

# CyberSec Certificate Tracker

## The Neighbors

Aayush Gautam

Anuska Pant

Mesbaul Khan

Sabina Adhikari

Yatin Gupta

Clients: John Romero, Kelly Ragusa

## Project Summary

Our clients John Romero and Kelly Ragusa from the Department of Cybersecurity, Texas A&M University were having issues managing and keeping track of the progress and certifications of students who are involved in various cybersecurity courses. To organize the information that comes from different portals, our clients wanted a functional web application platform for the administrators. They wanted to view all the student records and the relevant course details in an organized tabular format. They wanted a portal to upload CSV file which contains the student information, course information, vendors and many other necessary fields and a link to manually add that information to the system via a customized form. One of the major requirements of our clients was to perform a super search and extract the CSV file which contains the information that is shown on the screen.

We have a functioning web application software where the main users are the admins and the users. The admins have access to add student information, courses, vendors, and companies manually. In addition, they can upload the CSV file that contains information on all the necessary data fields and extract the CSV file with the data shown on the screen. However, on the other hand, the users have only a read-only feature where they can find the information but can make no changes to the database. Both the users and admins can utilize the super search and filter features.

## User Stories

---

**Feature: Log in to the system**

As the main admin,  
So that I can access the system with read/write privileges  
I want to securely login to the application

**Story Points:** 2

**Status:** Completed

The clients wanted a simple email-id and password login system since the application will be deployed in a private VPN anyways.

## Log in

Email

Password

### Feature: View the data in tabular format

As the general/main admin

So that I can keep track of all the records

I want to view the records in an organized tabular format

**Story Points:** 4

**Status:** Completed

This story involved receiving the data from the backend and displaying it in a tabular format with relevant columns for the client shown in front.

Full Name	Canvas Id	Title	Company Name	Email Address	Course Name	Registration I
<input type="text" value="Search 350 records..."/>	<input type="text" value="Search 350 records..."/>	<input type="text" value="Search 350 records..."/>	<input type="text" value="Search 350 records..."/>	<input type="text" value="Search 350 records..."/>	<input type="text" value="Search 350 records..."/>	<input type="text" value="Search 350 records..."/>
Sophia Davis	297248897		Netflix	sophia.davis@example.com	DCLDP - 2021	2012-10-16
Sophia Davis	297248897		Netflix	sophia.davis@example.com	O-CYBER: CompTIA Security+ Prep Course for DCLDP (501)	2012-09-07
Sophia Davis	297248897		Netflix	sophia.davis@example.com	O-CYBER: Cybersecurity Fundamentals for Teachers Summer Camp	2016-03-06
Sophia Davis	297248897		Netflix	sophia.davis@example.com	O-CYBER: CompTIA Network+ Certificate Prep (N10-007)	2009-04-11
Michael Jackson	642171178		Salesforce	michael.jackson@example.com	O-CYBER: CompTIA Security+ Certificate Prep (SY0-501)	2015-08-05
Michael Jackson	642171178		Salesforce	michael.jackson@example.com	DCLDP - 2021	2016-07-20
Michael Jackson	642171178		Salesforce	michael.jackson@example.com	O-CYBER: CompTIA Security+ Prep Course for DCLDP (501)	2012-08-10

### Feature: Super search the data

As the general/main admin,  
So that I can find specific entries by word  
I want to **search** for students, courses, etc.

**Story Points:** 6

**Status:** Completed

This story involved creating a common search field that can traverse all columns in one search.

### Examples:

#### *Searching by Company*

Search:  Reset Filters Download CSV

Full Name	Canvas Id	Title	Company Name
<input type="text" value="Search 43 records..."/>	<input type="text" value="Search 43 records..."/>	<input type="text" value="Search 43 records..."/>	<input type="text" value="Search 43 records..."/>
Sophia Davis	297248897		Netflix
Sophia Davis	297248897		Netflix

#### *Searching by Name*

Search:  Reset Filters Download CSV

Full Name	Canvas Id	Title	Company Name
<input type="text" value="Search 9 records..."/>	<input type="text" value="Search 9 records..."/>	<input type="text" value="Search 9 records..."/>	<input type="text" value="Search 9 records..."/>
Geralda Johnsona	258162908		Apple
Geralda Johnsona	258162908		Apple

