

302 Capstone Project

R&D

**Complexity of Zero Trust Architecture measured on
project management applications.**

Frederick Rive

Date: 08/12/2023

Supervisory Team: Rouwa Yalda

Contents

MVP Production	3
Abstract.....	3
Development Log.....	3
Bug Report.....	5
Summary	6
Notion Link.....	12
GitHub Link.....	12
R&D Report	12
Introduction.....	12
Access Management	12
Logs	13
2-Factor Authentication	13
Backups.....	14
Conclusion	15
Supervisor Notes and Signature.....	16
Bibliography	16
Table of Figures	16

MVP Production

Abstract

For my Capstone Project, I continued my study into the complexity added to a Project Management application by the addition of Zero Trust security practises. My focus for this paper was on administrator role functions and privileges, as well as intrusion response systems. A security administrator role is necessary for an applications security system maintenance; however, it also poses a significant risk should it be infiltrated itself. Intrusion response systems are also an important part of any security system. Given that no method of intrusion prevention is perfect, it should be regarded as inevitable that an application will be attacked. Intrusion response systems exist to help identify the attack and the attacker, as well as repair any damages they may have caused to the application.

Development Log

Week	Date	Goals	Review
Week 1	9/10/2023	<ul style="list-style-type: none">• Set up Vue Project• Create website frame• Learn Vue concepts	Vue project set up, succeeded at creating basic frame, css and figuring out Vue v-for lists
Week 2	16/10/2023	<ul style="list-style-type: none">• Create login widget• Create Kanban Board• Connect Node script to Vue script, allow for login and fetching tasks	Node script required some reworks, as it wasn't made to send JSON objects, instead sending strings that I would decode in C++. Once data is being imported, getting the login and kanban boards went smoothly, though each need more CSS styling
Week 3	23/10/2023	<ul style="list-style-type: none">• Complete Kanban Board• Add Task Widget dragging• Add logging system• Add admin page, log viewing	CSS styling for kanban board completed. Logging system was added quickly. Task widget dragging took

			most of the week, and is currently still very buggy. Since task dragging took longer than expected, admin page was not created
Week 4		<ul style="list-style-type: none"> • Add admin page, log viewing • Add admin role • Add 2-factor authentication • Add user management 	Admin page was created with logging, however lacks user management. Admin role added, without 2-factor authentication.
Week 5	30/10/2023	<ul style="list-style-type: none"> • Finish user management • Bugfixes • 2-Factor Authentication 	Admin page has user management added. Most of this week dedicated to fixing bugs from earlier systems. I couldn't get time to work on 2-factor authentication.
Week 6	6/11/2023	<ul style="list-style-type: none"> • Bugfixes • Backups • 2-Factor Authentication • Notifications 	Got system backups working, along with 2-factor authentication and notifications. More bugfixes.
Week 7	13/11/2023	<ul style="list-style-type: none"> • Bugfixes • Encryption • Write R&D Report • Presentation 	Added encryption and hashing to the application, otherwise worked on R&D report and presentation
Week 8	20/11/2023	<ul style="list-style-type: none"> • Bugfixes • Write R&D Report 	Presented, made progress on R&D report
Week 9	27/11/2023	<ul style="list-style-type: none"> • Bugfixes • Write R&D Report 	Worked on R&D Report and bugfixes
Week 10 4/12/2023	4/12/2023	<ul style="list-style-type: none"> • Bugfixes • Password Reset • Write R&D Report 	Finished R&D report

Bug Report

Week	Bug	File	Solution
Week 4	Kanban Widgets are not aligning with mouse when being dragged, instead the widget snaps to have the top left corner on the mouse	App.vue	Added an offset position value, that is equal to the relative position of the mouse upon clicking the widget. All subsequent widget positions include this offset.
Week 4	Kanban dragging has made opening the kanban modal unreliable, and often closes it instantly after opening it	App.vue	Kanban widgets will only be changed into draggable form after the user begins dragging, rather than immediately after clicking.
Week 5	Kanban dragging still unreliable, and the modal doesn't stick to the	App.vue	
Week 6	Data duplication occurs whenever data is saved.		Fixed by wiping the arrays containing the data before reloading them
Week 7	Access is not being restricted properly, regular users are being given access to all data.	Mongoconnector.js	Permission check query was returning an empty array, however permissions were only restricted if it was returning null. Changed to restrict when returning empty array
Week 8	Decrypting fails on certain ciphertexts	App.vue & mongoconnector.js	Sending ciphers with a '+' across HTTP get requests causes the '+' to become a ' ' (space), which naturally breaks the cipher. Currently creating new ciphertexts in a

			loop until the app creates one without a '+', hope to get a better fix in the future
--	--	--	--

Summary

For the first part of this project, I used Qt/C++ for the application development, Node.js for the backend, and MongoDB for the database. Since the Capstone Project paper had different requirements, I had to reconsider the tools I was using for the project. When choosing the tools and frameworks to use, I considered whether they fit with the project guidelines, my own familiarity with them, as well as any previous work I had done with them on the CS301 project.

