Western Norway
University of
Applied Sciences

BACHELOR'S THESIS

Monitorization Dashboard

Glenn Arnold Barosen
Charlie Edward Coulter
Marius Sunde Husevåg

Faculty of engineering and science
Department of Computer Science, Information Technology
Rogardt Heldal
02.06.2021

TITLE PAGE FOR THE MAIN PROJECT

Report title: Monitorization dashboard	*Date: 02.06.2021*
Author(s): Glenn Arnold Barosen, Charlie Edward Coulter, Marius Sunde Husevåg	*Number of pages w/o attachments: 27*
	Number of pages with attachments: 28
Field of study: Information Technology / Computer Science	*Number of discs/CDs:*
Supervisor at field of study: Rogardt Heldal	*Grading: none*
Notes:	

Employer: Wide Assessment AS	
Employer contact(s): Andreas Hammerbeck / CTO Kristoffer Perminow / Developer	*Phone:* 976 61 466 473 96 813

Summary:
The motivation behind this project comes from the complicated nature of handling security routines within a company. A company need to keep track of server uptime, employee access, along with other things. This project will make it easier to keep track of all these activities regarding its security routines.

The main goal of this project is to develop a monitoring dashboard that gives the company a better overview of the company's security routines by integrating checklists and critical tools in one place.

Motivasjonen bak dette prosjektet kommer av den vanligvis tungvinte måten å håndtere sikkerhetsrutiner i en bedrift. En bedrift må ha oversikt over status på servere, ansattes tilganger og mye mer. Dette prosjektet skal gjøre det enklere å holde oversikten over alle disse aktivitetene som inngår i en bedrifts sikkerhetsrutiner.

Hovedmålet med prosjektet er å utvikle et monitoreringsdashbord som gir bedriften en bedre oversikt over bedriftens sikkerhetsrutiner ved å integrere sjekklister og andre kritiske verktøy i et og samme sted.

keywords:

React, TypeScript, HCI, Agile, Kanban

Western University of Applied Sciences, Faculty of engineering and science
Post address: Postboks 7030, 5020 BERGEN Address: Inndalsveien 28, Bergen
Phone. 55 58 75 00 Fax 55 58 77 90 Email: post@hvl.no Homepage: http://www.hvl.no

Preface

We are now done with three years of studies and are ready for new adventures. This project has been challenging, mostly due to the situation regarding COVID-19, but we pulled through. We would like to thank Rogardt for tips and tricks this semester and the people at Wide Assessment for a fun project and cooperation this last year.

Table of contents

1 Introduction

The company Wide Assessment (WA.works) was established in 2016. The company has created a recruitment platform specialising in the IT industry, which simplifies the job searching process for both employee and employer.

1.1 Motivation and goal

The motivation behind this project comes from the complicated nature of handling security routines within a company. A company need to keep track of server uptime, employee access, along with other things. This project will make it easier to keep track of all these activities regarding its security routines.

The main goal of this project is to develop a monitoring dashboard that gives the company a better overview of the company's security routines by integrating checklists and critical tools in one place. The main goal is divided into two subgoals. The first subgoal is to create a checklist where the company can add and remove tasks. The checklist must have deadlines that are updated automatically when the task is done. The second subgoal is to integrate Github into the application using the Github API to create an overview of which employees have access to different parts of the company's source code.

1.2 Context

WA.works, like all other companies, have a lot of routines regarding the security and integrity of the company's systems. They are always looking for new opportunities to improve them, so they want a dashboard that makes it possible to get an overview of all security-related tasks and events all in one place. Today, these tasks are done manually and will therefore be something WA.works can save a lot of time on automating and will lead to the security of WA.works IT systems being improved. There was no existing code from WA to develop; therefore, our research is based on previous work from external entities. The reason for these routines is that the company has servers that they need to keep running and have backups of. The information about users also needs to be kept confidential. Because of this, there are certain routines in place to uphold the integrity of the system and the confidentiality of its users.

1.4 Limitations

The group’s most significant limitation is the group’s general understanding of human-computer interaction (HCI) and its relation to the project. This can affect the user’s interaction with the application regarding the human-to-human, open-ended dialogue (Interaction Design Foundation, n.d.).

Another limitation regarding the project is COVID-19. The current situation makes communication harder, and all work must be done from home. Not working from the company's offices creates a more significant divide between the group and company and makes reaching out for help more difficult. Our goal is to finish most of the initial requirements, but this could be difficult in the space of three months. The aim is to complete the two sub-goals within the deadline.

More minor limitations regarding the project are general inexperience in developing web applications and testability. The lack of experience can cause development time to increase because the group will have to do a lot of research. Since the only users who will test the application are our contacts at WA.works, the amount of testing can be limited. The project's timeframe can also impact the amount of time that can be set aside for testing.

1.5 Resources

To create this application, a code editor program called Visual Studio Code has been used. Visual Studio Code is a popular editor with extensions and plugins to support a wide variety of programming languages. To design the application, Figma has been used. Figma is a design and prototyping tool often used for creating sketches and prototypes for applications. The group members have used their own personal computers to develop the application, and no other special resources were needed.

1.6 Organization of the report

This report will go through the project description containing information about the initial requirements and initial solution. After this, there will be some information about the literature background for this project. Then, the group will go through the project design and the specification and what tools and methods have been used. After this, a chapter about the detailed design contains different figures that describe certain parts of the application. Then there is a chapter about the implementation of both the client-side application and the server.

After these parts explaining the development of the application, there is a chapter about which evaluations methods have been used, along with the results of these evaluations. Then there is a chapter discussing certain parts of the application and some of the group’s choices. At the end of the report is a conclusion talking about the results and what the group thinks about the project and what goals were reached, and further work that can be done to the application.

2 Project description

2.1 Practical background

There has been a collaboration between the project owner and the group before. This made understanding what the company wanted and the requirements of the project a lot easier. Two of the group members had an internship at WA.works the previous semester, which helped with the communication and progress of the project since the group and project owner was familiar with each other. During the internship, the company offered the group to create this project for our bachelor thesis.

2.2 Project owner

The CEO of the company is Stine Andreassen. The company has extensive experience in recruitment within the IT industry through their recruitment platform. The group's application is going to be used by the CTO of the company, Andreas Hammerbeck, who has a master’s degree in computer science from Western University of Applied Sciences.