### Searching by email

Search:

[Reset Filters](#)

[Download CSV](#)

Full Name	Canvas Id	Title	Company Name	Email Address
<input type="text" value="Search 179 records..."/>	<input type="text" value="Search 179 records..."/>	<input type="text" value="Search 179 records..."/>	<input type="text" value="Search 179 records..."/>	<input type="text" value="Search 179 records..."/>
Sophia Davis	297248897		Netflix	sophia.davis@example.com
Sophia Davis	297248897		Netflix	sophia.davis@example.com

#### Feature: Upload data using csv

As the main admin,

So that I can [view/search/filter](#) data I want to [store](#) the data using csv

**Story Points:** 17 (3 – frontend, 4 – CSV type 1, 4 – CSV type 2, 3 – adding an upload modal, 3 bug fixes)

**Status:** Completed

This story involved taking any CSV that client provides and respectively adding data into the database and showing in the frontend. This user story was challenging because the CSVs had no one format, and thus there are many different keys and types of data.

Example: One CSV might have “Name”, while the other would have “First Name and Last Name”. So, for the former CSV, a logic had to be written to separate into first and last name. Similarly, all other keys had differing formats.

#### Feature: Add data through forms

As the main admin,

So that I can enter the [user](#) data through a form-like [interface](#) I want to manually enter the respective data

**Story Points:** 24 (5-CRUD APIs, 15 - Form creations, 4 - Form validations)

**Status:** Completed

**Modification:** This story was modified later. Initially, a form was to be created to only add a student but further it was extended to have a form for all different fields namely student, course, company, certificate vouchers, vendors and users.

This story involved a lot of effort since different forms had to be created for all different fields. They have separate keys, formats, APIs, and even data validation techniques.

**Example:**

**Add Students** [X]

First Name [ ] Last Name [ ]

Email [ ] Canvas Id [ ]

Title [ ] Company [ Select Company v ]

[ Submit ] [ Cancel ]

### Feature: Filter student records by a column

As the general/main admin,

So that I can find specific entry by a column field I want to **filter** students by a column

**Story points:** 7

**Status:** Completed

**Modification:** This story was modified to filter on all columns rather than the earlier version where it can filter only on company name.

In the image below, we can see all columns having their own respective search fields.

First Name	Last Name	Email Address	Courses	Canvas Id	Company
Search 423 records...	Search 423 records...	Search 423 records...	Search 423 records...	Search 423 records...	Search 423 records...

### Feature: View, edit student profile

As a general/main admin,

So that I can get all the information on a specific student I want to **view or edit** the student's profile

**Story points:** 10

**Status:** Completed

**Modification:** Earlier the student profile was supposed to be read-only, but later we added edit functionality as well.

The student profile here shows all information relevant to a student, and option to edit them. Also courses information can be added and viewed as well.

Overview

Update Profile

First Name

Syed

Last Name

Shah

Email

syeddawood.shah93@gmail.com

Company

Unknown




Canvas Id

3084048

Courses

5 items

Add Course

	Canvas Course	Course Completed	Cert Name	Registration Date	Voucher Purchased	Test Re
	O-CYBER: CompTIA Security+ Certificate Prepa (SY0-501)	no		2020-05-21	no	
	DCLDP - 2021				no	
	O-CYBER: Cybersecurity Fundamentals for Teachers					

**Feature:** Navbar to access different components (Students, Exams, Vendor, Users, Company, Courses, Certificates)

As the general/main admin,  
So that I can access or view the information that I am looking for