I couldn't use Qt for the Capstone project, as the paper required a live website for its hand-in. Due to this, I have decided to replace Qt with Vue. Vue is a modern HTML/JavaScript framework used to create dynamic, responsive websites. I had very little experience with Vue, however I was interested in learning it and decided to use it in this project. I found Vue easy to learn and was able to develop new features relatively quickly, however my inexperience with the language caused me to make a lot of mistakes in early development, which caused a constant stream of bugs in later weeks. These bugs caused significant delays in the production of the application.

I continued to use Node.js and MongoDB, as I was familiar with these systems, they did not conflict with the goals of the project, and I had already created functional systems in these for the CS301 Project. Each required a small degree of reworking to fit with Vue, however this was not a significant factor during development.

The CS301 project had issues with its encryption. Initially, I had intended to use Crypto++, a well-regarded encryption library for C++. Unfortunately, this library had become incompatible with Qt, forcing me to look for a replacement. Due to time restraints, I was forced to settle for SimpleCrypt which had been built in Qt and therefore could be added to my project in a short timeframe. This encryption library did not produce sufficiently secure ciphers, so I decided to replace it with a new encryption system for my Capstone Project.

When selecting a new encryption library, I ensured that the library would be compatible with the application I was using. Fortunately, the JavaScript package manager npm, which I was already using for Node.js and Vue, offered a JavaScript library called cryptojs that used AES cryptography procedures. Getting this library to work in the application was significantly easier than it was for the CS301 project.

For project management, I moved to Notion. Notion is an advanced note-taking application that can be used to create Work Breakdown structures and create kanban boards using these WBS. I was

unsatisfied with my previous project management application TeamGantt as I would often forget to check or update it. I was much more successful with Notion, which I already frequently checked for tasks at work.

For web hosting, I used Github Pages. GitHub Pages is a service offered by Git, best known for the version control service GitHub. GitHub Pages allows users to host websites they have uploaded to GitHub. This allowed me to update the live website with minimal extra effort, as the host was based on the GitHub uploads, I was already required to make.

In-action Screenshots

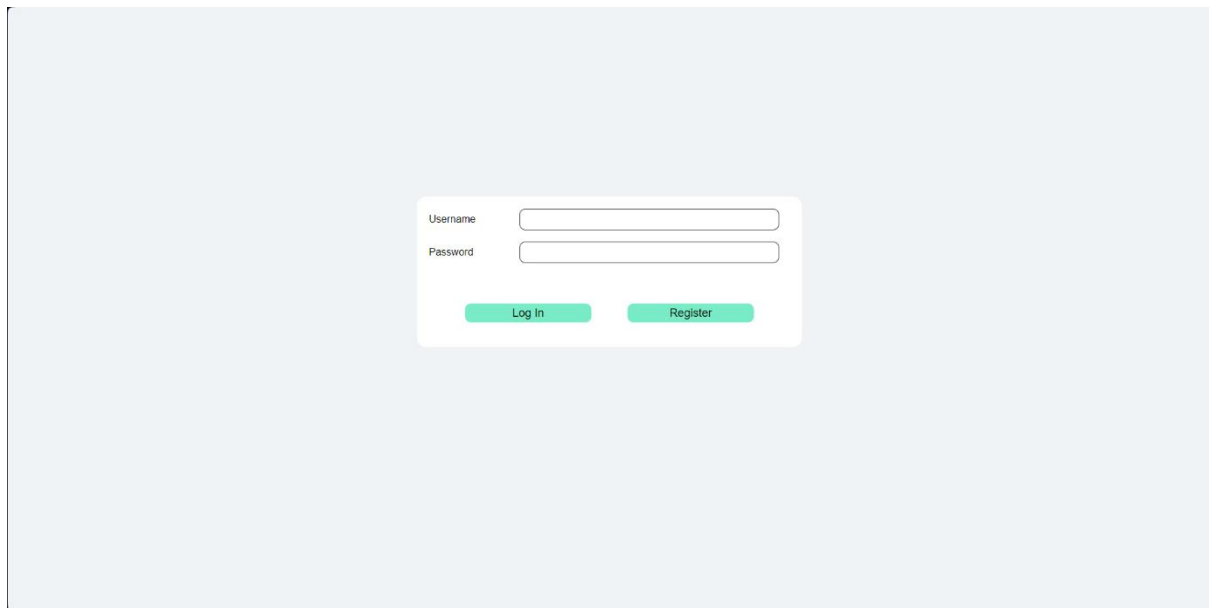
A screenshot of a login page with a light blue background. In the center, there is a white rounded rectangle containing a login form. The form has two input fields: 'Username' and 'Password', each with a light blue border. Below the 'Password' field, there are two green buttons: 'Log In' and 'Register'.