Our primary contact at the company is Kristoffer Perminow, which has the main responsibility for the bachelor projects this semester. The company has seven employees in daily operations but also has several consultants who are working for other businesses.

2.4 Initial solution idea

It was recommended to use React with Typescript for the client-side application and Node.js for the server. The initial solution idea was to use these technologies to create a web app containing a dashboard with different "cards" to complete various tasks.

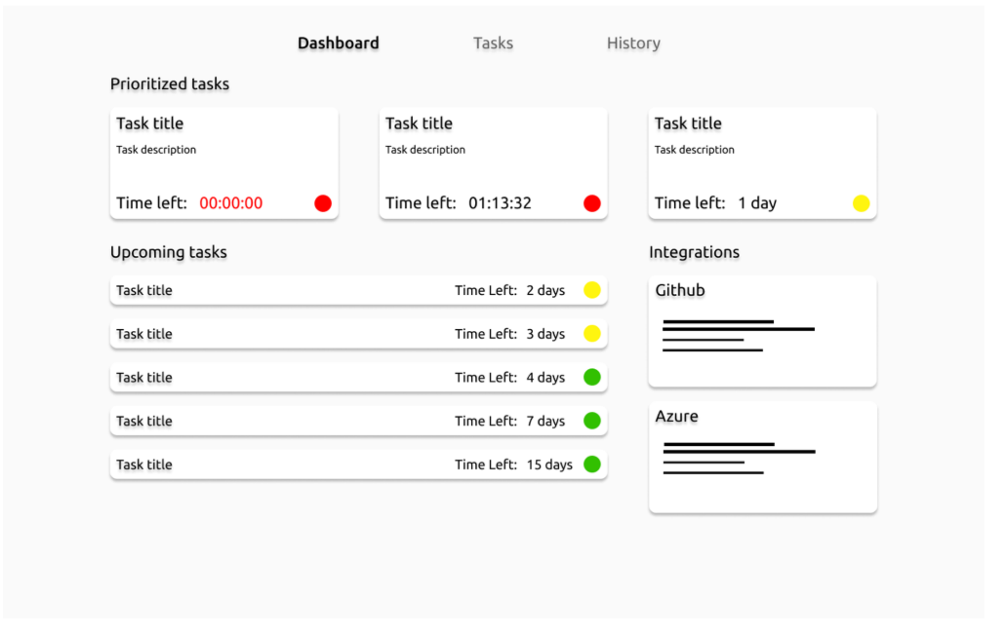


Figure 1: Initial design created in Figma.

2.5 Literature background

Human-Computer Interaction (HCI) is a multidisciplinary field of study focusing on the design of computer technology and the interaction between humans (the users) and computers (Interaction Design Foundation, n.d.). The reason why HCI is essential to our project is to be able to resemble a human-to-human relationship regarding the dashboard. Our focus is that the product user understands how the dashboard works to make their work more efficient. Research in the field of HCI is to improve human-computer interaction by enhancing the usability of computer interfaces. The research that is our focus is all the methods for designing, implementing, and evaluating computer interfaces.

There are several HCI guidelines that the application should follow to have correct HCI aspects (Katsabas, et al., 2005). The guidelines the group are using to evaluate the application are the following:

1. **Visible system state and security functions:** Applications should not expect users to search to find the security tools or have hidden features inside the application. Furthermore, the use of status mechanisms can keep users aware and informed about the state of the system. Status information should be periodically updated automatically and should be easily accessible.
2. **Suitable for advanced as well as first-time users.** Show enough information for a first-time user while not too much information for an experienced user. Provide shortcuts or other ways to enable advanced users to control the software more easily and quickly.
3. **Handle errors appropriately:** Plan the application carefully so that errors caused by the use of security features could be prevented and minimised as much as possible. However, when errors occur, the messages must be meaningful and responsive to the problem.

3 Project design

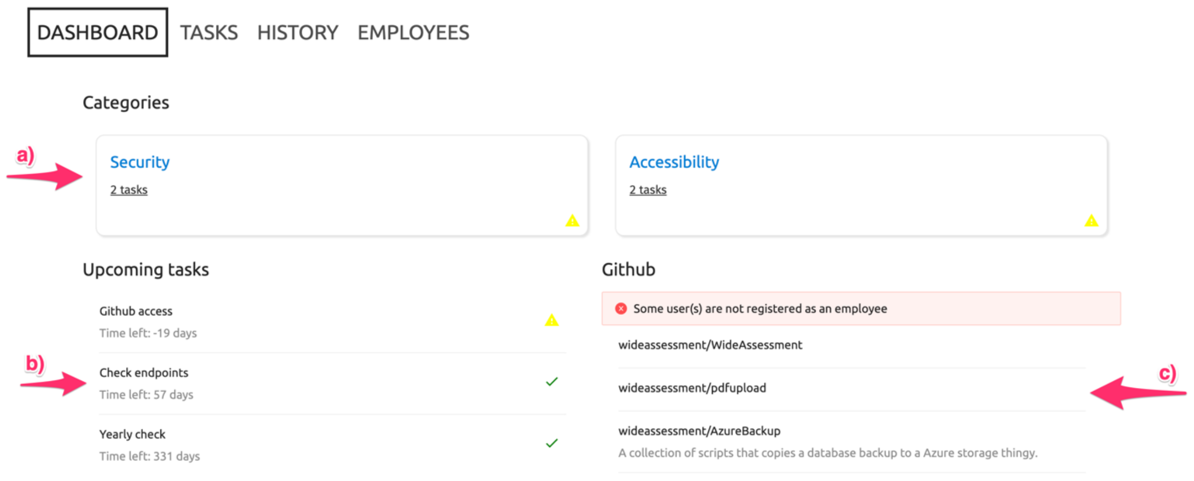
3.1 Possible approaches

In the first meeting between the project owner and the group, there was a discussion about possible approaches regarding the application's design. The project owner laid some ground rules regarding functionality and then asked the group to present a design at the next meeting. The group used Figma, which is a tool used to design prototypes, to create different designs.

3.2 Discussion of selected approach

At the second meeting between the group and project owner, the group presented the design created in Figma. After some discussions, the parts agreed on making some minor changes to the design. Figure 2 displays the final design of the dashboard. It contains a dynamic list of task categories such as accessibility, integrity and so on (Figure 2, a). This list gets bigger or smaller automatically based on the number of categories currently applied to the tasks.

