
onEditTable()	Method is to display when table has been updated
configurationHelp()	Displays a modal showing help information

Setup ⚙

ⓘ http://localhost:8080/#/setup-content-form

Setup Content (SetupContentForm.vue)

The page will prompt the analyst for their initials to be stored on the database and used as the current user.

The page them prompts the user to select whether they want to create a new event or if they will be performing a sync.

Methods

Name	Description
submit()	Displays a message when the user submits
reset()	Clears the submitted values
synced()	Determines if a sync should be started

Help i

http://localhost:8080/#/help-page

In order to update the content of this page, the changes need to be made to the `HelpContent.vue` file if you wish to change the text being displayed on the page.

Props

Name	Description	Type
icon	takes in a material icon to show on the page	String
title	takes in the text to display on hover	String

Methods

Name	Description
emitGoBack()	returns you to the previous picture

FRIC Git Service

Run service with Docker container □

FRIC needs this to be ran in a docker container. We have not tested it any other way and Docker has been working excellently.

In an effort to scale the application and be able to archive data efficiently and be able to restore from it. FRIC uses a free and open-source git service known as **Gogs** which can be self-hosted and completed customized.

This container can be ran on any linux-based machine that is connected to the local offline network. Even on a Raspberry Pi (which is what we used)

> To learn about Gogs check out their official repository here
> https://github.com/gogs/gogs

□To run Gogs using Docker follow these steps

Ensure you have docker installed on your machine with `docker -v`

1. Pull the latest image with the command `docker pull gogs/gogs`
2. Create a local volume where the data will go, use the command `mkdir -p /var/gogs`
3. Run the docker image with

```
docker run --name=fric-gogs-server -p 10022:22 -p 10080:3000 -v /var/gogs:/d
```

- You can now start the container instance with `docker start fric-gogs-server`

You should now be able to go through the initial setup of the service by visiting `http://ip:10022/` and just replace the word **IP** with whatever the IP address of your host machine is.

> Note that the name of the container, the ports, and the volume location can all be changed if need be. For more details visit https://github.com/gogs/gogs/tree/main/docker

ARM vs 64-bit Install

It is important to note, that the instructions above show how to run Gogs in the general case where there is a full linux machine to host the service on which is usually 64 bit. We used a Raspberry Pi, which is an ARM Architecture so there was a different process to install docker and a different version of **gogs** that had to be installed.

To run Gogs on Raspberry Pi devices follow these steps

1. Download the latest version of the *hypriot* operating system through this link https://blog.hypriot.com/downloads/
    - *hypriot* is a free and open-source, specially-built operating system designed to run Docker on ARM devices such as the Raspberry Pi.
2. Flash this image onto an SD card using a tool like Win-32 Disk Imager, Belena Etcher, or the `dd` command on Linux.
3. Once the image is loaded to the SD card, place the SD card into the Raspberry Pi and power it on.
4. Log in to the raspberry pi using the default credentials `pirate` and `hypriot` as username and password, respectively. **Note** that these might change in the future, please reference the documenation for the hypriot operating system here https://blog.hypriot.com/faq/ to see if the default credentials have been changed.
5. You should now be able to check that docker is installed with `docker info`
6. Now we can install gogs with

```
docker run -d --name fric-gogs-server --publish 8022:22 --publish 3000:3000
```

> The main difference in the docker command is that the Gogs image needs to be a different build because of the ARM architecture, so we install the Raspbian version

Success □

you should now be able to open a browser, visit the IP of the Raspberry Pi, followed by port `3000` in this case, and see the setup page for Gogs.