I want to find the detailed information about each component

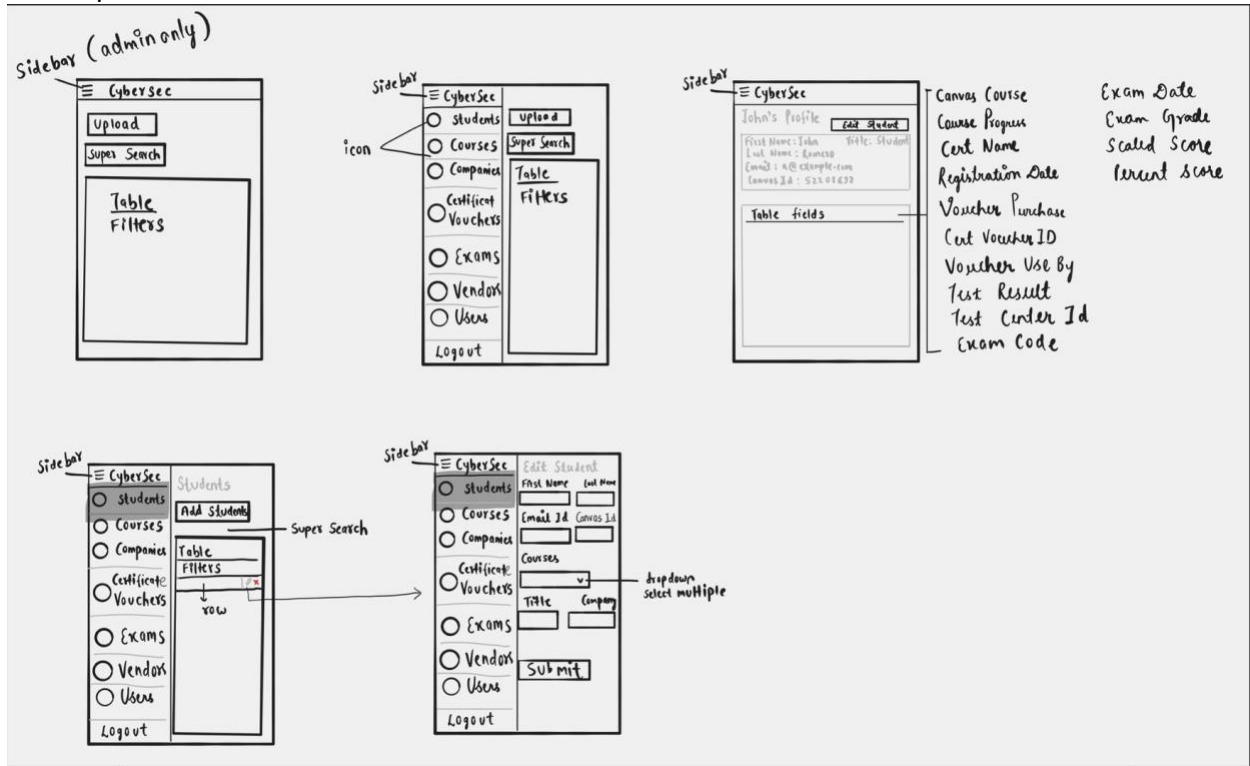
**Story points:** 5

**Status:** Completed

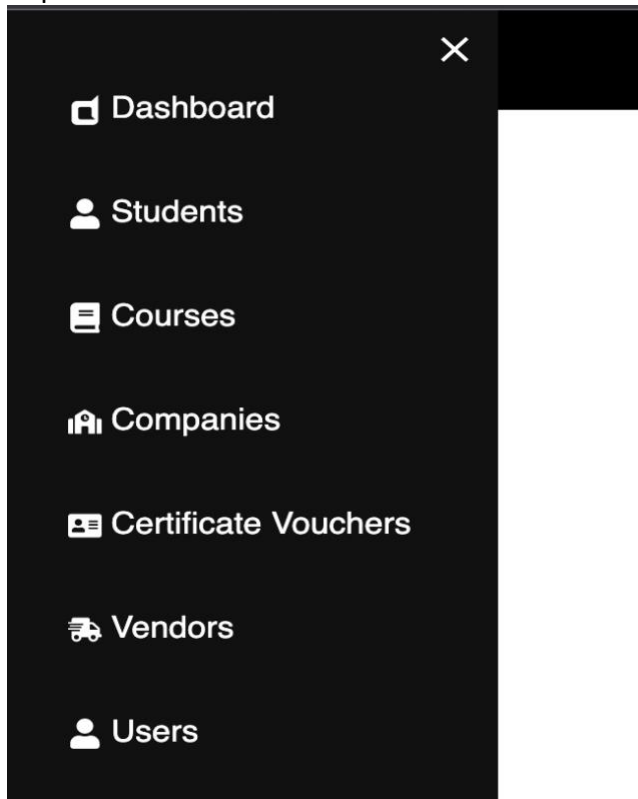
**Modification:** This story was not initially planned but decided upon after the 1<sup>st</sup> demo with the client

In this story we created different navigation components related to different type of data fields we have.