It also contains a list of five tasks that are close to or past their deadline (Figure 2, b). Lastly, the dashboard has an overview of the company's Github repositories and a status bar that tells the user any inconsistencies between the employee list and the list of accounts with access to the repositories. (Figure 2, c). The dashboard also has priority indicators on both the categories and tasks; these indicators are either green, yellow, or red and change depending on the time remaining.



Categories

Security
2 tasks

Accessibility
2 tasks

Upcoming tasks

Github access
Time left: -19 days

Check endpoints
Time left: 57 days

Yearly check
Time left: 331 days

Github

Some user(s) are not registered as an employee

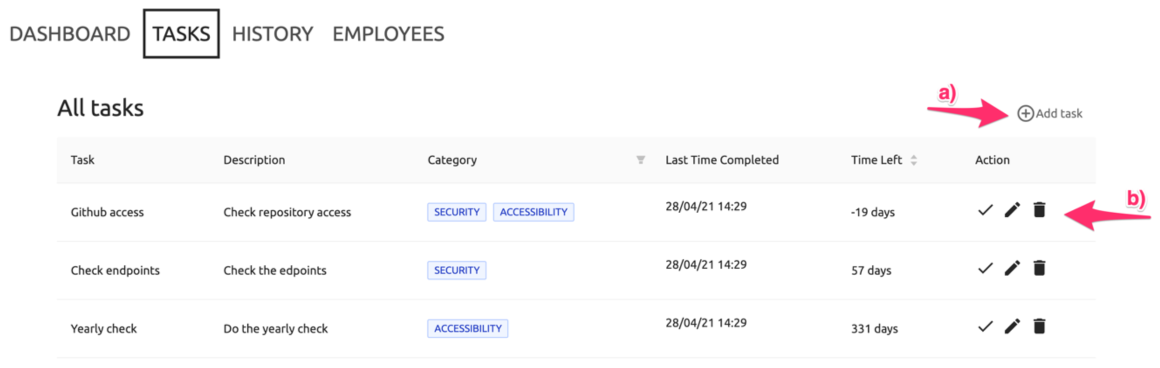
wideassessment/WideAssessment

wideassessment/pdfupload

wideassessment/AzureBackup
A collection of scripts that copies a database backup to a Azure storage thingy.

Figure 2: Final dashboard design used in the web application.

In addition to the main dashboard page, the web application consists of a "Tasks" page, a "History" page, and a "Employees" page. The "Tasks" page (Figure 3) consists of a table view of all existing tasks with the ability to sort the tasks depending on the time remaining and the ability to add new tasks (Figure 2, a). The task table also contains a column named "actions" that provide actions such as "complete", "edit", or "delete" (Figure 3, b). The "History" page is a list of already completed tasks with information about when the task was completed and whether the task was completed on time. The "Employees" page contains an employee list where the user can add or remove employees. This list is used to compare employees and user with access to the GitHub repositories.



All tasks

Task	Description	Category	Last Time Completed	Time Left	Action
Github access	Check repository access	SECURITY ACCESSIBILITY	28/04/21 14:29	-19 days	✓ ✎ 🗑
Check endpoints	Check the edpoints	SECURITY	28/04/21 14:29	57 days	✓ ✎ 🗑
Yearly check	Do the yearly check	ACCESSIBILITY	28/04/21 14:29	331 days	✓ ✎ 🗑

Figure 3: "Tasks" page containing a list of all tasks.

3.3 Specification

This chapter describes the functional and operational specifications which are meant to be implemented in our solution. These specifications are partly given to the group by the project owner and partially interpreted by the group through the project description provided by the project owner. Here are the specifications:

- The dashboard should be designed so that the tasks can be completed on the main page of the dashboard.
- It should be easy to see which tasks need to be done and each task's impact on the company's security.
- Must be able to see when the task is due.
- When a task is completed, or the deadline for the task expires, this must be logged in a task history.
- When a task is marked as done, the same task should automatically get created with a new deadline matching the interval given when the task was created.
- If the task is not completed within the time frame, this must be clearly displayed on the dashboard, and the task must have a colour code corresponding to the severity of this.
- A requirement to integrate the Github API with the dashboard to get an overview of people who have access to Wide Assessment's repositories on Github.

3.4 Selection of tools and programming languages

Due to the open nature of the project description, the group was not constrained to any specific technologies. The group decided to go with React and Typescript as recommended on the client-side of the project and Node.js on the server-side. The main reasoning behind these choices was that the members had previous experience using these technologies.

React is a JavaScript library for building user interfaces (React, n.d.). It uses JSX, a way to use HTML in JS and uses a virtual DOM that simplifies traditional DOM manipulation in JavaScript. Typescript could be seen as a superset of JavaScript. It introduces typing in JavaScript, which is traditionally a loosely typed programming language. Types provide a way to describe an object's shape, provide better documentation, and allow TypeScript to validate that your code is working correctly (TypeScript, n.d.).

Node.js is a backend JavaScript framework. It is an asynchronous event-driven JavaScript runtime, and it is designed to build scalable network applications (Node.js, n.d.).The group's prior experience with JavaScript makes this combination of technologies a great choice since the programming language remains relatively similar throughout the application.

As a database, the group decided to go for SQLite, which is a lightweight SQL database. The reason for using SQLite is because the data being stored is not complex, and advanced SQL database queries are not necessary. Knex.js was used to integrate the database in the backend, and it makes creating queries easy.

3.5 Project development method

The project has been developed using aspects from one of the most used development methodologies in software development, Agile development. Due to the nature of the project being worked on part-time, not all the aspects of Agile development are relevant for this project.

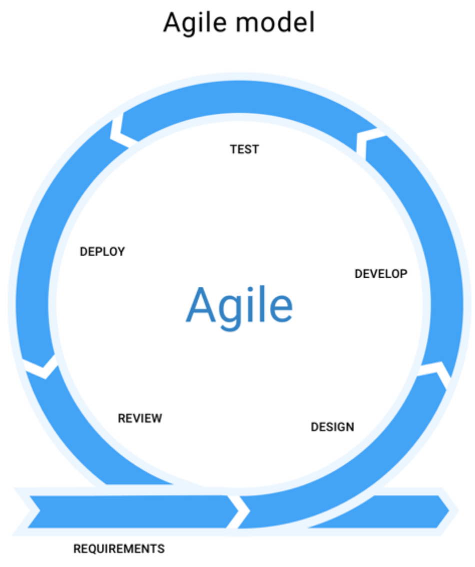


Figure 4: Agile lifecycle (Feoktistov, n.d.).