Figure 1: Login Page

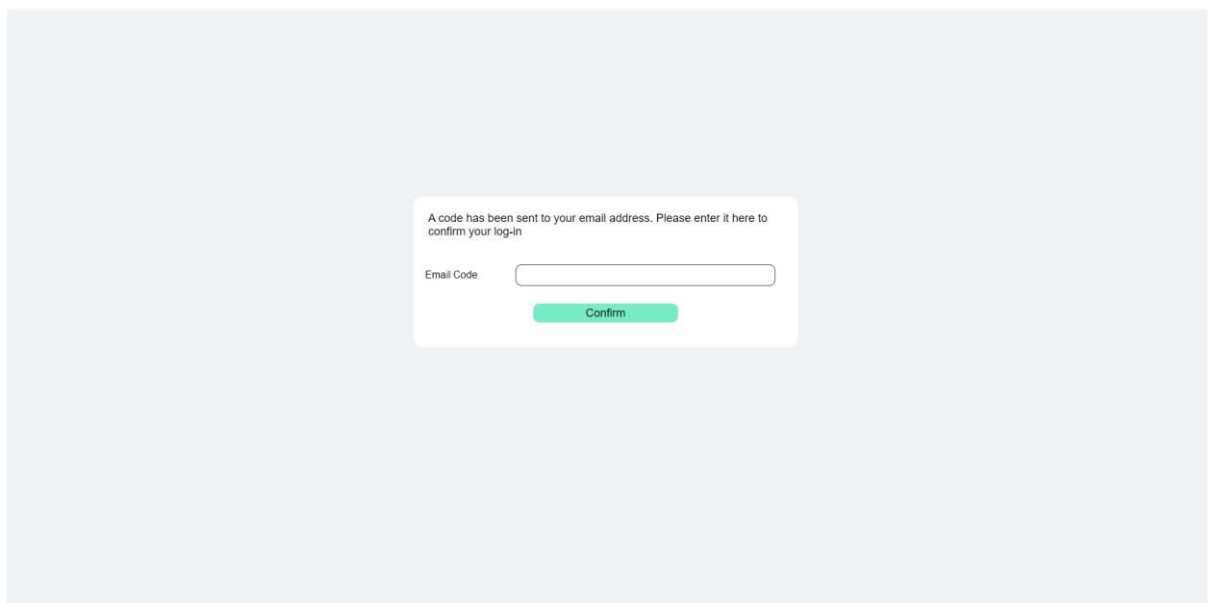
A screenshot of a two-factor authentication page with a light blue background. In the center, there is a white rounded rectangle containing a confirmation form. At the top of the form, it says 'A code has been sent to your email address. Please enter it here to confirm your log-in'. Below this text is an input field labeled 'Email Code' with a light blue border. At the bottom of the form is a green button labeled 'Confirm'.