## Mockup:



## Implementation





**Feature: Export data**

As the main admin,  
So that I can view, save, and share data  
I want to export data into csv format

**Story points:** 6

**Status:** Completed

**Modification:** This story was not initially planned but decided upon after the 1st demo with the client

The clients wanted an export to CSV feature where when they are filtering or looking up any data records, they can easily export them to a CSV for retention or sharing.

**Feature: Edit and delete records**


As the main admin,  
So that I can edit and delete data  
I want to have options on the respective tables

**Story points:** 5

**Status:** Completed

**Modification:** This story was not initially planned but decided upon after the 1st demo with the client.

The client wanted a quick edit and delete option on different tables we have for different fields. It looks like this

First Name	Last Name	Email Address	Courses	Canvas Id	Company	
Search 423 records...	Search 423 records...	Search 423 records...	Search 423 records...	Search 423 records...	Search 423 records...	
Syed	Shah	syeddawood.shah93@gmail.com	1. O-CYBER: CompTIA Security+ Certificate Prepa (SY0-501) 2. DCLDP - 2021 3. O-CYBER: Cybersecurity Fundamentals for Teachers Summer Campa 4. CYBER: CompTIA Security+ Prep Course (SY0-601) 5. O-CYBER: CompTIA Security+ Certificate Prep (SY0-501)	3084048	Unknown	

## Scrum master and product owner

We changed the scrum master and product owner every iteration. Following was the order of roles assumed:

Iteration	Scrum Master	Product Owner
-----------	--------------	---------------

0	Yatin Gupta	Mesbaul Alam Khan
1	Sabina Adhikari	Aayush Gautam
2	Anuska Pant	Yatin Gupta
3	Mesbaul Alam Khan	Sabina Adhikari
4	Aayush Gautam	Anuska Pant
5	Yatin Gupta	Mesbaul Alam Khan

## Iterations Summary

### Iteration 0: 36 points

The iteration 0 involved creating docker, frontend boiler, backend boiler, and setting up Postgres, RSpec and Cucumber. In addition, the database schema was designed and one of the features, login, was completed. The template for the table view page and login flow was achieved too. We used 36 points for our iteration 0.

### Iteration 1: 41 points

For iteration 1, we worked on the feature, and upload CSV in the frontend. At this point, we had multiple CSV files with different fields. We made our feature, upload CSV, compatible to read CSV files with different data fields and store their information on the database. In addition, the form to add data manually was created. Some of the points were utilized to run the cucumber test and create a CI/CD pipeline. One of the important features, super search, and basic filters were functioning in our table in the dashboard. In iteration 1, the setting up of docker was completed.

### Iteration 2: 47 points

An updated system design was created after meeting with our clients. There were multiple bugs on the front end as well as in the back end when uploading CSV files with inconsistent data fields. So, some of the points were utilized to fix these bugs. The API for the user table, auto-complete, and API modification for the student profile and sidebar tables were created in this iteration. The features like edit and delete records in the table, edit form for all the tables, a sidebar with access to index page of all tables and a student profile page were created. Some of the points were utilized to run the cucumber test.

### Iteration 3: 51 points

Some of the few important tasks were successfully implemented during this iteration. The features that were worked on during this iteration were cascade deletion, adding and editing form functionality, adding student course sidebar pages, assigning the permission to edit and delete only to the role: admin, resetting search and filters, creating dummy data for presentation, pagination, CSS fixes for production, CSS fixes to the sidebars and page margins and working on the UI of Student Profile page and add edit functionality. Some of the points were utilized to run cucumber test and check authentication errors for APIs.