3.5.1 Development method

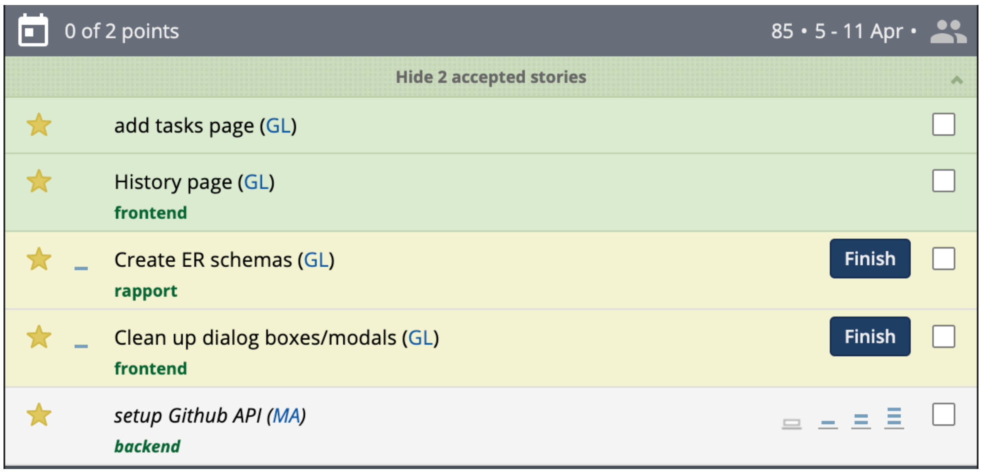
As mentioned, the development method used in this project is agile development. The agile development method is an iteration-based method. This means that every new feature in, for example, a web application goes through a development cycle ranging from getting feature requirements to reviewing the finished feature. The agile development method often has daily standups, which are meetings where the team talks about what they are working on each day.

In this project, the group will not use a lot of the methodology in agile development, such as daily standups and sprints that are common in scrum. The group will be using the Kanban method described in chapter 3.5.2. The group will work from a backlog of tasks on the Kanban board, implement each task, and test it before moving on to another task (Radigan, n.d.).

3.5.2 Kanban and Pivotal Tracker

The work of all Kanban teams revolves around a Kanban board, a tool used to visualise the outcome and optimi (Radigan, n.d.). The program that the group is operating as a Kanban board is called Pivotal Tracker. Pivotal Tracker has many features, including a point system for each task to give an idea of the importance of a task. This project will mainly use Pivotal Tracker in the traditional Kanban way by using it as a backlog of tasks. (Figure 4).

Pivotal Tracker visualises our project in the form of virtual cards on the board. This then encourages the group to break down the project into manageable parts. Pivotal Tracker gives a better overview of everything that needs to be done and indicates an approximate work timeframe. Due to the nature of continuously grabbing tasks from the backlog, any unexpected issues won't impact the work the group members are currently doing, but instead, the issue will be put in the backlog along with rearranging of the priority in the backlog.


0 of 2 points — 85 • 5 - 11 Apr •

Hide 2 accepted stories

- add tasks page (GL)
- History page (GL) — frontend
- Create ER schemas (GL) — rapport — Finish
- Clean up dialog boxes/modals (GL) — frontend — Finish
- *setup Github API (MA)* — *backend*

Figure 5: Backlog of tasks in Pivotal Tracker.

3.5.3 Other planning methods

In addition to using a Kanban board through Pivotal Tracker, the group has also created a Gantt chart that displays a greater overview of the project timeline. In the Gantt chart, there are different deadlines for all the tasks to be completed. This gives the group an indication of how much work needs to be put in to complete the tasks. It is important that the group set deadlines to have a better overview of time. The group come up with three main milestones for the project, these three milestones are:

- Finish the first iteration.
- Finish the second iteration.
- Finish the report.

The three milestones created are the most important tasks to be completed. In order to create a finished product for WA.works and to write a thorough report, the project needs to be appropriately planned.

3.5.3 Risk management

The project has several different risks throughout the development. The different risks could mainly occur due to time-consuming tasks like research of frameworks and APIs, lack of knowledge regarding integrating the Github API into the application, errors while developing and miscommunication with the project owner or within the group. To prevent these risks from happening, the group needs to locate our most common risks and find a solution to the problem for each risk.

The group have located most of our risks and made a risk list (Appendix 11.2). The risks are:

- Lack of time to get a working product.
- Lack of knowledge about APIs to finish Github iteration.
- Errors while developing.
- Unable to satisfy the project owner.
- Miscommunication with the project owner or within the group.

To prevent these risks from happening, the group have discussed mitigations to implement before these risks occur. The mitigations are as followed:

- Set aside more time for coding.
- Set aside enough time for learning about API implementation.
- Use TypeScript to set types for as much as possible. This will make the app less error-prone in the long run.
- Keep close communication with the project owner to make sure all the functions they want is included.
- Have frequent meetings to ensure everyone knows what they are doing.

4 Detailed design

This chapter describes how certain parts of the application work behind the scenes and is illustrated by various diagrams.

4.1 Use case diagram

Use case diagrams are drawn to capture the functional requirements of a system (Mule & Waykar, 2015). In this project's case, the use case diagram is not complex but does clearly state the main functionalities of the application.

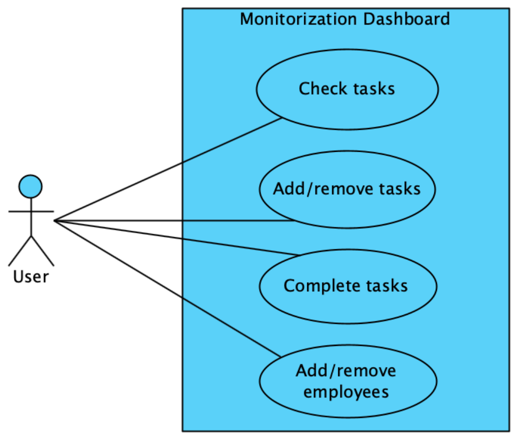


Figure 6: Use case diagram.