Figure 2: Two-Factor Authentication Page

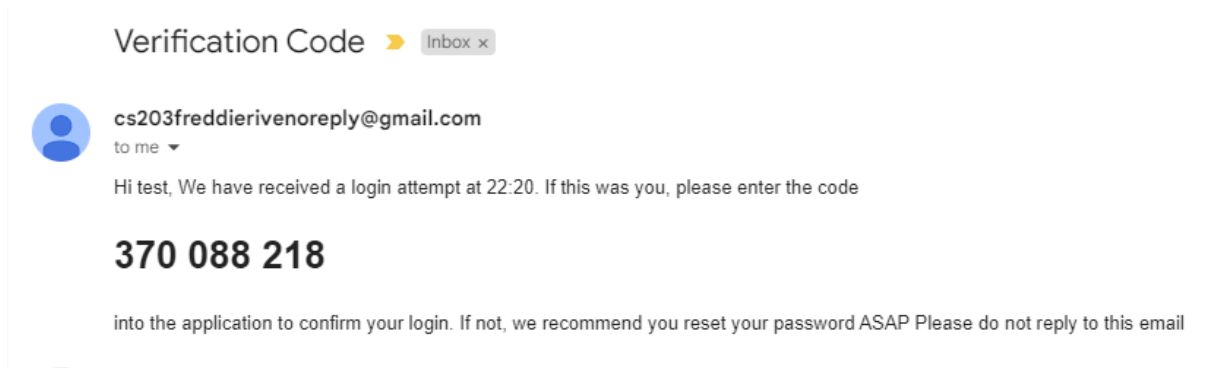


Figure 3: Two-Factor Authentication Email

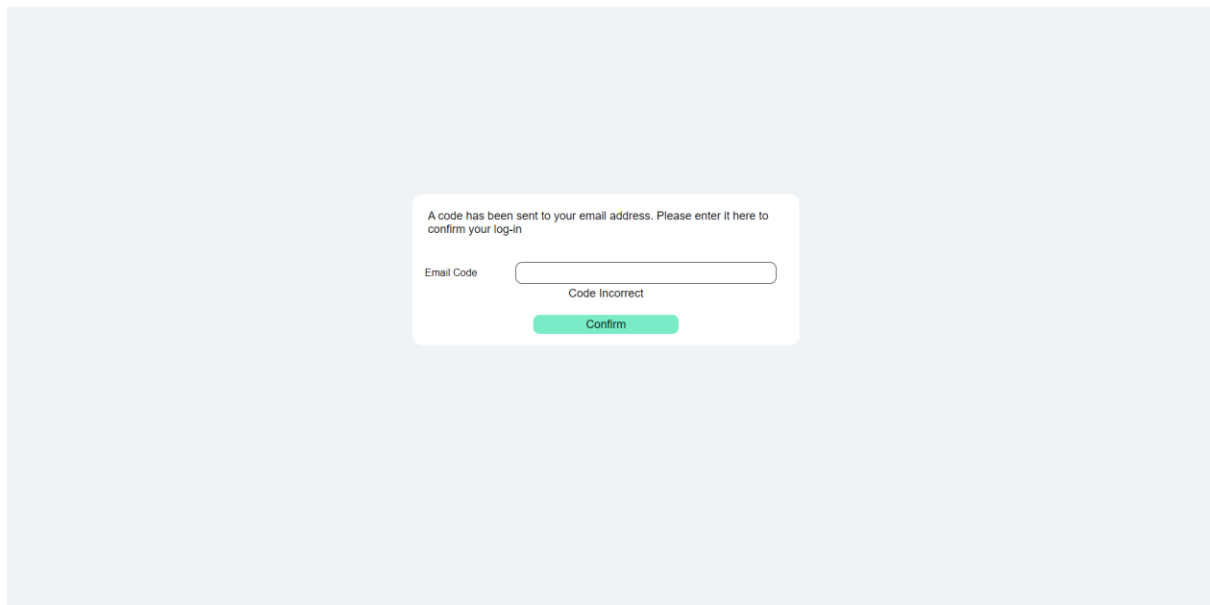


Figure 4: Two-Factor Authentication Failure

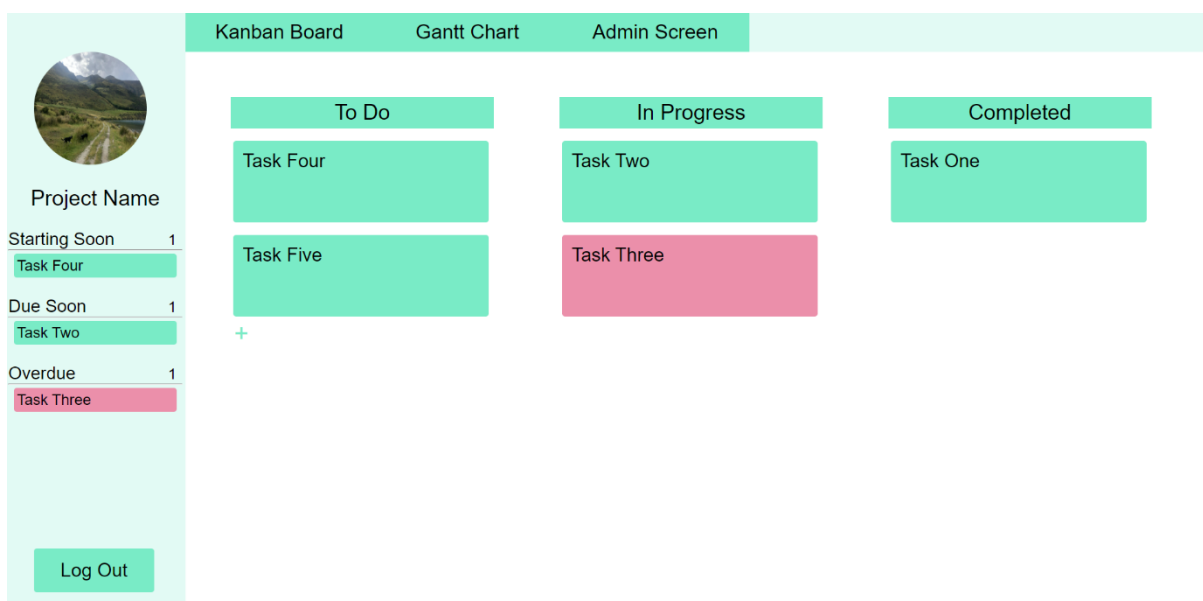


Figure 5: Kanban Board Page

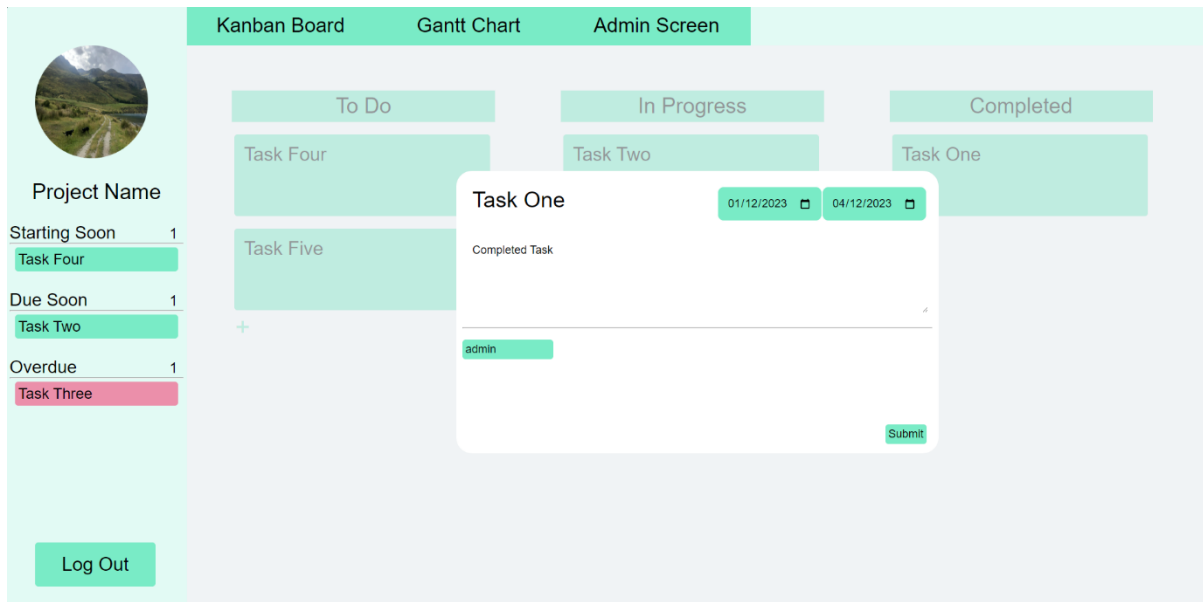


Figure 6: Task Modal

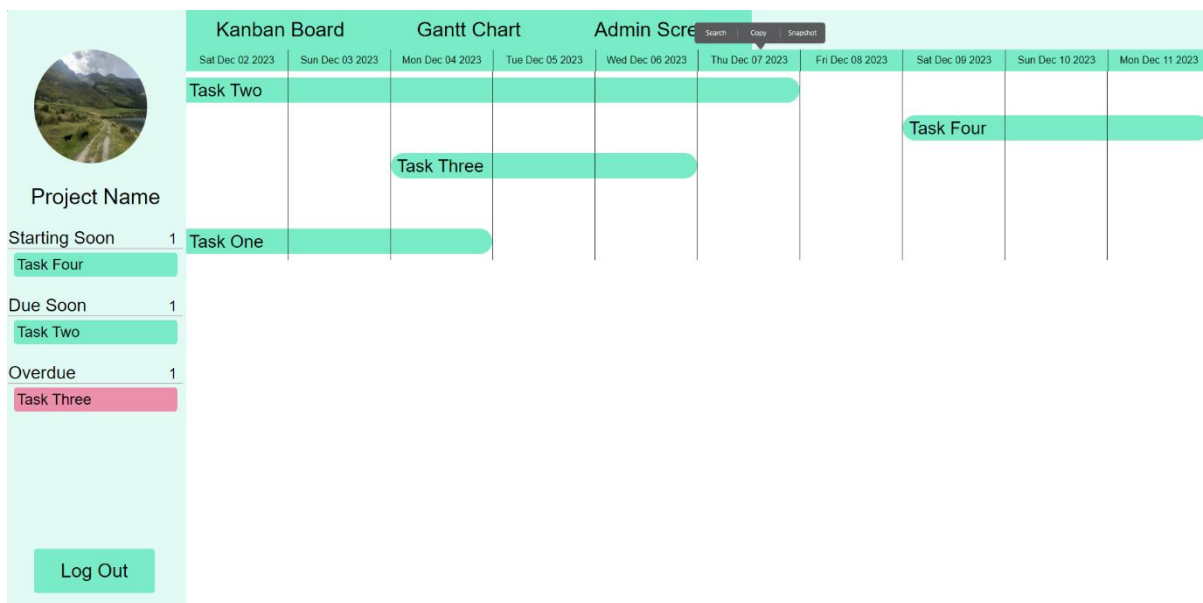


Figure 7: Gantt Chart Page

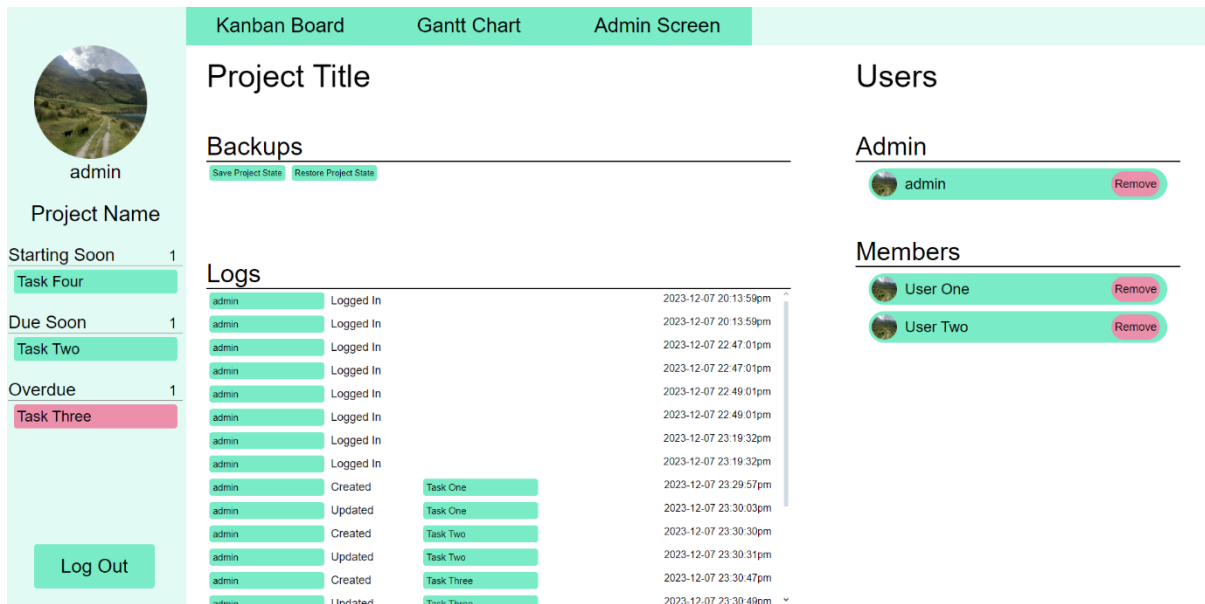


Figure 8: Admin Page