### Iteration 4: 48 points

Some of the tasks in this iteration were fixing toaster in the front end, adding calendar dates in the date fields, filtering modifications with autocomplete and forming a display error if field is missing an API call. The cucumber tests and RSpec tests are performed in each iteration. We had the updated schema, so we fixed the backend models and the form fields as per the new schema. The CSV export data feature was implemented during this iteration. In addition, the dummy data model was created to demo the clients in this iteration.

### **Iteration 5: 48 points**

Most of the points in this iteration are utilized for working on the project report, project demo, project poster video and project poster. In addition, the final touches on the web application are given such as fixing CSV upload with respect to the new template provided by the clients and validating the models using in the web applications.

### **Customer Meetings**

#### **1. Initial Meeting: 20<sup>th</sup> September 2022**

In this meeting, we discussed the requirements of the client. The client showed us the rudimentary management system they have using a lot of excel sheets, and what they wanted to have. We went over all their data, and what exactly they are looking for in the application. Most of the user stories were defined in this meeting, the approach to database schema was discussed and a big overview of the application was formed.

#### **2. ERD discussion: 27<sup>th</sup> September**

Based on the initial meeting, we developed an ERD schema and went back to the client to discuss the edge cases and come to a consensus on how the database should look. Since the client has a lot of different data points, we wanted to make sure we are aligned before work is started on designing the database schema.

#### **3. 1<sup>st</sup> Demo: 21<sup>st</sup> October**

All the earlier discussions led to this moment where we came prepared with a proper V1 of the application developed. We showed the client the deployed application on Heroku going through the various features we had. Particularly, we showed the following user stories:

- a. Log in to the system
- b. View the data in tabular format
- c. Upload data using csv
- d. Super search the data

The client was very impressed with the product, and it covered all the major features they wanted. Next up, we presented some of the future ideas we had to make the application better and client also told us some of the other features or improvements they wanted.

#### 4. 2<sup>nd</sup> Demo: 16<sup>th</sup> November

In this meeting, we showed all the features and improvements that they had asked after our last meeting. Particularly, we showed the following user stories:

- a. Add student data
- b. View student profile
- c. Filter student records by company name
- d. Navbar to access different components (Students, Exams, Vendor, Users, Company, Courses, Certificates)

The client was happy with the application. But later we noticed an inconsistency in the database schema due to miscommunication between the team and the client. It was a small error in the schema where the relationship between “exam” and “voucher” was misunderstood. We took note of this and suggested some design improvements to the client that we wanted to take up. The client also discussed with us V2 version of the application that they will work on in the next semester.

Afterward, we implemented all the changes, design improvements, and pending user stories. Due to the busy schedule of our clients, we have been communicating through email to show the finalized product.

#### BDD/TDD process

We followed a pairing process to do BDD/TDD. Instead of one person writing tests as well as the feature together, in our approach one would write the BDD test (cucumber), one would write the system tests (rspec) and finally other would do the feature development.

#### Benefits:

1. This way we removed the biases or blind spots a person may develop for a feature.
2. This also worked in pair programming fashion where 2-3 people are working on 1 feature-related code.
3. Code review became easier since 2-3 people had context on the entirety of the feature