4.2 Activity diagram

The group use Activity Diagrams to illustrate the flow of control in a system and refer to the steps involved in the execution of a use case (GeeksforGeeks, 2018). In this activity diagram, the starting point for every activity is the dashboard page; from there, the user can choose which activity to pursue and then every following step and decision is described through nodes in the diagram.

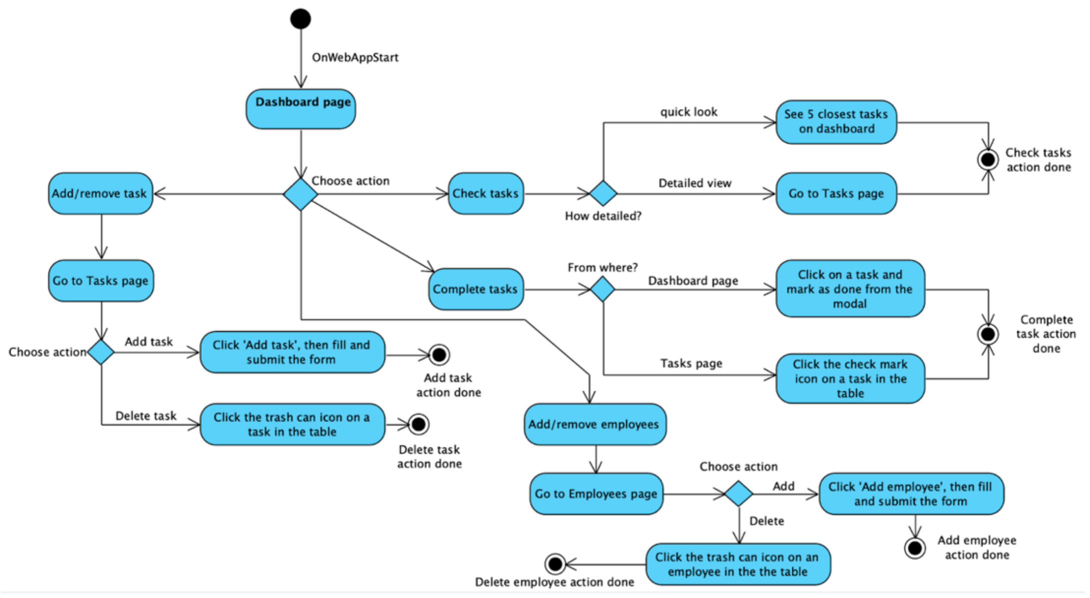

OnWebAppStart
Dashboard page
Choose action
Add/remove task
Go to Tasks page
Choose action
Add task
Click 'Add task', then fill and submit the form
Add task action done
Delete task
Click the trash can icon on a task in the table
Delete task action done
Check tasks
How detailed?
quick look
See 5 closest tasks on dashboard
Detailed view
Go to Tasks page
Check tasks action done
Complete tasks
From where?
Dashboard page
Click on a task and mark as done from the modal
Tasks page
Click the check mark icon on a task in the table
Complete task action done
Add/remove employees
Go to Employees page
Choose action
Add
Click 'Add employee', then fill and submit the form
Add employee action done
Delete
Click the trash can icon on an employee in the the table
Delete employee action done

Figure 7: Activity diagram.

4.3 Sequence diagram

Sequence diagrams describe how and in what order the objects in a system function (GeeksforGeeks, 2018). In this case, the sequence diagram illustrates the order of requests and responses between the client and server when a user submits a new task in the application. Sequence diagrams are helpful to get a deeper understanding of the underlying systems in an application and how these systems communicate with each other.

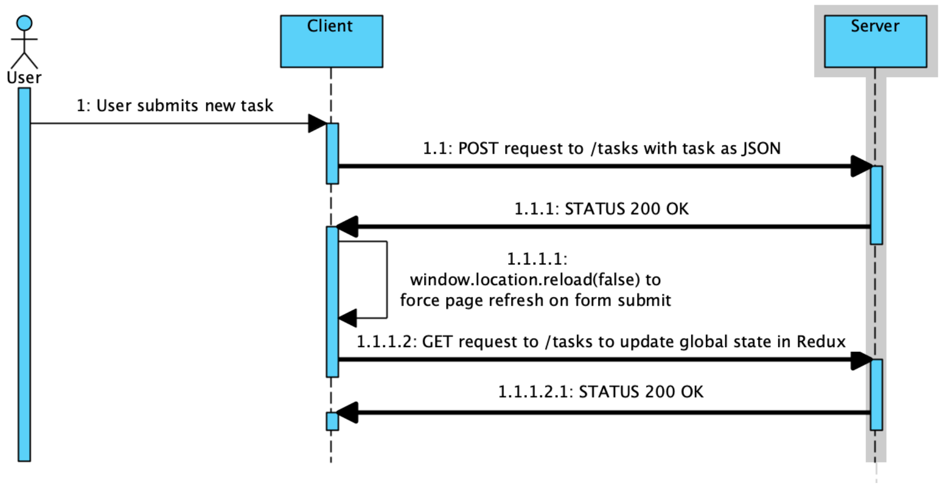

User
Client
Server
1: User submits new task
1.1: POST request to /tasks with task as JSON
1.1.1: STATUS 200 OK
1.1.1.1: window.location.reload(false) to force page refresh on form submit
1.1.1.2: GET request to /tasks to update global state in Redux
1.1.1.2.1: STATUS 200 OK

Figure 8: Sequence diagram.

5 Implementation

This chapter is about the actual implementation of the design and contains information about key aspects and technologies used throughout the development of the application.

5.1 Frontend development

As previously mentioned, the client-side application is built using React with Typescript. All the group members had experience in using these technologies from earlier projects, and because of this, the frontend development turned out to go relatively smoothly. For styling the application, Emotion was used, a CSS framework that fits well with React and uses CSS-in-JS.

During the application development, the group decided to change the appearance of some of the significant components, such as the tables and list styles. Due to this decision, Ant Design was added to the dependency list of the project. Ant Design is a component library that contains a lot of pre-built components. As the projects continued, more and more components were switched over to Ant Design components.