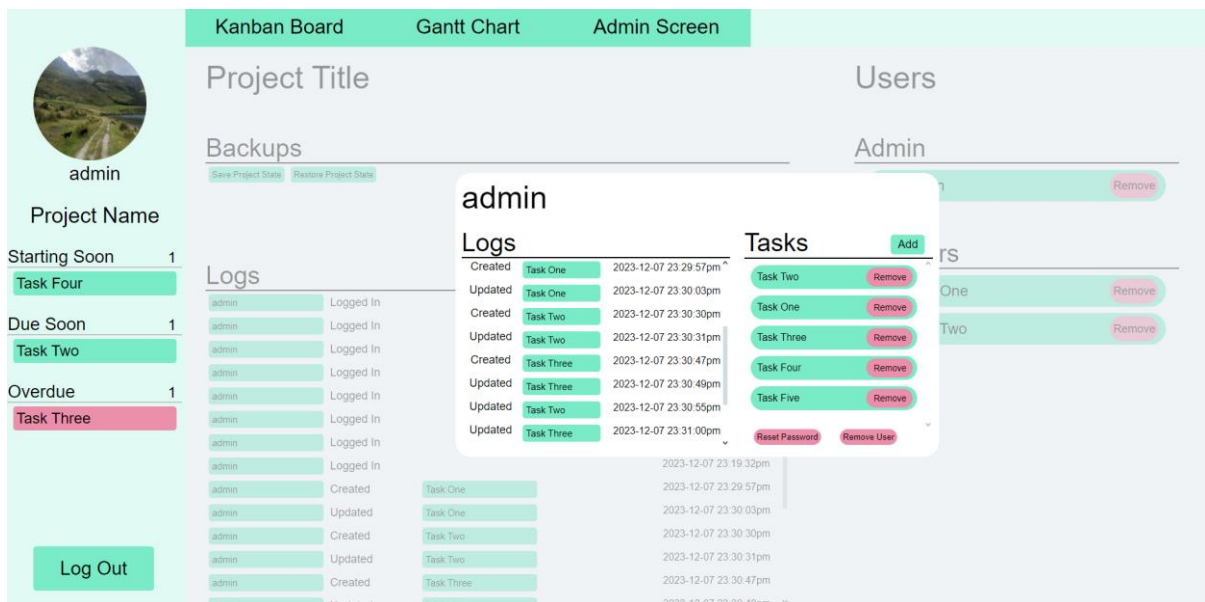


Figure 9: User Modal

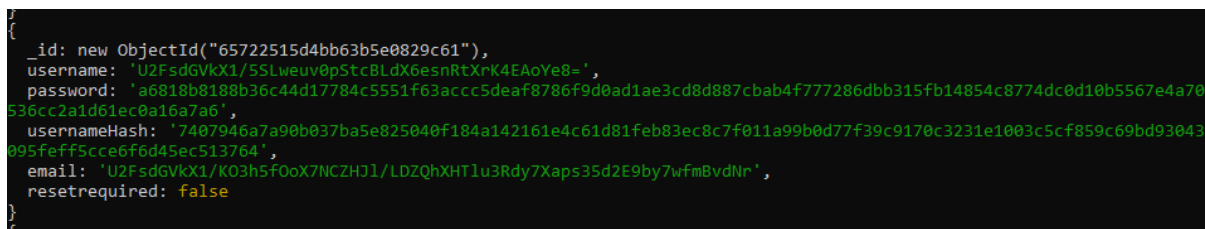


Figure 10: User Account in transit

```
[
  {
    _id: new ObjectId("65722515d4bb63b5e0829c61"),
    username: 'U2FsdGVkX1/5SLweuv0pStcBLdX6esnRtXrK4EAoYe8=',
    perm: 1
  },
  {
    _id: new ObjectId("65725659ebd1fb6e91f9bfff6"),
    username: 'U2FsdGVkX1/DCYUkh3auSG7Tj2xNA2uvPXA2J4CG8kc=',
    perm: 0
  },
  {
    _id: new ObjectId("65725667ebd1fb6e91f9bffe"),
    username: 'U2FsdGVkX1/8YiIB2gKRaqbCk3GEIRTPhFY1p1Arwj8=',
    perm: 0
  }
]
```

Figure 11: Array of usernames in transit

```
{
  _id: new ObjectId("657255c7ebd1fb6e91f9bfbb"),
  name: 'U2FsdGVkX19gg8BiUXZ43pOAMZnFx8uEkmZA7vFXDwQ=',
  description: 'U2FsdGVkX19yXdKcus+CO6Edx6QzkPRWmuDFkeM5A3A=',
  startDate: 2023-12-09T00:00:00.000Z,
  endDate: 2023-12-11T00:00:00.000Z,
  state: 'U2FsdGVkX18tWbZBEu+Q/fEpr8ioVPuL/A/Np/DRE88=',
  users: [
    new ObjectId("65722515d4bb63b5e0829c61"),
    new ObjectId("65725667ebd1fb6e91f9bffe")
  ]
}
```