#### Problem:

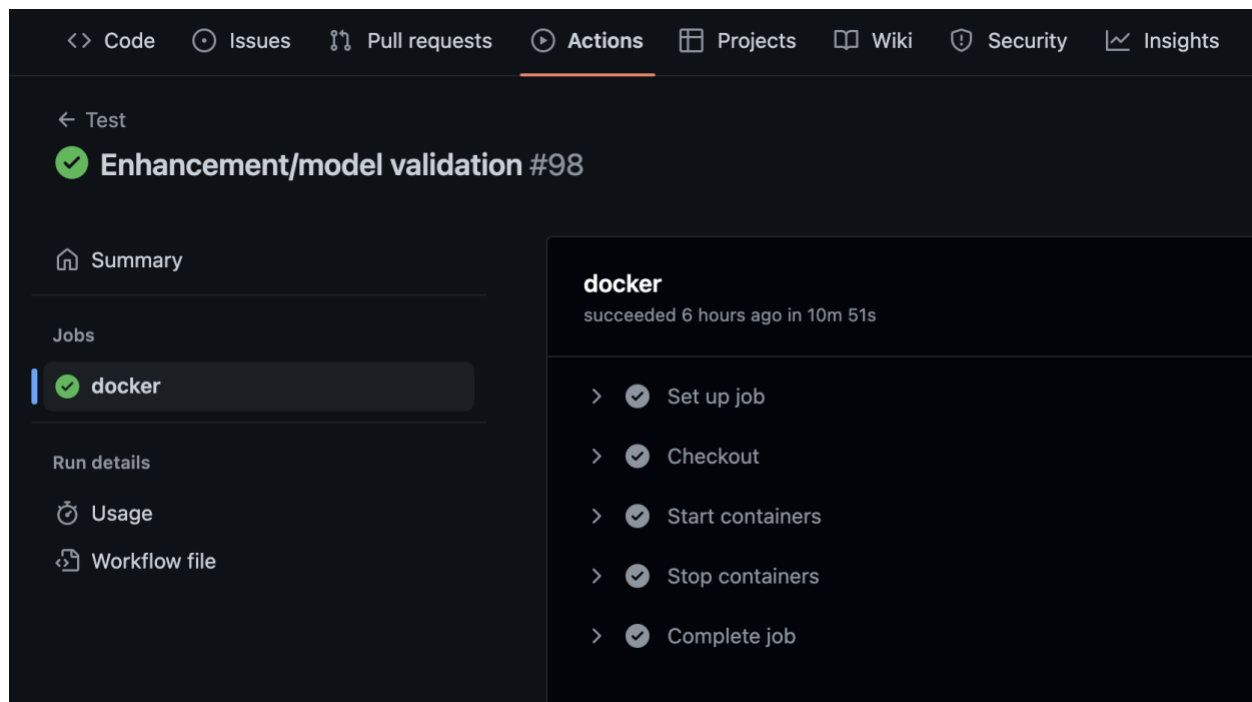
1. Different understandings of same feature caused delays since tests and features were developed by different people
2. Sometimes bad code was pushed since tests were being developed in a separate place

Although our approach had its benefits, the traditional TDD approach where tests are being written with the feature seems much better to avoid bad code causing issues.