Redux was also, as mentioned previously, used to manage the applications state. To handle state in React, the different components can all have their own props, which can be passed from their parent component. This can make state management complex, especially when working with components that are nested several layers deep. Redux makes this a lot easier by creating a global state that can be accessed by all components, no matter how nested they are.

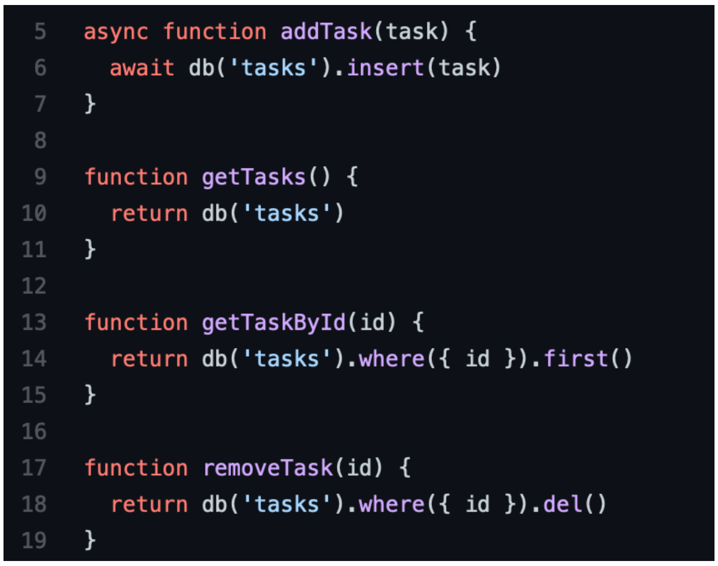
5.2 Backend development

The technology used on the backend is, as mentioned previously, Node.js combined with Express. The backend consists of three main parts, the routing, the database, and the Github API connection. The routing is done using Express to create the different endpoints such as /tasks or /history.

All routes are an extension of https://localhost:5000/api/v1/<YOUR_API_ROUTE_HERE>. The use of versioning in API can be advantageous if the server would be implemented on different platforms. If this was the case, then one platform could have more recent changes and version "v2" without affecting the app on another platform still using version "v1".

The database is an SQLite database. SQLite works great as the database engine for most low to medium traffic websites (SQLite, n.d.); this combined with the group's previous experience in using SQLite as a database solution, made choosing SQLite for the database a great choice. To avoid writing many SQL queries in the form of strings in the backend, Knex.js was used. Knex.js is a library for querying databases, and it has many out of the box setups for different database providers. The use of Knex.js lets the developer create functions without a lot of boilerplate SQL queries. These functions can then get called throughout the application to modify the database contents easily (see Figure 9).

One of the project goals was to implement the Github API in the application to get an overview of employee access and potentially more down the line. The group struggled a bit with setting up the API in the backend. This was mainly due to insufficient knowledge about authentication in request headers. To get information about the company's private information on Github, the requests to the API had to have a token with read and write access to the organisation's repositories. After receiving the correct token from the company contacts, the implementation went well. As with the database, routes or endpoints were created on the server, which fetched the data needed from the Github API, then a request can be made from the frontend to fetch employees from the server.



```
async function addTask(task) {
  await db('tasks').insert(task)
}

function getTasks() {
  return db('tasks')
}

function getTaskById(id) {
  return db('tasks').where({ id }).first()
}

function removeTask(id) {
  return db('tasks').where({ id }).del()
}
```

Figure 9: Knex.js database queries.

6 Evaluation

This chapter describes some of the evaluation methods used to evaluate the group's work and the application being developed, and the results of these evaluations.

6.1 Evaluation methods

6.1.1 User testing

User testing is the process through which the interface and functions of our dashboard service are tested by the actual users (Omniconvert, 2019), who in our case are our contacts WA.works. The users perform specific tasks on the dashboard that is created for them in realistic conditions (Omniconvert, 2019).

This method is helpful for the group because the user evaluates the usability of the dashboard. After testing, the users provide feedback regarding any changes, or if there are no issues, the application is ready to use. In a more extended time frame project, the group would use the feedback from the user testing to further improve the application, but in this case, any feedback would have to be taken care of by the company later.

User testing helps the group understand where improvements are needed and if the specification given by the project owner is met. The user testing is done manually by going through the application and testing the different functionalities in different scenarios.

6.1.2 Trial and error

The trial and error method has been used to determine if the application works during the programming phase of the project. This method helps to check if the code is running correctly continuously. One feature in React and other modern frameworks that can help detect errors during programming sessions is "hot reloading". Hot reloading keeps the app running and injects new versions of the files you edited at runtime (Bigio, 2016). This, along with the correct use of TypeScript, makes it so that the changes appear instantly on localhost every time the document gets saved. If TypeScript detects any type of error, the app crashes and displays information about where the error occurred. When developing without any unit testing libraries, it is essential not to make significant changes between saves so that the error is relatively easy to detect.

6.2 Evaluation results

6.2.1 User testing result

After our contacts at WA.works tested the application, feedback was given about the aspects that could be improved in the application. The group and testers had a meeting to discuss all the elements. Here is the feedback from WA.works about their findings in different parts of the application:

- Adding employees does not require a Github username, but a Github username is suddenly required when editing an employee.
- The repository status message says that everything is good when refreshing the page, even though the employee list does not correspond to the Github users.
- Needs clearer confirmation messages throughout the application, such as "Task added", "Task deleted", and so on.
- It should only be possible to add employees to the employee list who are a part of the company's organisation on Github. Currently, it is possible to add any Github user to the employee list.
- If a task is past the deadline, the priority colours do not match the severity. Currently has medium priority, should have high priority.
- Users should get clearer messages from the application when stuff goes wrong, i.e., wrong inputs.
- Medium priority colour should be darker, hard to see properly.
- Many components do not have cursors matching their action/use case. Button/links should have pointers; text fields should have text cursor.

The feedback received is helpful for the group if the group were to carry on developing the application. A lot of the feedback is highly relevant to HCI guidelines described in chapter 2.5. This makes sense as one of the group's main limitations is knowledge about HCI. This is relevant to UX design, which is a part of HCI. If the group had more knowledge about HCI, a lot of these issues would probably be negated.