Figure 12: Task in transit

```
_id: ObjectId('65722515d4bb63b5e0829c61')
username: "U2FsdGVkX1/5SLweuv0pStcBLdX6esnRtXrK4EAoYe8="
password: "a6818b8188b36c44d17784c5551f63accc5deaf8786f9d0ad1ae3cd8d887cbab4f7772..."
usernameHash: "7407946a7a90b037ba5e825040f184a142161e4c61d81feb83ec8c7f011a99b0d77f39..."
email: "U2FsdGVkX1/K03h5f0oX7NCZHJl/LDZQhXHTlu3Rdy7Xaps35d2E9by7wfmBvdNr"
resetrequired: false
```

Figure 13: User Account at rest

```
_id: ObjectId('65725575ebd1fb6e91f9bf5c')
name: "U2FsdGVkX1/s0xE/vGz2Hdtjs13t+Yvrbr2M9wfcHdY="
description: "U2FsdGVkX1/C0MOKudh40lNncIilVKk8UhXrqQ89g9c="
startDate: 2023-12-01T00:00:00.000+00:00
endDate: 2023-12-04T00:00:00.000+00:00
state: "U2FsdGVkX1/aj2xJUnZphYR8WT4BpJgnT2wXlCgS3xs="
► users: Array (2)
```

[Notion Link](#)

[GitHub Link](#)

R&D Report

Introduction

For the Capstone project, my aims were to study the increase in the complexity of development added by using Zero Trust Architecture (ZTA) security practices. To do this, I created a project management application that employed certain ZTA practices. In the CS301 project, I found that the project's development was significantly slowed by the addition of these practices, notably due to the incompatibility of different software.

For this paper, I continued my study by adding an administrator role to the application, which can modify user access, force users to reset passwords, view activity logs and save/reload the state of the application. I also wanted to add multi-factor authentication to secure these users accounts. Alongside these security features, I would continue the development of the application by adding a Gantt chart and notifications.

Access Management

In the prior paper, I had set up an access control system that restricted task data to the user who had created it, that I was able to import directly to the website. For this paper, I expanded on the access management system by incorporating administrative management of user access. This means that I needed to add an admin role that a user could be assigned, and I needed to add functionality to allow these admin users to grant or revoke access to task data.

Adding functionality to allow access management was relatively straightforward. The system I was using to check for access was giving each task object an array of users that had access to it. MongoDB queries allow queries to be restricted based on the contents of an array. By adding the id of the current user to queries for tasks, we can restrict the results to tasks that the user has access to. This access control will restrict data at the database level, meaning the system will not send any data to the user's network unless they have access to it. And, by allowing administrators to add a user's id to the task array, they will be able to grant that user access to the task.

User Management

In occasions where an account's credentials have been compromised to a malicious entity, it will be necessary for an admin to modify the account to mitigate the damage and prevent the attacker from having free access to the project. Initially, I had planned to do this by giving the admin three options for user management: forced password resets, account deletion, and IP blocking. The first two options were developed into the application, and can be used by an administrator to respond to compromised accounts. When an account is flagged with a password reset, they will be incapable of logging into the account until they have completed it. Deleting an account will remove it from the database, making it impossible for either the intended user or the malicious actor to log in using it ever again.

I had intended for IP blocking to be included as well; however the feature would have been far more complex to add than I had initially thought. Given an already lagging production schedule, I chose to cut it from the application.

Logs

To give administrators a comprehensive understanding of their projects, they are going to need to know how their users are using the application. The most effective way of keeping them up to date is via logs.

I intended to add user login logs to track the times & locations that users log in from, along with activity logs to track how users interact with the application. These logs would be displayed to administrators, who could use them to help identify attacks and repair data.

I added activity logs by adding a new step to the database upload process. Before any data is uploaded, the application will create a log object that stores relevant information, including the action, user, and timestamp. These logs are stored before the action is performed.

Logs are also stored whenever a user logs in or registers.

2-Factor Authentication

Since the administrator is given access to all data in the project, it created a new security risk I needed to account for. Multi-factor authentication is a common method for increasing the difficulty for a malicious actor to break directly into user accounts. For this project, I used 2-factor authentication, with admins authenticating themselves first through a password, and then by accessing a randomly generated code sent to the admins mail.

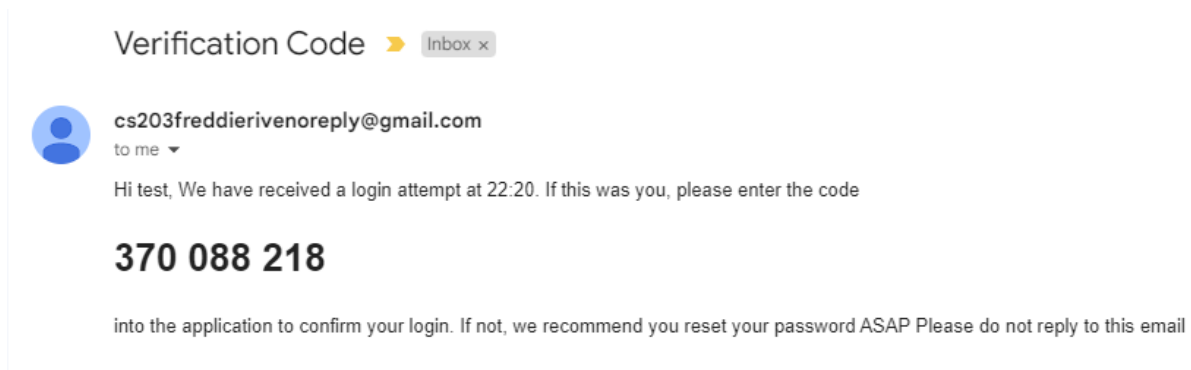


Figure 15: Two Factor Authentication email

A screenshot of a web form for Two Factor Authentication. The form has a light blue background. Inside, a white box contains the text: "A code has been sent to your email address. Please enter it here to confirm your log-in". Below this, there is a label "Email Code" and a text input field containing the code "370 088 218". Below the input field is a green "Confirm" button.