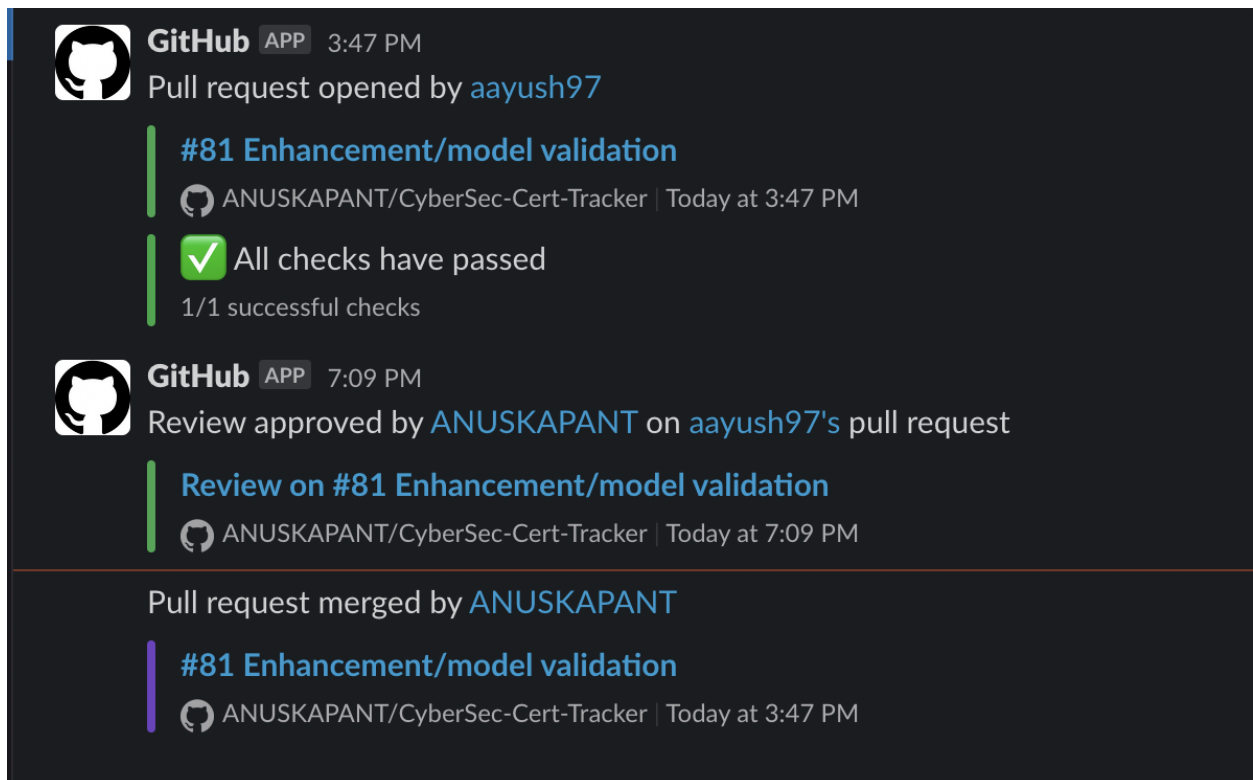
## Configuration and Release Management

We followed the following steps for managing releases:

1. All work was done in a separate feature/bug branch.
2. All Pull requests went through automatic **CI process** using github actions. Our Rspec and Cucumber tests would run in a docker container inside a runner provided by github actions.
3. A Merge request was raised to main which required atleast one approval before it can be merged.
4. If the tests fails or there are any review comments, the developer fixes those
5. If the CI pipeline successfully passes and approver is happy with the merge request, it is merged onto main branch
6. We did 1-2 releases per iteration depending on the changes. We tried to follow fast releases and verify the changes on our production URL.



To be updated on notifications of all merge requests, we integrated slack with GitHub.



## Production Release

There is an issue in the release to Heroku due to tool versions. We are using ruby version 2.7.2 and rails 6.0.6 version. Due to Heroku not supporting this version, we had to use Heroku 18 which has its end of life as April 30<sup>th</sup>, 2023.

This is not an issue with our client since the client wants to deploy the application in their own VPN. To help expedite the process for them, we **dockerized** the entire application so that the client can easily deploy it as a container.

## SimpleCov

To find out the code coverage for our tests, we used SimpleCov. It is a simple tool that succinctly integrates with rspec and cucumber. We only need to run both tests once; it merges the code coverage and gives us the final report. It shows the final code coverage in % as well as the lines we covered or missed. This was useful in identifying features/corner cases we forgot to test.

## All Files ( 95.94% covered at 9.17 hits/line )

64 files in total.

764 relevant lines, 733 lines covered and 31 lines missed. ( 95.94% )

Here the green line shows the successfully tested lines while the red ones show untested lines.

```
4. def create
5.   @file = CsvFile.new(file_params)
6.   if @file.save
7.     @file.create_records(params[:body].path)
8.     respond_to do |format|
9.       format.json { render json: CsvFileSerializer.new(@file).serialized_json, status: :created }
10.    end
11.   else
12.     respond_to do |format|
13.       format.json { render json: @file.errors, status: :unprocessable_entity }
14.     end
15.   end
16. end
17.
```