Point 1 in 2.5 talks about the importance of a visible state in an application. The group got feedback about the application displaying the wrong status of the repositories when the page refreshes; this could have been prevented if the group was more aware of these guidelines during development.

Point 2 in 2.5 is about the application is suitable for advanced as well as first-time users. The feedback about the application states that more precise messages are needed throughout the application, such as “Task added” and “Task deleted”.

It is important to think that first-time users need clearer instructions to function the application. This could have been prevented if the group had put themselves in the shoes of a first-time user.

The group also received feedback about the user not getting error messages where the user should get error messages. This is very relevant to point 3 in the HCI guidelines in chapter 2.5, which talks about the importance of correctly displaying error messages when working with security.

Many of these issues can be easily fixed, but without user testing, many of them would probably never have been found. All in all, the results from the user testing have clarified how actual users use the application.
User testing has been a successful method because it allows the group to improve the application to the standards our users expect. It helps the group improve on the HCI principles being kept on the application to enhance the experience for the users.

6.2.2 Trial and error results

Due to the fact the application is relatively small, trial and error have worked out. If the application had gotten any bigger, automated tests would be useful, but using TypeScript to its fullest is often enough for smaller projects like this one. The group believes that the amount of time spent setting up tests would outweigh the benefits in this case. Using the trial and error method when developing has given the results the group hoped for, reaching our main project goal.

As mentioned, relying on TypeScript to give type errors when something is wrong has worked perfectly. However, the group could have used TypeScript even more by defining more types to make the development more reliable in the long run. This, along with hot reloading, has worked perfectly. Instead of scrolling through code to find a small error, TypeScript logs out the error detailed. Small mistakes happen all the time when coding, and just placing a curly bracket wrong can cause a lot of hassle, so to avoid these minor formatting mistakes, Prettier was installed.

Prettier automatically formats code to a given format on every save. This negates nearly all misplacements of parentheses or brackets. All in all, the trial and error method has worked well in this project.

7 Discussion

7.1 Trial and error vs unit testing

In this project, trial and error have been used for testing and evaluating the application throughout the development process. Some people would maybe look down on this method of evaluation, but with the tools available today, the use for unit testing in an application like this minimal. The use of TypeScript along with reloading on each save makes detecting bugs and errors easier. Throughout our studies, Java and unit testing with Junit has been used extensively, but this is older technology, which is time-consuming.

Although test-driven development is a thing, the usefulness depends on the project. If this project were to be deployed, it would probably be deployed through Netlify, or AWS and these platforms have built-in testing when an application gets built. If the project were to get any bigger, the group might consider setting up automated tests, but because of the scale of the project, the group decided that time would be better spent on other things.

7.2 Usefulness of Kanban in small teams.

As mentioned, the group has used a Kanban board to keep track of a backlog of tasks needed to finish the project. While the group see the usefulness in using the Kanban method, the tasks in this project were straightforward, and the group members knew what their tasks were without checking the Kanban board. The use of the board decreased further into the project. It was used quite frequently during the planning phase but could probably have been used more.

Some of the group members use a Kanban board in their part-time job, and the usefulness is clear once the team and application get bigger, but in a team of three, the usefulness isn't as great.

7.3 Limitations

The limitation with the most significant impact on the project was COVID-19. The communication between the group and project owner took a hit because everyone had to work from home, and this made it so that some of the requirements were misinterpreted and had to be changed along the way. It also affected the communication internally in the group. The development process worked fine but discussing certain parts of the report would have been better in person to avoid misunderstandings.

The lack of knowledge about HCI didn't affect the development process, but the group noticed its lack of knowledge about HCI when receiving feedback from user testing. Having had the knowledge would most likely result in a better product that is more user friendly.

7.4 Libraries vs native features

During the development process, different libraries are used to help the group develop the application better and faster. Some libraries help with the client-side styling, while some libraries help with the logic on the server. One library worth mentioning is Ant Design. Using this library has saved quite some time while developing the UI in our application. As mentioned previously, Ant Design provides pre-built components that the developer can import into the project. The components come with logic and design options, so it's easy to implement. It's almost a "drag and drop" solution for developing. If this project were to be styled with CSS and all logic made from scratch, it would have taken a lot longer. There are many cases where one would want to style components from scratch and create logic from scratch, but in this case, choosing to use Ant Design has drastically reduced the development time.

Another library that has saved the group some time developing is date-fns. date-fns simplifies one of the most challenging parts of programming, handling date and time. This library comes with many functions used for manipulating date and time in JavaScript. The group used this library to handle everything regarding date and time in both the frontend and backend.

Using date-fns speeds up date manipulations by a lot compared to using the built-in features in JavaScript. Without this library, the group would have spent a lot of time creating functions for all the date manipulation and formatting.

The use of these libraries has massively decreased development time. At the end of the day, the use of libraries depends on the task at hand, but libraries will often make the development process smoother and with less boilerplate code and more reusable components.

9 Conclusion

9.1 Project goals

In the process of the project, there were a lot of goals to accomplish along the way. Briefly summarised these goals included making an application where all security tasks are collected.
The different tasks are going to indicate how important and how close the deadline of a task is. Another goal for the application was to integrate with the Github API to get an overview of different accesses for the project owner organisation. Finally, there should be an option to upload files to the application for the users to read.

9.1.2 Were the goals reached?

All the goals mentioned above were reached in time, except the option for users to upload files. There were some difficulties when starting this integration as it needed a lot more research than first expected. It also required another database than first were picked, which would leave the group with a lot more work to change our solution entirely just for this implementation to work. All this would delay the project deadline and was not worth the risk of having an incomplete application.

9.2 Usability for others

The work done in this project could, in fact, be used by others if some simple things were done. There are two main cases where this project could benefit others; the first case is if some other needs an application for doing repetitive tasks, then this project could benefit them. To make this a reality, some changes would have been made. First, the project owner would have to approve to sell or give away the application. After that, a simplified version of the application could have been made where you only got the task tracking and task history modules. More people would benefit from the application with a simplified version as the project is now customized to fit the project owner's wishes.

In the second use case of the project, there would be those who want to make their application track repetitive tasks. Here they could have used a lot of the same logic and code as in the application made in this project. The applications source code would have to be published as public source code for anybody to see to make this happen. In this way, everyone thinking of making a similar application could get help from the source code given here.