Figure 16: Two Factor Authentication being input

Since users tend to only adopt multi-factor authentication when forced to (Cristofaro, Du, Freudiger, & Norcie, 2014), I decided to make this system mandatory for admin users. The random codes being sent contain 9 digits, allowing for 360,880 possible combinations of numbers. To protect against brute-force attacks, the application will only allow for 5 attempted entries before it requires the user to re-enter their username and password.

Backups

Backups serve as an intrusion response system, that can be used to undo damages after a cyberattack. The way MongoDB handles data is by grouping it into 'collections', which themselves are grouped into databases. The backups are copies of the 'collections' inserted into another database. These copies themselves can be copied back onto the main database to replicate the state of the database when it was saved.

This system is effective at protecting against vandalism caused by attacks on the application or network layers, however it provides little protection against an attack on the database itself. The MongoDB database is protected by its own multi-factor authentication; however, a breach of this system is not inconceivable. Ideally, a database backup would be stored in a separate physical storage device that the application owners possess, that they can restrict access too on their own. More common is the practice of purchasing a backup server from your cloud server provider. Neither of these options were available to me, given my limited resources.

Conclusion


Adding the administrator role has greatly inflated the complexity of the security systems. Not only has this required more rigorous authentication, but it has also forced me to modify all existing security systems to account for its existence.

Throughout this project, the tools and practices I have used have been primarily dictated by my own resources and skills. This has created limitations on all of the features I have added to the application. When laying out the plans for this project, I found that it was unreasonable to include many features of a true Zero Trust environment, either due to a lack of finances or skill on my part. As such, the finished product cannot be said to be Zero Trust and suffers from several vulnerabilities that Zero Trust principles could have prevented.

This is representative of an issue with implementing a Zero Trust security system: it requires many resources to implement. Application developers cannot apply all ZTA techniques without purchasing expensive software and employing trained professionals who specialize in this field. IBM's Security Intelligence and Event Management system 'QRadar' costs between \$1,370 and \$5,980 USD per month (IBM QRadar Pricing, 2023). The salary for a Security Analyst in New Zealand typically ranges from \$120k – \$200k, with chief security managers sometimes costing as much as \$500k per annum to employ (careers.govt.nz Security Analyst, n.d.) A steep cost of security provides an incentive for companies, particularly startups, to attempt to cut costs by cutting security. This is particularly true in cases where the true costs of a security breach are not with the software provider (Moore, 2010).

The current era of cybersecurity, dominated by Zero Trust principles, has continued the trend of cybersecurity towards a role in the software industry that requires specialised professionals to handle, as opposed to an aspect of development that can be handled by a generalist. I have found that the additional complexity added to application development by the inclusion of Zero Trust practices is very significant. A team of software developers intending to develop an application using these practices will require a team of professionals dedicated solely to this task.

Supervisor Notes and Signature

Date and Signature 08/12/2023 	Supervisor's notes with bullet points of the major tasks -
--	---

Bibliography

careers.govt.nz Security Analyst. (s.d.). Récupéré sur careers.govt.nz:

<https://www.careers.govt.nz/jobs-database/it-and-telecommunications/information-technology/security-analyst/>

Cristofaro, E. D., Du, H., Freudiger, J., & Norcie, G. (2014). A Comparative Usability Study of Two-Factor Authentication. *arXiv*, 1-10.

IBM QRadar Pricing. (2023, December). Récupéré sur IBM: <https://www.ibm.com/products/qradar-siem/pricing>

Moore, T. (2010). The Economics of Cybersecurity: Principles and Policy Options. *International Journal of Critical Infrastructure Protection*, Volume 3, Issues 3-4,, 103-117.

Rive, F. (s.d.). *Project GitHub*. Récupéré sur <https://github.com/Frederick-Rive/project-management-phase-2>

Rive, F. (s.d.). *Project Notion*. Récupéré sur <https://www.notion.so/Project-Management-Phase-2-cb138b3dbc79437796ddb1bae98b6649>

Table of Figures

Figure 1: Login Page.....	7
Figure 2: Two-Factor Authentication Page	7
Figure 3: Two-Factor Authentication Email	8
Figure 4: Two-Factor Authentication Failure.....	8
Figure 5: Kanban Board Page	8
Figure 6: Task Modal	9
Figure 7: Gantt Chart Page	9
Figure 8: Admin Page	10
Figure 9: User Modal	10

Figure 10: User Account in transit	10
Figure 11: Array of usernames in transit.....	11
Figure 12: Task in transit	11
Figure 13: User Account at rest.....	11
Figure 14: Task at rest.....	12