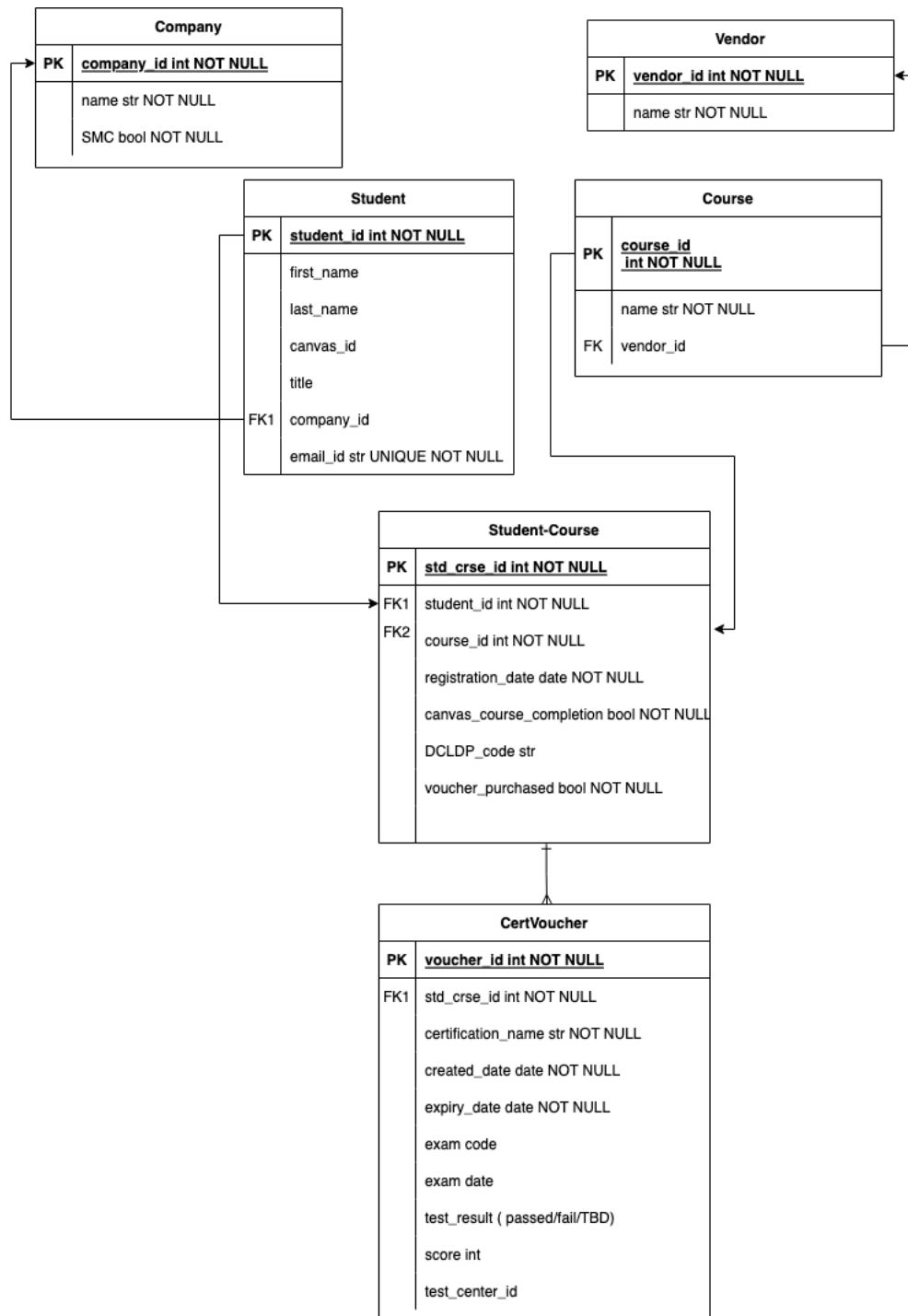
### [Repo contents](#)

We used the ruby-on-rails monolithic approach to design the application. The backend is rails, while in the frontend we replaced normal JavaScript or templating with react in the monolith. Standard rails folder structure is followed with some modifications/additions.

- **.github** folder has the yml file needed to run tests with github actions.
- App represents all the business logic code needed
- The frontend is in react, and code is present in app/javascript
- Documentation has respective tar files done for each iteration
- The features folder has the cucumber tests

For deploying to Heroku, it is just connecting with the project on heroku and running “git push heroku main”. But since our clients wanted to deploy the application in their own VPN, to help speed up the process we dockerized the entire application and added respective docker and docker-compose files in the repo.

## Database ERD



## Links

**Pivotal Tracker:** <https://www.pivotaltracker.com/n/projects/2598723>



**GitHub repo:** <https://github.com/ANUSKAPANT/CyberSec-Cert-Tracker>

**Heroku:** <https://cybersec-cert-tracker.herokuapp.com/dashboard>

For testing: use these credentials

### **Users**

Email: [test@example.com](mailto:test@example.com)

Password: test123

### **Admin**

Email: [admin@example.com](mailto:admin@example.com)

Password: test123

---

Presentation video and demo video: <https://www.youtube.com/watch?v=48T09IRUfqg>

---