One last helpful case this project can bring others is implementing the Github API into their application. As there was a lot of time researching implementing the Github API in this project, it could save some time for other developers later if they get their hands on the source code for the application. The same procedure as the last case with sharing the source code would have been done to accomplish this.

9.3 Further work

If the work was to be continued, there could have been done several improvements and new implementations to the program that were made. For instance, the group would first complete the incomplete remaining goals for the project. These goals include adding the opportunity to upload files to the program for the users to read. The file upload would give the end-user an option to share documents with the organisation's users for them to read.

Besides the remaining goals, the group would also suggest adding more functionality to the GitHub integration by adding and removing permissions for users to the company's GitHub repositories. The program would be more powerful and time-efficient for the end-user with these added functionalities than it already is.

Another helpful improvement to the program would be to add a login for each user and make restrictions for different user groups. In that way, "normal" users could only read documents provided to them and see tasks connected to them. In contrast, admin users could do everything such as look at permissions, adding and removing tasks, upload files and managing users.

Some other integration worth mentioning would be an integration to the Azure API. With this integration, the end-user could quickly overview the different servers, databases, and storage they got connected to the Azure platform.

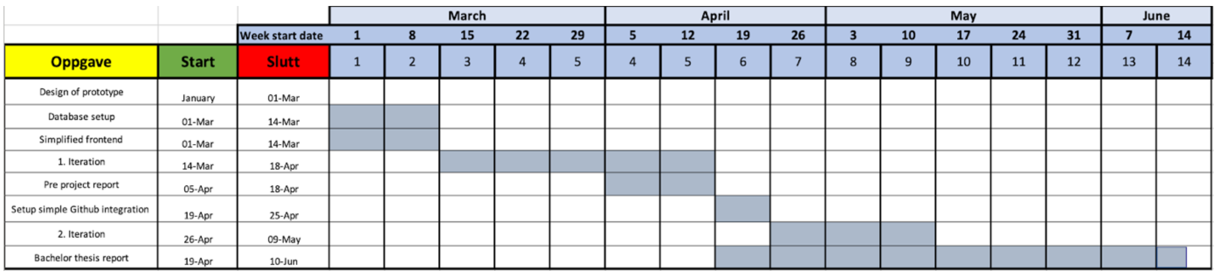
10 Bibliography

Bigio, M., 2016. *Introducing Hot Reloading.* [Online]
Available at: https://reactnative.dev/blog/2016/03/24/introducing-hot-reloading
[Accessed 2 May 2021].
Feoktistov, I., n.d. *Agile Software Development Lifecycle Phases Explained.* [Online]
Available at: https://relevant.software/blog/agile-software-development-lifecycle-phases-explained/
[Accessed 20 April 2021].
GeeksforGeeks, 2018. *GeeksforGeeks.* [Online]
Available at: https://www.geeksforgeeks.org/unified-modeling-language-uml-sequence-diagrams/
[Accessed May 2021].
GeeksforGeeks, 2018. *GeeksforGeeks.* [Online]
Available at: https://www.geeksforgeeks.org/unified-modeling-language-uml-activity-diagrams/
[Accessed May 2021].
Interaction Design Foundation, n.d. *Human-Computer Interaction (HCI).* [Online]
Available at: https://www.interaction-design.org/literature/topics/human-computer-interaction
[Accessed 15 May 2021].
Katsabas, D., Furnell, S. M. & Dowland, P. S. H., 2005. *Using Human Computer Interaction principles to promote usable security.* s.l.:s.n.

Mule, S. S. & Waykar, Y., 2015. *ROLE OF USE CASE DIAGRAM IN S/W DEVELOPMENT.* s.l.:International Journal of Management and Economics.
Node.js, n.d. *About | Node.js.* [Online]
Available at: https://nodejs.org/en/about/
[Accessed 17 April 2021].
Omniconvert, 2019. *What is User Testing?.* [Online]
Available at: https://www.omniconvert.com/what-is/user-testing/
[Accessed 18 May 2021].
Radigan, D., n.d. *Kanban - A breif introduction.* [Online]
Available at: https://www.atlassian.com/agile/kanban
[Accessed 2 April 2021].
React, n.d. *React- A JavaScript library for building user interfaces.* [Online]
Available at: https://reactjs.org
[Accessed 15 April 2021].
SQLite, n.d. *Appropriate Uses For SQLite.* [Online]
Available at: https://www.sqlite.org/whentouse.html
[Accessed 18 May 2021].
TypeScript, n.d. *TypeScript: Typed JavaScript at Any Scale.* [Online]
Available at: https://www.typescriptlang.org
[Accessed 15 April 2021].

11 Appendix

11.1 GANTT chart



			March					April				May					June	
		Week start date	1	8	15	22	29	5	12	19	26	3	10	17	24	31	7	14
Oppgave	Start	Slutt	1	2	3	4	5	4	5	6	7	8	9	10	11	12	13	14
Design of prototype	January	01-Mar																
Database setup	01-Mar	14-Mar	▨	▨														
Simplified frontend	01-Mar	14-Mar	▨	▨														
1. Iteration	14-Mar	18-Apr			▨	▨	▨	▨	▨									
Pre project report	05-Apr	18-Apr						▨	▨									
Setup simple Github integration	19-Apr	25-Apr								▨								
2. Iteration	26-Apr	09-May									▨	▨	▨					
Bachelor thesis report	19-Apr	10-Jun								▨	▨	▨	▨	▨	▨	▨	▨	▨

11.2 Risk list

ID	Risk	Category	Owner	Probability	Impact	Mitigation
1	Lack of time to get a working product.	Development, Project management	Glenn, Marius, Charlie	Low	High	Set aside more time for coding.
2	Lack of knowledge about APIs to finish Github iteration.	Development	Glenn, Marius, Charlie	Medium	High	Set aside enough time for learning about API implementation.
3	Errors while developing.	Development	Glenn, Marius, Charlie	High	Medium	Use TypeScript to set types for as much as possible. This will make the app less error prone in the long run.
4	Unable to satisfy the project owner.	Project management, Development	Glenn, Marius, Charlie	Low	Medium	Keep close communication with the project owner to make sure we include all the functions they want.
5	Miscommunication with project owner or within the group.	Project management	Glenn, Marius, Charlie	Low	Medium	Have frequent meetings to ensure everyone knows what they are doing.